UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE

Master's Thesis

Learning Quantum Optimization Games

Monte Carlo Tree Search as an alternative for quantum optimization

Andoni Agirre Arabolaza

Advisors: Evert P.L. van Nieuwenburg, Matteo M. Wauters and Michele Burrello

Submitted: August 8, 2023

Preamble

Abstract

Hard combinatorial optimization problems have an abundance of real-life applications, but quickly become intractable and too costly to solve. By encoding them into ground states of spin-1/2 Hamiltonians, such problems can be formulated as ground state preparation problems, which can be tackled by quantum algorithms. A lot of effort is currently being made to determine if the limited quantum devices of today could provide better and faster platforms to solve these hard problems.

In this thesis we work with 3-satisfiability, the prime example of such hard problems, and try to solve it with algorithms based on two of the main near-term quantum computational paradigms for quantum optimization: Adiabatic Quantum Computation and Variational Quantum Algorithms. In view of the difficulties of quantum optimization, such as having to navigate exceedingly complicated cost landscapes, we employ the Monte Carlo Tree Search algorithm, a very peculiar way of approaching quantum optimization from the viewpoint of board game artificial intelligence.

We first implement and study the performance of our own version of the Monte Carlo Tree Search algorithm, both within a conventional game domain and a simple quantum system with two spins. We then successfully use our algorithm as a tool to optimize annealing schedules to solve hard instances of 3-satisfiability, as well as presenting and testing different methods of optimizing the parameters of the Quantum Approximate Optimization Algorithm with it. We also devise a number of concrete modifications to the standard version of the algorithm, which make it an even more compelling tool for quantum optimization.

Contents

Chapter 1

Introduction

This introductory chapter has two sections. In section 1.1 we attempt to give a broad overview of the problems we will be tackling and the context in which they arise. In the outlook section 1.2 we introduce our approach to the problem and explain the layout of the thesis.

Throughout this work we will set $\hbar = 1$.

1.1 Background

The marvelous promises of quantum computation (QC) have drawn the attention of many since its original proposal in the 1980s. By integrating quantum mechanics into their very hardware, quantum devices hold the potential of cunningly simulating big quantum systems —notoriously hard to do in classical computers due to the exponential increase of the size of the Hilbert space—, as well as providing rampant speedups over classical algorithms — such as factorizing numbers [1], solving linear systems of equations [2] and searching unstructured data sets [3].

Most of these promises, however, may not be properly realized today. The required amount of qubits to implement these quantum algorithms is way beyond current capabilities, not to mention the lack of reliability of those qubits, as they need to be shielded from decoherence-inducing external noise while being reliably controlled. Similarly, although there exist ways that combat decoherence and other sources of error through the theory of Quantum Error Correction (QEC) [4], the required number of qubits as overhead to perform QEC is too large in practice.

Since fault tolerant QC seems not to be a thing of the near future, a great deal of effort is currently being made to create and develop algorithms for alternative computational paradigms that make use of existing quantum devices (often in conjunction with powerful classical computers) and achieve quantum advantage with them. Although these Noisy Intermediate-Scale Quantum (NISQ) devices, as they are known, are severely limited —from lower than

ideal qubit numbers to shallow circuit depths—, said limitations are not expected to necessarily impede the development of near-term algorithms that take advantage of them. The creation of usable and practical NISQ era algorithms is, in fact, a very promising field of research spanning a vast number of applications, from condensed matter physics and quantum chemistry to machine learning and classical optimization [5]. Despite its many challenges, it has seen some limited success and there have been experiments that have obtained exponential speedups over classical devices for certain tasks, albeit very contrived and inane ones [6, 7]. Any quantum advantage for practical applications has yet to be experimentally demonstrated. There are various ways one can try to create algorithms for NISQ era devices. These algorithms are divided into two main types: gate-based and analog.

The gate-based or digital kind fall within the framework of Variational Quantum Algorithms or VQAs. These consist of a combined effort of both a classical subroutine and a quantum one, and the classically hard part of the computation is delegated to the latter. Their objective is to solve an optimization problem where a cost function $C(\boldsymbol{\theta})$ needs to be minimized with respect to a number of variational parameters $\boldsymbol{\theta} = \{\theta_1, ..., \theta_L\}$. VQAs encompass a very broad class of algorithms, which can have very different applications [8]. Nevertheless, all VQAs are equipped with the same modular parts: a cost function $C(\boldsymbol{\theta})$ to be minimized, a circuit of universal quantum gates, known as the unitary Ansatz $U(\boldsymbol{\theta})$, which gives the $\boldsymbol{\theta}$ dependence of the cost function[1], and a classical optimizer that finds the set of parameters $\boldsymbol{\theta}^*$ associated with the global minimum of the cost function

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} C(\boldsymbol{\theta}), \tag{1.1}$$

as well as a quantum computer that computes the cost function as required by the classical optimizer along the minimization process. Usually the cost function will correspond to an expectation value of some observable, very often[2] a Hamiltonian H, and its $\boldsymbol{\theta}$-dependence will enter via transforming it with the unitary Ansatz, $C(\boldsymbol{\theta}) = \langle U^\dagger(\boldsymbol{\theta}) H U(\boldsymbol{\theta}) \rangle$. In those cases the minimization problem becomes a problem of finding the ground state of H. The general structure of VQAs can be seen in Fig 1.1.

The source of the possible speedup for a VQA with respect to a purely classical optimizer is the extremely significant role of the quantum computer: while the quantum computer works with the exponentially large, $2^{N_{\text{qubits}}}$-dimensional Hilbert space, the classical machine is solely concerned with the

[1] Because of the limitations of near-term quantum devices, the circuits cannot be too deep, and the gates that constitute the circuit can act at most on a few qubits only.

[2] A less restrictive cost function could be constructed as the linear combination of expectation values of many observables $\{O_k\}$, $C(\boldsymbol{\theta}) = \sum_k f_k \left(\text{Tr}\left[O_k U(\boldsymbol{\theta}) \rho_k U^\dagger(\boldsymbol{\theta})\right]\right)$, for some initial projectors $\{\rho_k\}$, although it is not even restricted to that. With loss of generality, here it is assumed that one minimizes expectation values of Hamiltonians, which is very common for VQAs.

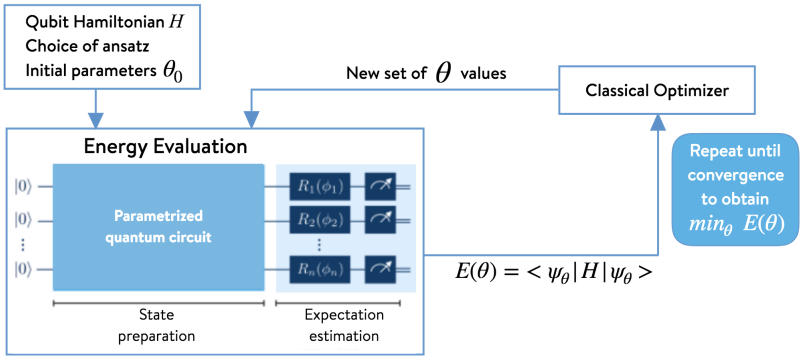


Figure 1.1: Structure of a Variational Quantum Algorithm (VQA). The classical optimizer optimizes the parameters of the quantum circuit. Figure adapted from [9].

L-dimensional space of the variational parameters. This allows the classical optimizer to harness the power of the quantum device and opens the door to solving problems thought to be impossible to solve by purely classical means. This, of course, requires that the unitary Ansatz be chosen smartly, such that the $\boldsymbol{\theta}$ cover the most relevant regions of the Hilbert space. The two most well known examples[3] of VQAs are the Variational Quantum Eigensolver (VQE) [10] and the Quantum Approximate Optimization Algorithm (QAOA) [11].

In stark contrast to VQAs, the other kind of NISQ era algorithms does away with gate-based quantum circuits altogether. They are instead based on analog devices that aim to simulate special-purpose systems. Chief among the analog quantum algorithms is Adiabatic Quantum Computation (AQC) or[4] Quantum Annealing (QA) [12]. QA was originally inspired by the classical optimization algorithm known as Simulated Annealing (SA) [13], employed for locating the global minimum of energy landscapes with many local minima. SA simulates thermal excitations by introducing a temperature parameter T which allows the system to visit energetically less favorable states with probability $e^{-\frac{\Delta E}{k_B T}}$, where the energy difference is ΔE, meaning that at higher temperatures the system is allowed to perform more thermal jumps over energy barriers to escape local minima. In analogy with annealing processes in material science[5], the SA process starts at a very high temperature, and is then slowly cooled down until the system gets frozen at a minimum (hopefully the global one) at zero temperature.

