Simulating on the D-Wave Two and Emulating its Behavior on an Ordinary Computer

von

Lukas Hobl

Masterarbeit in Physik

vorgelegt der

Fakultät für Mathematik, Informatik und Naturwissenschaften
der RWTH Aachen

im September 2015

angefertigt am

Jülich Supercomputing Center

bei

Prof. Dr. Kristel Michielsen

Contents

1 Introduction **3**

2 Adiabatic Quantum Computing **5**

2.1 Introduction 5

2.2 Quantum Annealing and Adiabatic Quantum Computation 6

2.3 Adiabatic Theorem and Landau-Zener Tunneling 8

3 Quantum Spin Dynamics **10**

3.1 Time-dependent Schrödinger Equation 10

3.2 General Aspects 11

3.3 Full Diagonalization Algorithm 12

3.4 Suzuki-Trotter Product-Formula Algorithm 13

3.5 Simulation of 2-SAT Problems 15

3.5.1 2-satisfiability (2-SAT) Problems 15

3.5.2 Mapping of 2-SAT Problems to the Ising Model 16

3.5.3 Simulation Results 17

4 Quantum Annealing and the D-Wave Two Processor **23**

4.1 Superconducting Flux Qubits 23

4.2 Inoperable Qubits, Calibration Errors and Programming Noise 26

4.3 Programming the D-Wave Two Processor 28

4.4 Quantum Annealing 28

5 Solving Random Spanning Trees on the D-Wave Two **30**

5.1 Randomly Generated Spanning Trees 30

5.2 Results 31

5.2.1 Z_2-Symmetry 31

5.2.2 Statistical Independence of Uncoupled Regions 35

5.2.3 Date/Time Dependency 38

6 Solving 2-SAT Problems on the D-Wave Two Processor **40**

6.1 Embedding 2-SAT Problems on the Chimera Graph 40

6.2 Parameters for Simulating the Quantum Annealing Process 41

6.3 Results for Solving the 2-SAT Problems 41

7 Conclusion **46**

A Graph of System 13 **50**

B Matlab Code for Operating on the D-Wave Two **51**

C C++ Code for Simulating the D-Wave Two Processor **52**

1 Introduction

Since the first theoretical proposals of a quantum computer in the 1980's and the subsequent discovery of several quantum algorithms suitable for execution on quantum computers in the 1990's, quantum computing has developed into a rapidly growing area of research. The discovery of algorithms like Shor's algorithm for number factoring and Grover's search algorithm [1, 2] put forward that quantum algorithms running on a quantum computer are able to perform specific computational tasks much faster than algorithms on a classical computer. The theoretical concept of a quantum Turing machine, also known as the universal quantum computer, was presented by David Deutsch suggesting that the internal state of a quantum computer could be controlled by quantum gates [3].

In general, quantum computers employ quantum mechanical processes such as quantum interference and/or quantum entanglement in order to perform computations. Instead of bits, as in a classical computer, quantum computers use quantum bits called qubits. Classical bits can only assume two classical distinct states, namely 0 and 1. The state of a qubit, however, can be described by a two-dimensional vector of length one in a two-dimensional vector space spanned by the basis vectors $|0\rangle$ and $|1\rangle$. Qubits can be in state $|0\rangle$, state $|1\rangle$ or in any superposition of the two basis states $a_0|0\rangle + a_1|1\rangle$ where a_0 and a_1 are complex numbers [4]. The complex numbers suggest that an infinite amount of information can be stored in a qubit. However, any actual measurement of a qubit will always yield either the result 0 or 1 i.e. the information encoded in the superposition is lost. In order for a quantum computer to be of any practical use, it needs to be able to store and process more than one single qubit. The internal state of a quantum computer consisting of L qubits is given by a unit vector in a D-dimensional space, wherein $D = 2^L$.

In a classical computer, information (i.e. bits) is processed by logical gates. In a quantum gate computer the qubits are manipulated by applying a series of quantum gates that correspond to unitary transformations acting on individual qubits. The internal state of a quantum computer evolves in time by applying a sequence of unitary transformations. Such a sequence is called a quantum algorithm. When all unitary transformations of a quantum algorithm have been executed, the time evolution of the quantum computer is interrupted by measuring the values of all individual qubits. The values for the qubits will either be 0 or 1 and the information encoded in the superposition of the state is lost. The computational power of a quantum computer results from the fact that each application of a unitary transformation can change all amplitudes of the unit vector simultaneously [4].

Building an actual quantum gate computer is a challenging task. The main problem is that during computation, quantum computers must be isolated from their surrounding environment, since quantum computers are susceptible to external noise which may lead

to decoherence [5]. The larger the number of qubits used in a quantum computer the more likely it is that the information encoded in the state of the quantum computer is lost into the surrounding environment. To the present day this fact restricts the number of qubits of a quantum gate computer to very small numbers.

In contrast to the limitations of quantum gate computers it is believed that quantum computers based on adiabatic quantum computation provide an inherent degree of robustness to decoherence [6]. In adiabatic quantum computing, a quantum system is prepared in a specific state and then evolved in time so that its final state holds the solution of the problem considered.

Due to the adiabatic quantum computers' robustness to decoherence, implementations using larger numbers of qubits have now become feasible. D-Wave Systems, Inc., a quantum computing company, has built such an adiabatic quantum computer. The newest model the D-Wave 2X employs over 1000 qubits. D-Wave Systems was founded in 1999 and in 2007 presented the first prototype of a quantum annealing processor, the Orion system employing a total of 16 qubits. The first commercially available system, the D-Wave One, was presented in 2010 employing 128 qubits [7]. Since the presentation of the D-Wave One, an intense discussion has developed in the quantum computation community as to whether the D-Wave processor actually uses quantum effects to find the solutions to problems and whether the expected quantum speedup could possibly be detected.

This work presents the results of optimazation problems solved on a D-Wave 2 Vesuvius V7 processor. The obtained results are discussed and compared to simulation results emulating the processor's physical behavior. Chapter 2 introduces the principle of adiabatic quantum computing. Chapter 3 presents a method for simulating the D-Wave Two processor by solving the time-dependent Schrödinger equation. Chapter 4 provides a technical description of the working principle of the D-Wave Two processor. Chapter 5 presents the results obtained by the D-Wave Two for random spanning tree problems. Lastly, in chapter 6, the results of 2-SAT problems obtained from the D-Wave Two processor are compared to the results of the simulations emulating the D-Wave Two processor.

2 Adiabatic Quantum Computing

In this chapter the concept of and motivation for adiabatic quantum computing is presented.

2.1 Introduction

Adiabatic quantum computing is based on the idea to use quantum mechanical processes to solve problems of optimization. In general, a problem of optimization arises when one has to choose the best available option from a host of options depending on many variables and conditions. In physics optimization represents a wide range of problems. Often the physical problem can be expressed in the form of an energy cost function, wherein the task of optimization then lies in finding the global minimum of this function. This corresponds to the state of the physical system with the lowest energy, called the ground state. The search for the ground state can be very challenging, as the cost function often depends on a large number of variables and constraints on the variables, which can lead to frustration in the system. Since the cost function may depend on a large number of variables, it possesses a large number of local minima, which makes the search for the ground state very complex [8, 9].

A well known physical optimization problem is finding the ground state of a system consisting of N Ising spins with frustrated interaction, a so called Ising spin glass. There are also many complex optimization problems known from computer science. Two examples are the traveling salesman problem, that is finding the shortest possible path that visits each city exactly once and the Boolean satisfiability problems (k-SAT problems), that are the problems of finding whether there is a solution in the form of an N-bit number that satisfies all Boolean imposed conditions [8].

Without going into detail, the computational complexity of these optimization problems can be classified. In computational science problem complexity is classified by the time and memory resources, that are needed to solve the worst case of the problem ensemble on a classical computer. The above given examples rank in the NP complexity class except for the two-dimensional Ising spin glass and the 2-SAT problem, which are both ranked P. For the problems belonging to the class P of polynomially solvable problems some algorithm can provide a solution in polynomial time. For the problems in the class of non-deterministic polynomially solvable problems NP it is impossible to verify the solution in polynomial time [10, 11].

The first idea to use the mechanism of annealing in simulations for optimization problems was introduced by Kirkpatrick [12] in 1983. The idea of simulated annealing (SA) is to solve the optimization problems by introducing a temperature variable, which is slowly lowered during a Monte Carlo simulation. Starting at a high temperature at the beginning

of the annealing process, the system is allowed to explore the whole configuration space of the problem. By slowly reducing the value of the temperature variable, the system will not be trapped in local minima, as the system is able to escape from the local minima by thermal fluctuations [8, 9].

The idea of quantum annealing (QA) and adiabatic quantum computing is based on the idea of simulated annealing but instead of employing thermal fluctuations, the method employs quantum fluctuations as described in the next section.

2.2 Quantum Annealing and Adiabatic Quantum Computation

Quantum annealing in contrast to simulated annealing is based on the idea that instead of using thermal fluctuations to escape local minima, quantum fluctuations are introduced to allow the system to escape from local minima by tunneling through the potential barriers.

The quantum system at the beginning of the quantum annealing process is prepared in the product state of all spin states, which is the ground state of the initial Hamiltonian as a strong transverse field is applied to introduce quantum fluctuations, which allows for quantum tunneling between states. During the so called annealing sweep the transverse field is gradually turned off. If the transverse field is turned off slowly enough, the system will evolve adiabatically and therefore stay in the ground state of the Hamiltonian. Adiabatic quantum computation can be seen as a special case of quantum annealing [9, 13]. When the transverse field is finally turned off completely the system should have reached the ground state of the classical Ising model that encodes the solution to the original optimization problem.

This idea is now presented in the context of the transverse Ising model, as this model will be the model studied throughout this work. The Hamiltonian of the classical Ising model with N spins can be written as

$$H_{cl} = -\sum_{i,j}^{N} J_{ij}\sigma_i^z\sigma_j^z - \sum_{i}^{N} h_i^z\sigma_i^z, \tag{1}$$

where the σ_i^z represent the spin projections along either the $+z$ or $-z$ direction and take values +1 (spin up) and -1 (spin down), respectively [10, 14]. Between two Ising spins located on lattice sites i and j there can be an interaction J_{ij} which will favor that the spins align parallel if $J_{ij} > 0$ (ferromagnetic interaction) or anti-parallel if $J_{ij} < 0$ (anti-ferromagnetic interaction). Further, every Ising spin σ_i^z can interact with an external field h_i. The task of optimization now lies in finding the state (configuration of all σ_i^z) that minimizes the energy cost for a given set of couplings J_{ij} and fields h_i^z.

In order to introduce quantum fluctuations to the classical Ising system, a transverse

field in the x-direction is applied. This results in the Hamiltonian

$$H = H_{cl} + \sum_{i=1}^{N} h_i^x \sigma_i^x, \tag{2}$$

where σ_i^α with α =x,y,z denote the Pauli spin matrices

$$\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma^y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{3}$$

The vectors spanning the basis for the Ising spins are defined as $|\uparrow\rangle = \begin{pmatrix} 1 & 0 \end{pmatrix}^{\mathrm{T}}$ and $|\downarrow\rangle = \begin{pmatrix} 0 & 1 \end{pmatrix}^{\mathrm{T}}$. Note that to introduce quantum fluctuations basically any term that does not commute with the classical Hamiltonian can be used (e. g. J couplings in the x-direction). The Hamiltonian used for adiabatic quantum computation reads

$$\begin{aligned} H(t) &= \lambda(t) H_{cl} + (1 - \lambda(t)) H_{trans} \\ &= \lambda(t) H_{cl} - (1 - \lambda(t)) \sum_i \sigma_i^x. \end{aligned} \tag{4}$$

During the course of annealing $\lambda = t/\tau$, where τ is the total annealing time, is slowly turned on (see Fig. 1) [15]. At $t = 0$, $\lambda = 0$ so that the quantum fluctuations dominate. The ground state of the system at $t = 0$ is given by the product state of the spins in the x-direction. Since the ground state is known at $t = 0$, the quantum system can be easily initialized in this state. If during the annealing course λ is turned on slowly enough, the system will evolve adiabatically and will therefore by the adiabatic theorem at all time remain in the ground state [16, 17]. At the end of the annealing sweep $\lambda = 1$ the Hamiltonian defined by Eq. (4) is reduced to H_{cl}. If the evolution of the quantum system is adiabatic, then the final state of the system is the ground state of the classical Hamiltonian H_{cl}. As H_{cl} represents the function to be optimized, its ground state encodes the optimal solution.

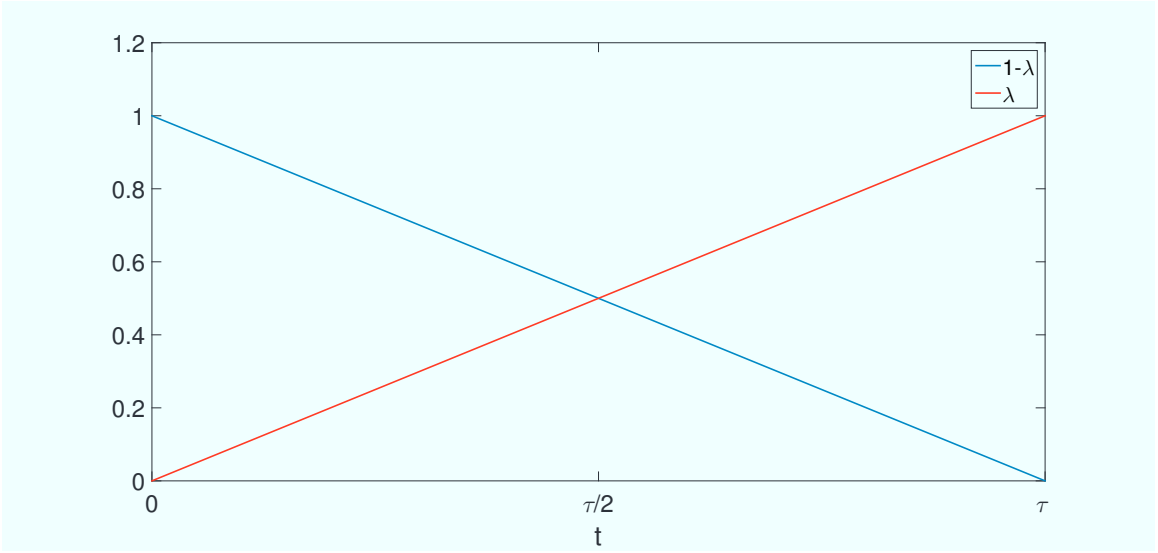


Figure 1: Linear annealing schedule used in the quantum annealing simulations. The annealing parameter is defined as $\lambda(t) = t/\tau$, where τ denotes the total annealing time.

2.3 Adiabatic Theorem and Landau-Zener Tunneling

In order to ensure, that at the end of the annealing sweep the final state is the ground state of the classical Hamiltonian, the evolution needs to evolve adiabatically. The adiabatic theorem states that if a quantum system is in an instantaneous eigenstate of a Hamiltonian $H(t)$ at some point in time it will remain in this eigenstate at all time, if the Hamiltonian $H(t)$ is varied sufficiently slow [17]. In the present case of quantum annealing, this can be ensured by taking sufficiently long annealing times τ. More precisely, the adiabatic theorem states that for a quantum system with a non-degenerate spectrum and a minimum energy gap Δ_{min} between the ground state and the first excited state the annealing time must satisfy the condition

$$\tau \gg \frac{|\langle H(t)\rangle|_{max}}{\Delta^2_{min}}, \tag{5}$$

with

$$|\langle H(t)\rangle|_{max} = \max_{0\leq t\leq\tau}\left(\left|\left\langle \phi_0(t)\left|\frac{dH(t)}{d\lambda}\right|\phi_1(t)\right\rangle\right|\right), \tag{6}$$

$$\Delta^2_{min} = \min_{0\leq t\leq\tau}\left[\Delta^2(t)\right], \tag{7}$$

where ϕ_0 and ϕ_1 are the instantaneous ground state and first excited state of the Hamiltonian during the annealing sweep at time t and $\Delta(t)$ is the instantaneous gap between the ground state energy of the state ϕ_0 and the energy of the first excited state ϕ_1 at time t [9, 18].

For a two-level quantum system Landau-Zener theory gives an estimate of the probability for a diabatic evolution (a non-adiabatic evolution), which means tunneling out

of the ground state into an excited state during the annealing sweep. The Landau-Zener formula can also be used for more general quantum systems (with more than two energy levels). Its application is well justified for a quantum system, for which the energy gap between its first excited state and its second excited state is relatively large compared to the gap between its ground state and its first excited state and if the first excited state is non-degenerate. Thus, assuming that the Landau-Zener formula gives at least a good indication of the probability of a diabatic evolution of the quantum system, then the probability of finding the ground state with quantum annealing is given by $P_{adiabatic} = 1 - P_{diabatic}$, where $P_{diabatic}$ is given by [19, 20, 21]

$$\begin{aligned} P_{diabatic} &= \exp(-\tau/\tau_c) \\ \tau_c &= (\hbar\alpha(\lambda)\Gamma_0) \,/\, \left(2\pi\Delta^2_{\text{min}}\right). \end{aligned} \tag{8}$$

As Eq. (8) shows, the probability for a diabatic evolution decreases for longer annealing times τ. Further, as can be seen from the characteristic tunneling time τ_c, the probability decreases with the minimum gap squared Δ^2_{min}. The characteristic tunneling time τ_c also depends on the relative slope α of the two levels as a function of $\lambda(t)$ and the amplitude of the transverse field Γ_0 at time $t = 0$ (in Eq. (4) $\Gamma_0 = 1$).

From the Landau-Zener formula it becomes apparent that the complexity of finding the solution to an optimization problem with quantum annealing depends on the minimum gap Δ_{min}. Therefore it is possible for a fixed annealing time τ to predict the success probability of quantum annealing if the minimum gap Δ_{min} for the specific problem is known.

3 Quantum Spin Dynamics

In this chapter two numerical methods to solve the time-dependent Schrödinger equation (TDSE) for quantum spin systems are discussed. By solving the TDSE, the dynamical behavior of the respective system can be computed. In sections 3.1-3.4 we closely follow Ref. [4]. Section 3.5 is devoted to the usage of both algorithms to simulate 2-SAT problems.

