BSc thesis APPLIED PHYSICS & APPLIED MATHEMATICS

"Using reinforcement learning for finding the optimal quantum annealing schedule"

MARC CALVIN DIJKSMAN

Delft University of Technology

Supervisors

Dr. E. Greplová

Dr.ir. L.J.J. van Iersel

Other committee members

Drs. E.M. van Elderen

Prof. Dr. A.F. Otte

14th July 2021

Delft

Acknowledgements

I would like to thank my supervisors Dr. E. Greplová and Dr.ir. L.J.J. van Iersel for making this project possible. Dr. E. Greplová has helped me lots along the way and provided me with the needed guidance to finish this thesis. Her research group, QMAI, has also been a very inspiring and nice group to be a part of. I would also like to thank Imelda Romero for her help and for putting up with all my questions about debugging and coding and also for brainstorming with me about the problems I faced. I would also like to thank Dr. E. van Nieuwenburg for following the project closely and giving useful feedback. Lastly I would like to thank Prof. Dr. A. F. Otte and Drs. E.M. van Elderen for agreeing to be part of the assessment committee for this thesis.

I hope you will enjoy reading this thesis.

Abstract

For NP-hard optimisation problems no polynomial-time algorithms exist for finding a solution. Therefore, heuristic methods are often used, especially when approximate solutions can be satisfactory. One such method is quantum annealing, a method where some initial Hamiltonian is slowly perturbed to anneal towards a problem Hamiltonian. The annealing schedule should follow the adiabatic theorem of quantum mechanics and should therefore be slow to yield the most accurate results. Finding a schedule that is both fast and still accurate has been referred to as a 'black art' and is usually just guessed. In this thesis reinforcement learning (RL) is explored to see if it can help with finding the optimal quantum annealing (QA) schedules. The Hamiltonians used are of the form of a transverse field Ising model. These are well studied Hamiltonians that can map to many NP-complete problems [1]. Finding a good balance between going both fast and being precise proved to be difficult and needs further research. Therefore this thesis mainly dealt with training an RL agent to find the most accurate QA schedule. The results show that the agent learns to find the most accurate (slowest) annealing schedule after 300,000 learning steps for a problem Hamiltonian with a single coupling constants J and no reward shaping. Annealing in an environment with a spin glass problem Hamiltonian proved to be difficult and has not shown good results in this thesis, this needs further investigation. Nonetheless, the result on the single coupling constant Hamiltonian shows the potential of an RL agent for learning in a quantum annealing environment with very sparse rewards. This paves the way for further research into an RL agent that can find a schedule that is both fast and accurate on a wide range of problem Hamiltonians.

Contents

1 Introduction

We solve optimisation problems on a daily basis. When commuting we optimise our route and when packing our bags we optimise the amount of things we can bring along. These problems often seem very trivial, but their simplicity may not be a reflection of their computational complexity. Optimization problems can sometimes be very hard to compute in a reasonable time. A notoriously hard subset of optimization problems are the so called NP-complete problems. This subset consists of variations of the famous travelling salesman problem (TSP), the knapsack problem, Boolean satisfiability and many more. Many NP-complete problems can be found in [2]. More about optimisation problems will be explained in §2.1.

To solve NP-hard problems efficiently we need to come up with smart ways of exploring the solution space to find the optimal solution. There are, namely, no known algorithms that solve these problems with N independent variables in a time bounded by a polynomial in N. They can therefore take exponentially long to compute. In this paper we will explore a heuristic way of solving optimization problems by translating them to a structured Hamiltonian problem, where the Hamiltonian is represented by the Ising model. Once we have found the ground state of the Hamiltonian it can be mapped back to the initial optimisation problem to give the optimal solution. Simulated annealing has been used for finding the ground state of such Hamiltonians. However, simulated annealing tends to get stuck in local minima when energy barriers become too high. Therefore, we will use quantum annealing (QA) in the hope to escape local minima efficiently via tunneling [3]. The adiabatic theorem of quantum mechanics states that that as long as you perturb a system that is in it's ground state 'adiabatically' it will remain in the corresponding ground state[4]. Therefore, we can find the ground state of a problem Hamiltonian by annealing adiabatically from the ground state of some initial Hamiltonian. This idea of quantum annealing has shifted the focus of many scientist in order to optimise the technique, some prominent papers from this niche are collected by Ghosh and Mukherjee [5].

For quantum annealing to work we need to know the 'annealing schedule'. The aim is to arrive close to the ground state while not taking too long, however, going too fast will be detrimental. Finding such an optimal schedule is very hard to do and we therefore employ reinforcement learning (RL) in this thesis in an attempt to find the optimal annealing schedule. Finding a suitable annealing schedule is often done heuristically and frequently just a simple linear schedule is used. RL has already been proven to work on the problem of finding an annealing schedule for simulated annealing (SA) but has never before been used for quantum annealing (to the best of my knowledge) [6]. That problem is much related to the problem of finding an optimal schedule for QA and therefore inspired us to implement it on our own problem. The perks of using RL are that an agent can be used to efficiently explore different actions without knowing anything about the problem. One only needs to define an environment which models the problem and give rewards or penalties for good or bad behaviour. In this way the agent can in principle learn to find the fastest annealing path and avoid local minima, if it is given the right reward scheme. The drawbacks are that training such a model is very time consuming and tuning the (hyper-)parameters is not straight forward. One must also be careful not to over-engineer the reward scheme, which could hamper the search for the optimal schedule. Therefore, the reward scheme used in this thesis gives only a reward when the anneal is finished as to not interfere with the agent's choices. The ends thus justify the means.

To sum up, this thesis will explore the possibility of using RL for finding the optimal annealing schedule for QA. In §2 all the theory is explained in more detail and the model is also described in §2.6. In §3 the annealing schedule that was constructed by an agent trained on this model is presented along with some samples of energyspectra that show the energy landscape of the problem. The results show how an RL agent can learn to do find the most accurate annealing schedule, given enough training steps and a sufficient reward scheme. The goal of this thesis was to train an agent to find the optimal annealing schedule. This was not achieved however due to a lack of time. Finding the balance between rewards for going fast and being accurate still needs to be done. The implications of the results are discussed in §4, where also the shortfalls of the model are further discussed and some recommendations are made for further research.

For the interested reader the full code can be found at: `https://gitlab.tudelft.nl/qmai/theses/calvindijksman`.

2 Theory

In this section the theoretical background for the method in this thesis is presented. First, some theory for optimisation problems is given, understanding this will explain the relevance of this thesis. After that the 'Ising model' is explained, this is the backbone of the heuristic optimisation technique used for solving optimisation problems. Then, simulated annealing is described, which serves as the inspiration for quantum annealing. Following the explication of quantum annealing, we develop the basis of reinforcement learning. Lastly, the eventual reinforcement learning model is presented.

2.1 Optimisation problems

Optimization problems are, as the name suggests, problems where one needs to find the optimal solution from the collection of all feasible solutions. The function to be optimized is called the cost function or the objective function. This objective function is subject to certain constraints. For example, the problem could be to pack a bag, however it can only hold a certain weight and a certain volume. For small problems, e.g. finding a route in a small district, one can often easily find the best solution by trial and error. But, for bigger problems this becomes impossible in finite time. Optimisation problems are divided into classes based on their complexity. There are problems for which there exist polynomial-time algorithms, these are in the class P [2]. These problems are generally the least complex. Every optimisation problem also has a decision problem variant.

A decision problem doesn't ask for a specific optimal solution, but rather asks whether a certain solution exists, these can be answered by 'yes' or 'no'. An example would be: Is there a route shorter than 5 Km? As opposed to: What is the shortest route? A certificate can be anything that certifies the answer to a decision problem, often it is just a solution, but sometimes this may not be enough. If for a decision problem there exists a polynomial-time algorithm to check whether a given certificate proves that the instance is a yes-instance, then the decision problem is in the so-called class NP for 'non-deterministic polynomial time'. The question whether P=NP remains unanswered although it is generally assumed very unlikely and therefore we often conjecture that P≠NP.

More interesting are the NP-hard and NP-complete classes. A decision problem is said to be NP-hard if there exists a polynomial-time reduction to the problem from each problem in the class NP. This is equivalent to saying that the problem is at least as hard as every problem in the class NP. This implies that if there exists a polynomial-time algorithm for any NP-hard problem, then NP=P, which would be a phenomenal discovery. The final class of decision problems is the NP-complete class. A problem is in this class if it is NP-hard and in the class NP. To illustrate the complexity classes see figure 2.1. There you can also see the implications that the P versus NP problem causes for the complexity of an optimisation problem. You can see that (assuming P≠NP) NP-complete problems constitute the hardest problems of the class NP, many of these are described in Ref.[2]. The NP-hard problems that are not NP-complete are the most complex in general. An important example is the halting problem which asks whether a given computer program will finish running or run forever [7].

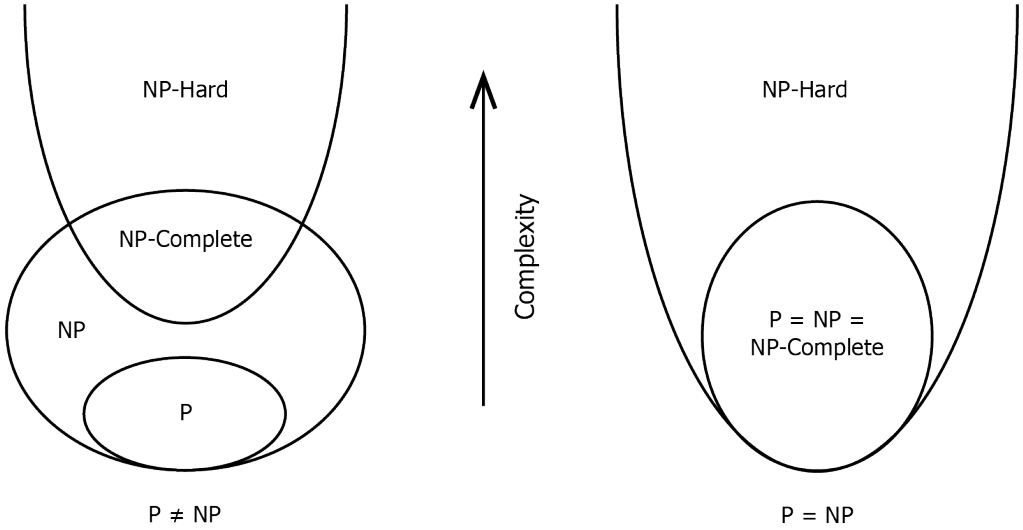

Fig. 2.1: A representation of the complexity classes.