[3] Both of these examples use the expectation value of a Hamiltonian as their cost function.

[4] In this work both terms will mostly be used as synonyms, although some texts consider AQC to be the theoretical framework on which QA is based.

[5] Annealing denotes a heat treatment for materials which consists of heating them to a suitable temperature followed by cooling them at a slow rate, primarily used to soften metallic materials [14].

QA is the quantum version of SA, used to find the ground state of a complicated target Hamiltonian H_{target}. Instead of using a temperature, it works by introducing a driving Hamiltonian[6] H_{driving} with an easy to construct ground state, multiplied by a time-dependent interaction strength $\Gamma(t)$,

$$H(t) = H_{\text{target}} + \Gamma(t)H_{\text{driving}}. \tag{1.2}$$

The new term fulfills the same role as the temperature in SA; the interaction strength $\Gamma(t)$ should initially be very large, such that the ground state of the system almost coincides with that of H_{driving}, and slowly die down until only the target Hamiltonian is left. The system is initialized in the ground state of the driving Hamiltonian, and by letting it evolve according to Schrödinger's equation, the energy barriers that emerge in the dynamic energy landscape can be crossed by quantum tunneling through them instead of jumping over them via thermal excitations. These differences are shown schematically in Fig 1.2.

A very nice advantage of QA is that one does not have to rely on the hope that the quantum effects that are present during the evolution of the system, such as being able to tunnel through barriers, might make the QA algorithm reach the required energy minimum: there is the theoretical guarantee that —provided some conditions are met— the global minimum will be reached via the adiabatic theorem. The adiabatic theorem states that, as long as the driving Hamiltonian is turned off slowly enough, the system will stay in its instantaneous ground state throughout the evolution; finally reaching the ground state of H_{target} after the total evolution time τ elapses. The slow enough condition is most commonly stated as

$$\tau \gg \frac{\xi}{\Delta^2}, \tag{1.3}$$

where ξ is the absolute value of the matrix element of the time derivative of the Hamiltonian (1.2) between the ground state and the first excited state, and Δ the minimum energy gap along the evolution path. This might not be a feasible τ for some Hamiltonians, but it is very convenient that the theorem is there as validation for QA[7].

It has been proven that there are particularly interesting and difficult types of problems that QA is able to solve much more efficiently than SA [17, 18]. Furthermore, QA has the advantage over SA of being able to run in quantum hardware. In fact, current analog quantum devices can have a much higher number of qubits than gate-based ones since not all of them need to be precisely controlled, and it has also been shown to be more resistant to noise [19]. Their disadvantage is that, in practice, they are not universal devices as they

[6] Any driving Hamiltonian H_{driving} can be chosen as long as $[H_{\text{target}}, H_{\text{driving}}] \neq 0$.
[7] Technically, an equivalent theorem can also be found for SA [16].

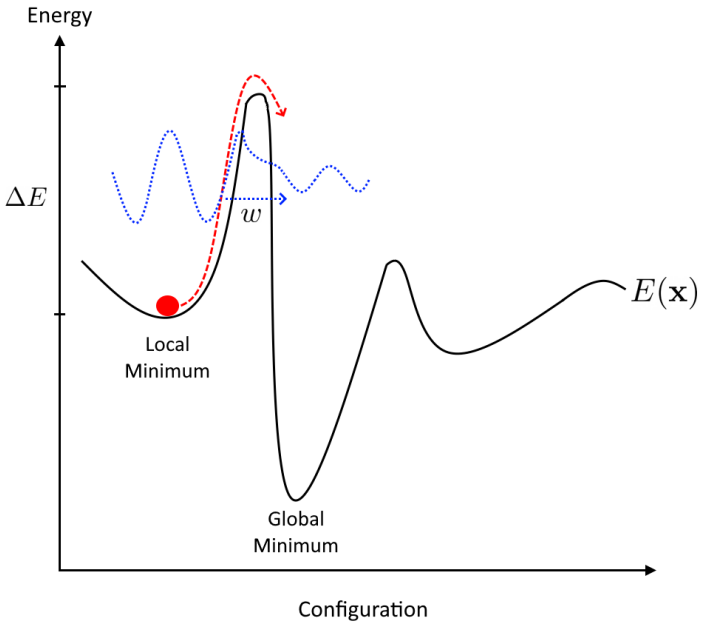


Figure 1.2: Schematic comparison of mechanisms to overcome barriers in an energy landscape $E(\mathbf{x})$. Simulated Annealing (red dashed line) and Quantum Annealing (Blue dotted line). The thermal jumps in SA occur with a probability $e^{-\frac{\Delta E}{k_B T}}$, whereas the probability for a tunneling event in QA can be crudely approximated as $e^{-\frac{w\sqrt{\Delta E}}{\Gamma}}$, where w is the width of the barrier. Because the height of the barrier enters the approximate tunneling probability with a square root, it was predicted that tunneling should work better when the energy landscape contains thin but high barriers, while thermal jumps might get stuck if the barriers are too high. This has been seen in [15], among others.

can only be used to tackle a very particular type of task.

To cut a long story short, there are two main computational frameworks to create NISQ era algorithms, and both can be used to find the ground states of quantum Hamiltonians. It should come as no surprise that a quantum algorithm able to find ground states for complicated systems has a huge number of applications in condensed matter physics and quantum chemistry [5]. But there is also another application of immense interest for ground-state-finding near-term quantum algorithms, although it is not quantum mechanical in nature: using them to solve difficult classical combinatorial optimization problems [8, 5], specifically so called NP-hard problems. Before exploring this deeper, it makes sense to make a brief detour to introduce a few concepts related to combinatorial optimization problems.

A combinatorial optimization problem deals with a classical cost function with a discrete set of possible solutions. Each instance of a problem has an associated parameter n that gives its size, this can be thought of as the number of discrete variables that enter the cost function. Such problems can be classified by difficulty into what are known as complexity classes [20], which give the dependence of the computational cost of solving the problem as a function of n. A problem is said to be in complexity class P if there exists an algorithm that can solve it in a time polynomial in n, such that the cost of solving the problem scales as $\mathcal{O}(n^k), k \geq 0$ in the worst case. This is considered a class of (mostly) efficiently solvable problems; P is made up of, in general, tractable, but not very interesting[8] problems. Conversely, a problem is said to be in NP if a proposed solution can be verified in polynomial time. From this definition it follows that all problems in P are contained in NP, $\mathsf{P} \subseteq \mathsf{NP}$, since if a problem can be solved in polynomial time, then a proposed solution can always be verified in polynomial time (by just solving the problem). The interesting problems in NP are those for which every existing algorithm that solves them scales worse than polynomially. An extremely famous unanswered question in mathematics asks whether polynomial algorithms exist for solving all the problems of NP, i.e., if $\mathsf{P} = \mathsf{NP}$. To study this, instead of checking if there exist polynomial time algorithms for each of the infinitely many problems in NP, there is another class of problems, NP-hard problems, which reduce[9] to every other problem in NP. Finding a polynomial time algorithm that solves any one NP-hard problem would be sufficient to prove that $\mathsf{P} = \mathsf{NP}$. If an NP-hard problem is also in NP, that problem is said to be NP-complete. This distinction can also be formulated as follows: NP-hard problems are at least as hard as all other problems in NP, whereas NP-complete problems are the hardest problems within NP. This can be nicely visualized in Fig 1.3.

NP-complete problems are most often formulated as "yes or no" questions or decision problems. Some examples include Decision Graph Coloring (can you color all the vertices of this graph with these many colors such that no adjacent vertices have the same color?) and Boolean Satisfiability (is there an assignment of bits that satisfies this boolean expression?). Instead, the harder NP-hard problems look like optimization problems. And there are often NP-hard variations of NP-complete problems and vice versa. For the two previous examples, the NP-hard versions are Optimization Graph Coloring (what is the least amount of colors one needs to color this graph like in Decision Graph Coloring?) or Max-Satisfiability (what is the assignment of bits that comes closest to satisfying this boolean expression?). Far from being mathematicians' delusional creations, NP-hard and NP-complete problems have a monumental number of tangible uses, as many real-life problems happen to be in those

[8] The word "interesting" should be understood as interesting within the context of computational complexity.

[9] This means a solution for an NP-hard problem implies a solution to all NP problems with the same computational cost. Particularly, a solution in polynomial time.