3.1 Time-dependent Schrödinger Equation

In order to study the dynamical behavior of a quantum spin model described by a Hamiltonian $H(t)$ one has to solve the TDSE

$$-i\frac{\partial}{\partial t}|\Phi(t)\rangle = H(t)|\Phi(\mathrm{t})\rangle, \tag{9}$$

in units such that $\hbar = 1$ and where $|\Phi(t)\rangle$ denotes the wave function of the system. The solution of this equation reads

$$\begin{aligned} |\Phi(t+\Delta t) \quad \rangle &= \quad U(t+\Delta t,t)|\Phi(t)\rangle \\ &= \quad \exp_+\left(-i\int H(u)du\right)|\Phi(t)\rangle, \end{aligned} \tag{10}$$

where Δt is the time step and the time evolution operator $U(t+\Delta t,t)$ is a unitary matrix which transforms the state $|\Phi(t)\rangle$ to $|\Phi(t+\Delta t)\rangle$. In order to compute the time evolution numerically, the time is discretized. To ensure the unitarity of the time evolution operator the time interval $[t,\, t+\Delta t]$ is split into n subintervals and $H(t)$ is assumed to be piecewise constant in these time intervals. For the subinterval $[t_k,t_{k+1}]$ the time t_k is defined as $t_k = t + \sum_{j=0}^{k}\tau_j$, where τ_j is the jth time interval over which $H(t)$ is constant. Then $U(t+\Delta t,t)$ can be written as

$$U(t+\Delta t,t) = U(t_n,t_{n-1})U(t_{n-1},t_{n-2})...U(t_1,t_0), \tag{11}$$

with $U(t_j,t_{j-1}) = \exp(-i\tau_j H(t_{j-1}+\tau_j/2))$. The number of time steps n that are needed to ensure that the approximation is unitary, must be chosen carefully. If n is chosen too big, the solution will not be correct and if chosen too small, it will take up more computational resources than necessary. From Eq. (10) it becomes clear, that to solve the TDSE one has to compute matrix exponentials. The solution is an approximation to the exact solution which becomes better with an increasing number of time intervals n.

In this work two methods for calculating $U(t)$ are discussed, the Suzuki-Trotter Product-Formula algorithm and the Full Diagonalization algorithm. Before these two methods are

explained, conventions about the notation and some general aspects are given in the next section.

3.2 General Aspects

In order to calculate the evolution of the quantum system with N spins described by Eq. (4) it is necessary to define the wave function $|\Phi\rangle$. The wave function is given by the linear superposition of the direct product states of the N single spin states in the z-basis, which yields a 2^N state vector. The wave function is given by

$$\begin{aligned} |\Phi\rangle &= a(\uparrow\uparrow \ ... \ \uparrow)|\uparrow\uparrow \ ... \ \uparrow\rangle + a(\downarrow\uparrow \ ... \ \uparrow)|\downarrow\uparrow \ ... \ \uparrow\rangle + ... + a(\uparrow\downarrow \ ... \ \downarrow)|\uparrow\downarrow \ ... \ \downarrow\rangle \\ &\quad + a(\downarrow\downarrow \ ... \ \downarrow)|\downarrow\downarrow \ ... \ \downarrow\rangle, \end{aligned} \tag{12}$$

where the complex amplitudes $a(\uparrow\uparrow \ ... \ \uparrow)$,...,$a(\downarrow\downarrow \ ... \ \downarrow)$ completely specify the state of the quantum system. The wave function should be normalized so that $\langle\Phi|\Phi\rangle = 1$. For this condition to hold the sum over the squares of the absolute values of the amplitudes must be $\sum_{\sigma_1...\sigma_N=\uparrow,\downarrow} |a(\sigma_1...\sigma_N)|^2 = 1$. In this representation the first label corresponds to the first spin of the system.

For the simulation it is more convenient to express the states as binary numbers where $|\uparrow\rangle$ corresponds to $|0\rangle$ and $|\downarrow\rangle$ corresponds to $|1\rangle$. Another useful convention from computer science is to consider the first spin (qubit) as the least significant bit of an integer index, that runs from 0 to 2^{N-1}. Using this notation the state $|\Phi\rangle$ defined by Eq. (12) can be rewritten as

$$|\Phi\rangle = a(0...00)|0...00\rangle + a(0...01)|0...01\rangle + ... + a(1...10)|1...10\rangle + a(1...11)|1...11\rangle. \tag{13}$$

This binary representation will be used throughout this work. For later use it is helpful to consider the effect of applying each of these terms on the wave function $|\Phi\rangle$. In the Hamiltonian defined by Eq. (4) there are three different kinds of terms, namely two single spin terms σ_i^x, σ_i^z and one two spin term $\sigma_i^z\sigma_j^z$.

Applying σ_i^z on $|\Phi\rangle$ ($|\Phi'\rangle = \sigma_i^z|\Phi\rangle$) reverses the sign of all amplitudes a of the states of $|\Phi\rangle$ for which the ith bit of the vector index has the value 1 $\left(|1\rangle = |\downarrow\rangle = \begin{pmatrix} 0 & 1 \end{pmatrix}^T\right)$. This yields for the amplitudes

$$\begin{aligned} a'(\bullet...\bullet \ 0 \ \bullet...\bullet) &= +a(\bullet...\bullet \ 0 \ \bullet...\bullet) \\ a'(\bullet...\bullet \ 1 \ \bullet...\bullet) &= -a(\bullet...\bullet \ 1 \ \bullet...\bullet), \end{aligned} \tag{14}$$

where $\bullet$ stands for the unchanged bits.

Applying σ_i^x on $|\Phi\rangle$ ($|\Phi'\rangle = \sigma_i^x|\Phi\rangle$) interchanges the up and down states. For the

amplitudes it follows that they are obtained by swapping pairs of $|\Phi\rangle$. This yields

$$\begin{aligned} a'(\bullet ... \bullet\ 0\ \bullet ... \bullet) &= +a(\bullet ... \bullet\ 1\ \bullet ... \bullet) \\ a'(\bullet ... \bullet\ 1\ \bullet ... \bullet) &= +a(\bullet ... \bullet\ 0\ \bullet ... \bullet). \end{aligned} \tag{15}$$

Applying the two-spin term $\sigma_i^z \sigma_j^z$ on $|\Phi\rangle$ can be calculated analytically. The signs of the amplitudes are now only changed if the ith and the jth bit of the amplitudes are different. This yields

$$\begin{aligned} a''(\bullet ... \bullet\ 0\ \bullet ... \bullet\ 0\ \bullet ... \bullet) &= +a(\bullet ... \bullet\ 0\ \bullet ... \bullet\ 0\ \bullet ... \bullet) \\ a''(\bullet ... \bullet\ 1\ \bullet ... \bullet\ 0\ \bullet ... \bullet) &= -a(\bullet ... \bullet\ 1\ \bullet ... \bullet\ 0\ \bullet ... \bullet) \\ a''(\bullet ... \bullet\ 0\ \bullet ... \bullet\ 1\ \bullet ... \bullet) &= -a(\bullet ... \bullet\ 0\ \bullet ... \bullet\ 1\ \bullet ... \bullet) \\ a''(\bullet ... \bullet\ 1\ \bullet ... \bullet\ 1\ \bullet ... \bullet) &= +a(\bullet ... \bullet\ 1\ \bullet ... \bullet\ 1\ \bullet ... \bullet). \end{aligned} \tag{16}$$

With the single and two-spin operations discussed here, the calculation of $|\Phi'\rangle = H|\Phi\rangle$ can be performed easily [4]. This is necessary for the full diagonalization of the Hamiltonian which is discussed in the next section.

3.3 Full Diagonalization Algorithm

The full diagonalization approach is based on the fact that the Hamiltonian H is an Hermitian operator represented by a $D \times D$ Hermitian matrix with $D = 2^N$. This implies that the matrix H has a complete set of eigenvectors and real valued eigenvalues. The diagonal matrix of the eigenvalues Λ is then given by $V^\dagger H V = \Lambda$, where V is the unitary matrix of the eigenvectors. The unitary time evolution can then be calculated with $U(t) = \exp(-itH) = V \exp(-it\Lambda)V^\dagger$. This means that if V and Λ are known, the time evolution can be obtained by matrix multiplications. Therefore, the most straightforward approach to compute the time evolution $U(t)$ is by diagonalizing H numerically.

The matrix elements of H can be calculated by repeatedly using $|\Phi'\rangle = H|\Phi\rangle$ for all basis vectors $|\Phi\rangle$. The new vectors $|\Phi'\rangle$ then give the columns of the matrix H. With the obtained matrix H, the time evolution of the system can be calculated.

This exact approach is strongly limited by computational resources and therefore can only be applied for small systems. For small systems it can however be used to check the results of the Spin Dynamics Simulation based on the Suzuki-Trotter Product-Formula algorithm introduced in the next section, in order to ensure the correct implementation of the algorithm.

3.4 Suzuki-Trotter Product-Formula Algorithm

The Suzuki-Trotter Product-Formula algorithm is based on the expansion of the unitary matrix exponential [22]

$$U(t) = \exp(-itH) = \exp(-it(H_1 + H_2 + ... + H_K)) = \lim_{m \longrightarrow \infty} \left(\prod_{k=1}^{K} \exp(-itH_k/m) \right)^m . \tag{17}$$

Equation (17) implies that

$$\widetilde{U}_1(t) = e^{-itH_1} e^{-itH_2} ... e^{-itH_K}, \tag{18}$$

is a good approximation to $U(t)$, if the time step t is chosen sufficiently small. The Taylor series of $U(t)$ and $\widetilde{U}_1(t)$ are identical up to the first order and if all H_i for $i = 1, ..., K$ are Hermitian the algorithm is unconditionally stable. $\widetilde{U}_1(t)$ is therefore a first order approximation of $U(t)$. The accuracy of the approximation can be increased by going to higher orders. For the simulations in this work, the second order approximation for $U(t)$ is used

$$\widetilde{U}_2(t) = \widetilde{U}_1^{\mathrm{T}}(t/2)\widetilde{U}_1(t/2) = e^{-itH_K/2} ... e^{-itH_1/2} e^{-itH_1/2} ... e^{-itH_K/2}. \tag{19}$$

Since $\widetilde{U}_1(t)$ is unitary, $\widetilde{U}_2(t)$ is also unitary and from Eq. (19) it is clear that to calculate the time evolution it is necessary to calculate the matrix exponentials $\exp(-itH_k/2)$. Therefore, the main question is how to choose the Hermitian matrix H_k so that the matrix exponential can be calculated efficiently .

In this work the Hamiltonian for the quantum system is given by Eq. (4). The Hamiltonian contains one two-spin term and two single spin terms. It is convenient to compose $U(t)$ (see Eq. (17)) into [23]

$$\widetilde{U}(t) \;=\; \exp(-itH_1/2) \exp(-itH_2) \exp(-itH_1/2), \tag{20}$$

which is a second order approximation with

$$\exp(-itH_1/2) \;=\; \exp\left(-it \left(-\sum_{j=1}^{N} \sum_{\alpha=x,z} h_j^{\alpha}(t) \sigma_j^{\alpha} \right) /2 \right) \tag{21}$$

$$\exp(-itH_2) \;=\; \exp\left(-it \left(-\sum_{jk=1}^{N} J_{jk}(t) \sigma_j^z \sigma_k^z \right) \right), \tag{22}$$

where $h_j^x(t) = (1 - \lambda(t))h_j^x$, $h_j^z(t) = \lambda(t)h_j^z$ and $J_{jk}(t) = \lambda(t)J_{jk}$. Since the spin operators with different labels commute the sum over the spins can be written as a product.

Equation (21) can be rewritten as

$$\exp\left(-it\left(-\sum_{j=1}^{N}\sum_{\alpha=x,z}h_j^{\alpha}(t)\sigma_j^{\alpha}\right)/2\right)=\prod_{j=1}^{N}\exp\left(\frac{it}{2}\sum_{\alpha=x,z}h_j^{\alpha}(t)\sigma_j^{\alpha}\right). \tag{23}$$

Each factor in this expression can be calculated analytically using

$$e^{i\boldsymbol{v}\cdot\boldsymbol{S}}=\mathbb{I}\cos(\frac{v}{2})+\frac{2i\boldsymbol{v}\cdot\boldsymbol{S}}{v}\sin(\frac{v}{2}), \tag{24}$$

where $e^{i\boldsymbol{v}\cdot\boldsymbol{S}}$ describes a rotation of the vector $\boldsymbol{S}$ about the vector $\boldsymbol{v}$ and where v denotes the norm of $\boldsymbol{v}$. Here $\exp(-itH_1/2)$ describes a rotation of spin j about the vector $\boldsymbol{h_j}=\big(h_j^x(t),0,h_j^z(t)\big)$. The analytical solution yields

$$e^{it\boldsymbol{\sigma}_j\cdot\boldsymbol{h}_j}=\begin{pmatrix}\cos\left(\frac{th_j}{2}\right)+\frac{ih_j^z(t)}{h_j}\sin\left(\frac{th_j}{2}\right) & \frac{ih_j^x(t)}{h_j}\sin\left(\frac{th_j}{2}\right)\\ \frac{ih_j^x(t)}{h_j}\sin\left(\frac{th_j}{2}\right) & \cos\left(\frac{th_j}{2}\right)-\frac{ih_j^z(t)}{h_j}\sin\left(\frac{th_j}{2}\right)\end{pmatrix}, \tag{25}$$

where h_j is the norm of the vector $\boldsymbol{h_j}$. Equation (25) shows that the time evolution comes down to calculating the elements of 2×2 matrices and applying these matrices on the state $|\Phi\rangle$. This can be done by picking the corresponding pairs (cf. section 3.2) of states and applying Eq. (25) thereon.

The two-spin time evolution operator given by Eq. (22) can also be decomposed into products due to the fact that spin operators with different labels commute

$$\exp\left(it\sum_{j,k=1}^{N}J_{jk}\sigma_j^z\sigma_k^z\right)=\prod_{j,k=1}^{L}\exp\left(it(J_{jk}\sigma_j^z\sigma_k^z)\right). \tag{26}$$

This expression can also be worked out analytically and yields

$$e^{itJ_{jk}\sigma_j^z\sigma_k^z}=\begin{pmatrix}e^{itJ_{jk}} & 0 & 0 & 0\\ 0 & e^{-itJ_{jk}} & 0 & 0\\ 0 & 0 & e^{-itJ_{jk}} & 0\\ 0 & 0 & 0 & e^{itJ_{jk}}\end{pmatrix}. \tag{27}$$

The time evolution for the two-spin coupling Hamiltonian H_2 consists of calculating the matrix in Eq. (27) and applying the rotation matrices on $|\Phi\rangle$ by picking the four respective states (cf. section 3.2). With this method, it is now possible to apply all parts of $\widetilde{U}_2(t)$ on $|\Phi(t)\rangle$ and therefore the time evolution of the wave function can be calculated.

With the two algorithms (Full Diagonalization and Suzuki-Trotter Product-Formula algorithm) it is now possible to simulate quantum annealing on a classical computer and to compare the results to the results obtained from the D-Wave Two adiabatic quantum

computer, which will be discussed in the next chapter.

3.5 Simulation of 2-SAT Problems

For the purpose of understanding 2-SAT problems an 8 spin 2-SAT problem is considered and the results of the Full Diagonalization and Suzuki-Trotter Product-Formula algorithm are shown in the following subsection.

3.5.1 2-satisfiability (2-SAT) Problems

The task in the 2-SAT problem is to determine whether a set of binary variables with a set of constraints (clauses) on pairs of variables (therefore the name 2-SAT) can be chosen such that all constraints are fulfilled. The problem is known to be of the complexity class P in contrast to 3-SAT and higher SAT problems that are NP-complete [8, 11].

The 2-SAT problems considered in this work have been obtained by a Monte Carlo search algorithm that is designed to find the computationally hardest problems sampled from the ensemble of 2-SAT problems with unique satisfying assignments (non-degenerate ground states) and a clause to spin ratio $\alpha = (N + 1)/N$ where N is the number of binary variables or spins in the spin 2-SAT problems. This ratio is the smallest possible ratio for 2-SAT problems that have a unique satisfying assignment. In this work 2-SAT problems with 8, 12, 18 spins will be considered [24].

The computational hardness of the problems is reflected in the very small minimal energy gaps between the ground state energy and the energy of the first-excited state during the annealing process of the 2-SAT problem and a high density of states of the first excited state of the 2-SAT problems. The small minimal gaps make the problems hard to solve by quantum annealing as described by the Landau-Zener formula Eq. (8). The high density of states of the first excited state makes a classical search hard as the probability of getting trapped in a local minimum during the course of simulated annealing increases linearly with the degeneracy (or density of states) of the first excited state.

This hardness is the key feature of the problems we study as they are hard to solve by quantum annealing even so the number of spins is still that small that the solution can be obtained easily by a brute force method. The small number of spins also allows for calculating the gaps and by that the computational complexity for quantum annealing of each problem is known.

3.5.2 Mapping of 2-SAT Problems to the Ising Model

2SAT problems are defined by binary variables x_i and clauses on these variables. Here we consider the following 2-SAT problem with 8 binary variables and 9 clauses [24]

$$(x_6 \vee x_3) \wedge (x_5 \vee \neg x_6) \wedge (x_8 \vee \neg x_4) \wedge (8_8 \vee \neg x_7) \wedge (x_1 \vee \neg x_3) \wedge \\ \wedge(\neg x_5 \vee \neg x_1) \wedge (x_6 \vee \neg x_3) \wedge (\neg x_8 \vee x_3) \wedge (\neg x_2 \vee \neg x_5), \tag{28}$$

where $\wedge$, $\vee$ and $\neg$ denote the logical AND, OR and negation, respectively. To give a better understanding of the problem, the first two conditions $(x_6 \vee x_3)$ and $(x_5 \vee \neg x_6)$ are discussed. The first condition $(x_6 \vee x_3)$ is fulfilled if either x_6 or x_3 is true (has a value of 1) or both are true. The second clause $(x_5 \vee \neg x_6)$ is fulfilled if either x_5 is true (has a value of 1) or x_6 is not true (has a value of 0) or if x_5 is true and x_6 is not true.

In order to be able to use the quantum annealing algorithm to solve this type of problems, the problems have to be mapped to the Ising model. As mentioned in the first chapter, an Ising spin can take the values 1 and -1. Therefore it is quite natural to identify the Ising spins as Boolean or binary variables where 1 stands for true (1) and -1 for false (0). The mapping is shown in Fig. 2 for the first two clauses $(x_6 \vee x_3)$, $(x_5 \vee \neg x_6)$ but is the same for all 9 clauses.

	√	√	√	-
x_6	1	1	0	0
x_3	1	0	1	0

$\Rightarrow$

	√	√	√	-
σ_6	1	1	-1	-1
σ_3	1	-1	1	-1
m=$\sigma_6 + \sigma_3$	2	0	0	-2

	√	√	√	-
x_5	1	1	0	0
$\neg x_6$	0	1	0	1

$\Rightarrow$

	√	√	√	-
σ_5	1	1	-1	-1
σ_6	-1	1	-1	1
m=$\sigma_5 - \sigma_6$	2	0	0	-2

Figure 2: Mapping of the 2-SAT clauses to the Ising variables. A negation of a binary variable adds a minus to the Ising spin.

For one clause, an Ising Hamiltonian can now be constructed with a magnetization m, so that the solution of the 2-SAT problem is the ground state of this Ising Hamiltonian. The Hamiltonian with magnetization $m = \sigma_i + \sigma_j$ is given by

$$\begin{aligned} H &= m \cdot (m-2) \\ &= \sigma_i^2 + \sigma_j^2 + 2\sigma_i\sigma_j - 2\sigma_i - 2\sigma_j \\ &= 2\sigma_i\sigma_j - 2\sigma_i - 2\sigma_j + \text{const.} \end{aligned} \tag{29}$$

Comparison with the classical Hamiltonian given in Eq. (1) allows identification of

the fields h_i^z and couplings J_{ij} to be $h_i^z = 2$, $h_j^z = 2$ and $J_{ij} = -2$. This procedure can be repeated for all clauses and gives a set of h_i^z and J_{ij}. The corresponding Ising spin Hamiltonian has the solution of the 2-SAT problem as ground state. The mapping to the Ising model allows to solve the 2-SAT problem with quantum annealing on the D-Wave Two processor and by quantum annealing simulations on a classical computer.