Since each type of optimisation problem asks for a different approach many algorithms exist for solving different kinds of optimization problems, like the simplex method for solving linear optimization problems. However, those algorithms (including the well-known simplex method) don't always perform as desired. The general consensus is that a good algorithm scales at most polynomially with the system size while exponential scaling indicates that the computational time explodes with the system size. To date, no algorithm is known to scale polynomially on all optimization problems and so there does not (yet) exist a *general* optimization algorithm.

2.2 Ising Model

Since for NP-hard problems the computation cost is unlikely to scale polynomially with the problem size heuristic approaches are often considered. One of them is annealing, where an optimization problem is mapped to the Ising model. Solving for the ground state of the Ising model is a promising idea to solve the problem quicker. Here the Ising model will be further explained.

The Ising model is a mathematical model in statistical mechanics. It consists of a lattice of sites that can have spin up or down. The model can be used to represent multiple things, most commonly a magnet [8]. In the simplest model each lattice site can only interact with its nearest neighbours and can additionally be affected by an external magnetic field. The total energy of the system is then described by the following Hamiltonian:

$$H = -\sum_{\langle ij \rangle} J_{ij} s_i s_j - \sum_j h_j s_j \tag{2.1}$$

The s_i in Eq.2.1 represent the states of the individual lattice sites, these can be either up or down, $+1$ or -1 respectively. The system then has 2^N possible configurations where N denotes the number of spins. The J represents the coupling constant that governs the interaction between the neighbouring lattice sites. The magnitude of J tells us how strongly two neighbours are coupled and the sign tells us the type of coupling. The positive couplings are called ferromagnetic, while negative couplings are called anti-ferromagnetic, respectively preferring alignment or anti-alignment of the spins. Summing over all pairs of nearest neighbours gives the coupling interaction contribution to the total energy of the system. The h in Eq.2.1 models the strength of a potential longitudinal magnetic field, the sign tells us whether spin up or down is preferred. Positive h's prefer spins in the up position. The sum over all sites gives the contribution to the energy of the external magnetic field.

An important type of Ising model is one where the coupling constants are a mix of positive and negative values, this is then called an 'Ising spin glass'. These are more complicated to solve than an Ising model

with only positive or negative couplings. Additionally more complex models can include long range interactions so the first term in Eq.2.1 becomes $-\sum_{i<j} J_{ij} s_i s_j$. Solving for the ground state of a spin glass is a so-called 'QUBO' (quadratic unconstrained binary optimization) problem [1]. This problem is known to be NP-complete for the two-dimensional case in a magnetic field and in the case of more than two dimensions [9, 10]. Because of this property analogies between NP problems and Ising spin glasses have been frequently studied. Since the decision problem of finding the ground state of an Ising spin glass is NP-complete we can find a mapping between any NP-complete problem to the Hamiltonian of an Ising spin glass. Many of these mappings are presented in [1]. These serve as a framework for translating many NP problems into Ising models.

In this thesis we will focus on a special case of the Ising model called the transverse field Ising model. We will in fact use the Sherrington-Kirkpatrick (SK) model in a transverse field, where we also include long-range interactions thus making it more general [11]. This is actually the general quantum analog of the classical Ising model that was just described. The transverse and longitudinal fields will both be tuned down via an annealing schedule. The longitudinal field is added to break trivial degeneracies of the ground and excited states following the works of Rajak and Chkrabarti in Ref.[12]. The SK model was also shown to be NP-hard. The Hamiltonian that was used is given as follows:

$$H = -\sum_{i<j} J_{ij}\sigma_i^z\sigma_j^z - \Gamma\sum_i \sigma_i^x - h\sum_i \sigma_i^z \tag{2.2}$$

These spin states are represented by the Pauli spin matrices σ_i. For a multi-spin system these are defined as: $\sigma_i^\alpha \equiv \mathbb{1}_1 \otimes \cdots \otimes \mathbb{1}_{i-1} \otimes \sigma_i^\alpha \otimes \mathbb{1}_{i+1} \otimes \cdots \otimes \mathbb{1}_N$. The transverse field term, $\Gamma\sum_i \sigma_i^x$, acts as the driver for quantum tunneling between various states and is the control parameter of the system. This is the quantum analogue of the temperature which causes disorder in the classical Ising model. The J_{ij} are i.i.d. random variables that follow the normal distribution with mean zero and a variance of one.

2.3 Simulated annealing

Annealing is a technique that is commonly used in metallurgy where a material is heated and subsequently cooled in a controlled manner to change the material's properties like hardness and ductility [13]. This way of manipulating materials to reorganise the internal structure to have certain properties inspired scientists to use a similar method, but for solving optimization problems. This method is called *simulated annealing*. The main idea is that the configurations of the spins in this Ising model then map to the optimisation problem. By finding the ground state energy we can then find the optimal solution to our optimisation problem.

The problem we have now is: How do we find this ground state of the Ising model? To achieve this we first heat the system to initialize it in a random configuration. We then gradually cool the system to relax it into a lower energy configuration while trying to escape from any local minimum [14]. The simulation is done using the Metropolis-Hastings algorithm to simulate the configuration of the Ising model at arbitrary temperatures [15, 16]. At each step of the algorithm the system is perturbed and the resulting change in energy ΔE is calculated. If $\Delta E \leq 0$, the new configuration is accepted. However, if $\Delta E > 0$ the acceptance is treated probabilistically with $P(\Delta E) = exp(-\Delta E/k_B T)$. A random number is chosen from a uniform distribution of $(0, 1)$ and if this number is less than $P(\Delta E)$ the new configuration is retained, else the original configuration is kept. Getting stuck in local minima can therefore be avoided, since there is always a slight chance of a transition out of the current state even though the energy might not decrease. However, as the temperature drops this provability decreases and the system therefore settles in a state. Settling too soon must be avoided, because the system then becomes non-ergodic preventing it from exploring the full configurational space. Since there is no way of knowing a priori whether we will find the lowest energy state this is considered a heuristic method for optimization. If the temperature is decreased too quickly the system might settle in a local minimum while going too slow results in unnecessary computational expense. Finding the optimal schedule that anneals to a state sufficiently close to the ground state while not taking too long remains an open question. In their paper Geman and Geman have proved a theorem on the annealing schedule for a generic combinatorial optimization problem. It states that the ground state will be reached if the temperature is decreased as $T(t) \geq c/log(t)$, where c is a constant determined by the system size

and other structures of the model [17]. In practice, however, good results can be achieved with faster schedules.

One of the main examples where *simulated annealing* can be used is in the travelling salesman problem (TSP) [14]. This is a multivariate optimisation problem where a travelling salesman has to pass through a list of cities and return back home. Where we define the distance between cities as the cost for travelling between them. The problem is then to find the shortest possible route. All known exact methods require a computing time that scales exponentially with the number of cities. Hence, the heuristic method of simulated annealing might yield satisfactory results.

2.4 Quantum annealing

In systems with many local minima where the minimum energy configurations are separated by high barriers the system may become non-ergodic. A finite temperature will then not be able to make the system ergodic, so the full phase-space wont be explored by the system. The outcome will be far off the optimum in this case. A different approach must be considered. The control parameter in SA is the temperature. However, there are more ways of driving a system of spins into a certain ground state. Quantum tunneling for example can cause transitions between states and thus play the same role as thermal fluctuations in SA [18]. The idea of using quantum fluctuations is implemented using the transverse Ising model (Eq. 2.2), where the transverse field term is a function of time. The transverse term starts of highly mixed into the system and is then slowly decreased via the annealing schedule. Again, finding the optimal annealing schedule poses some difficulties. In quantum annealing the process is governed by the adiabatic theorem of quantum mechanics. This theorem states the following:

Theorem 1 (Adiabatic theorem)**.** *If one changes the Hamiltonian gradually (adiabatically) from some initial form $\hat{H}(0)$ in the nth eigenstate to some final form $\hat{H}(T)$ then the system will be carried into the nth eigenstate of the final Hamiltonian [4].*

Using the adiabatic theorem we first start with a system that has a Hamiltonian for which we know how to put it in the ground state. If we then put this system in the ground state and we can change the Hamiltonian slowly enough to some difficult non-commuting Hamiltonian, the adiabatic theorem ensures that we end up in the ground state of this final Hamiltonian. Therefore, if we want to know the ground state of some Ising model we could start in the ground state of some non-commuting Hamiltonian and anneal adiabatically to the Hamiltonian of the Ising model. We will then end up in the ground state and can thus solve the optimisation problem that maps to this specific Ising Hamiltonian. The full Hamiltonian is given by Eq.2.2 with some modification for the monotonic decrease of the transverse term. We arrive at the following Hamiltonian:

$$H = (1 - s(t)) \cdot H_0 + s(t) \cdot H_p \tag{2.3}$$

Here the H_0 is the initial Hamiltonian which is given by the transverse and longitudinal fields. The H_p term is the final Hamiltonian which maps to the optimisation problem we're trying to solve. The $s(t)$ term gives us the schedule for how the Hamiltonian evolves towards the problem Hamiltonian. This schedule should increase monotonically towards 1 and is thus bounded by the interval [0,1]. At $t = 0$ the Hamiltonian is initially equal to H_0 and since it is non-commuting with H_p it is in a superposition of all possible states in the eigenbasis of H_p. The time evolution is done via the unitary evolution of the Hamiltonian:

$$|\psi(t)\rangle = \hat{U} |\psi(t=0)\rangle = \sum_{i=0}^{2^N} e^{-iE_i t/\hbar} |\psi_{E_i}\rangle \langle\psi_{E_i}|\psi(t=0)\rangle \tag{2.4}$$