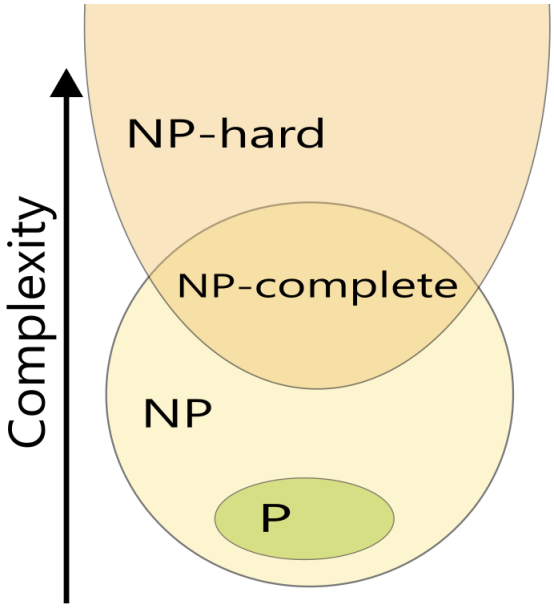


Figure 1.3: Diagramatic representation of the mentioned complexity classes.

complexity classes [20].

The interest of solving hard problems in quantum devices really took of after the discovery of the celebrated Shor's algorithm [1]. Integer factorization, an NP problem, was proven to be solvable in polynomial time in a quantum computer[10,11], giving an exponential speedup over the best existing classical factorization algorithms. For reasons that will become more apparent in subsequent chapters, it is unlikely that NISQ era algorithms will give exponential speedups for NP-complete or harder problems. However, the true potential of near-term devices is still unknown.

But how exactly can one solve NP-hard problems in quantum computers? The answer to this question stems from one simple fact: finding the ground state of a classical spin-glass Hamiltonian is an NP-hard problem (or NP-complete if it is formulated as a yes or no question), and so is the quantum version[12]. This has the immediate extremely convenient consequence that any NP-complete problem and many NP-hard ones can be formulated

[10]This has no dramatic effect on the current understanding of complexity classes, unless someone could prove integer factorization to be NP-complete, from which P = NP would follow.

[11]This lead to the definition of other complexity classes for quantum algorithms. Specifically, integer factorization is in Bounded Quantum Polynomial (BQP) time. The problems in BQP can be solved by a quantum device in polynomial time, with a success probability at least as high as 2/3 [20, 5].

[12]Technically, the quantum version is what is called QMA-complete [21], but this is not relevant to the present discussion.

as ground state preparation problems for problem-specific target Hamiltonians. Naturally, Variational Quantum Algorithms or Quantum Annealing can be employed to attempt to solve the quantum version of just such problems. Indeed, a huge number of Hamiltonian representations that have the solution to an NP-hard problem encoded in their ground state have been created [22], making attempting to solve them with NISQ era devices possible.

The classical Hamiltonians that encode optimization problems are made out of classical spins or bits s_i which can only take the value 1 or -1. For quantum optimization, the target Hamiltonians are instead built with σ_i^z Pauli operators. The hope is that by transitioning into quantum spins, $s_i \rightarrow \sigma_i^z$, the new degrees of freedom of the system might give rise to new faster and/or better ways to attempt to solve these problems.

One might ask how good NISQ era algorithms that were mentioned earlier are at solving these kinds of combinatorial optimization problems. Well, in the case of gate-based algorithms, there is one very powerful and promising VQA for this, the Quantum Approximate Optimization Algorithm (QAOA). It has successfully obtained limited polynomial speedups for a few NP-hard problems with respect to the best existing classical algorithms[13] [23, 8]. The main problem of VQAs, and the reason that no general quantum advantage over classical algorithm has been discovered yet, is that the energy landscapes which need to be navigated for these problems are exceedingly complicated with many local minima and barren plateaus [24]. Most classical optimizers for VQAs hopelessly struggle to navigate such landscapes, especially since the cost function evaluations from the quantum device have significant sources of errors such as noise. This makes the job of gradient-based optimizers particularly difficult.

In the case of the analog Quantum Annealing machines, solving hard problems does not get any simpler. Issues emerge because, as the size of the combinatorial optimization problem grows, the energy gap between the ground state and the excited states along the evolution path becomes narrower. For NP-hard problems this happens as a first order phase transition [25], which means the gap closing happens exponentially, $\Delta \sim \mathcal{O}(e^{-kn})$. Seeing this should trigger all kinds of alarm bells, because a look at the adiabatic condition (1.3) reveals that if Δ decreases at such rates the required evolution time will very quickly grow beyond any remotely reasonable time. However, it turns out that the particular way in which the magnetic field $\Gamma(t)$ is turned off during QA, usually making it slow down during the most delicate parts of the evolution, can greatly reduce the required annealing time [26]. Finding the optimal way to do this, however, is far from easy: the location of the gap minimum and the appearance of the energy spectrum of the instantaneous Hamiltonian (1.2) are unknown. This calls for alternative ways to optimize the evolution of the

[13] This does not mean that no one will be able to come up with a new classical algorithm for a specific problem that beats QAOA. In fact this has happened in some cases [8].

system and find the optimal way of traveling from the driving Hamiltonian to the target one. This optimization can be formulated by giving a variational Ansatz for $\Gamma(t)$ with respect to a number of tunable parameters; this, in turn, runs into similar problems as the classical optimizer in VQAs.

In summary, one can engineer promising ways of solving difficult and interesting combinatorial optimization problems on both gate-based and analog NISQ era paradigms. However, they require to navigate very complex and noisy landscapes, as well as escaping non-adiabatic transitions. It is therefore of great relevance to search for ways of solving these problems.

1.2 Outlook of this work

Now that all of the pieces of the puzzle have been laid out, we set out to explain our intentions with this thesis within the context that was just introduced.

As a compelling candidate algorithm to navigate the complex energy landscapes of quantum optimization, we choose the Monte Carlo Tree Search algorithm [27]. It is an algorithm that has become exceptionally well known in the context of board games such as Go and chess, and is being found to be exceptionally adaptable for more and more applications within other contexts in recent years [28]. As we will see, it is also applicable to quantum optimization, if we succeed in formulating the optimization task as a board game that it can play. Its possible advantages include its gradient-free nature and very original approach to the problem, as well as its versatility, resistance to noise and the option of straightforwardly integrating the algorithm within deep learning frameworks. Chapter 2 will be devoted to introducing, discussing, as well as benchmarking our implementation of the Monte Carlo Tree Search algorithm.

The goal is to test the algorithm in both existing NISQ era computational paradigms that were introduced in the last section. Because of the particular quantum algorithms that we will choose, it makes sense for us to start out with Adiabatic Quantum Computation first, which we explain more in-depth in Chapter 3. We also try the Monte Carlo Tree Search with our implementation of Quantum Annealing for a simple system with two spins.

After that, we move onto the hard combinatorial optimization problem that we will focus on: 3-Satisfiability or 3-SAT, the first problem that was proven to be NP-complete [20], as well as its NP-hard variant. We discuss 3-SAT and its Hamiltonian representation in Chapter 4.

We then try to see how the Monte Carlo Tree Search algorithm holds up when trying to solve difficult instances of 3-SAT with quantum optimization algorithms. In Chapter 5 we present and discuss the results for the case of Quantum Annealing.

Only then do we move onto the gate-based Variational Quantum Algorithms in Chapter 6, where we introduce the Quantum Approximate Optimization Algorithm as well as displaying the results obtained in our implementation of this second computational paradigm. Following these results, a discussion will follow where we propose and test a set of applications and modifications for our algorithm in the context of the Quantum Approximate Optimization Algorithm.

Chapter 7 will conclude this thesis. In it, we will summarize the main results of this work and the conclusions we can extract from them, along with discussing the various doors that said results opened, as promising future directions in which the present work can be further extended.

Chapter 2

Monte Carlo Tree Search

In this chapter we introduce one of the most central parts of this thesis, the Monte Carlo Tree Search (MCTS) algorithm. The aim of this section is to understand how MCTS works and why it could help us with quantum optimization algorithms. We very briefly explore the context in which the MCTS algorithm originated in Sec. 2.1, moving on to the technical details of the algorithm in Sec. 2.2. We then benchmark our own version of MCTS with a test game and comment on its versatility, strengths and possible weaknesses in Sec. 2.3. Further MCTS-related discussion can be found in Appendix A.

2.1 Background

Before applying the Monte Carlo Tree Search algorithm to the case of quantum optimization, we should first try to understand it in its most natural habitat: game artificial-intelligence creation. We are particularly interested in games that can be represented by decision trees such as the one in Fig 2.1.