Results of the Full Diagonalization and Suzuki-Trotter Product-Formula algorithm of the 8 spin 2-SAT problem defined by Eq. (28) are presented in the next subsection.

3.5.3 Simulation Results

The Full Diagonalization algorithm allows for calculating the energy spectrum of the system during the annealing run. Figures 3 and 4 show the energy spectrum of the 8 spin 2-SAT problem given in Eq. (28) during the annealing run from $\lambda = 0$ to $\lambda = 1$ and show that the ground state is unique. From the spectrum it can be seen that during the annealing run there is a critical λ_{crit} at which the energy gap between the ground state and the first excited state becomes very small. The minimum gap Δ for this particular problem is 0.3973. The gaps of all 8 spin problems can be obtained by this method. For larger problems (in this work 12 and 18 spin problems) different algorithms can be used to obtain the gaps, as for example the Lanczos algorithm.

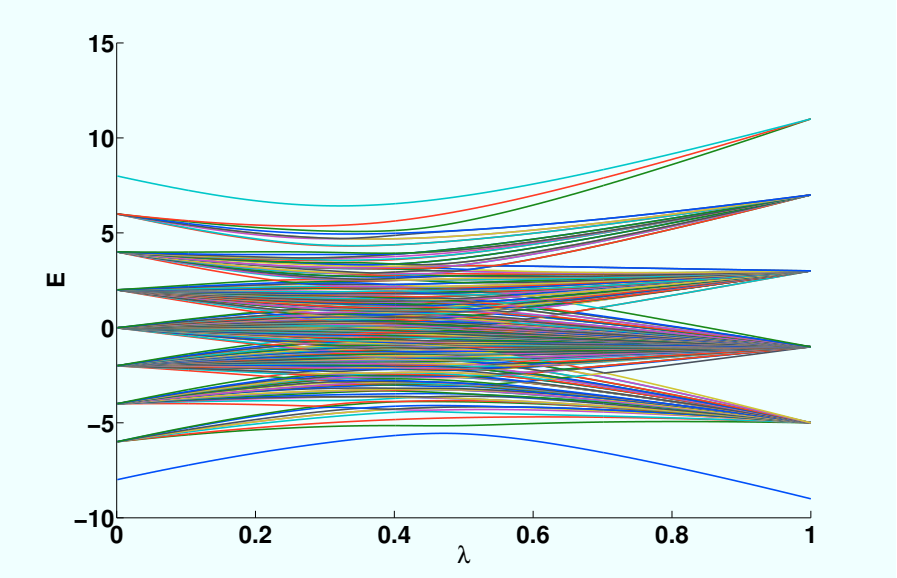


Figure 3: Full energy spectrum of the 256 states of the 8 spin 2-SAT problem defined by Eq. (28) as a function of the annealing parameter λ.

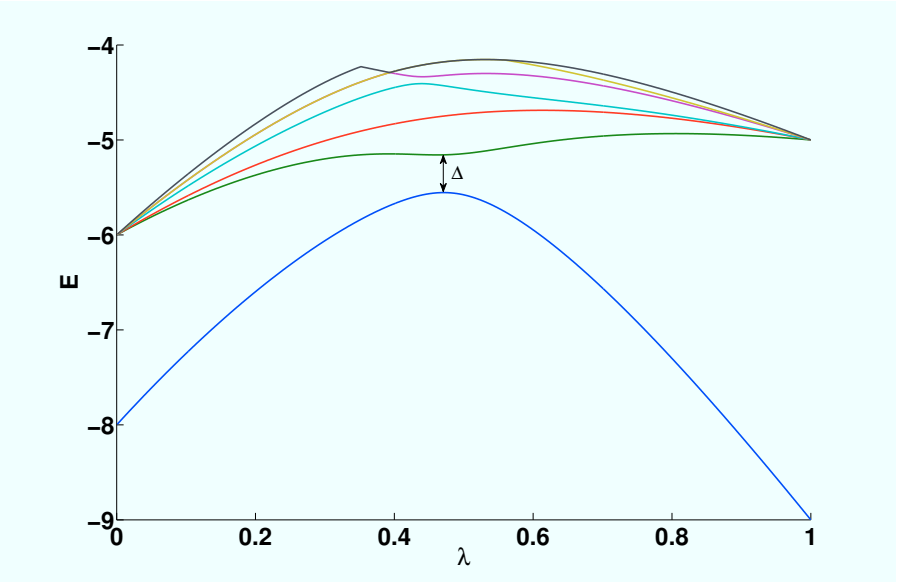


Figure 4: Spectrum of the lowest lying energy states for the same problem as used for Fig. 3. The ground state (blue) is unique and the minimum gap between the ground state and the first excited state (green) is indicated by Δ.

The probability, that the simulated spin system is in the ground state, is given by the overlap of the ground state wave function $\Phi_{\text{GS}}(\tau)$ at the end of the annealing sweep and the exact ground state ψ_{GS}, $\langle\Phi_{\text{GS}}(\tau)|\psi_{\text{GS}}\rangle$. According to Eq. (8) the probability for an adiabatic evolution (successful annealing run) is proportional to $P \propto 1 - e^{-\tau}$. Figure 5 shows the overlap $\langle\Phi_{\text{GS}}(\tau)|\psi_{\text{GS}}\rangle$ for various annealing times τ obtained by the Suzuki-Trotter algorithm. The figure shows that with longer annealing times the probability for finding the exact ground state increases and converges to 1. This fact also translates into a decreasing residual energy per spin, which is defined by

$$\epsilon_{res} = \frac{E_{sys} - E_{GS}}{N}, \tag{30}$$

with the energy of the ground state E_{GS} and the energy of the system $E_{sys} = \langle\Phi(\tau)|H|\Phi(\tau)\rangle$ at the end of the annealing run. As the probability for the system being in the ground state increases with the annealing time τ, the residual energy decreases and converges to 0, as the energy of the system approaches the ground state energy with longer annealing times (see Fig. 6).

Further the Landau-Zener Formula Eq. (8), shows that the probability of finding the ground state (adiabatic evolution) is proportional to $P \propto 1 - e^{-\Delta^2}$. Therefore problems with larger gaps Δ should have a higher probability to be solved than problems with smaller gaps. All gaps of the 8-spin 2-SAT problems can be computed. Therefore it is possible to show the scaling behavior of P with the gap Δ as shown in Fig. 7 for an annealing time $\tau = 10$. The figure indeed shows that with larger gaps the probability for

finding the solution increases.

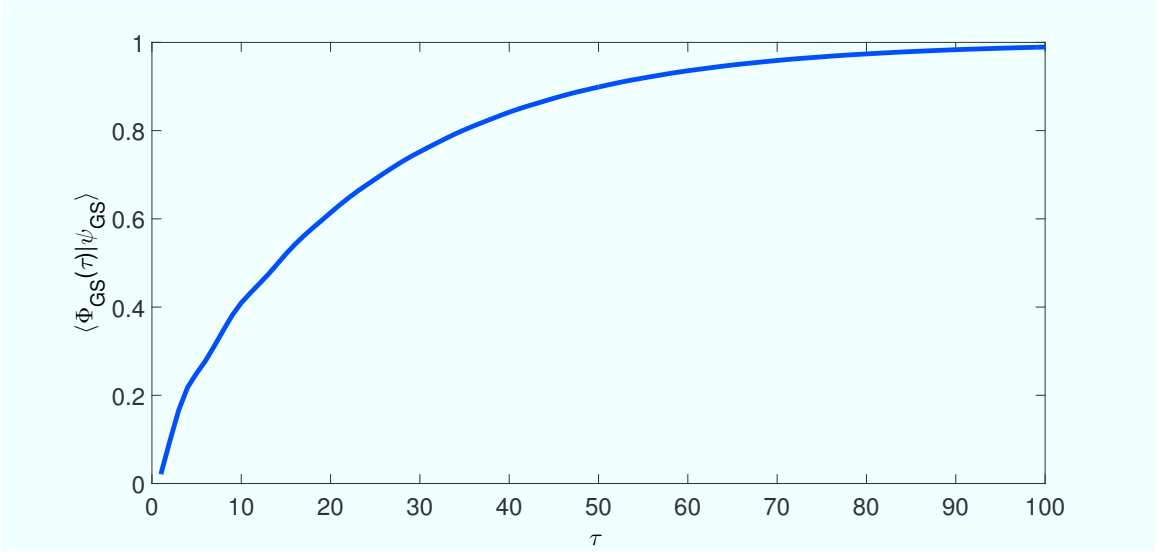


Figure 5: Overlap$\langle\Phi_{\rm GS}(\tau)|\psi_{\rm GS}\rangle$ of the ground state wave function $\Phi_{\rm GS}(\tau)$ at the end of annealing sweep with the exact ground state $\psi_{\rm GS}$ for various annealing times τ.

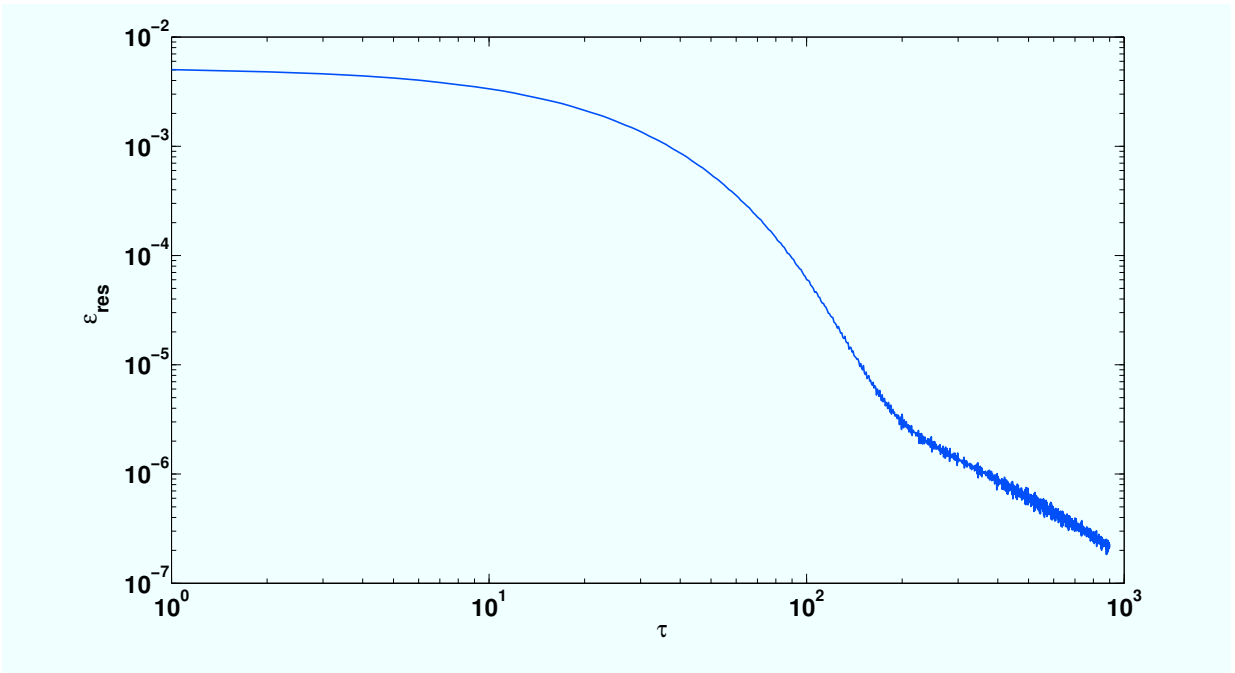


Figure 6: The residual energy ϵ_{res} per spin as a function of the annealing time τ.

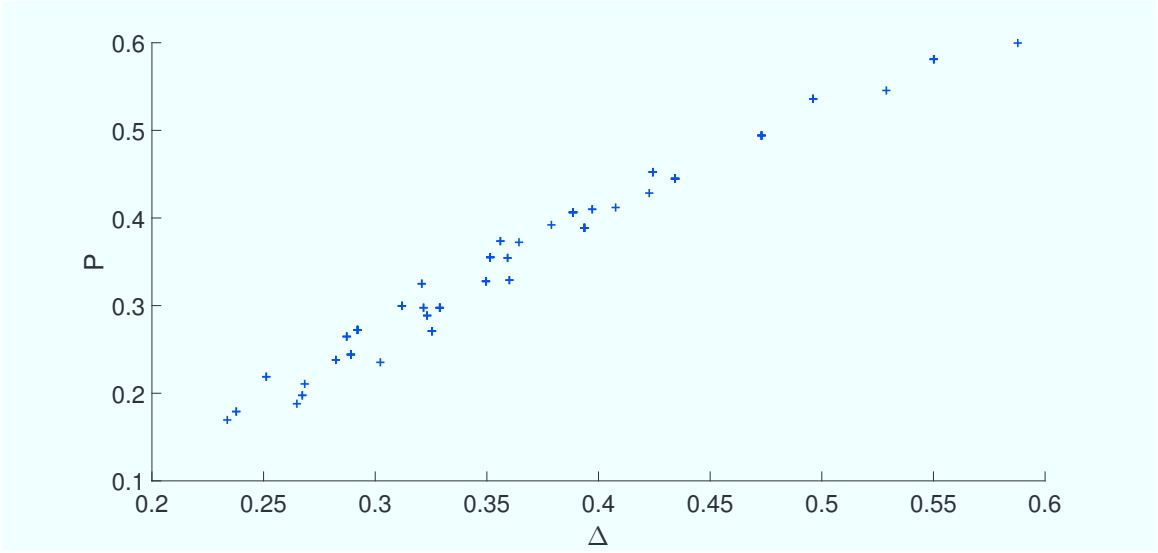
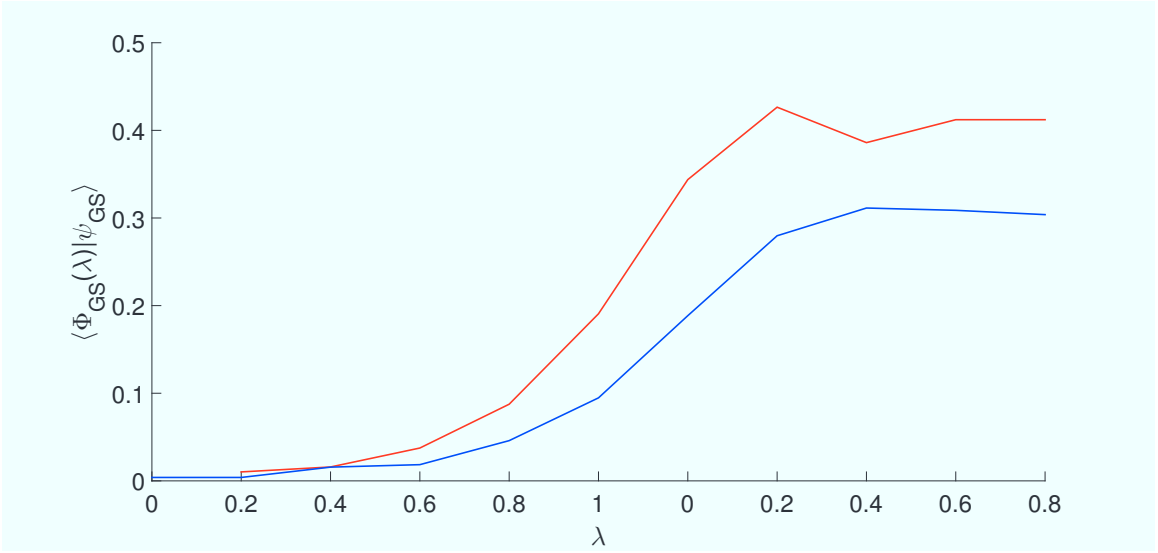


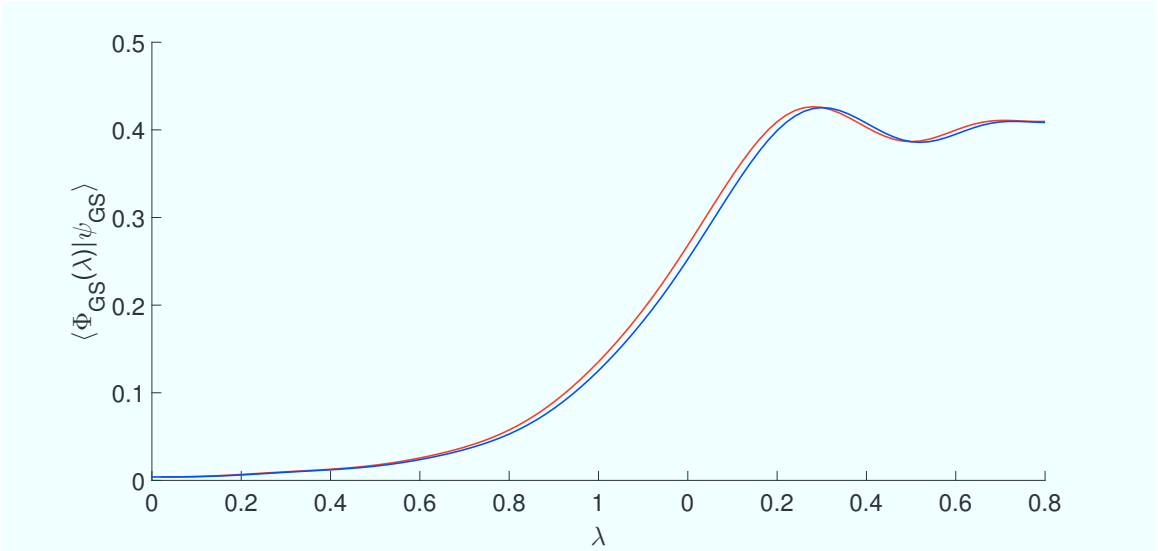
Figure 7: The probability P for the system being in the ground state as a function of the gap Δ is shown for an annealing time $\tau = 10$.

Finally the two algorithms are compared to ensure that the Suzuki-Trotter algorithm has been implemented correctly. Figure 8 shows the overlap during the annealing sweep of the exact ground state ψ_{GS} of the Hamiltonian with the ground state wave function $\Phi_{\text{GS}}(\lambda)$ for different time steps Δt. Figure 8a shows the overlap for a time step $\Delta t = 1$. The results obtained by the Full Diagonalization and Suzuki-Trotter algorithms differ very much, indicating that the time step $\Delta t = 1$ is too big to provide an exact solution. Figure 8b shows the overlap for a time step $\Delta t = 0.1$. The solutions are now close to being identical, but some smaller difference can still be seen. In Fig. 8c a time step of $\Delta t = 0.01$ is chosen. For this time step value the solutions are identical up to very small, nearly unnoticeable differences.