One thing to be aware of is the energy spectrum. This spectrum, shown in Fig.2.2, indicates how different energy levels are separated and restrict the speed of the annealing. When the energy gap is small the system is more likely to transfer to a higher eigenenergy due to thermal fluctuation or Landau-Zener transitions [19, 20].The adiabaticity condition is given by the following equation [21]:

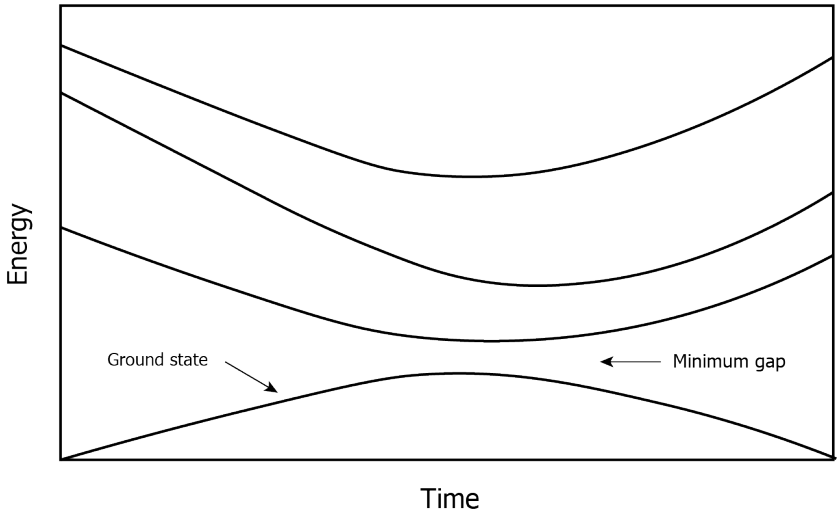

Fig. 2.2: An example of an eigenspectrum where the energy gap is indicated.

$$\frac{1}{\Delta_j(t)^2}\left|\langle j(t)|\,\frac{dH(t)}{dt}\,|0(t)\rangle\right| = \delta \ll 1 \tag{2.5}$$

This equation clearly shows that the anneal must slow down close to a small energy gap, Δ_j namely depicts the size of the energy gap between the ground state and the jth excited state, $\Delta_j(t) \equiv \epsilon_j(t) - \epsilon_0(t)$.

In systems where barriers between states are high SA may struggle. In SA the probability of escaping a barrier is of the order $exp(-\frac{\Delta V}{T})$, where ΔV denotes the height of the barrier and T denotes the temperature [22]. For QA this probability of the order $exp(-\frac{w\sqrt{\Delta V}}{\Gamma})$, where w denotes the width and Γ denotes the strength of the quantum fluctuations (how much the transverse term is mixed in). This shows the theoretical advantage of QA over SA due to the factor $\sqrt{\Delta V}$ instead of ΔV in the exponent. Since the quantum tunneling probability only decreases with the width of a barrier using quantum tunneling should speed up the relaxation time [3]. In 1999 this advantage of QA over SA was experimentally shown by Brooke et al.[23]. In Ref.[24] it was observed that for 3-D Ising spin glasses QA finds a low energy local minimum quickly, while increasing the annealing time leads to SA finding a lower energy state. The search for problems where quantum speedup is achieved goes on, in Ref. [25] some benchmark problems are proposed while Ref.[26] defines how to measure this speedup and avoid false claims of quantum speedup. Since the method proves to be faster for some problems while it is slower for others it should be used as a supplement to classical annealing in the right circumstances [6, 18, 24, 27–30].

2.5 Reinforcement Learning

Since optimising the annealing schedule can dramatically accelerate the computational process this is a much investigated topic [20, 24, 31, 32]. The technique used in this thesis for finding an optimal annealing schedule is reinforcement learning (RL). This is a machine learning technique that employs a so called agent to solve a problem. Instead of training on a given data set, like in supervised and unsupervised learning, the RL agent explores the environment by itself. The agent is put in an environment and is given an observation, for example the state of the system. The agent is then free to choose from a given set of actions which change the environment when performed. Based on whether it does something good it will receive a reward, if it does not it can also receive a penalty. In this way the agent explores the environment and is able to learn a good strategy, also called a policy ($\pi(a, s)$), by maximising the rewards. The policy tells us how likely an action is in a given state. In the standard RL setting, the model is a Markovian Decision Process (MDP) and the rewards should be Markovian as well. This means that the rewards depend only on the current state and not how the agent got there.

Sometimes it is not beneficial to try to get immediate rewards, since in the long term rewards might be better for another action. The agent might then get a higher cumulative reward in the long run if opting for a low immediate reward. The agent can learn this if given enough time [33]. Additionally a discount function γ can be used to communicate how important future rewards are as opposed to immediate rewards. Some RL methods use a value function for this, the value function determines which actions yield good results in the future as opposed to the rewards which give immediate feedback. The main goal of learning is to maximise the expected discounted cumulative reward, $\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right]$. Here $\gamma \in (0, 1]$ is the the discount factor which regulates the bias towards future rewards, the higher γ is the more it favours future rewards, r_t is the reward at time t. How an RL agent learns is shown schematically in Fig.2.3 below.

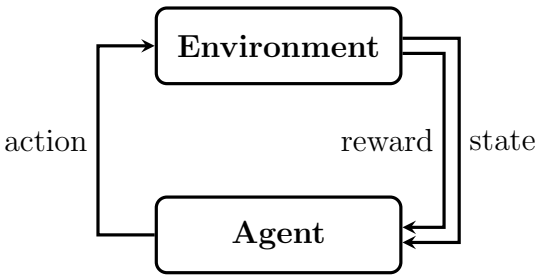

Fig. 2.3: Diagram showing how an RL agent learns.

The policy that is returned by the agent after training determines which actions will be chosen based on the state of the environment that is observed. It comes in two variants, deterministic and non-deterministic. The latter returns a stochastic policy, where a distribution $\pi(a, s)$ determines the probability for the agent taking a certain action a given a state s. The deterministic policy function on the other hand maps a given state to an action [34]. Many algorithms exist for finding the optimal policy function, but this thesis will focus on proximal policy optimisation (PPO) [35]. PPO is a model-free approach which means that the agent tries to estimate the value function or the policy directly from experience. PPO is a type of policy gradient method which means that the coefficients of the policy are incrementally improved towards a local minimum or maximum. The gradient of the objective function is usually estimated by the function in Eq.2.6 below[35]:

$$\nabla_\theta \hat{\mathbb{E}}_t \left[\sum_{t=0}^{\infty}\gamma^t r_t\right] = \hat{\mathbb{E}}_t \left[\nabla_\theta log \pi_\theta(a_t, s_t)\hat{A}_t\right] \tag{2.6}$$

Here $\hat{A}_t$ denotes the advantage function which is a measure of how good a certain action is given the state. It is given by $A(a, s) = Q(a, s) - V(s)$, where $Q(a, s)$ gives the Q-value and $V(s)$ is the value function. The Q-value gives the expected cumulative reward for the full episode after an action is taken in a certain state, while the value function gives the expected cumulative reward before an action is taken in a certain state. If $A > 0$ the action is good, ensuring a better policy, while $A < 0$ denotes a bad action. The advantage function is thus a very noisy estimate of the real advantage. With PPO we don't strive for only positive advantage functions, to stimulate exploration and efficiency negative advantage functions are allowed [35]. Normal gradient methods suffer from two pitfalls, it can take very long by taking small steps or if it can fall off the best trajectory by taking destructively large policy updates. PPO has a solution for this based on a trust region policy optimisation method. A surrogate objective function is maximised, this objective function is constrained by a penalizing large changes to the policy. This will constrict the size of updates to the policy and therefore make it more stable. The main training objective function used by the PPO algorithm is given by three terms in Eq. 2.7 below [35]:

$$L_t^{CLIP+VF+S}(\theta) = \hat{E}_t \left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right] \tag{2.7}$$

L_t^{clip} is the main objective given by Eq.2.8. L_t^{VF} is the squared-error loss $\left(V_\theta(s_t) - V_t^{targ}\right)^2$, it estimates

how good it is to be in the current state and $S[\pi_\theta]$ denotes an entropy bonus that stimulates exploration. Having a high $S[\pi_\theta]$ pushes the policy to behave more randomly until the other parts of the objective start dominating. Here c_1 and c_2 are coefficients that weigh the contributions of the L^{VF} and $S[\pi_\theta]$ terms. The objective function in Eq.2.7 is optimised via stochastic gradient descent (or Adam) where a random training sample is used for updating the parameters via gradient descent [36].

$$L^{CLIP}(\theta) = \hat{E}_t \left[min \left\{ \frac{\pi_\theta(a_t, s_t)}{\pi_{\theta_{old}}(a_t, s_t)} \hat{A}_t, clip \left(\frac{\pi_\theta(a_t, s_t)}{\pi_{\theta_{old}}(a_t, s_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right\} \right] \tag{2.8}$$

In $L^{CLIP}(\theta)$ we clip the $r(\theta) = \frac{\pi_\theta(a,s)}{\pi_{\theta_{old}}(a,s}$ function, since if the probability of an action in the current policy has a positive advantage function and is much more likely in the new policy we might destroy our old policy based on this result. Alternatively for a negative advantage function the $r(\theta)$ function is clipped by $1-\epsilon$ since otherwise the policy probability could be reduced to zero, again this is not desirable. Remember that the advantage function is only a noisy estimate so we don't want to destroy our policy based on a single estimate, that is why we clip the function and in effect apply a 'trust region'. The updates to the policy thereby stay within a close proximity to the old policy.

Since the model can have many possible state action pairs, we need a way of storing the policy efficiently. One can not simply save a list of all possible state and the required action when the state space becomes too large. PPO uses neural networks to solve this. Neural networks are in fact function approximators and any function can be approximated by a neural network that is sufficiently large [37]. This function approximator can be used to save the policy, since similar states should yield similar actions. Neural networks are a way of connecting observations (input) to the action that needs to be taken (output). For these connections 'neurons' are used that are connected to one another. This is analogous to how the human brain works, where different neurons light up based on the input [38]. Fig. 2.4 shows the diagram for RL using deep neural networks. The state that is observed produces an action chosen by the neural network.