Each node in the tree represents a particular state of the game. The branches that connect the different nodes are the allowed paths between the states or the legal moves of the game, which progresses downwards until it reaches what is called a *leaf node*, representing a terminal or game-over state. Leaf nodes have a score or reward associated with them, often given by a real number, that depends on the particular outcome of the game. For a particular node, the nodes below it that it is directly connected to are called its *children nodes*, or simply its *children*, and the nodes that are directly connected to it from above are called its *parent nodes* or *parents*.

Consider now that we start at the very top of the tree of a particular game, and want to create an algorithm to find the path down the tree that gives us the best possible outcome in the game (depending on the game, this could be a win or a maximum score). The most obvious way of doing just that is to perform a brute force search, which has to be conducted differently depending on the number of players, and is commonly known as the *Minimax*

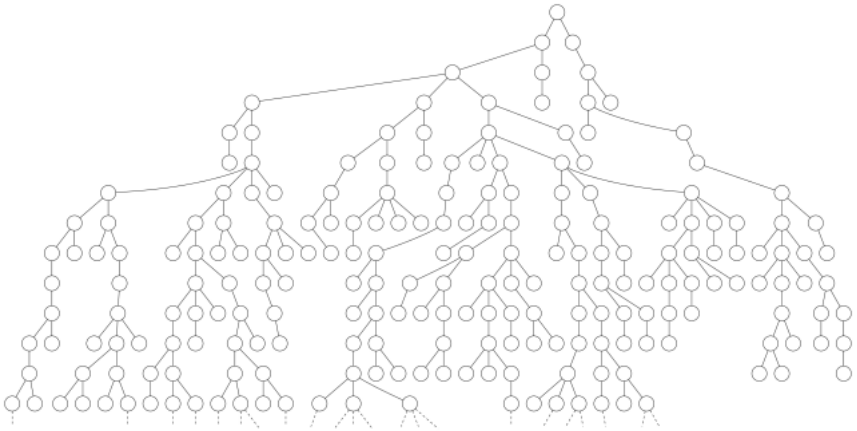


Figure 2.1: An example of part of a decision tree.

search for two player games. It consists of exploring each and every possible path down the tree, calculating the scores of all of the leaf nodes and selecting the optimal one by assuming that every player plays the game perfectly, i.e., by *mini*mizing our loss when the opponent plays the moves that give us the *max*imum loss possible [29].

Evidently, this will quickly become unfeasible in all but the simplest games, as the search spaces for most games can get absurdly big. In the case of chess, for example, one can estimate[1] the number of leaf nodes to be 10^{123}, all of which would need to be visited by the Minimax search! There exist more intelligent ways of searching trees, such as the very popular alpha-beta pruning [29], where the parts of the tree which are found to be unquestionably worse than a previously examined path are excluded or *pruned* from the search[2]. Still, to make alpha-beta pruning usable for big search spaces, it is usually necessary to stop the search when a particular depth is reached, before it arrives at a leaf node. At that point, some game-specific heuristic position evaluation function can be used, which estimates the goodness of the position from that non-terminal state. These could be hand crafted to take into account various criteria in the position[3], or even estimated by neural networks [30].

We can find a vast number of enhancements to alpha-beta pruning, such as

[1] From the average branching factor of the tree (the average number of legal moves in a position) b, which is around 35 for chess, and the average length of a game d, which is approximately 80 half-moves: $b^d \approx 10^{123}$.

[2] In the best of cases, the required number of leaf node evaluations is reduced by $\mathcal{O}(b^d) \rightarrow \mathcal{O}(b^{\frac{d}{2}})$ with respect to Minimax. This depends on the order in which the tree is explored; specifically, how early in the search we explore the best move.

[3] In the case of chess, the position evaluation function could take into account the material imbalance in the position via the relative value of the pieces, and other factors such as pawn structure, center control, king safety and more, all of which would have their corresponding weights in the calculation.

Transposition Tables (using symmetries of the tree to further reduce the size of the search space), Iterative Deepening (starting from a low depth search and gradually increasing it, combining the advantages of depth-first and breadth-first searches), and many more [29]. Most advanced game playing machines use a mix of these techniques to navigate decision trees.

The algorithm central to this thesis, the Monte Carlo Tree Search algorithm, is a way to achieve the same thing. And its way of going about is not fundamentally all that different from the more traditional techniques mentioned earlier. Its main advantage over alpha-beta pruning is that it does not require a game-dependent heuristic position evaluation function. It can make very meaningful evaluations of states just from sampling random paths down the tree in a clever way. This implies that it is extremely versatile, in that no strategic or tactical knowledge about the particular game is required for it to function, which makes it especially useful for games lacking a good heuristic to properly evaluate non-terminal positions. It has also been demonstrated to work extremely well when paired with Neural Networks that guide the search [28].

MCTS was first proposed in [27], and its power was demonstrated a few years later with the board game *Go*. The alpha-beta pruning variants, which worked decently well for games like chess and checkers, really struggled with Go because of the much larger size of the search-space, and the difficulty of finding good heuristics to evaluate positions. However, with the emergence of MCTS based algorithms [31], the level of play of Go playing algorithms increased sharply until in 2015 Deepmind's AlphaGo, which combined MCTS with neural networks, was able to win a match against one of the best human players. Since then, many MCTS based algorithms have been created and able to obtain superhuman playing strength for other board games, such as chess [32] and Scrabble [33], and even for more complicated, real-time and nondeterministic games [34].

Furthermore, currently more and more applications outside of game playing are being found for MCTS or variations thereof. These include scheduling problems, simulations of physical systems and more [35, 34].

2.2 Technical details

We will now see how exactly the Monte Carlo tree search works. MCTS is a search algorithm that explores a given search space via random samples guided by a given selection strategy. The MCTS search starts from a particular node in the decision tree called the *root node*, and, as it is running, gradually constructs parts of the tree, focusing on the paths it deems most promising. The search can be interrupted at any time to suggest the best action or move from the root node. It consists of a 4-step cycle, called the *MCTS cycle*, which is repeated for a set number of times, or for a set time. As the number of

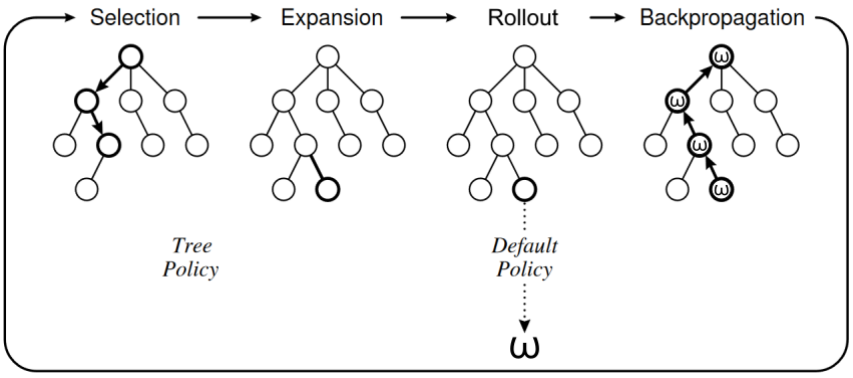


Figure 2.2: Schematic representation of one MCTS cycle. Figure modified from [34].

cycles grows, the suggested best action improves until it eventually converges to the optimal Minimax result. Because MCTS builds a replica from parts of the tree, to differentiate the actual full tree that we want to explore and the tree that the MCTS constructs and is saved to memory, we will use *tree in memory* when referring to the latter.

At the start of the game, MCTS starts with a tree in memory that only contains the initial or root node. The MCTS cycle is run a number of times, after which the algorithm outputs the move that it believes to be the best out of all the possible legal moves in the first turn. The MCTS algorithm can be run again for a subsequent turn by taking the chosen node as the new root node.

MCTS assigns two numbers to each node i in the tree in memory, which it builds as it runs: a score w_i, to keep track of how well node i has scored so far in the random simulations, and a visit count n_i, which keeps track of the number of times the algorithm has passed through the node i in the first of the four stages. After each MCTS cycle, the scores and visit counts of the nodes are updated, and that affects the decisions on the subsequent cycles, as well as the final chosen move.

The MCTS Cycle

The four stages of an MCTS cycle are *selection*, *expansion*, *rollout* or simulation, and *backpropagation*. The particular strategies used in the selection and expansion stages for an MCTS algorithm make up the *tree policy* of the algorithm, and the particular way the rollout happens defines its *default* or *playout policy*[4]. We now explore each of the four stages in more detail. Fig 2.2

[4]This makes the definition quite lenient, giving the freedom to develop many versions and variants of MCTS for specific types of search spaces by adjusting said policies.

contains a schematic representation of one MCTS cycle.