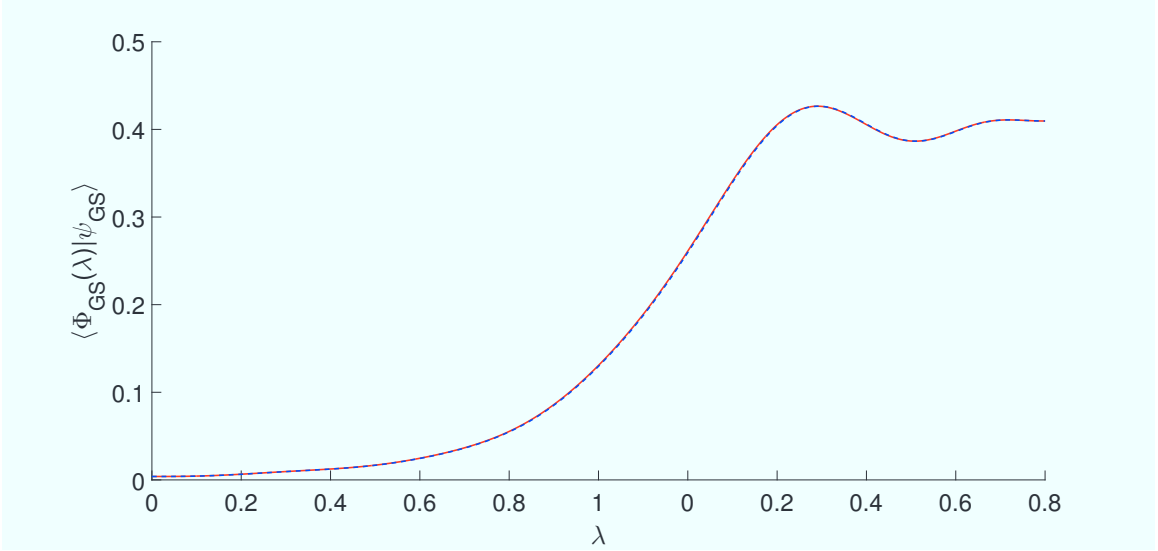
The choice of the time step not only has an influence on the exactness of the Suzuki-Trotter algorithm, it defines also the time interval in which the Hamiltonian $H(t)$ is assumed to be constant. Therefore it is expected that choosing a too large time step also introduces errors in the results obtained with the Full Diagonalization algorithm because of the time discretization. This property is illustrated in Fig. 9, which shows that the solution for a time step $\Delta t = 1$ differs strongly from the solutions for $\Delta t = 0.1$ and $\Delta t = 0.01$. Since the solutions for $\Delta t = 0.1$ and $\Delta t = 0.01$ are very similar, the choice of $\Delta t = 0.1$ is considered to be sufficient for simulating the 2-SAT 8 spin problems.



(a) $\Delta t = 1$



(b) $\Delta t = 0.1$



(c) $\Delta t = 0.01$

Figure 8: Comparison of the Suzuki-Trotter Product Formula algorithm results (blue curve) to the Full Diagonalization algorithm results (red curve) for the overlap of the exact ground state with the ground state wave function $\langle \Phi_{GS}(\lambda)|\psi_{GS}\rangle$ as a function of the annealing parameter λ for different time steps Δt.

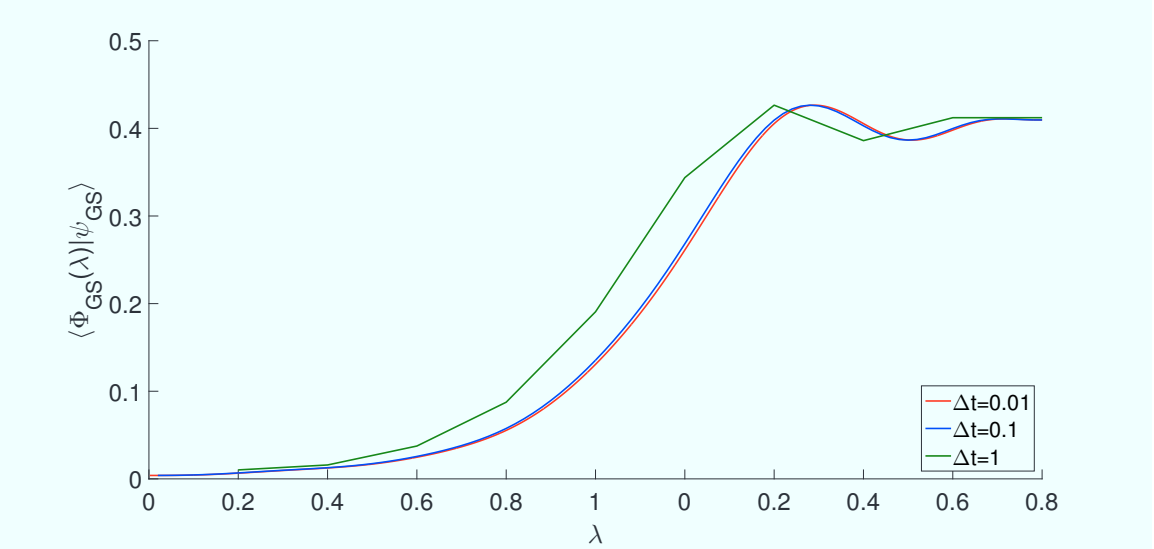


Figure 9: Comparison of the overlap $\langle\Phi_{\mathrm{GS}}(\lambda)|\psi_{\mathrm{GS}}\rangle$ obtained with the Full Diagonalization algorithm for various time steps Δt.

4 Quantum Annealing and the D-Wave Two Processor

In this chapter the implementation of the transverse Ising spin system on the D-Wave Two Vesuvius V7 processor is presented. The D-Wave Two processor implements the quantum annealing Hamiltonian defined by Eq. (4) by using superconducting flux qubits (rf SQUID qubits). The processor is designed to hold 512 qubits, for which the couplings J_{ij} and external fields h_i^z can be chosen. The processor performs quantum annealing sweeps to find the ground state for a given set of J_{ij} and h_i^z. The choice of the couplings J_{ij} is hereby restricted to the topology of the Chimera graph (Fig. 13) implemented on the processor chip.

4.1 Superconducting Flux Qubits

A schematic view of two coupled superconducting flux qubits (rf SQUID qubits) is shown in Fig. 10. In the following the basic principals of a rf SQUID is presented. A rf SQUID is a superconducting loop, that corresponds to an ordinary LC-resonator, forming a quantum harmonic oscillator [5, 25, 26]. The charge Q at the capacitor and the magnetic flux Φ_{qj} at the inductance obey the commutation relation $[\Phi_{qj},Q_j] = i\hbar$. As Φ_{qj} and Q are canonically conjugated variables the Hamiltonian of the harmonic oscillator reads

$$H_{LC} = \frac{\Phi_{qj}^2}{2L} + \frac{Q^2}{2C}. \tag{31}$$

where L is the inductance and C the capacitance of the LC-resonator. In order to get an anharmonic oscillator the superconducting loop is interrupted by a dc SQUID, which is a smaller superconducting loop (see Fig. 10) with thin insulating layers in between two sections of the loop (marked in blue in Fig. 10). The tunneling charge across the Josephson junctions adds a cosine term to the potential energy term $\Phi_{qj}^2/2L$ in H_{LC} [5, 26]. The Hamiltonian for the anharmonic oscillator then becomes

$$\begin{aligned} H_{rf} &= \frac{Q^2}{2C} + U^2\left(\frac{\varphi_{qj}^2}{2L} - \beta\cos(\varphi_{qj})\right), \\ \varphi_{q,j} &= U 2\pi\Phi_{qj}/\Phi_0\ , U = \frac{\Phi_0}{2\pi}, \end{aligned} \tag{32}$$

and $\beta = 2\pi L I_c/\Phi_0$. Hence I_c denotes the critical current on the Josephson junctions and Φ_0 the flux quantum. In order to control the constructed qubit two external fluxes Φ_{qj}^x and Φ_{ccjj}^x are applied. The flux Φ_{qj}^x threads through the main superconducting loop. The flux Φ_{ccjj}^x threads through the dc-SQUID allowing to tune the critical current I_c and therefore control β. If the device is constructed such that $\beta > 1$ and the flux bias is tuned to $\varphi_{1x} = 2\pi\Phi_{qj}^x/\Phi_0 \approx \pi$, then the potential energy becomes bistable (forms a double

well potential, see Fig. 11). The potential barrier δU can be controlled by increasing β via $\varphi^x_{ccjj} = 2\pi\Phi^x_{ccjj}/\Phi_0$ (φ^x_{ccjj} controls δU in Fig. 11). With increasing β two local minima form in the potential, through which the two lowest lying states of the device may couple via quantum tunneling. Changing φ^x_{qj} around the value π allows to bias the flux φ_{qj} (therefore φ^x_{qj} controls $2h$ in Fig. 11). The two lowest energy states then form the basis for the qubit, which allow to write an effective low energy Hamiltonian for the qubit

$$H_{qubit} = -\frac{1}{2}\epsilon\sigma^z + \Delta\sigma^x, \tag{33}$$

where σ^x and σ^z denote the Pauli matrices and ϵ and Δ are the magnetic fluxes produced by the persistent currents in the superconducting loop. The currents, which are flowing in opposite directions can now be treated as Ising spins. The restriction to the two lowest energy states is a valid approximation as the operating temperature of the D-Wave Two processor is about 17 mK [26].

The interaction between the qubits on the processor is also controlled with an external magnetic flux $\Phi^x_{co,ij}$ that is provided to every qubit-to-qubit coupler by a programmable digital to analog converter (DAC).

The flux qubits discussed above as well as the couplers are macroscopic superconducting loops, that can be stretched and routed up to a certain degree, as needed to design the processor. Based on this feature the unit cells holding 8 qubits are realized on the D-Wave Two processor (see Fig. 12). Figure 12 shows that in the unit cell 4 qubits run horizontally through the cell and 4 qubits vertically. All 4 horizontal qubits intersect with all 4 vertical qubits. At the intersections the qubits are coupled to each other. Therefore, the horizontal qubits are only coupled to the vertical qubits and vice versa. Each horizontal qubit is coupled to the corresponding horizontal qubits in the neighboring unit cells above and below and each vertical qubit is coupled to the corresponding vertical qubit in the neighboring left and right unit cells. This design leads to the topology of the implemented graph, called Chimera graph, depicted in Fig.13. The figure shows that the Chimera graph is a non-planar bipartite graph [25, 27].

The qubits in the first unit cell (see Fig. 13) are labeled by $i = 1...8$ with the labels for the qubits in the first column being $i = 1...4$ and the qubits of the second column being $i = 5...8$. For the second unit cell along the first row of unit cells in Fig. 13 the index i runs from 9 to 16. At the end of the row the next unit cell is the first cell in the second row and so on [25].

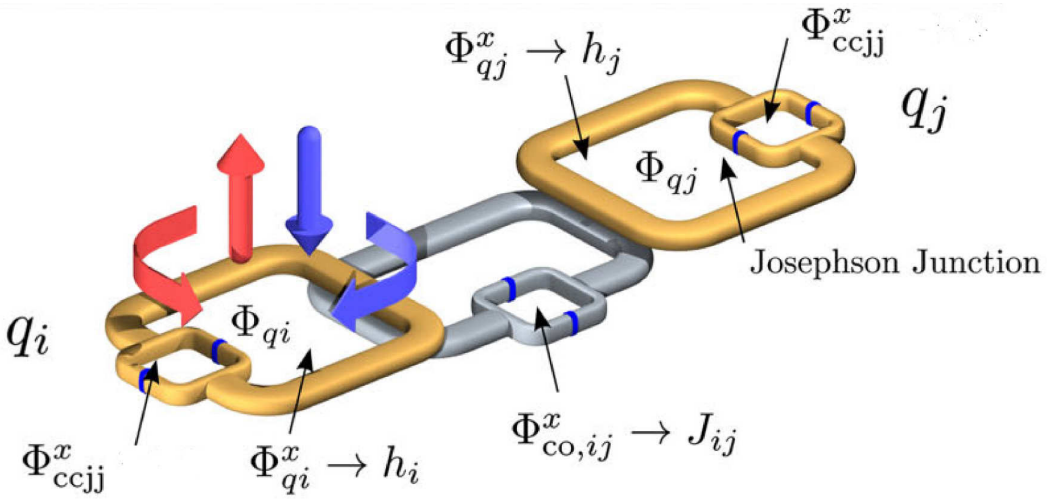


Figure 10: Schematic representation of two coupled superconducting flux qubits q_i and q_j, with Φ_{qi} and Φ_{qj} threading through the body of the qubits, which are externally controlled by $\Phi^x_{cc,jj}$, Φ^x_{qi} and Φ^x_{qi}. The inductive coupling of the qubits is controlled by the external flux $\Phi^x_{co,ij}$. From: Lanting et al. [27].

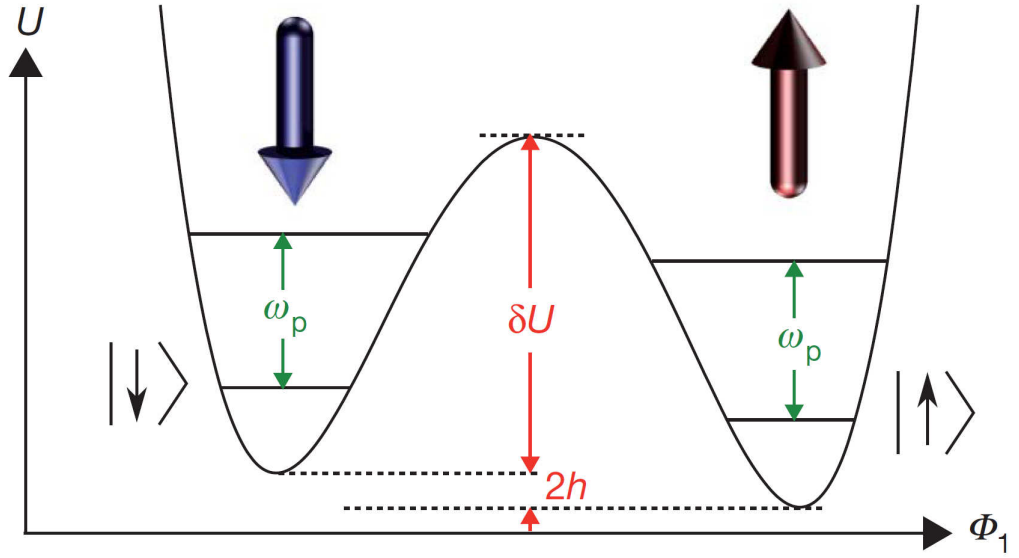


Figure 11: Diagram of the double well potential for the two lowest energy levels ($|\uparrow\rangle$ and $|\downarrow\rangle$) of a flux qubit. The barrier height δU and the bias $2h$ is controlled by external fields. From: Johnson et al. [28].

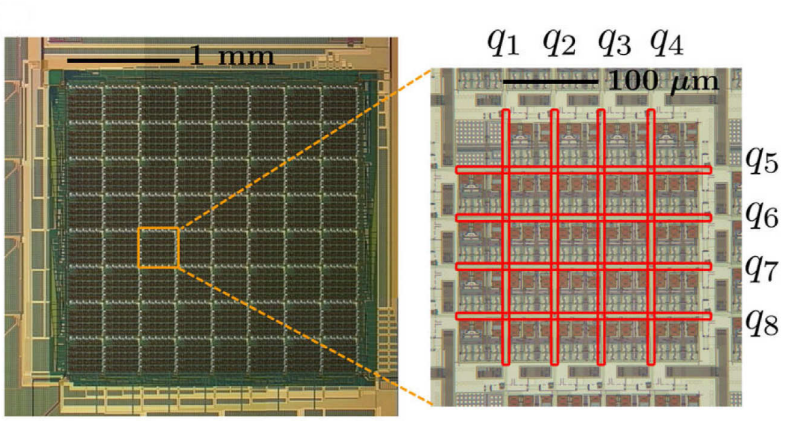


Figure 12: Picture of the D-Wave Two processor with 64 unit cells comprising 8 qubits per cell and a single unit cell of the processor. The qubits in the unit cells only interact at the intersections. From: Lanting et al. [27].

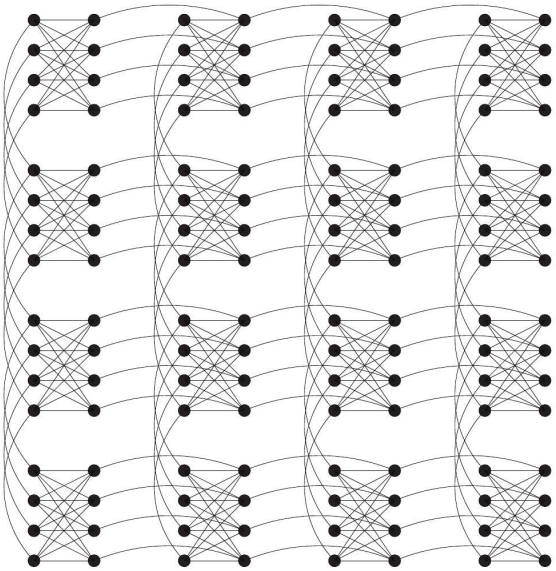


Figure 13: Representation of the topology of the Chimera graph implemented on a D-Wave processor with 16 unit cells. The dots represent the qubits and the lines the possible interactions. From: Katzgraber et al. [29].

4.2 Inoperable Qubits, Calibration Errors and Programming Noise

During the implementation of the flux qubits on the D-Wave Two processor errors can occur for a number of reasons. First, the implemented rf SQUIDS, DACs and the couplers are all electrical components that are subject to failure. Depending on what component is broken, this can lead to inoperable qubits and couplers. As a consequence the graph of the actual processor, on which the simulations were performed does not look as regular as depicted in Fig. 13 but looks like the one presented in Fig. 14. In this work two processors are used namely System 6 and System 13. The graph in Fig. 14 shows the

graph for System 6 and the graph for System 13 is shown in the Appendix A.

Second, the calibration of such a processor comprising 512 qubits is a very complex task. Therefore a variation of $\sim$5% of the programmed J_{ij} and h_i^z is common. These calibration errors can cause the ground states of the programmed model to differ from the actual ground state of the processor. Besides these calibration errors that are systematic errors (will be the same for every annealing run) there are also statistical errors caused by magnetic flux noise and thermal noise [30, 31].