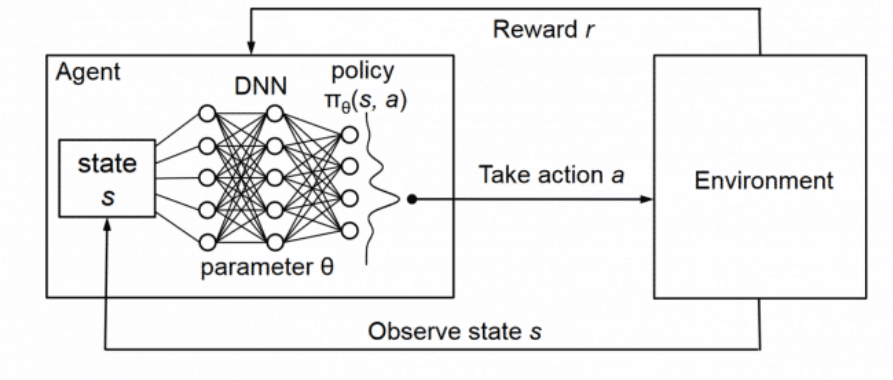

Fig. 2.4: Reinforcement learning with deep neural network to map a state to an action. Retrieved from [39].

One crucial thing we haven't discussed yet is the reward scheme. This might be the most important aspect of RL and greatly determines the outcome. The reward scheme should encourage the agent to take good actions. By giving intermediate rewards the agent might be shaped to show a particular behaviour while giving sparse rewards leaves agent free. Sparse reward settings thus generate agents that can have a very unorthodox and unique way of completing tasks [31].

But why use reinforcement learning? Reinforcement learning is a method well suited for problems where the model is very complex or that lack precise descriptions as long as the reward scheme correlates with the objective. Also, RL agents can be optimised to become more efficient as they are trained for longer and they can become resilient towards varying conditions [39]. The technique of reinforcement learning was recently used to create *AlphaZero*, an algorithm that teaches itself to become better and better. It

has self-taught the game of Go so well that it can beat the famous *AlphaGo* algorithm, which defeated the world champion at the game of Go in 2017 [40]. This shows how RL can effectively learn complicated tasks without any prior knowledge or human interference and become better than humans. In Ref.[6] reinforcement learning was used to find the optimal schedule for simulated annealing. The technique proved to be very successful and even resilient towards local minima. It therefore inspired us to use the same technique for optimising quantum annealing schedules.

2.6 RL model for finding optimal annealing schedule

Here the model that was used for optimising the annealing schedule will be explained. The model describes the environment for the agent, the actions it can take and the rewards it will receive. The environment models the quantum annealing of a transverse field Ising model and is used by the RL agent to learn the annealing schedule. It is written as a custom *gym* environment, *gym* is a *Python* toolkit for implementing RL algorithms. The environment is built using four functions, the initialisation function, the step function, the reset function and the render function. The latter has no use in our implementation. The pseudocode for the definition of the environment is shown in Pseudocode 1 in the Appendix.

The initialisation function defines the starting point for our agent. It first includes the definitions of the problem Hamiltonian and the initial Hamiltonian. For this thesis a longitudinal magnetic field was used to eliminate trivial degenerate states, as was done in Ref.[12] and Ref.[18]. Here we will follow the works of Rajak and Chkrabarti [12] where we will tune down the longitudinal field along with the transverse field to eliminate the error that the longitudinal field otherwise creates in the final outcome. We will thus use the Hamiltonian from Eq.2.2 and anneal following Eq.2.3 with $H_0 = -\sum_i \sigma_i^x - h\sum_i \sigma_i^z$. Without loss of generality the transverse field strength, Γ, is set to 1 here. The h is determined via a normal distribution, but with mean 0 and variance 0.3 to introduce only a slight symmetry breaking. The final problem Hamiltonian is given by $H_p = -\sum_{i<j} J_{ij}\sigma_i^z\sigma_j^z$. The coupling constants J_{ij} were used in two settings, one with all equal couplings and one with random i.i.d. couplings following the normal distribution with mean 0 and variance 1. Besides these formulations of the Hamiltonian the initial state of the model is also specified, this is done by finding the ground state of H_0. To do this we used the open source *Python* framework called '*QuTip*' for solving quantum dynamics [41]. Also the time, $t = 0$, and the annealing gradient, $z = z_{initial}$, are initialised in this function.

The next function is the step function, where the actions for the agent and their consequences to the environment are defined. We used three different actions, increasing z, keeping z unchanged, or decreasing z. Here, increasing z corresponds with annealing faster. The actions therefore define the annealing schedule by increasing or decreasing the slope. The annealing schedule in the model is described by $s(t)$, a function of the time t and the gradient or annealing speed z. As mentioned the annealing schedule is bounded, $s(t) \in [0, 1]$, and should be monotonically increasing, $s(t_1) < s(t_2)$ for $t_1 < t_2$. After each step the z-value can change according to the action that is taken, this would mean that after one step the value of $t \cdot z_{old} \neq t \cdot z_{new}$, this could therefore result in non-monotonic behaviour. To combat this, the difference $a = t \cdot z_{old} - t \cdot z_{new}$, is calculated after each step and added to a residue, $res = res + a$ where $res_{initial} = 0$. After this the time step added can be determined. This is done by partitioning the interval. At first $s(t = 0) = 0$, so the schedule must still grow by $tz_{res} = 1$, this is divided up into the amount of maximum steps. The time step added is then calculated by:

$$t_{step} = \frac{tz_{res}}{z_{new} \cdot (steps_{max} - steps_{current})} \tag{2.9}$$

After this the new tz_{res} is calculated by $tz_{res} = tz_{res} - (z \cdot t_{step})$ where initially $tz_{res} = 1$. Now the time-evolution of the quantum state with the time varying Hamiltonian is computed over the duration of the time step using the *Qutip* framework via the *mesolve* function. This function essentially solves Eq.2.4 and returns the state ψ. The final state of this time-evolution serves as the new initial state for the time-evolution in the next step. After the time-evolution the expectation value of the final Hamiltonian, H_p, in the current state is calculated by $\langle\psi| H_p |\psi\rangle$. For the training we know the real ground state of the problem Hamiltonian and can therefore calculate how close we are to it by calculating the difference, $|\epsilon_0 - \langle\psi| H_p |\psi\rangle|$. This difference serves as an observation for the agent together with the current time. After the action is taken and the state is evolved, the step function enters into the reward scheme. In

our reward scheme the agent only receives a reward or a penalty at the end of a full anneal, if it has not finished only a 1 is added to the step counter. This was done to minimise interference with the agent thus letting it explore and find a schedule that it thinks works best without pushing it towards a schedule that we assume to be good. This scheme has also shown to be effective in the study by Mills et al. [6]. The ends justify the means in this case. Since the reward is so sparse the agent will need to train quite long before any reasonable behaviour is observed. The reward is based on the difference between the true ground state energy and the expectation value of the energy, where ϵ_0 denotes the known ground state energy:

$$\frac{1}{|\langle\psi| H_P |\psi\rangle - \epsilon_0|} \tag{2.10}$$

Since this reward is the only feedback the agent gets it should now learn to go as slow as possible since that would be the most adiabatic and hence most accurate path and thus yield the highest rewards. To accommodate for a reasonable computing time the z values are clipped below 0.0001, this also ensures that the z values cannot become negative. Speeding up the agent to find a faster optimal schedule would mean that the reward schedule should strike a very intricate balance between accuracy and speed. Finding this balance has not worked out during this research so this will not be considered in the rest of this thesis.

The next function is the 'reset' function. This is called after each full anneal when $done = True$. The reset function re-initialises the environment by resetting the time, z and the Hamiltonian with the corresponding initial ground state. After each reset the Hamiltonian is thus different due to the normally distributed coupling constants and h. This enables the agent to learn the dynamics of quantum annealing better. With this limited information it is then set out to learn the annealing schedule. As discussed above we are using the PPO algorithm for this. We used the *Stable-Baselines* toolset for implementing their 'PPO2' algorithm [42].

Tuning and finding good working hyperparameters for an RL algorithm is often done heuristically. Since there are so many of them at play it is very hard to find a good balance. Some guide the algorithm towards more exploration, like the entropy coefficient, while others guide it towards convergence, like the learning rate. The hyperparameters used in this thesis were also chosen by trial and error. The neural network is an important one which determines how the policy can be saved. We used a neural network with three hidden layers of 16 nodes to do the job. This seemed like a manageable size neural network with enough depth to express somewhat complicated functions. Having a deeper network would make it more difficult to train and could potentially overfit the data.

3 Results

Here the final results will be presented. First, we present four random samples of the energy spectrum of the SK spin glass Hamiltonian that was used in this thesis. Each sample has random couplings and longitudinal field strengths respectively drawn from the normal distributions $N(0,1)$ and $N(0,0.3)$. Fig.3.1 shows how the energy landscape depends strongly on these two parameters, producing different behaviours of the eigenstates. The 3-spin systems have 2^3 eigenenergies representing the energies for each configuration. The bottom lines indicate the ground state while each subsequent higher line indicate the excited states as the Hamiltonian changes linearly from $H_{initial}$ to $H_{problem}$ from left to right. Following the lines one can read off the eigenenergies in each state of the anneal where the Hamiltonian is fully determined by the transverse field on the left and fully determined by the problem Hamiltonian on the right. It is clear that some samples have an easier energy landscape, as in Fig.3.1d, while other can become very difficult to anneal when the energy gap closes soon, as in Fig. 3.1a.