1. **The selection stage**

   In this first stage, the algorithm travels through the tree in memory that it has built from previous cycles. This process always starts at the root node, goes downwards and ends upon arriving at a node which has at least one child that is not yet stored in the tree in memory. In the very first cycle, the selection ends at the root node itself since it is the only node in the tree in memory.

   In order to choose a path, the algorithm requires a *selection policy*: a function that takes the scores and visit counts of the nodes as inputs, and assigns a potential of exploration to each node. The algorithm descends through the tree in memory level by level, at each point choosing the child with the maximum potential for exploration according to the selection policy. A good selection policy should maintain a proper balance between the exploitation of well scoring nodes and the exploration of nodes that lead to unknown or less explored paths. By far the most widely used selection policies are the Upper Confidence Bounds applied to Trees (UCT). Specifically, the UCB1 policy, which for node i is

   $$\mathrm{UCB1}_i = \frac{w_i}{n_i} + C\sqrt{\frac{2\ln n_i^{(\mathrm{p})}}{n_i}}, \tag{2.1}$$

   where $n_i^{(\mathrm{p})}$ is the visit count of the parent node of i, and C is a constant that we can choose to adjust the weight of each of the two terms.

   Intuitively, (2.1) is easy to understand: the first term is the average score of node i, so that it is higher for nodes which are believed to be good. The second term vanishes as node i is selected more often, because in that case $n_i \sim n_i^{(\mathrm{p})}$, and $\frac{\ln n_i}{n_i} \to 0$ as n_i increases. Instead, this term will increase logarithmically for nodes that are not getting selected as often by the algorithm in relation to their parent, as $n_i^{(\mathrm{p})}$ will increase in those cases while n_i stays the same. In summary, the first term encourages the algorithm to select the best scoring child, while the second one encourages it to also consider its more neglected siblings, which might be hiding some yet undiscovered potential, and the C balances both contributions. This is precisely the exploitation/exploration trade-off that was mentioned earlier. There is much more to the UCB1 expression than what was mentioned here, its very interesting origin can be found in [36], and there is some discussion about alternative policies in Appendix A.

   Therefore, our algorithm should, at each step, select the node that maximizes (2.1) to make sure that the algorithm is making the choice which will be the most beneficial to learn about the best path down the tree. In case of a tie scores, the tie is broken randomly.

Once the selection arrives at a node with some child in the decision tree not yet present in the tree in memory (an unexpanded child), the algorithm is done selecting the path with the most promise for exploration within the tree in memory. The selection stage ends and we enter the expansion stage.

2. **The expansion stage**

   We are at the selected node with at least one unexpanded child. In the expansion stage, we simply add to the tree in memory one (or more) previously unexpanded children of the selected node.

   There exist many different ways to perform this expansion [34]. In our case we will follow the most common approach of expanding only one child per cycle[5], selected randomly out of all the available unexpanded children. The newly expanded node is initialized with a null score and visit count. After adding the new node to the tree in memory, the algorithm travels to the newly expanded node and the expansion stage ends.

   With these two stages, the *Tree policy* of our algorithm has been defined. We now move onto the rollout or simulation stage.

3. **The rollout stage**

   This third stage is the point where the random sampling enters, and where the algorithm takes its "Monte Carlo" part of the name from.

   Starting from the newly expanded node, random moves are played by the algorithm until a terminal node is reached, and the reward associated to the terminal node's state is evaluated. That reward will serve as a way of updating the values of the nodes in the selection path, which will in turn affect the UCB1 scores of many nodes in the tree in memory.

   Note that the random simulations happen at the decision tree level and none of the nodes below the newly expanded node are saved into the tree in memory. Additionally, the way the playouts happen can also be semi-guided, instead of being completely random, although this negatively affects the versatility of the algorithm because game-specific criteria need to be taken into account in that case. Formally, the way the game plays out from the newly expanded node defines the *default policy* of the MCTS algorithm.

[5] This, as it will become apparent later, will have the added benefit that the number of function evaluations to obtain rewards from leaf nodes will equal the number of MCTS cycles.

Once we have a reward from the random simulations, the rollout stage terminates and we enter the backpropagation stage.

4. **The backpropagation stage**

   In the last stage of the cycle, we travel from the node expanded in stage 2, all the way back up to the root node along the route that was selected in the first stage. Along the way, the visit counts n of the nodes we pass through are increased by one, and their scores w are updated according to the reward obtained in the third stage.

   The way the scores are updated depends on the particular game we are dealing with[6]. However, these will be fairly straightforward for the case of quantum optimization, as it will become obvious later.

The MCTS cycle concludes after the backpropagation stage. Whenever the algorithm gets to this point, the final choice on the best move from the root node can be made with the statistics obtained so far. Alternatively, another cycle can be run to further expand the tree in memory and update the values of the nodes to make the final decision slightly more accurate.

We might think of some exceptional cases, where a terminal node is expanded in the second stage. This most commonly happens if the root node is close to the bottom of the tree or a very large amount of MCTS cycles is run, but it is not a problem. In those cases, the rollout is trivial and the reward is the score of the leaf node. If that terminal node is selected during a selection stage in a later cycle, the expansion stage can simply be skipped since leaf nodes have no children, so that the rollout once again happens from the leaf node itself. Note that no node other than a leaf node can be the starting point of a rollout more than once unless the tree in memory is reset.

Making the final choice

When the MCTS search is terminated, either because the desired amount of cycles have been executed or because the maximum allowed time has been exceeded, the best child of the root node has to be chosen. There exist a number of criteria to make this choice [34]. The one chosen for our algorithm is the one known as *max_child*, where we choose the child with the highest average score. This is the same as choosing the child that maximizes (2.1) with $C = 0$.

[6] In the very common case of a two player game where the only outcomes are {win, loss, tie}, the corresponding rewards could be chosen to be {1, -1, 0}. If we arrive at a leaf node where the MCTS player wins (loses), we should add a winning score of 1 to all the nodes along the path where the MCTS (opponent) player played the last move, and a losing score of -1 to every node where the opponent (MCTS) player played last. Whereas, in the case of a tie, all of the nodes along the path should get a tying score of 0. This makes sure that the MCTS understands that the opponent is also trying to make good moves.

Further discussion on different criteria, and the reasons behind our choice of using that particular one, can be found in Appendix A.2.

Conclusions

Once we understand how the MCTS algorithm works, the comments in Sec. 2.1 praising its adaptability should seem quite reasonable. For any game tree, we require nothing other than a way to get the legal moves from a given position and a way to extract the corresponding rewards from terminal nodes, and MCTS will be able to learn to play the game without any other information about the domain to be searched. It also has a very straightforward behavior with its cycles, each iteration having a very similar time and resource cost to the next.

There is also interesting discussion to be had about what should happen to the tree in memory when one wants to use the MCTS algorithm for a later turn after using it for a previous one. The advantages and disadvantages of recycling or resetting the tree in memory are explored in Appendix A.4, as well as proposing an alternative memory subtree recycling method which might be interesting to study.

2.3 Benchmarking MCTS

Our implementation of MCTS [37] should, in principle, be able to play any game as long as it has a way of obtaining a list of legal moves from positions and rewards from leaf nodes. Tic-tac-toe, the two player game where the players win by placing three of their marks lined up in vertical, horizontal or diagonal on a 3×3 grid, is a simple yet not too trivial game, and a very typical test for such game-playing algorithms. We test it here to see that MCTS is working as intended before moving on to using it for quantum optimization.

To see that MCTS is able to properly learn this game, we present it with various challenges: we first test the performance of the MCTS player against a player that simply plays random moves in Figure 2.3, which it is able to consistently beat or tie with ease, especially in the case that MCTS plays first. We also made it play against a Minimax algorithm that looks at the entire tree and plays the game perfectly in Figure 2.4. We found that MCTS plays at a comparable level when running 500-600 cycles. Another very interesting and common thing to test is to make MCTS play against another version of itself, and to see whether the games become of higher level for a higher number of MCTS cycles, which we confirm in Figure 2.5.

We also ran further tests where we gave the MCTS player a set of specific prearranged positions and told it to choose the move that it would play in such position, but to keep this section from becoming too lengthy, the results can be found in Appendix A.1.