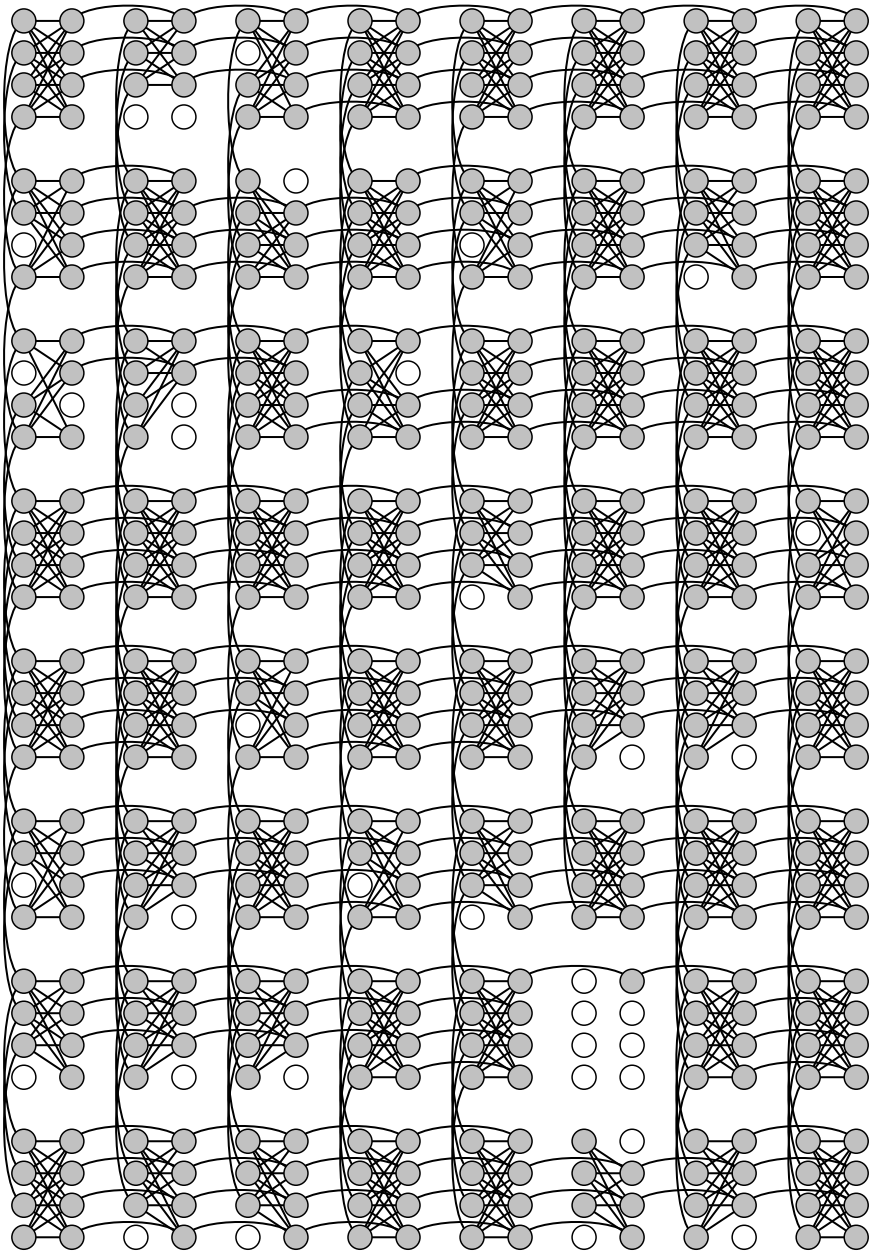


Figure 14: Schematic picture of the graph implemented on the D-Wave Two processor (System 6) that was used in the work. Gray (White) spots indicate (in)operable qubits. Lines indicate qubit-qubit interactions.

4.3 Programming the D-Wave Two Processor

For the user the programming of the D-Wave Two processor is simple. The problems, that can be put on the processor must fit the Chimera graph. If this is the case the problem set of fields h_i^z and couplings J_{ij} can be sent to the processor via a Matlab interface. The fields h_i^z are given as a vector $h_D(i)$ of length 512. Hereby the vector $h_D(i)$ stores the external fields applied to σ_i^z. The J_{ij} couplings between σ_i^z and σ_j^z are stored in a 512×512 adjacency matrix J_D. The machine returns an error, if an inoperable qubit is coupled or a non-existing coupling between two Ising spins is chosen, as well as if an external field is applied to an inoperable qubit. The programming on the processor itself is done by 6 DACs per qubit and one DAC per coupling. An example Matlab code is shown in Appendix B.

4.4 Quantum Annealing

As described above the flux qubits on the D-Wave Two processor implement an Ising spin model that can be controlled by programming the DACs controlling the qubits. The Hamiltonian for the Ising spin system on the D-Wave Two processor is given by

$$H(t) = B(t)\left(-\sum_i^N h_i^z \sigma_i^z + \sum_{i,j}^N J_{ij}^z \sigma_i^z \sigma_j^z\right) - A(t)\sum_i^N \sigma_i^x, \tag{34}$$

which can be identified to be the same as Eq. (4) except for the fact that the energy scales $A(t)$ and $B(t)$ may not evolve linearly as $\lambda(t)$ in Eq (4) (see Fig. 15). The annealing parameter $\lambda(t) = t/\tau$, where τ is the total annealing time, can be related to the external fields $\Phi_{cc,kk}^x$ threading through the Josephson junction by the formula [27]

$$\lambda(t) = \frac{\left(\Phi_{cc,kk}^x(t) - \Phi_{cc,kk,\text{initial}}^x\right)}{\left(\Phi_{cc,kk,\text{final}}^x - \Phi_{cc,kk,\text{initial}}^x\right)}. \tag{35}$$

The energy scale is hereby set for the Hamiltonian given by Eq. (34) by the macroscopic quantities of the superconducting qubits. A typical course of the annealing schedule for the D-Wave Two is shown in Fig. 15. The operating temperature of 17 mK is depicted as a black dashed line in the figure. The energy scales are $B(\tau) = 22$ GHz and $A(0) = 34$ GHz with an default annealing time of 20μs [16, 31].

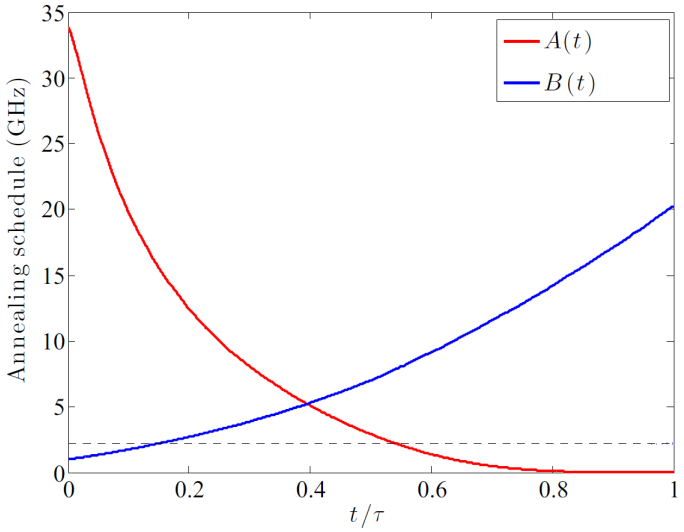


Figure 15: Diagram of a typical annealing schedule for the two energy scales $A(t)$ and $B(t)$ on the D-Wave Two. The operating temperature T=17 mK is visualized with a black dashed line. From: Albash et al. [16].

5 Solving Random Spanning Trees on the D-Wave Two

In this chapter the qubit (spin) configurations obtained by the D-Wave Two processor for randomly generated spanning tree problems are presented.

5.1 Randomly Generated Spanning Trees

In general, a spanning tree of an undirected graph G with n vertices is a subgraph of G, that includes all vertices of G without any cycles. Therefore the tree is always a minimal set of edges that connects all vertices, i. e. the tree has $n-1$ edges. We consider trees on the graph, or on a part of the graph, shown in Fig. 14.

The trees are generated by an algorithm that is based on a random walk on the graph G. A random starting point (vertex) of the graph is chosen at the beginning. From there the algorithm performs a random walk on the graph (along the edges) until all vertices are visited at least one time. Every time a vertex is visited for the first time the edge by which it was entered is added to a list of edges. At the end of the random walk the list holds $n-1$ edges that form the spanning tree. It was shown that the trees randomly generated by the algorithm are uniformly distributed in the ensemble of spanning trees of an undirected graph G [32].

The edges obtained from the algorithm for generating a spanning tree are taken to be the couplings J_{ij} for the problems that are solved on the D-Wave Two processor. The couplings J_{ij} are all chosen to be (anti-) ferromagnetic with value 1 (-1). The fields h_i are all chosen to be 0. Since a spanning tree is a one dimensional object (it has no loops and therefore no frustration) choosing antiferromagnetic couplings or ferromagnetic couplings results in the same problem (with $h_i = 0$) since the problems are invariant under a Z_2-gauge. This of course only holds, if the spin system has a Z_2-symmetry. The ground state energy of a (anti-) ferromagnetic spanning tree is minus the number of edges $n-1$ times $|J_{ij}|$. As J_{ij} is chosen to be 1 (ferromagnetic) or -1 (antiferromagnetic) the ground state energy is thus $-(n-1)$.

The nice feature of the spanning tree problems is that the solution is known. For the ferromagnetic case the ground state is the state with all spins up or all spins down. For the antiferromagnetic case the solution is given by the state with spins alternately pointing up and down along the tree. This makes it easy to check whether the D-Wave Two processor is able to find the solution and to identify in which region of the processor problems arise if it does not find the solution. But even so the solution is known for all the problems, it might still be hard to solve the problem with quantum annealing.

5.2 Results

5.2.1 Z_2-Symmetry

As a first test, a ferromagnetic spanning tree is put on the entire D-Wave Two processor. On the used chip (System 6) 476 qubits of the 512 qubits are operable which means that the ground state energy of the problem is -475. The processor performs by default 1000 annealing runs for one solving request, i.e. the processor returns 1000 solutions for every problem submitted. We submit 1000 of such requests giving a total of one million solutions. The percentage of correct solutions found in these thousands annealing runs yields information about the complexity of the problem. As mentioned above, there are two solutions to the ferromagnetic spanning tree problem, as the ground state is two fold degenerate with the two ground states having all spins up and all spins down, respectively. If the chip implements an Ising spin system perfectly the probability of finding one ground state is as high as finding the other one, as the system is Z_2-symmetric. This means that the average magnetization of all spins should be zero ($m_{\sigma_i} = \sum_{k=1}^{1000} \sigma_k^z$ and $m_{sys} = \sum_{i=1}^{N} m_{\sigma_i}$). This test is performed for various randomly generated spanning trees. In the following results for one particular ferromagnetic tree are discussed but the results for other trees are similar.

For the particular ferromagnetic tree the D-Wave Two processor found 237 times a ground state with $E_{GS} = -475$. This gives a relative frequency of finding the ground state of about $0.2‰$. First excited states with energy $E_1 = -473$ were found 159387 times ($15{,}9\%$) and the second excited states with energy $E_2 = -471$ were found the most, namely 362632 times ($36{,}3\%$). In Fig. 16 the relative occurrence frequency of the different energies for all obtained states is shown. Important to note is that all 237 times the D-Wave Two found a ground state it was the ground state with all spins up.

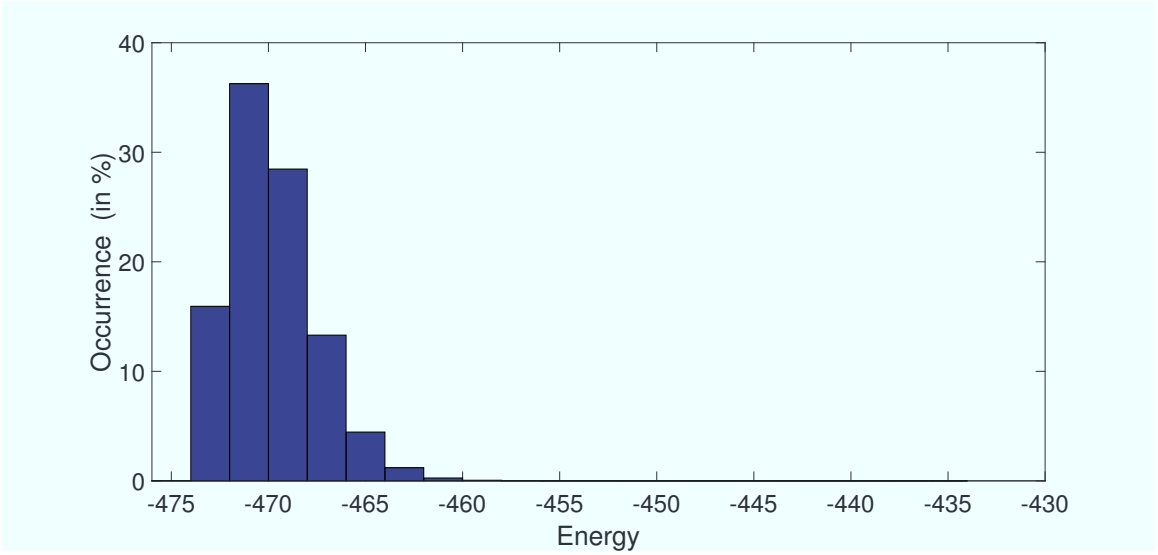


Figure 16: Relative occurrence frequencies of the solutions with different energies in percent for a ferromagnetic random spanning tree generated on the entire system. The ground state energy is -475 and has an relative occurrence frequency of $0.2‰$.

In order to better understand why the processor faces such difficulties of finding the correct solution the average single spin magnetization is considered. The obtained average magnetizations for the single spins (qubits) are shown in Fig. 17 with the qubits numbered from 1 to 512. For the inoperable qubits the magnetization is set to 0. The figure shows that the processor is inhomogeneous as it exhibits two different areas. In one region the spins have a tendency to point down and in a second region the spins tend to point up. The region in which the spins tend to point up is located in the lower part of the processor (qubit numbers larger than 250). The region in which the spins tend to point down is located mainly in the upper right part of the processor. In these two regions a bias-field appears to be acting, so that the spins are forced to point up/down even so it might not be favorable according to the Hamiltonian. As the bias-fields are in opposite directions in the two regions of the processor it is favorable for the spin system on the processor to ”break” a ferromagnetic coupling (an antiparallel alignment) between two spins, so that the spins in the respective regions can point in the direction of the bias-field.

This observation gives an explanation for the fact that the ground state is rarely found, as the two regions compete against each other and therefore favor an antiparallel alignment between two coupled qubits, which finally leads to an excited state. In Fig. 18 the relative frequency of the qubits to have an antiparallel alignment to a coupled qubit is shown in order to detect ”broken” couplings on the processor. The figure demonstrates that the antiparallel alignment mainly occurs (77%) at one and the same qubit namely the one with number 15. In the considered tree qubit 15 is connected to qubits 10 and 23, which have a relative frequency of an antiparallel aligned coupled qubit of 55% and 32%, respectively. In total only six qubits have such a relative frequency higher than 15%.

This shows that the ”breaking” of a ferromagnetic coupling mainly takes place between six qubits. This feature was observed for all different spanning tree problems submitted to the processor, however the qubits involved in the ”breaking” were different but in the same area. We also tried to put a bias field h_i on the couplings involved in the ”breaking” in order to force the qubits to be aligned, but this moved the ”breaking” along the chain to the next qubit.

From this it can be concluded that the bias-fields in the two regions are so strong that a ”breaking” of a ferromagnetic coupling between qubits nearly always occurs and that it is not an error of single qubits. Further to note is that the bias-field at the bottom part of the processor is stronger than the one at the upper part. This can be concluded from the fact that the 237 ground states found by the D-Wave Two were the ground states with all spins up.

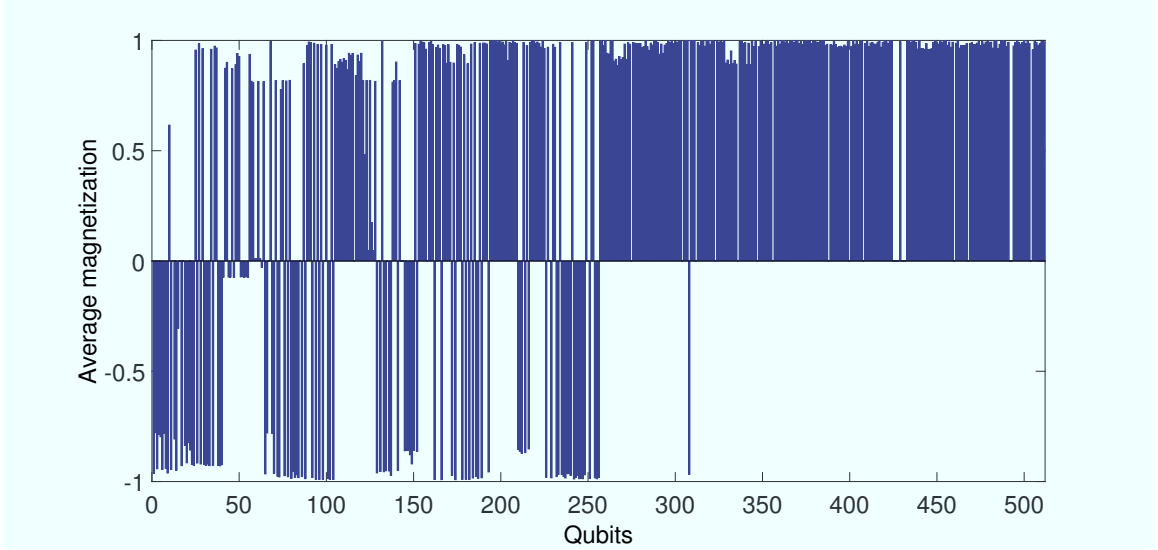


Figure 17: Average single spin magnetization for a ferromagnetic spanning tree of all working 476 qubits. The processor has two regions where spins are biased to point up or down, respectively and the Z_2-symmetry is broken.