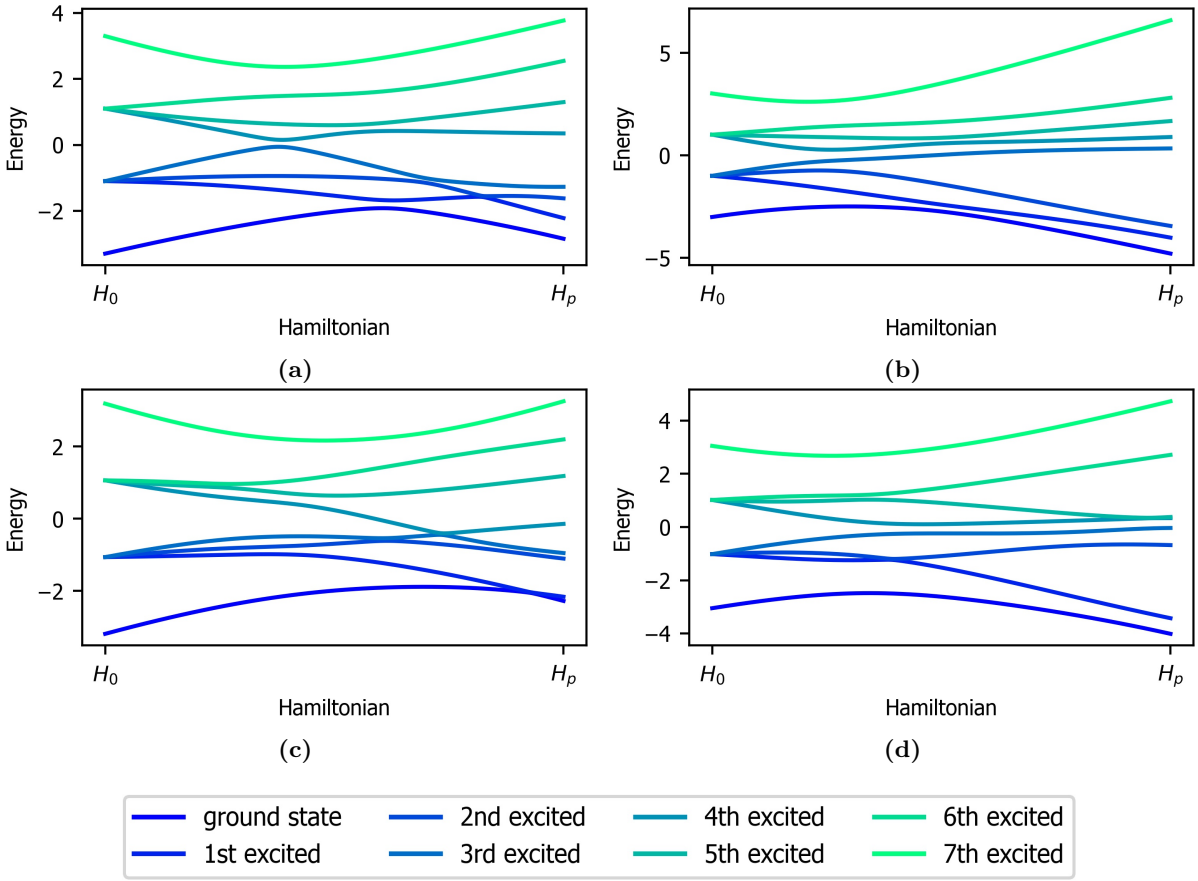

Fig. 3.1: Four random samples of energy spectra for 3 spin SK spin glasses with a transverse and longitudinal field where the Hamiltonian is linearly changed to the problem Hamiltonian over a time of 100 seconds.

The parameters used for training can be viewed in table 1 for future reference. The hyperparameters that were used in Ref.[6] were used as a basis for the tuning. These parameters were tuned by trial and error and should not be considered as the optimal hyperparameters.

Table 1: The hyperparameters used for training the PPO2 algorithm.

parameter	value
γ	0.99
n_steps	100
vf_coef	0.5
$learning\_rate$	10^{-5}
ent_coef	0.01
$cliprange$	0.1
$nminibathes$	10

Apart from these hyperparameters used by the PPO2 algorithm, the actual environment also needs some parameters. These are shown in table 2. Two different settings for J_{ij} were used, producing different results. These parameters depend highly on the computational power and precision that is desired. Choosing a lower clipping value for z can result in slower schedules and will therefore yield more accurate results. Also playing with the initial z-value and the step size will produce different results.

Table 2: The parameters needed for the environment.

parameter	value
N_spins	3
$z\_initial$	0.02, 0.001
z_step	0.00025, 0.00001
$max\_counter$	100
z_clip	0.00001
J	-1, $N(0,1)$

After a training run of $300,000$ steps the agent managed to learn the best annealing schedule given the reward scheme. With the reward scheme where after each full anneal the agent receives a reward of $\frac{1}{|\langle psi|H_P|psi\rangle - \epsilon_0|}$ the agent should learn to choose the slowest annealing schedule which optimises the precision and thus yield the highest rewards. This result was indeed achieved on the problem with equal coupling constants J. The progression during training can be seen in Fig. 3.2. The figure shows the accumulation of how often each available action is picked at each training step. At first the agent chooses completely random actions which is resembled by the overlapping of the lines in the first $50,000$ steps or so. After that an increasingly stronger convergence towards the 'slow' action is observed. This indicates that the policy tends to choose the slow action exclusively at the end of training, which is exactly what we desire. The policy is thus optimised with respect to the reward scheme.

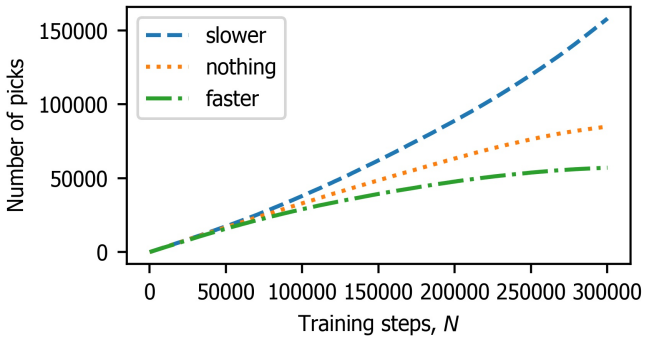

Fig. 3.2: Training data showing how often an action is chosen at each step with $J = -1$.

After the training phase the policy should be validated, this is done by running the policy on randomly initialised samples. The chosen schedule is the same for each sample, namely the slowest possible

schedule in the model that was defined by the parameters. As can be seen in Fig.3.4 the z value decreases linearly until it is clipped just above 0. The lower this value is the slower the anneal will be and therefore the more adiabatic, thus the lowest z value gives the highest precision.

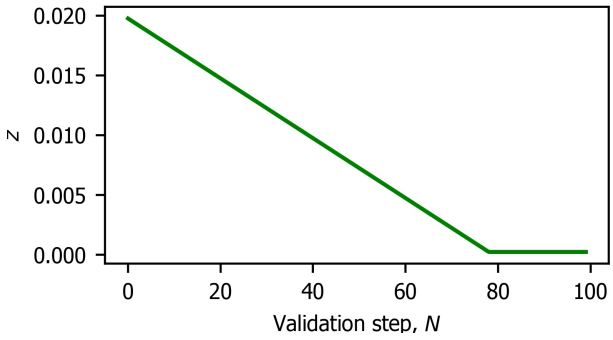

Fig. 3.3: Validation run showing decay of z value starting at $z = 0.02$ with $z\_step = 0.00025$ until clipped at $z = 0.00025$.

This z-schedule results in the annealing schedule $s(t) = t \cdot z + res$ as was explained in §2.6. Since lower z-values need longer to anneal the time steps for smaller z-values are bigger, following Eq.2.9. It can thus be seen how the decrease of the gradient z produces the annealing schedule in Fig.3.4. The slope of the schedule is decreased at each step and in order to finish at $t \cdot z + res = 1$ the time steps increase slightly at each step until the z-values are clipped and remain equal.

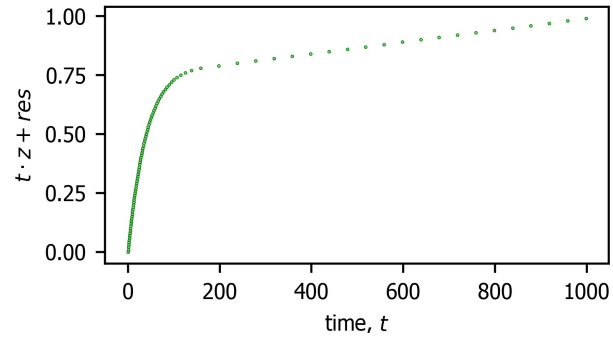

Fig. 3.4: Validation run showing the resulting annealing schedule $t \cdot z + res$ as a function of time again with $z\_intial = 0.02$ and $z\_step = 0.00025$.

The convergence of the energy to the ground state is perhaps the most important for validating the schedule and can be seen in Fig.3.5. The plot shows the dynamics of the annealing very nicely. In the inset the precision of the anneal can be read off on a logarithmic scale. In Fig.3.5a that particular instance yielded a precision of the order 10^{-4}, as well as Fig.3.5b. These results were derived using the parameter values from Table 2. This shows how different instances of the Ising spin glass have different energy landscapes that can will give rise to slightly different convergence with the same annealing schedule.

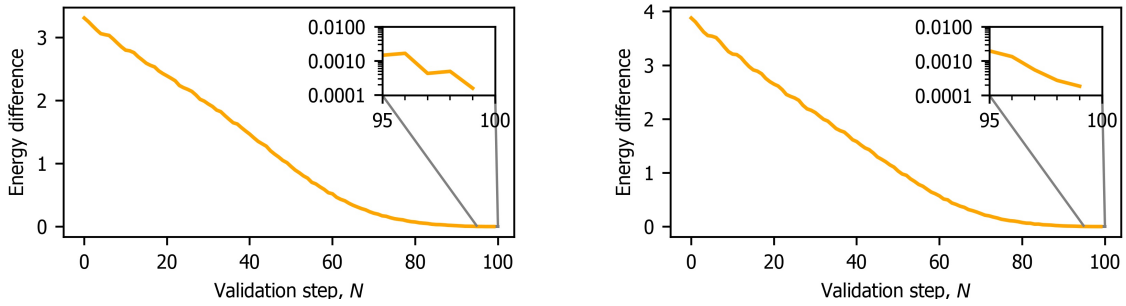

Fig. 3.5: Energy validation plots for two random instances of the environment thus yielding different Hamiltonians that produce slightly different convergences to the ground state.

Using lower starting values for z, a lower clipping value and a different step size the accuracy could be increased a lot. This is shown in Fig.3.6 where a smaller initial z-value along with a smaller step size results in increased accuracy to the order 10^{-7}. The annealing schedule that resulted from the different choice of *z_initial* and *z_step* is shown in Fig.3.6b.