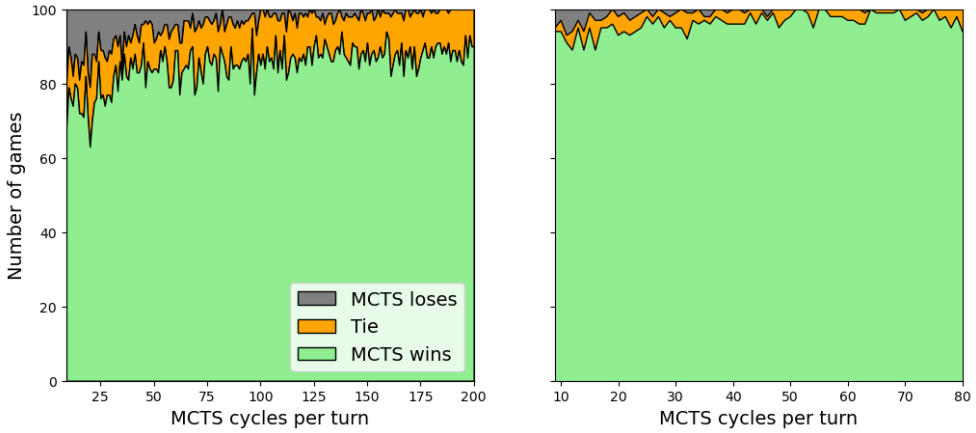


Figure 2.3: MCTS plays tic-tac-toe against a completely random player. Results with varying number of MCTS cycles when MCTS plays second (left) and first (right). This shows that the advantage of the player that plays first is considerable, as the number of required simulations for perfect play goes from around 80 MCTS cycles when playing first to around 200 when playing second. Notice that no matter how much we increase the number of cycles, MCTS is unlikely to win every game since the random player occasionally just happens to play well and manages to score a tie, which is unavoidable.

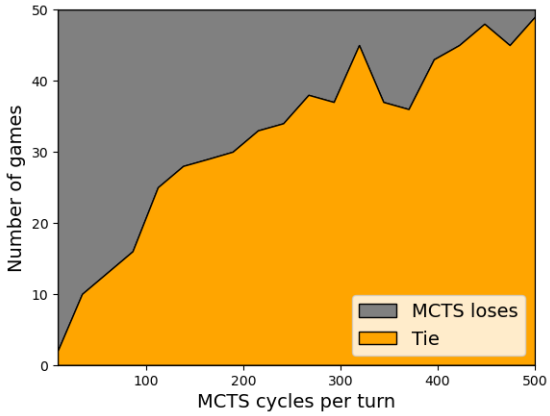
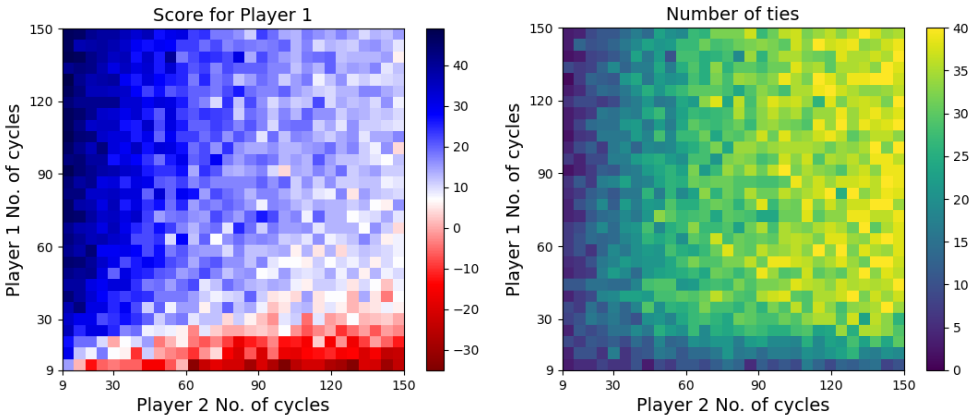


Figure 2.4: MCTS plays tic-tac-toe against the perfect player (Minimax) that we implemented. Results with varying number of MCTS cycles, where, because of the computational cost of running the Minimax algorithm, 50 games were played per cycle number instead of 100. The Minimax player always plays optimally, so that it is impossible to win against it, scoring a tie is the best that can be done. The perfect player plays first, making this the hardest possible challenge for MCTS.



(a) (b)

Figure 2.5: MCTS plays tic-tac-toe against itself. The grid shows different numbers of cycles used, with 50 games played for each pixel. In (a) we display the score defined as $N(\text{wins}) - N(\text{losses})$ for the MCTS player that plays first. The players win more and lose less the more MCTS cycles they run in comparison to their opponent's number of cycles. When both players have a high number of cycles, the scores get closer to zero as the games played are of higher level and more games are tied. This also shows that player 1 (the one who goes first) has a clear advantage over player 2, since, if the game was more balanced, the color map would be more symmetric around the diagonal where the number of cycles for both players are equal. (b) Shows that the density of games that end in ties is significantly higher for the cases where the players play with high numbers of MCTS cycles, indicating higher level play as expected. Conversely, the games that end decisively are much more common when the level of play is lower or there is a vast difference in playing strength between the players.

With these very promising results, we move on to explaining the type of challenging games that the MCTS algorithm will be faced with, starting with Adiabatic Quantum Computation.

Chapter 3

Adiabatic Quantum Computation (AQC)

In this chapter we study Adiabatic Quantum Computation, one of the currently realizable computational paradigms for quantum optimization, and its experimental realization, Quantum Annealing. At the heart of any kind of quantum analog simulation is the resemblance of the physical phenomenon to be simulated to the simulating device; in a way, the device itself becomes the system to be simulated. In the case of AQC, the quantum annealer simulates a time dependent quantum mechanical system which evolves according to adiabatic Schrödinger dynamics for a high enough evolution time, with the objective of obtaining the complicated ground state of the system.

In Section 3.1 we build upon the explanations of the introductory Section 1.1 by formulating Quantum Annealing in a more concrete way and studying it further. In Section 3.2 we look more in-depth at the adiabatic expansion, the mechanism which makes Quantum Annealing work, and we test our findings in a simple Ising chain in Section 3.3. In the remaining section Section 3.4, we discuss how Quantum Annealing path optimization can be formulated as a board game, such that MCTS learns to play it and we test it for the simple toy model.

3.1 Quantum Annealing (QA)

We have a Hamiltonian H_{target} the ground state $|\Psi_0\rangle$ of which we are dying to find. Either because we are intrigued by the low energy dynamics of the system or because it would provide us with the result of a fascinating optimization problem. First, for our convenience, as we saw that in classical optimization H_{target} will be made out of σ^z operators, we refer to the target Hamiltonian as H_z from now on. Similarly, the driving Hamiltonian H_{driving} will be nicknamed

H_x, as it is conventionally chosen to be the transverse field Hamiltonian[1]

$$H_x = -\sum_j \sigma_j^x, \tag{3.1}$$

where the index j on a Pauli operator indicates that it acts non-trivially only on the jth spin. For the case of n spins we would have

$$\sigma_j^x = \underbrace{\overbrace{\mathbb{1}}^{1} \otimes ... \otimes \overbrace{\mathbb{1}}^{j-1} \otimes \overbrace{\sigma^x}^{j} \otimes \overbrace{\mathbb{1}}^{j+1} \otimes ... \otimes \overbrace{\mathbb{1}}^{n}}_{n}, \tag{3.2}$$

with all of the operators on the right-hand side being 2×2 dimensional. In Quantum Annealing the system is usually constructed by interpolating both Hamiltonians as

$$\tilde{H}(t) = (1 - f(t))\, H_x + f(t)\, H_z, \tag{3.3}$$

where $f(t)$ is a function $f : [0, \tau] \to [0, 1]$ known as the annealing schedule[2], with τ being the total evolution time of the system —commonly referred to as the annealing time. The most intuitive, albeit naive, schedule we can choose is the linear schedule $f(t) = \frac{t}{\tau}$, which brings the system from $\tilde{H}(0) = H_x$ to $\tilde{H}(\tau) = H_z$ at a constant rate. For most schedules we will still have that $f(0) = 0$ and $f(\tau) = 1$, such that $\tilde{H}(t)$ starts from H_x and ends up at H_z, even if the interpolation is not linear[3].

The way to proceed with QA is to start the system in the easy-to-construct ground state of (3.1) at $t = 0$, with all of the n spins lined up in the $+x$ direction

$$|\psi(0)\rangle = \frac{1}{2^{n/2}} \bigotimes_{j=1}^{n} (|\uparrow\rangle_j + |\downarrow\rangle_j). \tag{3.4}$$

As the Hamiltonian changes with time, the evolution of the system will be governed by the Schrödinger equation

$$i\partial t\, |\psi(t)\rangle = \tilde{H}(t)\, |\psi(t)\rangle\,, \tag{3.5}$$

such that after the evolution the system will end up in the state

$$|\psi(\tau)\rangle = U(\tau)\, |\psi(0)\rangle\,, \qquad U(\tau) = \mathcal{T} e^{-i \int_0^\tau \tilde{H}(t')dt'}, \tag{3.6}$$

with $\mathcal{T}$ being the time-ordering operator. The idea is that we would like this state to be as close as possible to the ground state of the target Hamiltonian,

[1] To see why there is often a minus sign in front, see [38].