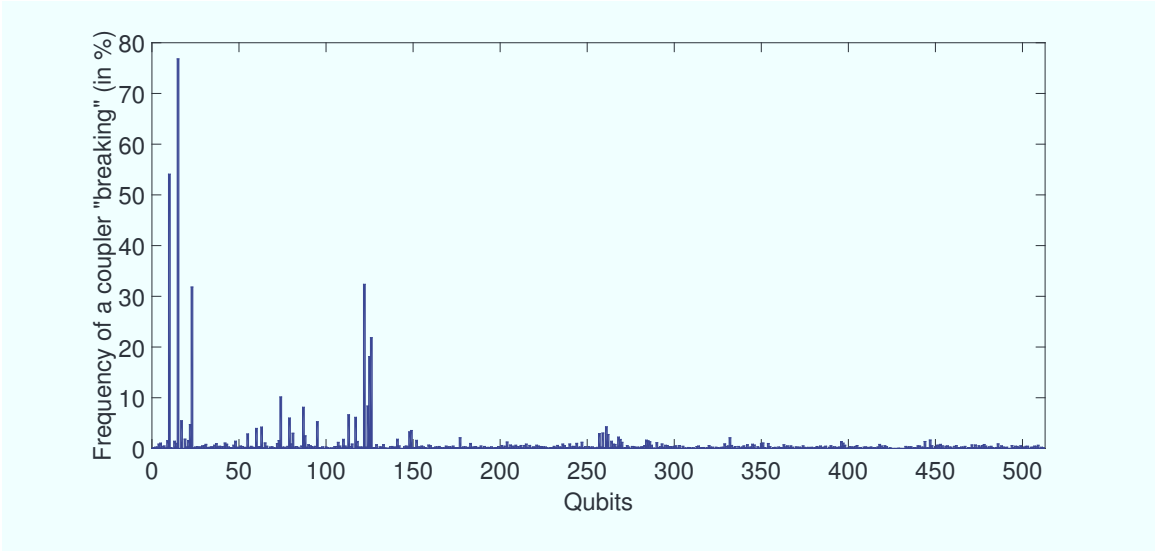
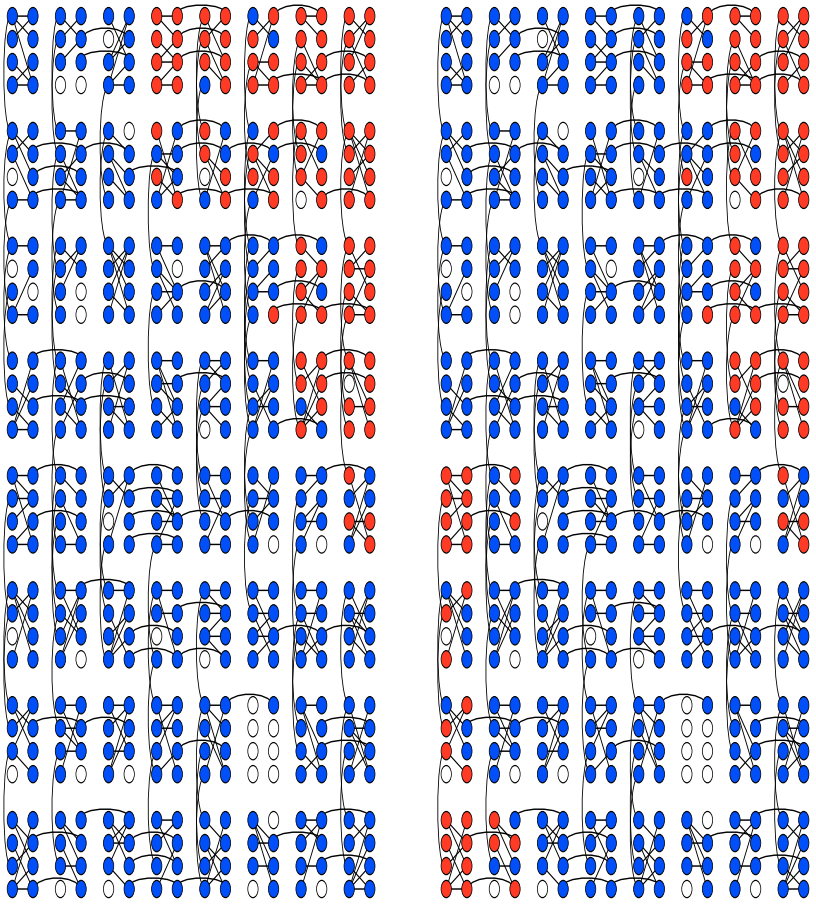


Figure 18: Relative frequency of an antiparallel alignment between coupled qubits in a ferromagnetic spanning tree with 476 qubits. Most of the antiparallel alignment occurs between 6 qubits.

To further illustrate this fact a field bias $h_{452} = -1$ is applied to qubit 452, which is the qubit at the bottom left corner of the processor. This in theory should break the Z_2-symmetry and make all spins in the spanning tree to point down (thereby lifting the ground state degeneracy). The ground state energy is $E_{GS} = -476$. The lowest energy states found by the processor are second excited states which have the energy $E_2 = -472$. Two of the lowest energy states found by the processor are shown in Fig. 19. In Fig. 18a it can be seen that instead of flipping all the spins in the bottom left side of the processor (and also on the entire chip) the field $h_{452} = -1$ is ignored and the spins still point up (blue). In the second state depicted, the field bias $h_{452} = -1$ locally flips some spins (42 spins) but the spin flipping does not extend over the entire bottom region. This also shows that the regions have strong bias-fields and that therefore the Z_2-symmetry is broken severely.



(a) (b)

Figure 19: Two lowest energy solutions ($E_2 = -472$) found by the D-Wave Two processor for a random spanning tree that has a field bias $h_{452} = -1$ applied to qubits 452. In (a) the applied field bias is ignored in favor of the regional bias-field as spin 452 aligns antiparallel to the direction of the applied field bias. In (b) some spins are flipped locally but the overall bias-field of the region is too strong to flip the spins in the entire bottom part of the processor.

5.2.2 Statistical Independence of Uncoupled Regions

In this subsection the processor is divided into four regions (top-left, top-right, bottom-left, bottom-right) with the same number of unit cells. The same spanning tree is put onto the four regions. This is done for ferromagnetic as well as for antiferromagnetic spanning trees. As the results are similar, the ferromagnetic spanning tree is discussed in the following. Inoperable qubits in one region must also not be used in the other regions in order to be able to put the same spanning tree on all four regions. This reduces the number of used qubits per region to 99. Hundred spanning trees are considered and for each spanning tree 1000 solution requests are processed yielding one million solutions

per tree.

The first idea is to test whether regions with more inoperable qubits operate in a similar way as the regions with less inoperable qubits. If a different way of operation would be observed, then this would mean that there is a difference between unused and inoperable qubits and that there is a difference in the way they influence the entire system. Tests in this direction did however not show any significant differences.

Another interesting question is, whether the presence of unused couplers and qubits influence the neighboring used qubits and couplers. To answer this question one has to test whether switched off qubits and couplers are actually off i.e. do not influence the results. This was tested by checking whether the frequencies of finding the solution in the different regions are statistically independent from each other. All four regions have a relative frequency P_i of finding the solution to the spanning tree ($i = 1, 2, 3, 4$). If the four regions are statistically independent, then the relative frequency for finding the solution in region i and region j ($i \neq j$) at the same time is given by $P_{ij} = P_i \cdot P_j$. For finding solutions in three regions at the same time the relative frequency is given by $P_{ijk} = P_i \cdot P_j \cdot P_k$ ($i \neq j \neq k$) and for four regions by $P_{1234} = P_1 \cdot P_2 \cdot P_3 \cdot P_4$.

Figure 20 shows the relative frequency of finding the solution in the regions P_i (yellow bars labeled 1 to 4) on the D-Wave Two processor. The relative frequency of finding solutions in different regions at the same time is represented by the yellow bars that are labeled with P_{ij} for finding the solution in region i and j at the same time, P_{ijk} for finding the solution in regions i, j, and k at the same time and P_{1234} for finding the solution in all four regions at the same time. The red bars plotted over the yellow bars in Fig. 20 represent the frequencies calculated from the single region relative frequencies P_i. From Fig. 20 it can be seen that the calculated relative frequencies of finding solutions at the same time in different regions are the same as the frequencies obtained on the D-Wave Two processor taken into account the statistical errors (error bars indicate one σ deviation). The relative frequencies are also listed in Table 1. The results suggest that the regions can be treated as statistically independent, as they appear not to influence each other.

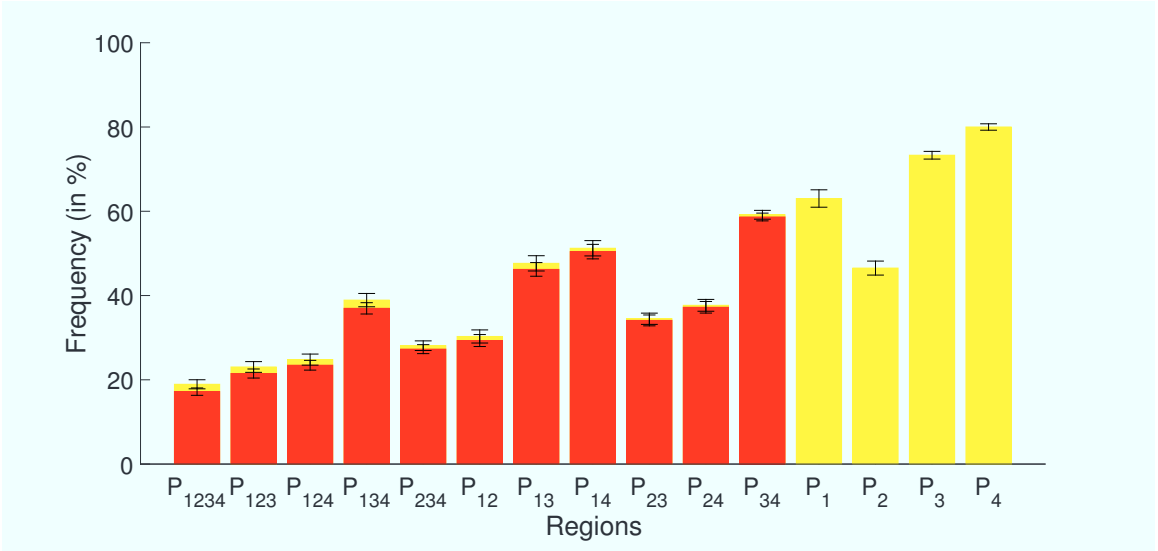


Figure 20: Relative frequencies of finding the solution to a random spanning tree problem in four disconnected regions (yellow bars labeled by P_1 to P_4). The relative frequency of finding solutions in different regions at the same time (yellow bars labeled by P_{ij}, P_{ijk}, P_{1234}) is given by the product of the frequencies in the single regions (red bars), if the regions are statistically independent. As the yellow and red bars overlay (taking statistical errors into account), the four regions can be considered to be independent.

	Calculated [%]	D-Wave [%]
P_{1234}	17.2 ± 0.9	18.9 ± 1.1
P_{123}	21.5 ± 1.1	23.0 ± 1.3
P_{124}	23.5 ± 1.2	24.8 ± 1.3
P_{134}	37.0 ± 1.3	38.9 ± 1.6
P_{234}	27.3 ± 1.1	28.1 ± 1.1
P_{12}	29.3 ± 1.4	30.3 ± 1.6
P_{13}	46.2 ± 1.6	47.6 ± 1.8
P_{14}	50.4 ± 1.7	51.2 ± 1.8
P_{23}	34.1 ± 1.3	34.5 ± 1.3
P_{24}	37.2 ± 1.4	37.7 ± 1.4
P_{34}	58.6 ± 1.0	59.1 ± 1.1
P_1	-	63.0 ± 2.1
P_2	-	46.5 ± 1.7
P_3	-	73.3 ± 1.0
P_4	-	80.0 ± 0.8

Table 1: Relative frequencies of finding the solution to a random spanning tree problem in four disconnected regions and the relative frequency of finding solutions in different regions at the same time.

5.2.3 Date/Time Dependency

In this section the right half of the processor was divided into two regions with each using 93 operable qubits. Note that this chip is a different one (System 13) as the one used before was taken offline (a graph of the chip is shown in the appendix). Again identical spanning trees are put on the two regions. The spanning trees are antiferromagnetic trees. This was also originally done to test for the influence of inoperable qubits, which did not yield conclusive results as much more data would be needed in order to get enough statistics to draw a clear conclusion. 30 different spanning trees are put on the processor again with 1000 solution requests yielding one million solutions per tree. The key point of the observations made is that the first ten trees were submitted on January 15, 2015 and the other 20 trees on January 20, 2015. The average relative frequencies of finding the solution in the single regions and in both regions at the same time at the different days are listed in Table 2. The relative frequencies obtained on January 15 differ strongly from the frequencies obtained at January 20. The relative frequencies for finding the solution in each region independently are more than 30% lower on January 15 as on January 20 with a statistical error less than 0.3%. For finding the solutions in both regions at the same time the relative frequency difference is nearly 50%. In Fig. 21 the relative frequency for finding a solution for all 30 spanning tree problems for the two regions is shown. A large increase in the relative frequency for finding the solution can be seen for the random spanning tree problems submitted on January 20.

This result shows that the D-Wave Two performs different from day to day. This might be caused by different operating temperatures at specific days or by magnetic flux fluctuations in the system. This suggests that in order to obtain statistical stable results it is necessary to submit problems on different days over a longer time period.

	15.1.2015	20.1.2015
Finding both solutions [%]	36.57 ± 0.15	85.54 ± 0.06
Solution for Region 1 [%]	62.32 ± 0.19	92.45 ± 0.02
Solution for Region 2 [%]	59.10 ± 0.23	92.35 ± 0.03

Table 2: Relative solving frequency for antiferromagnetic random spanning tree problems on two different days for two regions on the chip.

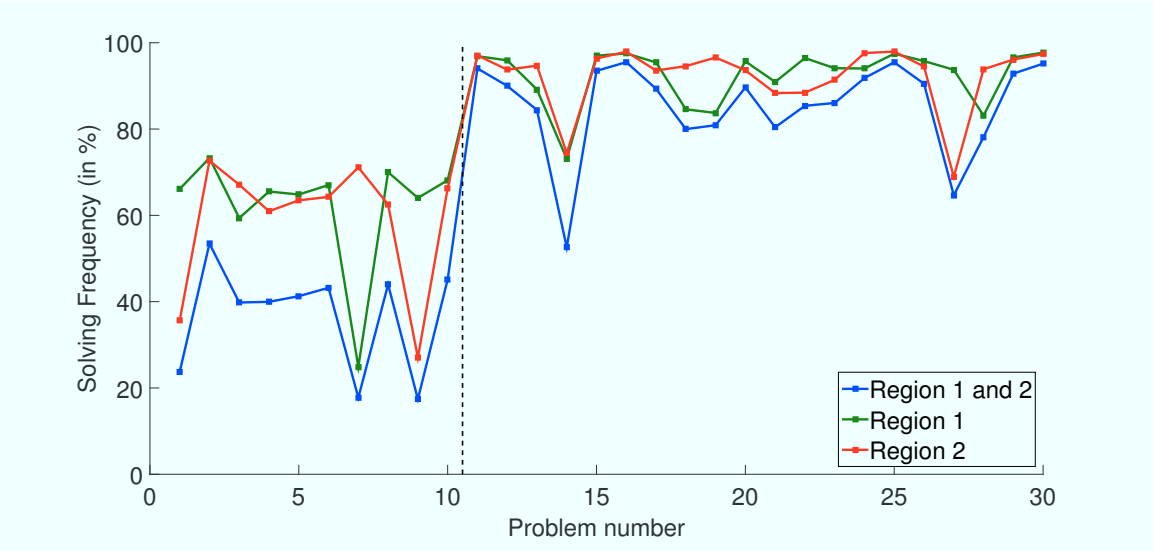


Figure 21: Relative solving frequencies for antiferromagnetic random spanning tree problems in two regions of the processor as a function of the problem number . The first ten problems (on the left of the dashed line) were submitted on January 15, 2015 and the other 20 (on the right of the dashed line) on January 20, 2015. A strong increase in the relative solving frequency is observed for the spanning tree problems submitted on January 20.

6 Solving 2-SAT Problems on the D-Wave Two Processor

In this chapter 2-SAT problems are studied on the D-Wave Two processor and the results are compared to the results obtained with the Suzuki-Trotter Product Formula algorithm for simulating the quantum annealing process.

6.1 Embedding 2-SAT Problems on the Chimera Graph

The 2-SAT problems introduced in section 3.5 are solved on the D-Wave Two processor (System 13). As mentioned in section 3.5, the 2-SAT problems considered in this work have been obtained by a Monte Carlo search algorithm, designed to find the computationally hardest problems. The problems have a unique satisfying assignment (non-degenerate ground state) and a clause to spin ratio $\alpha = (N + 1)/N$. The 2-SAT problems are 8, 12, 18 spin problems. There are a total of 100 8-spin problems and 1000 12- and 18-spin problems, respectively [24].

They can be mapped to the Ising model as shown in subsection 3.5.2. The problems are embedded directly on the processor (no logical qubits). This means that only the problems that are subgraphs of the Chimera graph fit on the processor. The search for a mapping of the problems onto the Chimera topology was done computationally by a kind of sophisticated random walk. This reduces the number of problems drastically. The algorithm mapped 65 8-spin problems, 334 12-spin problems and 262 18-spin problems on the Chimera graph. Note that a better algorithm possibly can map more problems on the graph.

As mentioned before, the problems are computationally the hardest of the 2-SAT problems because of their small minimal gaps Δ during the annealing sweep at λ_{crit} and their high degeneracy of the first excited state Ω_1. Another key feature of the problems is that the correlation between the gaps Δ and the degeneracy Ω_1 of the first excited state are nearly completely uncorrelated as they have a correlation coefficient of 0.12. Figure 22 shows the degeneracy Ω_1 plotted against $1/\Delta$ for the full set of 18-spin problems. The red markers indicate the problems that were studied on the D-Wave Two.

Since the problems have known minimum gaps Δ, it is expected that the scaling of the solution frequency with the gap corresponds to the scaling properties of the quantum annealing process simulated with the Suzuki-Trotter Product Formula algorithm.

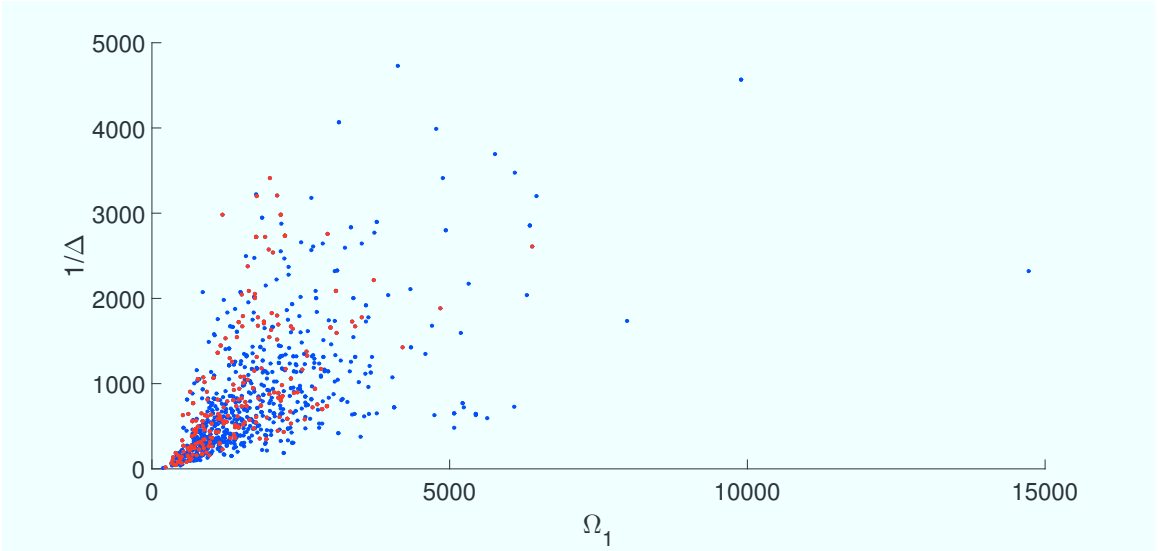


Figure 22: Inverse gap $1/\Delta$ as a function of the degeneracy Ω_1 of the first excited state for all 18 spin problems. The red markers indicate the problems that could be mapped on the graph of the D-Wave Two processor.