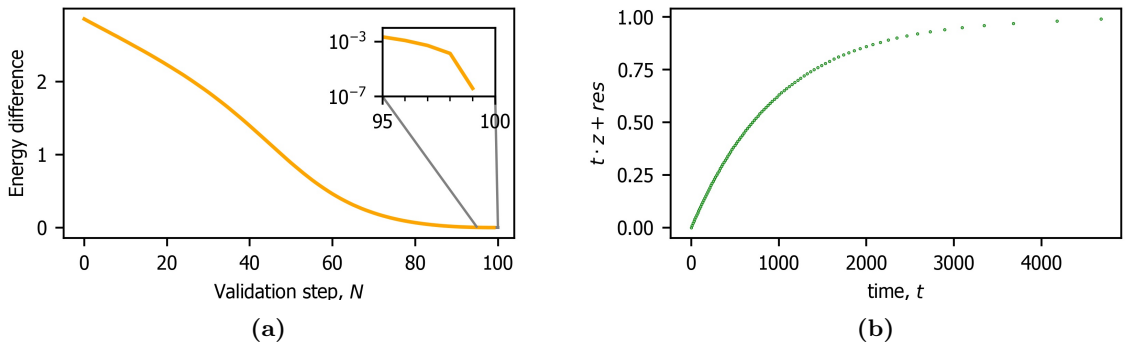

Fig. 3.6: Validation run with *z_initial* = 0.001 and *z_step* = 0.00001 to show increased accuracy. Showing the energy difference plot in plot (a) and the corresponding annealing schedule in plot (b).

The agent was also employed with the problem for a 'glassy' Hamiltonian where the coupling constants ,J_{ij}, are thus both positive and negative drawn from the normal distribution, $N(0, 1)$. This environment proved to be very hard to anneal on, possibly due to the more difficult energy landscape where the energy gap might close very soon in the anneal thus resulting in an excited state after the anneal if it was not slow enough. The training of an agent on such an environment thus did not work out, because the rewards for taking the slow action did not differ much from the one for taking the faster action as shown in Fig.3.7. The training as a result doesn't show a convergence to the slow action as we would like to see.

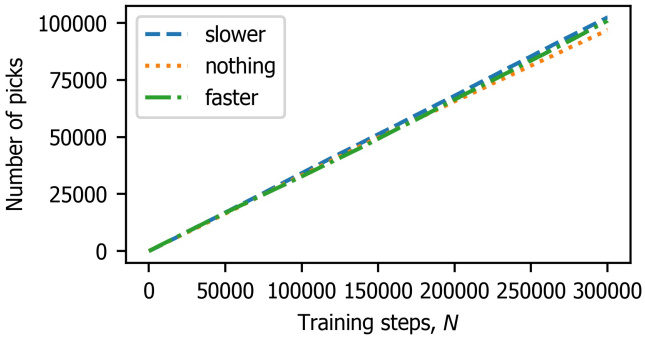

Fig. 3.7: Training plot on Hamiltonian with randomly picked coupling constants J_{ij}.

Since training the agent in this environment did not work out well, the agent that was trained on the previous 'easier' environment was used for annealing in the new glassy environment. It used the slowest annealing schedule as it had learnt on its own environment and this shows clearly why training the agent on the random J_{ij} environment did not work out well. Even when choosing the slowest actions the agent can not manage to anneal to the ground state and will consequently not receive any larger rewards for good actions than for bad actions. The energy difference during a randomly initialised anneal in a glassy environment is shown in Fig.3.8:

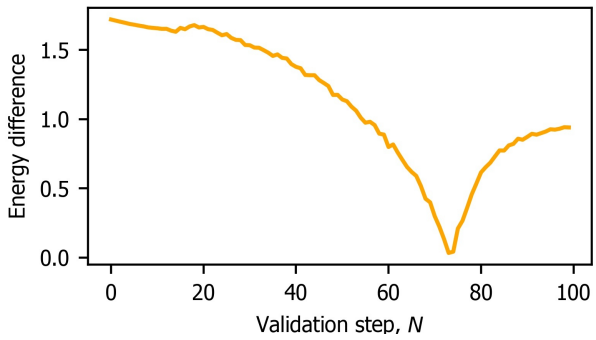

Fig. 3.8: Error difference plot using the same annealing schedule on a random instance of a glassy Ising Hamiltonian with the same parameters for the environment.

The environment might need a much slower annealing schedule in order to avoid jumping to higher energy levels. This could be achieved by decreasing the *z_initial*, the *z_clip* and the *z_step*. This was not done with z-values smaller than $z\_initial = 0.001$ and $z\_step = 0.00001$ since the computational time becomes too long in that instance. Satisfactory results on these Hamiltonians have thus not yet been found.

4 Discussion

The results show that reinforcement learning can be used to find accurate annealing schedules for quantum annealing on some Ising models. When training an agent on an environment with an Ising model with equal coupling constants and a reward given by Eq.2.10, it has been shown to learn the slowest annealing schedule. This corresponds to the desired outcome given the reward scheme. As of yet the reward scheme only favoured the most accurate solution and it therefore doesn't conclusively answer the question whether we can find a better annealing schedule by using RL than by guessing it ourselves. However, the results do show that the agent can learn to optimise its reward in difficult quantum annealing environment with very limited rewards and information. In this thesis a 100 steps were used for each anneal, this means that the agent could find $3^{100} \approx 10^{47}$ different schedules. Trying all possible paths would therefore be impossible in finite time. Hence, the RL agent uses the clever PPO algorithm to optimise the annealing schedule. It learns to converge to the slowest annealing schedule after a $300,000$ step training cycle as shown in Fig.3.2. The slowest schedule yields the most accurate results and is therefore the expected outcome from the reward schedule that was used. Upon validating the trained agent the annealing schedule shown in Fig.3.4, where the z value is chosen to decrease at each step, resulted in an energy difference of 10^{-4} on an annealing schedule that used approximately 1000 seconds. This residual energy could be further reduced by annealing slower as is suggested by the theory in §2.4 and backed up by the plot in Fig.3.6a where slower initial z was used and an overall annealing time of over 4000 seconds was used.

Finding a good balance between rewards for going fast and for yielding accurate results also has not worked out. Therefore only a reward for the accuracy was used and the agent should learn to find the most accurate and hence the slowest schedule. In a future pursuit of an optimal annealing schedule one should take the time to find a good balance in the reward scheme between rewards for a fast and an accurate schedule. Before such a balance can be found one should define clearly what the goal is for the speed and the accuracy. One could consider the linear schedule as a good benchmark for this. Some ideas could then be to add a bias towards the 'fast' action or to add a reward based on the final annealing time. Having a reward too biased towards fast actions will result in a scheme that doesn't produce accurate results, on the other hand too few bias would lead to either a linear or a slow schedule. Hence, one could consider adding a reward for having many different actions in the policy to stimulate exploration and avoid finding one of the extreme schedules where it is either too fast, linear or too slow. A generic reward scheme to be considered for future research is suggested in the appendix A.2.

The method thus works on the small 3-spin problem with equal couplings. But annealing on the more difficult spin glass where the couplings are all different has shown to be more difficult. As shown in Fig.3.8 the agent doesn't manage to anneal to the ground state even when given the slowest annealing schedule. Also on attempts with slower schedules it showed similar results. This problem results in an inability for the agent to learn the best annealing schedule when training on this environment, since rewards will not be better when good actions are taken as opposed to when bad actions are taken. This could be resolved by initialising the agent with a slower z-value, but this will dramatically increase the annealing time and is therefore not feasible on a regular computer.

This thesis has only considered 3-spin systems. To know whether it would work for more complicated spin glass systems with more spins is still not certain. However, given that the agent can learn to take the best action for 100 steps where the number of possible schedules is as great as 10^{47} gives some confidence that it will work on more complicated problems as well. Also, more spins do not necessarily mean that the energy gaps decrease, so it doesn't guarantee a more difficult annealing environment. Especially when the goal is to find the most accurate annealing schedule on a Ising model with equal coupling constants no problems should be expected. However, when trying to find the most optimal (read: fast and accurate) schedule the problem will most definitely become more difficult as more spins are considered, since the energy landscape will become more wobbly with more narrow energy gaps. Additionally the research by Irsigler and Grass on locating the minimal energy gap could help with difficult energy landscapes. They derived a method for revealing the minimal annealing gap which could then be used to optimize annealing schedules [43]. By using their method one could essentially go fast where the gap is large and slow down where the gap is minimal thus finding a faster and still accurate schedule.

In order to find more practically useful results the agent should produce a schedule that is generally faster at the start of the anneal and slows down as the energy gap closes [24]. Slowing down is namely only needed where the the energy gap is small and the probability of exciting the state is thus larger. Going fast where the energy gap is big and slowing down where it is small would produce a schedule that is both fast and accurate. Achieving this is possible by shaping the rewards in such a way that the agent has an incentive to go faster where possible but slow where needed. Using the sparse reward approach used in this thesis might still work, but is expected to need many more learning iterations. Therefore trying to give intermediate rewards should be considered. Also the observation that is used must be considered. For this paper only the energy difference between the expectation value of H_p in the current state and the ground state of H_p was considered along with the time. This doesn't provide the agent with any real knowledge of the energy landscape. Thus one could argue to use more detailed information for training the agent, since we essentially know everything about the system we are modelling. However, one must be careful not to give too much information as this would destroy the applicability of the algorithm to real life problems where the Hamiltonians have unknown characteristics. Yet, using more information to train the agent on certain classes of Hamiltonians could work to teach the agent the dynamics of those Hamiltonians. The trained agent could then be used on difficult instances and might still show the desired behaviour. Another intricate remark here is that in a real life experiment one would of course not have all the knowledge that we have here. Measurements would collapse the state and also the ground state of the problem Hamiltonian is unknown. Since we are training the agent we know what state it is in at each time step and we can extract the state from the model without collapsing the function. If we were to model a real life experiment we could use the setup for destructive measurements that was proposed by Mills et al. in Ref.[6]. They proposed to run multiple replicas of the system that have evolved through the same annealing schedule, but with different initialisations. Then each replica can be measured and used as data for the agent only once after which it has collapsed. This way the policy can be improved without collapsing the state. In their study Mills et al. showed that this works almost as well as with normal observations. Additionally the agent could be given the choice to perform a costly observation during training, it could then learn when observations are needed and when they could better not be performed. One could also argue that after training on a certain class of Hamiltonians the agent won't need observations to control the annealing schedule, since they would need a similar treatment.