[2] It is very common to use $s(t)$ for the schedule instead [39]. Here we reserve the letter s to represent the dimensionless time $s = t/\tau$ to avoid confusion, just as in [26].

[3] In fact, if this is not the case, it is impossible for the evolution to be adiabatic. Non-adiabatic schedules of this kind are desirable in some cases, such as when the evolution operator is discretized or some sort of digital bang-bang schedule is employed [39].

$\langle\Psi_0|\psi(\tau)\rangle \to 1$. Fortunately, this is guaranteed by the quantum adiabatic theorem [40], which states that, provided the evolution time τ is long enough, $|\psi(\tau)\rangle$ will end up arbitrarily close to $|\Psi_0\rangle$. As was mentioned in the introductory Chapter 1, the adiabatic theorem gives us the condition of the evolution time τ being large enough as

$$\tau \gg \max_{j,t} \left\{ \frac{\xi_j(t)}{\Delta_j(t)} \right\}, \tag{3.7}$$

where $\xi_j(t)$ is the matrix element of the time derivative of the Hamiltonian (3.3) that connects the ground state and the jth excited state at time t, and $\Delta_j(t)$ the energy difference between the ground state and the jth excited state at t.

We now explore this condition in more depth to get some interesting and useful results.

3.2 Adiabatic Perturbation Theory

We assume we have a Hamiltonian of the form (3.3), with the very important[4] caveat that its time dependence always enters as t/τ. In this section we thus work with the Hamiltonian written in terms of the dimensionless time parameter $s = t/\tau$, $\tilde{H}(t) \to H(s)$, with $s \in [0, 1]$ being the only time dependence that it has. Another way of stating the condition is that the Hamiltonian $\tilde{H}(s\tau)$ must not explicitly depend on the total evolution time τ. This is the case for most interpolated Hamiltonians in QA, and for all of the Hamiltonians we will consider throughout this work.

The Schrödinger equation now gets an extra factor τ as

$$i\partial_s |\psi(s)\rangle = \tau H(s) |\psi(s)\rangle . \tag{3.8}$$

We will be working with the instantaneous eigenstate basis $\{|\varepsilon_i(s)\rangle\}$, the elements of which satisfy the time-independent Schrödinger equation at every time s during the evolution,

$$H(s) |\varepsilon_j(s)\rangle = \varepsilon_j(s) |\varepsilon_j(s)\rangle . \tag{3.9}$$

It is also important to mention that we assume that the eigenvalues never cross along the evolution path, such that, for all times and all j, we have $\varepsilon_j(s) \leq \varepsilon_{j+1}(s)$; and, in the case of the ground state $j = 0$, there should be a non-zero gap, $\varepsilon_0(s) < \varepsilon_1(s)$. The only exception to this is that we can have a degenerate ground state at $s = 1$, after the evolution.

[4] This ensures that there do not exist two different timescales during the evolution, which can break adiabaticity even when (3.7) is met. See [41] to see an example of such a case.

The underlying idea of this derivation will be to figure out how the system changes for long evolution times if we are initially at the instantaneous ground state $|\psi(0)\rangle = |\varepsilon_0(0)\rangle$. We can do so by performing an asymptotic expansion of the coefficients of the instantaneous eigenstates in powers of $1/\tau$.

We first expand the state in the basis of the instantaneous eigenstates as

$$|\psi(s)\rangle = \sum_j c_j(s) e^{-i\int_0^t ds' \varepsilon_j(s')} |\varepsilon_j(s)\rangle \,. \tag{3.10}$$

The exponential in this Ansatz, which corresponds to a quantity known as the dynamical phase, is not strictly necessary, but is there for simplification reasons. Inserting (3.10) into (3.8) gives the differential equation that the $c_j(s)$ coefficients obey:

$$\dot{c}_k(s) = -\sum_j c_j(s) e^{-i\tau\int_0^s ds'(\varepsilon_k(s')-\varepsilon_j(s'))} \langle\varepsilon_k(s)|\dot{\varepsilon}_j(s)\rangle \,, \tag{3.11}$$

where the dot represents differentiation with respect to the dimensionless time parameter s. We also acted from the left with another state $\langle\varepsilon_k(s)|$ and used the orthonormality of the instantaneous eigenbasis. In order to proceed, we should notice that we can rewrite the $\langle\varepsilon_k(s)|\dot{\varepsilon}_j(s)\rangle$ terms as

$$\langle\varepsilon_j(s)|\dot{\varepsilon}_k(s)\rangle = \frac{\langle\varepsilon_j(s)|\dot{H}(s)|\varepsilon_k(s)\rangle}{\varepsilon_k(s) - \varepsilon_j(s)} \tag{3.12}$$

for $j \neq k$, by differentiating (3.9) with respect to s. For the special $j = k$ case, we have that the overlap $\langle\varepsilon_j(s)|\dot{\varepsilon}_j(s)\rangle$ is a purely imaginary quantity, because

$$\partial_s \left(\langle\varepsilon_j(s)|\varepsilon_j(s)\rangle\right) = \langle\varepsilon_k(s)|\dot{\varepsilon}_j(s)\rangle^* + \langle\varepsilon_k(s)|\dot{\varepsilon}_j(s)\rangle = 0, \tag{3.13}$$

where the last equality comes from $\langle\varepsilon_j(s)|\varepsilon_j(s)\rangle = 1$. We can also check that if we add a time dependent phase to the instantaneous eigenstates $|\varepsilon_j(s)\rangle \to e^{i\gamma_j(s)} |\varepsilon_j(s)\rangle$, the overlap will transform by an imaginary amount,

$$\langle\varepsilon_j(s)|\dot{\varepsilon}_j(s)\rangle \to i\dot{\gamma}_j(s) + \langle\varepsilon_j(s)|\dot{\varepsilon}_j(s)\rangle \,. \tag{3.14}$$

Therefore, since both terms in the right-hand side are imaginary, we can always choose a basis such that the eigenstates are appropriately rotated to make sure that for every j, $\langle\varepsilon_j(s)|\dot{\varepsilon}_j(s)\rangle = 0$ holds. This is ensured with $\gamma_j(s) = i\int_0^s ds' \langle\varepsilon_j(s')|\dot{\varepsilon}_j(s')\rangle$, known as the Berry phase[5].

With these considerations, we may rewrite (3.11) as

$$\dot{c}_k(s) = \sum_{j\neq k} c_j(s) \frac{\mathcal{A}_{jk}(s)}{\Delta_{jk}(s)} e^{i\tau\int_0^{s'} d\tilde{s}\Delta_{kj}(\tilde{s})}, \tag{3.15}$$

[5]We could have also put this extra phase factor into the Ansatz itself in (3.10) to begin with and arrive at the same result.

where we have defined the energy difference $\Delta_{jk}(s) \equiv \varepsilon_j(s) - \varepsilon_k(s)$, which has absorbed the minus sign, and wrote the matrix elements of $\partial_s H(s)$ as $\mathcal{A}_{jk}(s) \equiv \langle \varepsilon_j(s)|\partial_s H(s)|\varepsilon_k(s)\rangle$. We integrate the last expression from an initial time $s' = 0$ to a time $s' = s$ to obtain

$$c_k(s) = c_k(0) + \sum_{j\neq k} \int_0^s ds' c_j(s') \frac{\mathcal{A}_{jk}(s')}{\Delta_{jk}(s')} e^{i\tau \int_0^{s'} d\tilde{s} \Delta_{jk}(\tilde{s})}. \tag{3.16}$$

At this point, a way to qualitatively arrive at the adiabatic condition would be to argue that, as one increases τ, the integral oscillates faster and faster and the contribution from the second term becomes negligible, since the term at the saddle-point $j = k$ is not in the sum. From this it would immediately follow that at high enough τ the coefficients of different levels do not mix and that the system remains in the ground state for all times. We now see this in slightly more detail.