6.2 Parameters for Simulating the Quantum Annealing Process

In order to simulate the behavior of the D-Wave Two processor, the energy scales and the run time need to be chosen accordingly. The energy scales on the D-Wave Two processor are $B_{DW}(\tau) = 22$ GHz and $A_{DW}(0) = 34$ GHz (see Fig. 15) with an annealing time of $\tau_{DW} = 20\ \mu s$. In the simulation $A_s(0)$ is set to one which leads to $B_s(\tau) = 22/34$. In this scale the annealing time of 20 μs of the D-Wave Two processor corresponds to $\tau_{\text{sim}} = 1000$ in the simulation ($\tau_{DW} A_{DW}(0) \equiv \tau_{sim} A_s(0)$) . The operating temperature is $T_{DW} = 17\ mK$. For the simulation this corresponds to $T_s = 0.058$.

6.3 Results for Solving the 2-SAT Problems

In this section simulation results for the 2-SAT problems are compared to the results obtained with the D-Wave Two processor. The simulations on the D-Wave Two were performed in three different regions of the processor. Since the results of the three regions are similar only the results of the first region are presented in the following. Further, in order to average out the effects of a possible bias-field in the region (see section 5), all problems are also run with flipped fields $h_i^z = -h_i^z$ and the results are averaged over both field directions. The data for each specific problem is collected over several different days. This accounts for the possibility of obtaining different results witnessed on different days (see section 5).

As the set of problems is limited, the problem Hamiltonians are rescaled $H_{cl}^{'} = a \cdot H_{cl}$, in order to generate more problems with different gaps. The 18-spin problems are rescaled

with $a = 0.425$ and $a = 0.425/2$. The 12-spin problems are rescaled with $a = 0.425$ and the 8-spin problems with $a = 0.1, 0.2, 0.4, 0.6, 0.8$. The rescaling leads to smaller gaps (harder problems) and also slightly shifts the critical point λ_{crit}, at which the minimal gap Δ occurs during the annealing process.

All results from the D-Wave Two and the spin simulations are shown in Fig. 23, which depicts the frequency P for finding the solution as a function of the gap Δ. Spin simulation results for 12-spin problems rescaled with $a = 0.2$ and 18-spin problems are included in the plot, as they extend to the left bottom corner of the plot in the line formed by the brown and violet markers. The operating temperature is indicated by the black vertical line at $\Delta = 0.58$. The figure shows that the frequency obtained by the D-Wave Two processor for the 8-spin and the non-rescaled 12-spin problem sets, lie in the same range as the frequencies obtained by the spin simulation. But for the 18-spin problem sets and the rescaled 12-spin problem set ($a = 0{,}425$) this is not the case. This is now studied in more detail.

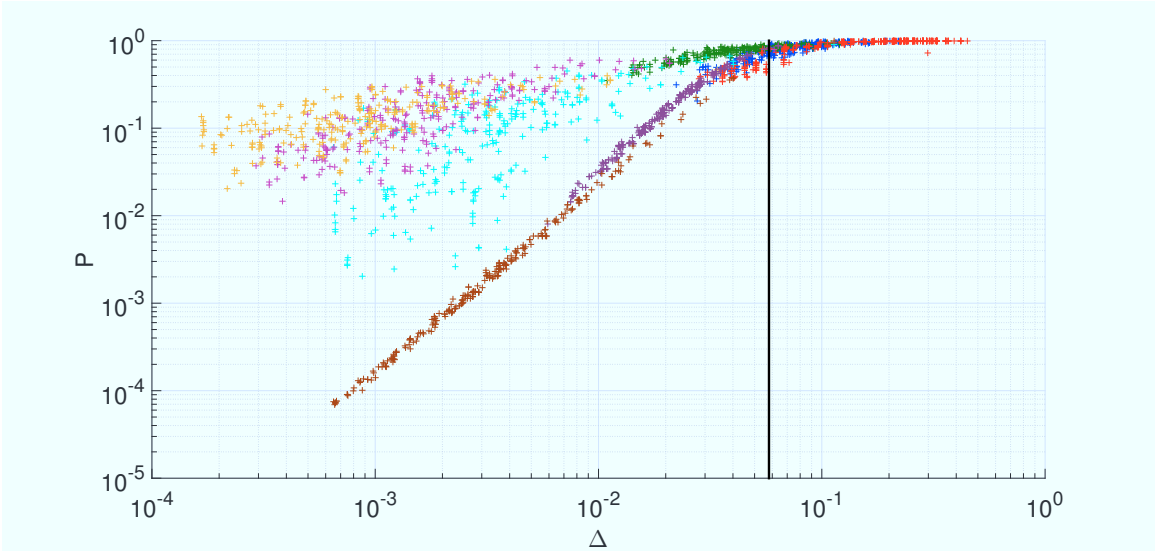


Figure 23: Frequency P for finding the solution of 2-SAT problems with the D-Wave Two Processor and by simulated quantum annealing. The operating temperature is indicated by the black vertical line at $\Delta = 0.58$. Results for 18-spin problems are indicated with yellow ($a = 0.425/2$), pink ($a = 0.425$) and light blue ($a = 1$) markers. Results for 12-spin problems are indicated with green ($a = 0.425$) and blue ($a = 1$) markers and 8-spin problems ($a = 0.1, 0.2, 0.4, 0.6, 0.8$) with red markers. Simulation results for 12-spin problems ($a = 0.2$) are incdicated with violet markers and simulation results for 18-spin problems ($a = 1$) with brown markers.

In Fig. 24 the results of the 12-spin problems are omitted. When comparing the simulated quantum annealing results and the D-Wave results for the 18-spin problems it becomes apparent that the frequencies of finding a solution on the D-Wave Two processor are significantly higher than expected from the simulations. The rescaling of the

18-spin problems, which was originally intended to create harder problems as the gap gets smaller, seems to help the D-Wave processor to find the solutions to the problems. The non-rescaled problems (light blue markers) have lower frequencies as the problems rescaled by $a = 0.425$ and the problems rescaled by $a = 0.425/2$ have an even higher solving frequency. This result is unexpected as rescaling makes the gaps smaller and therefore harder to solve by quantum annealing. This indicates that a process, which is not of quantum nature, is helping the D-Wave processor to find the solutions for the 18-spin problems. A reasonable assumption might be, that thermal fluctuations help the processor to find the solutions and that rescaling the problems makes the barriers, that are to overcome thermally, smaller. This assumption also makes sense with respect to the operating temperature of the processor. The energy scale of thermal fluctuation is larger than the gaps of the 18-spin problems.

In Fig. 24 the results for the 8-spin (rescaled and non rescaled) problems are also shown (red markers). The gaps are large compared to the ones of the 18-spin problems. Most of the gaps are larger than the operating temperature and only some are smaller. The D-Wave results (red markers) match the simulation results (violet markers). This indicates that in the area around the operating temperature the dominating process for solving 8-spin problems is quantum annealing. From this it can be assumed that for 2-SAT problems with minimum gaps larger than the operating temperature quantum annealing is the dominating process. Around the operating temperature must be a transition area where the quantum annealing process gets dominated by another process, probably thermal fluctuations, but which does not seem to influence the 8-spin problems.

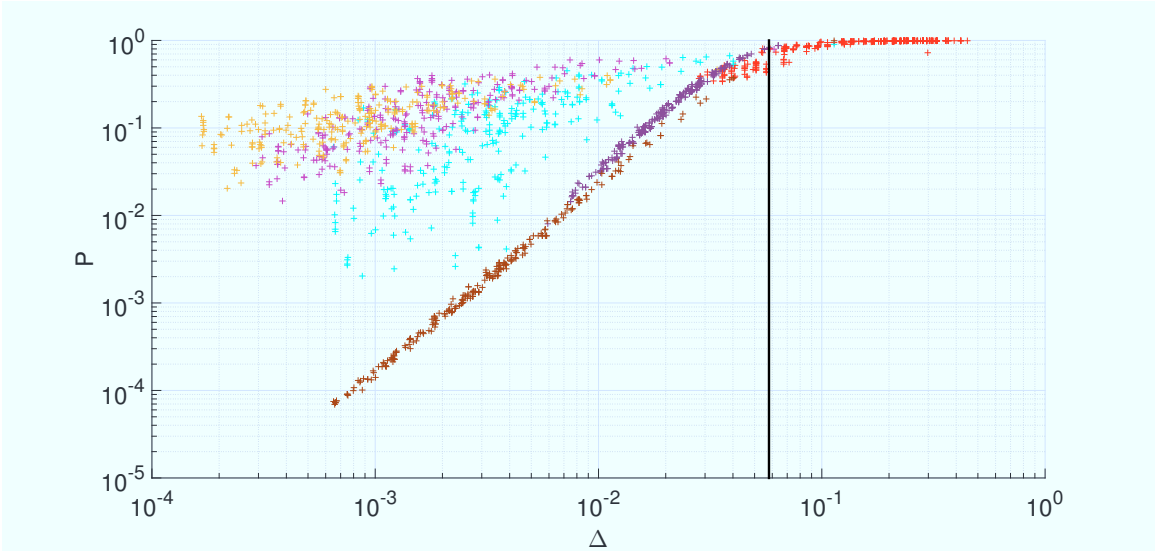


Figure 24: Same as Fig. 23 but the results of the 12-spin problems have been omitted.

This transition area around the operating temperature can be studied further by focusing on the solution frequencies as a function of the gap for the 12-spin and the rescaled

12-spin ($a = 0.425$) problems as depicted in Fig. 25. The solution frequencies of the non-rescaled and the rescaled 12-spin problems with a gap larger than the operating temperature have a similar value range. At the operating temperature the frequencies for the rescaled (green markers) and the non rescaled (blue markers) problems start to exhibit a different behavior. The frequencies of the non-rescaled problems obtained with the D-Wave Two processor still follow the behavior of frequencies obtained by the spin simulations but for the rescaled problems the frequencies obtained with the D-Wave Two processor begin to be higher than the simulated ones. Therefore the transition between the two regions, the region where quantum annealing is dominating (gaps larger than the operating temperature) and the region where thermal fluctuations dominate (below the operating temperature), takes place around the operating temperature. Rescaling for problems with gaps of the size of the operating temperature already helps the processor to solve the problems and suggests that the solutions are obtained by a mixture of thermal fluctuations and quantum annealing. For the non rescaled 12-spin problems however the quantum annealing process still seems to dominate.

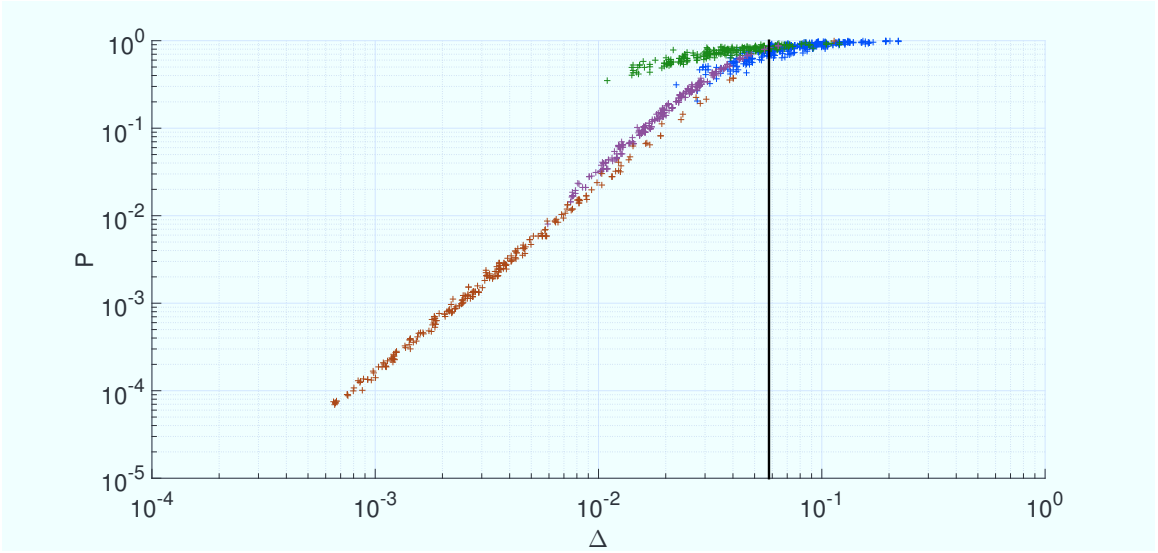


Figure 25: Same as Fig. 23 but only the 12 Spin problems are depicted.

From the results of the 2-SAT problems it can be assumed that the D-Wave machine performs quantum annealing for problems with gaps that are larger than the operation temperature. Problems with much smaller gaps seem to be not solved by quantum annealing and the assumption that thermal fluctuations play a major role can be made. For gaps around the operating temperature the quantum annealing process seems to dominate as long as the problems are not rescaled. Rescaling increases the frequency for finding the solution and is therefore assumed to be at least thermally assisted.

Also the size (number of spins) of the problems seems to play a role in the transition region. This is indicated by the fact that rescaled 8-spin problems seem to follow the

simulated quantum annealing behavior, but the rescaled 12-spin problems do not.

7 Conclusion

This work presents the results of random spanning tree and 2-SAT problems, which were solved using a D-Wave Two processor.

Firstly, the results for random spanning tree problems, translated to Ising spin problems are discussed. The results suggest that the Z_2-symmetry of the Ising spin system implemented on the D-Wave Two processor is severely broken. It is furthermore shown, that isolated regions of qubits on the D-Wave Two processor behave statistically independent. A date and time dependency of the solution frequency is observed as the solution frequency of the same problem on different days differs significantly with respect to the statistical errors.

When starting to operate on a new D-Wave processor, the random spanning tree problems presented in this work can be used as good test cases that allow obtaining basic information on the processor's characteristics. The results of the random spanning tree problem also suggest that, in order to get statistically reproducible results, computations on the D-Wave Two processor need to be performed over long time periods and in different regions of the processor.

Secondly, the solving frequencies for 2-SAT problems with very small minimum gaps solved on the D-Wave Two processor are compared to those obtained via emulation i.e. simulation of the actual physical behavior of the D-Wave Two quantum processor on a classical computer. A comparison of the solving frequencies shows, that 2-SAT problems with minimum gaps larger than the operating temperature of the D-Wave Two processor are solved by quantum annealing. For problems with minimum gaps significantly smaller than the operating temperature of the D-Wave Two processor, the results suggest that finding a solution is assisted by thermal effects. For problems with minimum gaps close to the operating temperature of the D-Wave Two processor, a transition between the quantum annealing process and the thermally assisted process can be assumed.

In order to gain a more detailed insight into the behavior of the D-Wave Two processor, further studies of the 2-SAT problems are planned. An investigation of additional problems in the area of transition is expected to deliver a better understanding of the prevailing processes. It should also be of interest to investigate at what point a problem with a smaller minimum gap than the processor's operating temperature would lead to a decrease rather than an increase of the solution frequency. A decrease of the solution frequency is expected since the energy barriers between states are so small that the likelihood of thermal occupation of exited states increases and also that calibration errors of the D-Wave Two processor become more dominant.

References

[1] L. K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search", in *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pp. 212–219. ACM, New York, NY, USA, 1996.

[2] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", *SIAM J. Comput.* **26** no. 5 (1997) 1484–1509.

[3] D. Deutsch, "Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer", *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* **400** no. 1818 (1985) 97–117.

[4] H. De Raedt and K. Michielsen, "Quantum and Molecular Computing, Quantum Simulations", in *Handbook of Theoretical and Computational Nanotechnology*, M. Rieth and N. Schommers (eds.), vol. 3, ch. 1, pp. 2–48, American Scientific Publisher, Los Angeles, 2006.

[5] T. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, and J. OBrien, "Quantum computers", *Nature* **464** no. 7285 (2010) 45–53.

[6] T. Albash and D. A. Lidar, "Decoherence in adiabatic quantum computation", *Phys. Rev.* **A91** (2015) 062320.

[7] http://www.dwavesys.com/our-company/meet-d-wave *D-Wave Systems, Inc* .

[8] G. E. Santoro and E. Tosattim, "Optimization using quantum mechanics: quantum annealing through adiabatic evolution", *J. Phys. A: Math. Gen.* **39** no. 36 (2006) R393–R431.

[9] A. Das and B. K. Chakrabarti, "Quantum annealing and analog quantum computation", *Rev. Mod. Phys.* **80** (2008) 1061–1081.

[10] P. Ray, B. K. Chakrabarti, and A. Chakrabarti, "Sherrington-Kirkpatrick model in a transverse field: Absence of replica symmetry breaking due to quantum fluctuations", *Phys. Rev.* **B39** (1989) 11828–11832.

[11] D. Johnson, "The NP-completeness column: an ongoing guide", *Journal of Algorithms* **6** no. 3 (1985) 434–451.

[12] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing", *Science* **220** no. 4598 (1983) 671–680.

[13] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, "Quantum Computation by Adiabatic Evolution", *arXiv:quant-ph/0001106* (2000).

[14] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse Ising model", *Phys. Rev.* **E58** (1998) 5355–5363.

[15] Y. Matsuda, H. Nishimori, and H. G. Katzgraber, "Quantum annealing for problems with ground-state degeneracy", *Journal of Physics: Conference Series* **143** no. 1 (2009) 012003.

[16] T. Albash, W. Vinci, A. Mishra, P. A. Warburton, and D. A. Lidar, "Consistency tests of classical and quantum models for a quantum annealer", *Phys. Rev.* **A91** (2015) 042314.

[17] M. Born and V. Fock, "Beweis des Adiabatensatzes", *Zeitschrift für Physik* **51** no. 3-4 (1928) 165–180.

[18] M. S. Sarandy, L.-A. Wu, and D. A. Lidar, "Consistency of the Adiabatic Theorem", *Quantum Information Processing* **3** no. 6 (2004) 331–349.

[19] C. Zener, "Non-Adiabatic Crossing of Energy Levels", *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* **137** no. 833 (1932) 696–702.

[20] N. G. Dickson, M. W. Johnson, M. H. Amin, R. Harris, F. Altomare, A. J. Berkley, P. Bunyk, J. Cai, E. M. Chapple, P. Chavez, F. Cioata, T. Cirip, P. deBuen, M. Drew-Brook, C. Enderud, S. Gildert, F. Hamze, J. P. Hilton, E. Hoskinson, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Lanting, T. Mahon, R. Neufeld, T. Oh, I. Perminov, C. Petroff, A. Przybysz, C. Rich, P. Spear, A. Tcaciuc, M. C. Thom, E. Tolkacheva, S. Uchaikin, J. Wang, A. B. Wilson, Z. Merali, and G. Rose, "Thermally assisted quantum annealing of a 16-qubit problem", *Nat. Commun.* **4** no. 1903 (2013).

[21] H. De Raedt, S. Miyashita, K. Saito, D. García-Pablos, and N. Garcia, "Theory of quantum tunneling of the magnetization in magnetic particles", *Phys. Rev.* **B56** (1997) 11761–11768.

[22] M. Suzuki, S. Miyashita, and A. Kuroda, "Monte Carlo Simulation of Quantum Spin Systems. I", *Progress of Theoretical Physics* **58** no. 5 (1977) 1377–1387.

[23] H. D. Readt, "Product Formula Algorithms for Solving the Time Dependent Schrödinger Equation", *Comp. Phys.* (1987).