Some drawbacks of the method are that it is very time consuming to train such an agent and only after training can the agent be evaluated. If it then performs sub-optimally some parameters can be tweaked and the algorithm can run again. The algorithm needs many iterations of improving the hyperparameters before it can work. Running times for the algorithm can also get excessively long based on the choice of step size and the cliprange. For smaller z-values the annealing time is longer and the computer needs to calculate the time-evolution over longer time periods. Thus, using a fast computer or running on a cluster is recommended for the best results. Running training sets with different hyperparameters in parallel can also be advised, given that there is enough computational overhead.

The hyperparameters that were used with the PPO algorithm were not optimally tuned. They were chosen via trial and error and eventually found to work acceptably well. There do exist hyperparameter tuning algorithms like *Optuna* and these could be used for a faster and better convergence of the PPO algorithm [44]. For the relative simplicity of the task to be learned by the agent in this thesis, that did not seem necessary. Nevertheless, one should keep in mind that the hyperparameters greatly determine the performance of an agent. When training an agent on more complicated problems one should consider using such tools to optimise the hyperparameters for better performance. Apart from the hyperparameters like the discount factor γ, entropy coefficient c_2 and clip range ϵ, which were all discussed in §2.5, the topology and size of the neural network used as the policy function approximator was also chosen heuristically. The chosen neural network should be easy to train on, but approximate functions well enough. Having too many layers could overfit the data which is not desired, but having too few would result in a loss of information. Therefore a more sophisticated tuning of the neural network could yield better results as well. Although the optimization of neural networks is often considered a 'dark art' there exist methods that automate the calibration like *HyperNOMAD* [45].

In a future experiment one should also consider changing the z-value percentage-wise. This would remove the need for clipping of the z-value to keep it above 0 and only clip it for faster computing.

The way of bounding the annealing schedule in the interval [0,1] while also guaranteeing monotonicity should also be reconsidered. Here I should note that considering a function $s(t)$ where the t itself is also bounded by the interval [0,1] seems like a nice representation comparable to the representation used by Morita [20]. He also suggested some annealing schedules that bound the residual energy error, these can serve as further inspiration for finding the optimal annealing schedule via RL.

5 Conclusion

No algorithm exists that can solve NP-hard problems in polynomial-time. Therefore heuristic methods have been developed that can yield fast approximate results. Simulated annealing presented such a heuristic method and later served as the inspiration for quantum annealing, a method where a transverse field Ising model is used to anneal. Quite some research has already been done in the field of quantum annealing and it is certainly not a new method, a collection of high-impact papers is documented in Ref.[5]. By using QA the hope is to pass through high barriers in the energy landscape with a higher probability via quantum tunneling or fluctuations, while classical annealing would struggle to pass over the barrier by thermal excitation [3]. One common problem with annealing in general is to find a well suited annealing schedule. This should be both fast and achieve a satisfactory precision in the final result. Some research has already aimed to solve this problem, but none (to the best of my knowledge) have used reinforcement learning in the process. Therefore, the aim of this thesis was to explore the possibility of using RL for finding an optimal quantum annealing schedule. It had already proven to work very well for simulated annealing by Mills et al. in Ref.[6]. Using RL for finding the optimal schedule which should be fast and accurate has not yet been accomplished. However, it has been shown that such an RL agent can learn in the QA environment and can find the most accurate schedule. More research is needed to verify whether RL can be used for finding a schedule that is also faster than a linear schedule while still remaining more accurate.

With high expectations we set out to formulate the RL environment and train an agent that can learn to find an optimal annealing schedule on a complex Sherrington-Kirkpatrick model. The method proved to be quite unforgiving towards bad hyperparameters and reward schemes. Therefore the formulation of the algorithm took a lot of iterations of tuning. The result is an agent that can learn to find the most accurate annealing schedule, hence the slowest one, where the problem Hamiltonian has one coupling constant J and is thus not 'glassy'. This is shown in Fig.3.2 where the convergence of the policy towards choosing the slowest action can be seen. This is quite remarkable considering the fact that each step the agent can choose three actions and thus produce a set of different policies of the order 10^{47}. Training the agent in an environment with the more difficult glassy Hamiltonian with random coupling constants has not worked out yet. The agent could not manage to converge its policy to the slowest one since the environment did not yield higher rewards with the slowest actions as can be seen in Fig.3.8.

The initial aim of finding the optimal annealing schedule was thus not accomplished. Finding the right balance between giving the agent rewards for speeding up while still remaining accurate still needs more iterations of trial and error. This balance is an intricate one where a speedup will by definition compromise the accuracy. So one should first determine a benchmark for the optimal schedule and determine how the balance should be.

The difficulty of using RL to optimise annealing schedules is mostly due to finding the right hyperparameters and reward schedule. These could be optimised using automated methods [44, 45]. Once this is all set it should be able to produce an annealing schedule for a multitude of Hamiltonian problems. Since this thesis only considered a 3-spin system is doesn't conclusively tell whether the method would work in bigger, more difficult, settings. When the goal is to find the optimal schedule there might be more and narrower minimal energy gaps thus making it more difficult to anneal. For these kinds of systems using a smaller initial z-value and smaller step sizes should help in the training process along with many more training steps. All of this should be considered in future experiments.

If one could manage to train an RL agent for finding good optimal annealing schedules this could help as a universal tool for finding fast approximate solutions to many NP-hard problems. The method seems promising so far but needs more validation especially on larger systems and on the glassy Ising model where the coupling constants are randomly distributed. The setup and results gathered in this thesis can be used as a basis for further expansion of the Hamiltonians considered. The results show the potential that RL has for finding good annealing schedules that can aid in solving optimisation problems.

References

[1] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2:1–14, 2014. ISSN 2296424X. doi: 10.3389/fphy.2014.00005.

[2] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Mathematical Sciences Series. W. H. Freeman, 1979. ISBN 9780716710448. URL https://books.google.nl/books?id=fjxGAQAAIAAJ.

[3] P. Ray, B. K. Chakrabarti, and Arunava Chakrabarti. Sherrington-Kirkpatrick model in a transverse field: Absence of replica symmetry breaking due to quantum fluctuations. *Physical Review B*, 39(16):11828–11832, 6 1989. ISSN 01631829. doi: 10.1103/PhysRevB.39.11828. URL https://journals.aps.org/prb/abstract/10.1103/PhysRevB.39.11828.

[4] David J. Griffiths and Darrell F. Schroeter. *Introduction to Quantum Mechanics*. Cambridge University Press, 8 2018. ISBN 9781316995433. doi: 10.1017/9781316995433. URL https://www.cambridge.org/core/product/identifier/9781316995433/type/book.

[5] Asim Ghosh and Sudip Mukherjee. Quantum Annealing and Computation: A Brief Documentary Note *. *SCIENCE AND CULTURE (Indian Science News Association)*, 79:485–500, 2013. URL http://blogs.scientificamerican.com/guest-blog/2013/05/17/is-it-quantum-computing-or-.

[6] Kyle Mills, Pooya Ronagh, and Isaac Tamblyn. Controlled Online Optimization Learning (COOL): Finding the ground state of spin Hamiltonians with reinforcement learning. *arXiv*, pages 1–10, 2020. ISSN 23318422. doi: 10.1038/s42256-020-0226-x.

[7] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. A Correction. *Proceedings of the London Mathematical Society*, s2-43(1):544–546, 1 1938. ISSN 1460-244X. doi: 10.1112/PLMS/S2-43.6.544. URL https://londmathsoc.onlinelibrary.wiley.com/doi/full/10.1112/plms/s2-43.6.544.

[8] The Ising Model. URL https://stanford.edu/~jeffjar/statmech/intro4.html.

[9] F. Barahona. On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241–3253, 10 1982. ISSN 13616447. doi: 10.1088/0305-4470/15/10/028. URL https://iopscience.iop.org/article/10.1088/0305-4470/15/10/028.

[10] Barry A Cipra. The Ising Model Is NP-Complete. Technical Report 6.

[11] David Sherrington and Scott Kirkpatrick. Solvable Model of a Spin-Glass. *Physical Review Letters*, 35(26):1792, 12 1975. doi: 10.1103/PhysRevLett.35.1792. URL https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.35.1792.

[12] A Rajak and B K Chakrabarti. Quantum Annealing search of Ising spin glass ground state(s) with tunable transverse & longitudinal fields. *Indian Journal of Physics*, 88(9):951–955, 5 2014. doi: 10.1007/s12648-014-0483-9. URL https://arxiv.org/abs/1405.3905v1.

[13] John D Verhoeven. *Fundamentals of physical metallurgy*. Wiley, 1975.

[14] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 5 1983. ISSN 00368075. doi: 10.1126/science.220.4598.671. URL http://science.sciencemag.org/.

[15] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 12 1953. ISSN 00219606. doi: 10.1063/1.1699114. URL https://aip.scitation.org/doi/abs/10.1063/1.1699114.

[16] W. K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):97, 4 1970. ISSN 00063444. doi: 10.2307/2334940.

[17] Stuart Geman and Donald Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6 (6):721–741, 1984. ISSN 01628828. doi: 10.1109/TPAMI.1984.4767596.

[18] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum Annealing in the Transverse Ising Model. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, 58 (5):5355–5363, 4 1998. doi: 10.1103/PhysRevE.58.5355. URL http://arxiv.org/abs/cond-mat/9804280.

[19] Clarence Ze. Non-adiabatic crossing of energy levels. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 137(833):696–702, 9 1932. ISSN 0950-1207. doi: 10.1098/rspa.1932.0165. URL https://royalsocietypublishing.org/.

[20] Satoshi Morita. Faster annealing schedules for quantum annealing. *Journal of the Physical Society of Japan*, 76(10):1–4, 2007. ISSN 00319015. doi: 10.1143/JPSJ.76.104001.

[21] Satoshi Morita and Hidetoshi Nishimori. Mathematical Foundation of Quantum Annealing. *Journal of Mathematical Physics*, 49(12), 6 2008. doi: 10.1063/1.2995837. URL http://arxiv.org/abs/0806.1859http://dx.doi.org/10.1063/1.2995837.

[22] Sudip Mukherjee and Bikas K. Chakrabarti. Multivariable Optimization: Quantum Annealing & Computation. *European Physical Journal: Special Topics*, 224(1):17–24, 8 2014. doi: 10.1140/epjst/e2015-02339-y. URL http://arxiv.org/abs/1408.3262http://dx.doi.org/10.1140/epjst/e2015-02339-y.

[23] J. Brooke, D. Bitko, T. F., Rosenbaum, and G. Aeppli. Quantum Annealing of a Disordered Magnet. *Science*, 284(5415):779–781, 4 1999. ISSN 0036-8075. doi: 10.1126/SCIENCE.284.5415.779. URL https://science.sciencemag.org/content/284/5415/779.

[24] Daniel Herr, Ethan Brown, Bettina Heim, Mario Könz, Guglielmo Mazzola, and Matthias Troyer. Optimizing schedules for quantum annealing, 2017. ISSN 23318422.

[25] Helmut G. Katzgraber, Firas Hamze, and Ruben S. Andrist. Glassy chimeras could be blind to quantum speedup: Designing better benchmarks for quantum annealing machines. *Physical Review X*, 4(2):021008, 4 2014. ISSN 21603308. doi: 10.1103/PhysRevX.4.021008. URL https://journals.aps.org/prx/abstract/10.1103/PhysRevX.4.021008.

[26] Troels F. Rønnow, Zhihui Wang, Joshua Job, Sergio Boixo, Sergei V. Isakov, David Wecker, John M. Martinis, Daniel A. Lidar, and Matthias Troyer. Defining and detecting quantum speedup. *Science*, 345(6195):420–424, 7 2014. ISSN 10959203. doi: 10.1126/science.1252319. URL http://science.sciencemag.org/.

[27] Bettina Heim, Troels F. Rønnow, Sergei V. Isakov, and Matthias Troyer. Quantum versus classical annealing of Ising spin glasses. *Science*, 348(6231):215–217, 4 2015. ISSN 10959203. doi: 10.1126/science.aaa4170. URL http://arxiv.org/abs/1401.7320.

[28] Lauren Pusey-Nazzaro and Prasanna Date. Adiabatic Quantum Optimization Fails to Solve the Knapsack Problem. 8 2020. URL http://arxiv.org/abs/2008.07456.

[29] Rolando D. Somma, Daniel Nagaj, and Mária Kieferová. Quantum speedup by quantum annealing. *Physical Review Letters*, 109(5):050501, 7 2012. ISSN 00319007. doi: 10.1103/PhysRevLett.109.050501. URL https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.109.050501.

[30] A. B. Finnila, M. A. Gomez, C. Sebenik, C. Stenson, and J. D. Doll. Quantum Annealing: A New Method for Minimizing Multidimensional Functions. *Chemical Physics Letters*, 219(5-6):343–348, 4 1994. doi: 10.1016/0009-2614(94)00117-0. URL http://arxiv.org/abs/chem-ph/9404003.

[31] Yu-Qin Chen, Yu Chen, Chee-Kong Lee, Shengyu Zhang, and Chang-Yu Hsieh. Optimizing Quantum Annealing Schedules: From Monte Carlo Tree Search to QuantumZero. *arXiv*, 4 2020. URL http://arxiv.org/abs/2004.02836.

[32] Oscar Galindo and Vladik Kreinovich. What Is the Optimal Annealing Schedule in Quantum Annealing. *2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020*, pages 963–967, 2020. doi: 10.1109/SSCI47803.2020.9308407.

[33] Beining Han, Zhizhou Ren, Zuofan Wu, Yuan Zhou, and Jian Peng. Off-Policy Reinforcement Learning with Delayed Rewards. 6 2021. URL http://arxiv.org/abs/2106.11854.

[34] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction Second edition, in progress*. MIT Press Ltd, 2 edition, 2015. URL https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf.

[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. 7 2017. URL http://arxiv.org/abs/1707.06347.

[36] S. Sra, S. Nowozin, and S.J. Wright. *Optimization for Machine Learning*. Neural information processing series. MIT Press, 2012. ISBN 9780262016469. URL https://books.google.nl/books?id=JPQx7s2L1A8C.

[37] Shiyu Liang and R. Srikant. Why Deep Neural Networks for Function Approximation? 10 2016. URL https://arxiv.org/abs/1610.04161v2.

[38] Titus Neupert, Mark H Fischer, Eliska Greplova, Kenny Choo, and Michael Denner. Introduction to Machine Learning for the Sciences. 2 2021. URL http://arxiv.org/abs/2102.04883.

[39] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource Management with Deep Reinforcement Learning. 2016. doi: 10.1145/3005745.3005750. URL http://dx.doi.org/10.1145/3005745.3005750.

[40] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 12 2018. ISSN 0036-8075. doi: 10.1126/SCIENCE.AAR6404. URL https://science.sciencemag.org/content/362/6419/1140.

[41] J. R. Johansson, P. D. Nation, and Franco Nori. QuTiP 2: A Python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 184(4):1234–1240, 4 2013. doi: 10.1016/J.CPC.2012.11.019.

[42] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.

[43] Bernhard Irsigler and Tobias Grass. The quantum annealing gap and dynamical quantum phase transitions in complex optimization problems. 6 2021. URL http://arxiv.org/abs/2106.08101.

[44] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2623–2631, 7 2019. URL http://arxiv.org/abs/1907.10902.

[45] Dounia Lakhmiri, Sébastien Le Digabel, and Christophe Tribes. HyperNOMAD: Hyperparameter optimization of deep neural networks using mesh adaptive direct search. 7 2019. URL https://arxiv.org/abs/1907.01698v1.

A Appendix

A.1 Code

Here the pseudocode for defining the environment used in this thesis is presented. This could serve as an initial framework for further research. When considering to train an agent to be both fast and accurate one should contemplate most about the reward scheme, §A.2 could help with this.

Pseudocode 1: Defining the Quantum Annealing environment

Result: Environment

```
class QA_env(gym.Env):
    init(self, N_spins, z_initial, z_step, max_counter, J):
        define Hamiltonians;
        define initial state;
        initialise parameters (z, tlist, counter);
        define action space (slow, nothing, fast);
        define observation space (|⟨ψ| H_p |ψ⟩ − ε_0|, tlist[−1]);
    return
    step(self, action):
        if action == fast then
            z+ = z_step;
        else if action == nothing then
            z = z;
        else if action == slow then
            z+ = −z_step;
        add time step;
        time-evolve Hamiltonian;
        observe energy difference and time;
        if final step then
            reward = 1/|⟨ψ|H_p|ψ⟩−ε_0|;
            done = True;
        else
            counter+ = 1;
            done = False;
        end
    return obs, reward, done
    reset(self):
        re-initialise Hamiltonian;
        re-initialise initial state;
        reset parameters;
    return obs
```

A.2 Reward scheme suggestion

In this section a reward scheme for optimising an annealing schedule for both speed and accuracy will be suggested. Fig.A.1 shows an idea of what one could try in the reward scheme. Adding to the sparse reward scheme that was used in this thesis one could consider giving rewards at each step or at a subset of intermediate steps to increase the learning speed. The rewards based on the annealing time could also be in the form of a penalty where a long time would result in a bigger penalty. The annealing time could also be compared to that of a linear schedule. Additionally, an extra reward could be given both after the anneal or during the anneal for taking different actions. This could counteract the tendency of the agent to find 'dull' schedules that work to find a minimum, but do not show the desired behaviour.

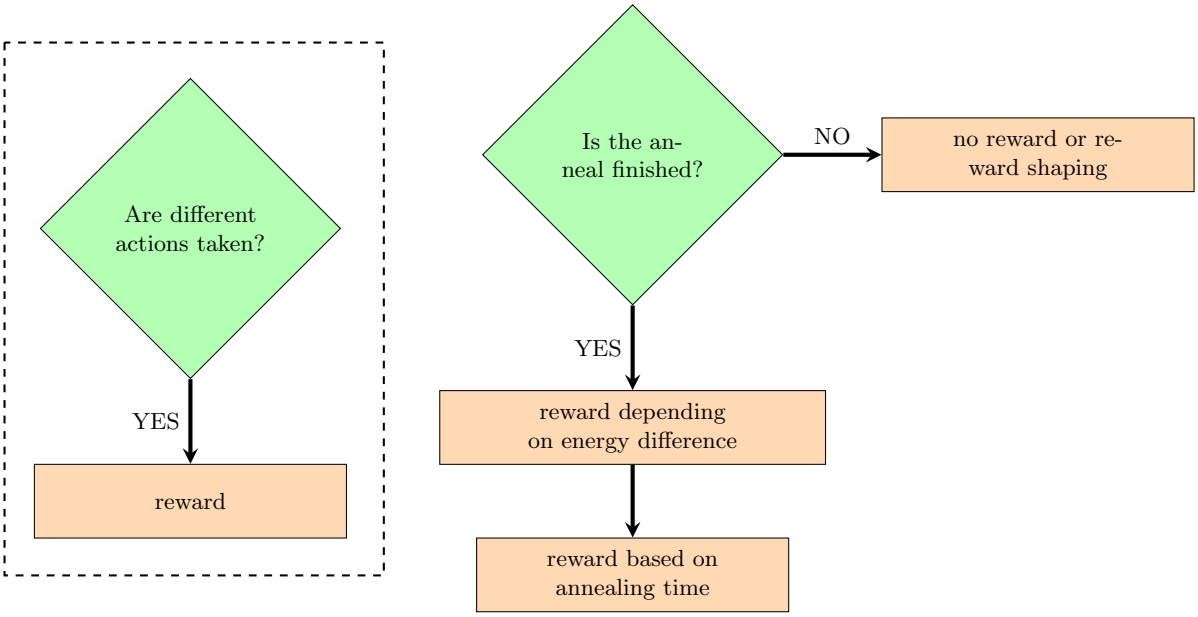

Fig. A.1: Flowchart of suggested reward scheme for speeding up the annealing schedule.