[24] T. Neuhaus, "Quantum Searches in a Hard 2SAT Ensemble", *arXiv:1412.5460 [quant-ph]* (2014).

[25] P. Bunyk, E. Hoskinson, M. Johnson, E. Tolkacheva, F. Altomare, A. Berkley, R. Harris, J. Hilton, T. Lanting, A. Przybysz, and J. Whittaker, "Architectural Considerations in the Design of a Superconducting Quantum Annealing Processor", *IEEE Transactions on Applied Superconductivity* **24** no. 4 (2014) 1–10.

[26] R. Harris, J. Johansson, A. J. Berkley, M. W. Johnson, T. Lanting, S. Han, P. Bunyk, E. Ladizinsky, T. Oh, I. Perminov, E. Tolkacheva, S. Uchaikin, E. M. Chapple, C. Enderud, C. Rich, M. Thom, J. Wang, B. Wilson, and G. Rose, "Experimental demonstration of a robust and scalable flux qubit", *Phys. Rev.* **B81** (2010) 134510.

[27] T. Lanting, A. J. Przybysz, A. Y. Smirnov, F. M. Spedalieri, M. H. Amin, A. J. Berkley, R. Harris, F. Altomare, S. Boixo, P. Bunyk, N. Dickson, C. Enderud, J. P. Hilton, E. Hoskinson, M. W. Johnson, E. Ladizinsky, N. Ladizinsky, R. Neufeld, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, S. Uchaikin, A. B. Wilson, and G. Rose, "Entanglement in a Quantum Annealing Processor", *Phys. Rev.* **X4** (2014) 021041.

[28] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose, "Quantum annealing with manufactured spins", *Nature* **473** no. 7346 (2011) 194–198.

[29] H. G. Katzgraber, F. Hamze, and R. S. Andrist, "Glassy chimeras could be blind to quantum speedup: Designing better benchmarks for quantum annealing machines", *Phys. Rev.* **X4** (Apr, 2014) 021008.

[30] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, "Defining and detecting quantum speedup", *Science* **345** no. 6195 (2014) 420–424.

[31] E. Snow-Kropla, "COMPILING PROGRAMS FOR AN ADIABATIC QUANTUM COMPUTER", Master's thesis, Dalhousie University, 2014.

[32] A. Broder, "Generating Random Spanning Trees", in *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, SFCS '89, pp. 442–447. IEEE Computer Society, Washington, DC, USA, 1989.

A Graph of System 13

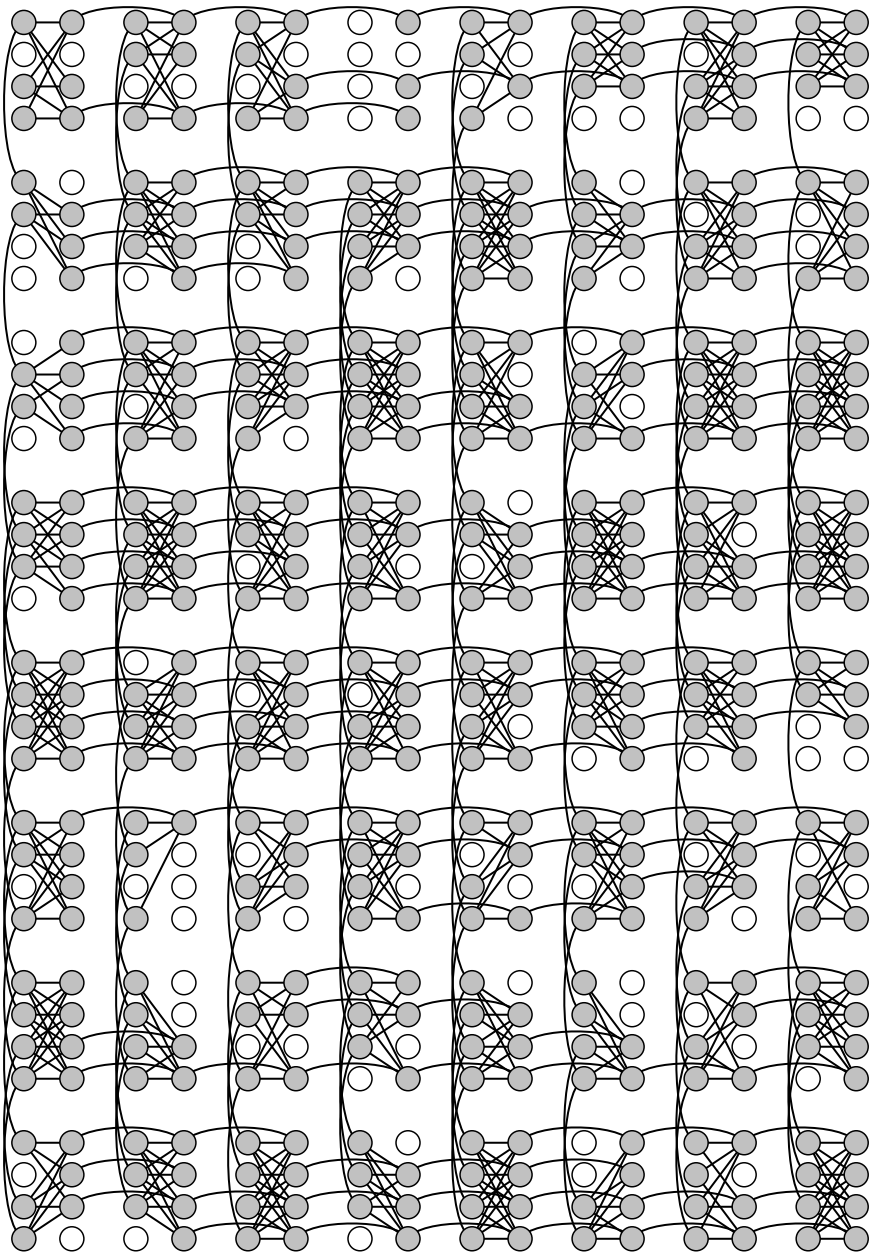


Figure 26: Schematic picture of the graph implemented on the D-Wave Two processor (System 13) that was used in the work. Gray (White) spots indicate (in)operable qubits. Lines indicate qubit-qubit interactions.

B Matlab Code for Operating on the D-Wave Two

```
function [ ]=datacoll ( )
        % D-Wave auth stuff
        url = 'https://qubist.dwavesys.com/sapi/';

        % Set token provided by D-Wave
        token = 'token';

        % Handle
        conn = sapiRemoteConnection(url, token);

        % Chosing solver
        solver = sapiSolver(conn, 'SYSTEM13');

        % geting properties from solver
        props = sapiSolverProperties(solver);

        % Load the h-fields
        load('h.mat')

        % Load the J-couplings
        load('J.mat')

        % sending the sovling request to the processor
        answer = sapiSolveIsing(solver, h, J);

        % saving the answer to a certain location
        filename = ['/home0/l.hobl/Documents/','filename','.mat' ];
        save (filename,'answer');

end
```

C C++ Code for Simulating the D-Wave Two Processor

main.cpp

```
#include "spinsys.h"
#include <string>
#include <iostream>
#include <iomanip>
#include <math.h>
#include <vector>
#include <fstream>
#include <stdio.h>
#include <stdlib.h>
#include <sstream>

using namespace std;

int main() {
        // Get startet
        start();
}

void start() {
    //take input arguments timesteps and Size
    int N;
    int timesteps;
    int start;
    int steps;

    cout<< "Input starting Tau"<<"\n";
    cin>> start;
    cout<< "Input end Tau"<<"\n";
    cin>> timesteps;
    cout<< "Input Tau step size"<<"\n";
    cin>> steps;
    cout<< "Input System size"<<"\n";
    cin>> N;

    // Output to file
    ofstream file;
    file.precision(14);
    file.setf(ios::fixed);
    file.setf(ios::showpoint);
    file.open("psi.txt",std::ofstream::app);

// Run the program for different timesteps
    for (int i=start; i< timesteps ; i=i+steps){
```

```
        //create an object of class spinsys
        spinsys theSimulation(i, N);

        //Run method of class spinsys
        theSimulation.run();
        //Getterfunctions for energy and wavefunction
        double* psi=theSimulation.getpsi();
        double* psii=theSimulation.getpsii();
        double Energy=theSimulation.Energy();

        //output to file
        file << Energy << "\n";
    }
}
```

spinsys.cpp

```
#include <string>
#include <iostream>
#include <iomanip>
#include <math.h>
#include <vector>
#include <fstream>
#include <stdio.h>
#include <stdlib.h>
#include <sstream>
#include "spinsys.h"

using namespace std;

/*-------------------------------------------------------------------*/
// Initialisierung
void spinsys::vInitialisierung(int T,int Size) {
    // Time functions
    gamma=0;
    hz=0;
    delta=0;
    test=0;
    E=0;
    sec=0;

    // Timesteps
    timesteps=T;

    // Timestep size
    tau=0.01;

    // Couplingstrength
    J=0;
    // # off spins
    N=Size;
    // # of states
    Nstates= (int) pow(2,N);

    // Initialise the wave function
    psi = new double[ (int) pow(2,N)];
    for(int i=0; i<pow(2,N); i++){
        psi[i]=1/(double) pow((double) 2,((double) N)/2);
    }

    // inialise imaginary part
    psii = new double[ (int) pow(2,N)];
```

```
    for(int i=0; i<pow(2,N); i++){
        psii[i]=0;
    }
    // initialize rotation matrix
    Fevo = new double *[2];
    for(int i = 0; i < 2; i++) {
        Fevo[i] = new double[2];
        for(int j = 0; j < 2; j++){
            Fevo[i][j] =0;
        }
    }
    // initialize rotation matrix
    Fevoi = new double *[2];
    for(int i = 0; i < 2; i++) {
        Fevoi[i] = new double[2];
        for(int j = 0; j < 2; j++){
            Fevoi[i][j] =0;
        }
    }
    // initialize coupling matrix
    Jevo = new double[4];
    Jevoi= new double[4];
    for(int i = 0; i<4; i++){
        Jevo[i] =0;
        Jevoi[i]=0;
    }
    // initialize Adjazenmatrix
    CMatrix = new double *[16];
    for(int i = 0; i < 16; i++) {
        CMatrix[i] = new double[16];
        for(int j = 0; j < 16; j++){
            CMatrix[i][j] =0;
        }
    }
    // inialise z-field vector
    hfield = new double [ 16];
    for(int i=0; i<16; i++){
        hfield[i]=0;
    }
    // read in Adjazenmatrix and fileds
    readM();
    readH();
}

/*-------------------------------------------------------------*/
// FMatrix time update
void spinsys::FMatrixupdate(){
```

```
    // update the rotation matrix
    gamma=1-(double) t/timesteps;

    double norm = sqrt(gamma*gamma+hz*hz);

    if (norm ==0){
        Fevo[0][0] = cos(tau*norm/2);
        Fevo[1][1] = cos(tau*norm/2);
        Fevoi[0][0] = 0;
        Fevoi[1][1] = 0;
        Fevoi[1][0] = 0;
        Fevoi[0][1] = 0;
    }
    else {
        Fevo[0][0] = cos(tau*norm/2);
        Fevo[1][1] = cos(tau*norm/2);
        Fevoi[0][0] = hz/norm*sin(tau*norm/2);
        Fevoi[1][1] = -hz/norm*sin(tau*norm/2);
        Fevoi[1][0] = gamma/norm*sin(tau*norm/2);
        Fevoi[0][1] = gamma/norm*sin(tau*norm/2);
    }
}

/*-------------------------------------------------------------*/
// JMatrix time update
void spinsys::JMatrixupdate(){
    // update the Coupling matrix
    delta= (double) t/timesteps;
    Jevo[0] = cos(tau* J*delta/1);
    Jevo[1] = cos(-tau*J*delta/1);
    Jevo[2] = cos(-tau*J*delta/1);
    Jevo[3] = cos(tau* J*delta/1);
    Jevoi[0] = sin(tau* J*delta/1);
    Jevoi[1] = sin(-tau*J*delta/1);
    Jevoi[2] = sin(-tau*J*delta/1);
    Jevoi[3] = sin(tau* J*delta/1);
}

/*-------------------------------------------------------------*/
// aplly Xfield rotation
void spinsys::Xfieldrotation(){
    // define local variables
    double R1=0;
    double R2=0;
    double I1=0;
    double I2=0;
    // update all spinstates for all spins
```

```
    for (int l=0 ; l<N ; l++){
                // calculate entry distance
        int i1= (int) pow(2,l);
        hz=hfield[l];

        // update the time dependent Hamiltonians
        FMatrixupdate();

        // calculate the rotation around the x axis (h_x field)
        for (int k=0 ; k<Nstates ; k += 2){
            // get index
            int i2=k&i1;
            int i=k-i2+i2/i1;
            int j=i+i1;
            // Get the real and imaginary part
            R1 = psi[i];
            I1 = psii[i];

            // Get the real and imaginary part
            R2 = psi[j];
            I2 = psii[j];

            // perform matrix vector multiplication and store
            psi[i] = R1*Fevo[0][0]-I2*Fevoi[0][1]-I1*Fevoi[0][0];
            psii[i] = I1*Fevo[0][0]+R2*Fevoi[0][1]+R1*Fevoi[0][0];

            // second wave function entry
            psi[j] = R2*Fevo[1][1]-I1*Fevoi[1][0]-I2*Fevoi[1][1];
            psii[j] = R1*Fevoi[1][0]+I2*Fevo[1][1]+R2*Fevoi[1][1];
        }
    }
}
/*-------------------------------------------------------------*/
// aplly Coupling shift
void spinsys::Couplingphaseshift(){
    // define local variables
    double R1=0;
    double R2=0;
    double R3=0;
    double R4=0;
    double I1=0;
    double I2=0;
    double I3=0;
    double I4=0;
    // calculate the coupling phase shift (J_z coupling)
    for (int l=0; l<N; l++){
        for (int m=l; m<N; m++){
```

```
            // check if the two spins are couples
            J=CMatrix[l][m];
            if (J!=0){

                // update matrix
                JMatrixupdate();
                int nii= (int) pow(2,l);
                int njj= (int) pow(2,m);

                // run through all states for the coupling
                for (int k=0; k<Nstates; k +=4){
                    // get the indexes
                    int n3= k & njj;
                    int n2=k-n3+(n3+n3)/njj;
                    int n1=n2 & nii;
                    int n0=n2-n1+n1/nii;
                    n1=n0+nii;
                    n2=n0+njj;
                    n3=n1+njj;
                    // get the wave functions entry
                    R1=psi[n0];
                    I1=psii[n0];
                    R2=psi[n1];
                    I2=psii[n1];
                    R3=psi[n2];
                    I3=psii[n2];
                    R4=psi[n3];
                    I4=psii[n3];

                    // update the wave function
                    psi[n0] = Jevo[0]*R1-Jevoi[0]*I1;
                    psii[n0] = Jevo[0]*I1+Jevoi[0]*R1;

                    psi[n1]= Jevo[1]*R2-Jevoi[1]*I2;
                    psii[n1]= Jevo[1]*I2+Jevoi[1]*R2;

                    psi[n2] = Jevo[2]*R3-Jevoi[2]*I3;
                    psii[n2]= Jevo[2]*I3+Jevoi[2]*R3;

                    psi[n3]= Jevo[3]*R4-Jevoi[3]*I4;
                    psii[n3] = Jevo[3]*I4+Jevoi[3]*R4;

                }
            }
        }
    }
}
```

```
/*-------------------------------------------------------------------*/
// Next step (time)
void spinsys::vNextStep() {

    // aplly exp(-i*t*H_x/2)
    Xfieldrotation();

    // aplly exp(-i*t*H_z)
    Couplingphaseshift();

    // aplly exp(-i*t*H_x/2)
    Xfieldrotation();
}

/*-------------------------------------------------------------------*/
// Main function
void spinsys::run() {
    // run the program for the number of timesteps
    for (int j=0; j<timesteps+1; j++){
        // timesteps passed
        t=j;

        // Begin the next update Step
                vNextStep();
    }
}

/*-------------------------------------------------------------------*/
// read in a matrix into array void spinsys::readM(){
    string line;
    int row,col;
    ifstream File ("/home0/l.hobl/Documents/NP/J.txt");

    if (File.is_open()){
        row=0;
        while(!File.eof()){
            getline(File, line);
            stringstream ss(line);
            col=0;
            while(ss >> CMatrix[row][col]){
                col++;
            }
            row++;
        }
        File.close();
    }
```

```
    else{
        cout << "Unable to open file";
    }

    for(int i=0;i<N;i++)      {
        for(int j=0;j<N;j++)          {
            cout<<CMatrix[i][j]<<"\t";
        }
        cout<<"\n";
    }
}
/*-------------------------------------------------------------*/
// read in a field into array
void spinsys::readH(){
    string line;
    int row,col;
    ifstream File ("/home0/l.hobl/Documents/NP/h.txt");

    if (File.is_open()){
        row=0;

        while(!File.eof()){

            getline(File, line);
            stringstream ss(line);
            while(ss >> hfield[row]){
            }
            row++;
        }
        File.close();
    }
    else{
        cout << "Unable to open file";
    }
    for(int i=0;i<N;i++)      {
            cout<<hfield[i]<<"\t";
        cout<<"\n";
    }
}

/*-------------------------------------------------------------*/
// Standartkonstruktor
spinsys::spinsys() {
}

/*-------------------------------------------------------------*/
// Konstruktor 1 (used)
```

```
spinsys::spinsys(int ti, int Systemssize) {
    vInitialisierung(ti,Systemssize);
}
/*-------------------------------------------------------------*/
// Destruktor spinsys::~spinsys() {
    for(int i = 0; i < 2; i++){
        delete [] Fevo[i];
    }
    delete [] Fevo;

    for(int i = 0; i < N; i++){
        delete [] CMatrix[i];
    }
    delete [] CMatrix;
    delete [] Jevo;
    delete [] Jevoi;
    delete [] psi;
    delete [] psii;
}

/*-------------------------------------------------------------*/
// Getter functions
double* spinsys::getpsi()const{
    return psi;
}
double* spinsys::getpsii()const{
    return psii;
}
double spinsys::getTime()const{
    return sec;
}
```

spinsys.h

```
#include <iostream>
#include <vector>

using namespace std;

class spinsys {

private:
    //Klassen Variablen
    double* psi;
    double* psii;
    double* hfield;
    double** Fevo;
    double** Fevoi;
    double* Jevo;
    double* Jevoi;
    double** CMatrix;
    double hz;
    double gamma;
    double delta;
    int timesteps;
    int t;
    double J;
    double E;
    int N;
    double tau;
    int Nstates;
    double test;
    double sec;

public:
    //Konstruktoren
    spinsys();
    spinsys(int,int);

    //Destruktor
    ~spinsys();

    //Klassen-Methoden
    void vNextStep();
    void vInitialisierung(int, int);
    void run();
    void Couplingphaseshift();
    void Xfieldrotation();
    void FMatrixupdate();
```

```
    void JMatrixupdate();
    void readM();
    void readH();
    double Energy();
    double* getpsi()const;
    double* getpsii()const;
    double getTime()const;

};
```

Ich versichere, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, den 9. September 2015