Let us first consider an integral $\int dx g(x) e^{i\lambda f(x)}$ for some functions f, g, and a real number λ that we can tune, repeatedly integrating by parts gives

$$\int dx g(x) e^{i\lambda f(x)} = -i \frac{g(x)}{\lambda f'(x)} e^{i\lambda f(x)} + \mathcal{O}(\lambda^{-2}). \tag{3.17}$$

We can see all other terms are higher powers in $1/\lambda$ because in the integral that remains after integrating by parts we have the $e^{i\lambda f(x)}$ factor again, which after each subsequent integration gives an extra λ^{-1} factor. Then, the condition on the required largeness of λ to make this expansion valid and the integral small due to the fast oscillations is $|\lambda| \gg |g(x)/f'(x)|$, as long as f' does not become very small and g does not become very big at some point, and the higher order terms behave nicely. This, applied to the integrals in (3.16), will give us something close to the adiabatic criterion (3.7). After after saying $\lambda = \tau$, $g(s) = c_j(s)\frac{\mathcal{A}_{jk}(s)}{\Delta_{jk}(s)}$ and $f'(s) = \Delta_{jk}(s)$, by (3.17), we need

$$\tau \gg \frac{|\mathcal{A}_{jk}(s)|}{|\Delta_{jk}^2(s)|}, \tag{3.18}$$

which follows from the fact that $|c_j(s)|^2 \leq 1$ and the multiplicativity of the absolute value of complex numbers[6] and the fact that $\tau > 0$. This needs to be true for every single $j \neq k$ in the sum of (3.16), but, as we will see, there are some terms in that sum which are actually of the order $\mathcal{O}(\tau^{-2})$ for the case we are interested in. For that, consider the case where initially the system is in the ground state, such that $c_k(0) = \delta_{k,0}$ (we could have chosen any arbitrary phase), and, by recursively substituting the expression (3.16) for the coefficient in the second term on the right-hand side of (3.16) itself,

[6] Even if they were only submultiplicative, like the operator norm, this analysis would be valid, as we would instead get an upper bound on the condition.

we see that each j except $j = 0$ gets an extra τ^{-1} factor, because they get another integral! All of the terms in the new $\mathcal{O}(\tau^{-1})$ sum except the $j = 0$ contribution vanish because of the Kronecker delta. Then, we have that $c_0(s)$ has no first order correction, $c_0(s) = 1 + \mathcal{O}(\tau^{-2})$, while all the $c_{k\neq 0}(s)$ do, $c_{k\neq 0}(s) = \mathcal{O}(\tau^{-1})$. We can calculate the τ^{-1} correction term explicitly by integrating by parts just like before:

$$\begin{aligned} c_{k\neq 0}(s) &\sim \int_0^s ds' \frac{\mathcal{A}_{0k}(s')}{\Delta_{0k}(s')} e^{i\tau \int_0^{s'} d\tilde{s}\Delta_{0k}(\tilde{s})} + \mathcal{O}(\tau^{-2}) \\ &= \frac{i}{\tau}\left[\frac{\mathcal{A}_{0k}(0)}{\Delta_{0k}^2(0)} - \frac{\mathcal{A}_{0k}(s)}{\Delta_{0k}^2(s)} e^{i\tau \int_0^s d\tilde{s}\Delta_{0k}(\tilde{s})}\right] + \mathcal{O}(\tau^{-2}), \end{aligned} \tag{3.19}$$

where the extra term after integrating by parts has been ignored as it is proportional to τ^{-2} as discussed. This is the adiabatic approximation to first order. Now, if we say that the condition for the adiabatic approximation needs to hold for all times, so that $c_{k\neq 0}(s)$ is, for all k, s, vanishing, we get to the adiabatic condition

$$\tau \gg \frac{\max_{k,s}\{\mathcal{A}_{0k}(s)\}}{\min_{k,s}\left\{\Delta_{0k}^2(s)\right\}}. \tag{3.20}$$

As we will see in later chapters, it will be precisely the inverse-of-the-gap-squared dependence that will make trying to achieve adiabatic evolution as dictated by (3.20) unfeasible for many hard optimization problems.

With this analysis, we have also found how exactly the evolved state approaches the ground state: for all levels $k \neq 0$, when we are in the adiabatic regime, the residual energy $\varepsilon^{\text{res}} = \langle\psi(\tau)|H_z|\psi(\tau)\rangle - \langle\Psi_0|H_z|\Psi_0\rangle$ after the evolution will vanish as τ^{-2}; specifically, as the square of the modulus of the leading-order term in (3.19). We will use this in the following section.

One could also calculate higher order corrections to the adiabatic solution by computing more terms from (3.16); and, although not particularly relevant for this work, this method has been used in fascinating ways to create improved annealing schedules by making terms from the adiabatic expansion vanish by choosing schedules with vanishing derivatives at the endpoints in [26].

But what happens in the case where we cannot say that we are in the adiabatic regime?

Faster evolution regimes

When we evolve the system $H(s)$ for an evolution time τ which does not fulfill (3.20), possibly because there exists a very tight avoided gap crossing along the evolution path making high enough τ values unfeasible, and the evolution time is instead comparable to the inverse square of the gap $\tau \sim \frac{\mathcal{A}}{\Delta^2}$, the previous analysis no longer applies. In this τ regime the evolution of the system is governed by Landau-Zener theory [42, 43], which states that the probability of excitation of the system to the first excited state after crossing

the gap is

$$|\langle \varepsilon_1(s=1)|\psi(s=1)\rangle|^2 \approx e^{-\tau\frac{\pi\Delta^2}{4v}}, \tag{3.21}$$

where $v = \partial_s f(s)|_{s=s^*}$ is the rate of change of the schedule at the gap minimum time s^*. This is, in theory, only valid for two-level systems that evolve for an infinite time. And although there exist ways to calculate the equivalent expressions for more complicated systems [44], and finite times [45], many systems can be remarkably well approximated by an effective two level system near the avoided gap crossing, and the Landau-Zener formula (3.21) is often employed to approximate the extent of the excitations [46, 47]. It is also nice to notice that we even recover the inverse of the gap squared adiabatic condition from the requirement that the exponent be small to minimize the probability of excitations.

If we go to even lower τ, we enter a fast regime where the change of the system happens so quickly that other mechanisms describe the dynamics. In cases where the system is driven through a second order phase transition, we can describe the residual energy as a function of the annealing time by a power law via the Kibble-Zurek mechanism [48]. This is the case, for example, of 1-dimensional Ising chains with nearest neighbor interaction [49].

In the remaining sections of this chapter, we test quantum annealing in a simple system. We first see that the behavior of the system agrees with our analysis, and then move on to using MCTS for annealing path optimization.

3.3 Toy model: flipping two spins with QA

Before moving on to systems which represent hard optimization problems, we check the results up to this point for a smaller-scale system with just two spins.

The toy model

We consider the following target Hamiltonian:

$$H_z = \frac{1}{4}\left(3 - \sigma_1^z - \sigma_2^z - \sigma_1^z\sigma_2^z\right), \tag{3.22}$$

where the 3 is implicitly multiplying the identity matrix. In the σ_z basis it has the form

$$H_z \doteq \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.23}$$

The ground state of H_z corresponds to the state where both spins are pointing up in the z-direction, $|\Psi_0\rangle = |\uparrow\uparrow\rangle$. We will try quantum annealing with this

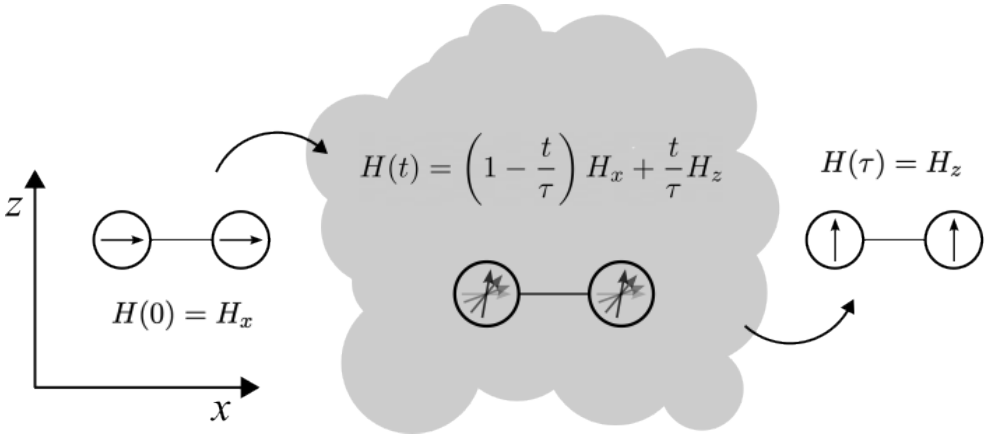


Figure 3.1: Quantum annealing for the simple model with two spins. The system must adiabatically transition from an initial paramagnetic state to a ferromagnetic one during the annealing process.

very simple system first. To that end, we choose the following —slightly modified from (3.1)— driving Hamiltonian:

$$H_x = \frac{1}{2} \sum_j \left(\mathbb{1} - \sigma_j^x\right). \tag{3.24}$$

The reason we choose this driving Hamiltonian is that it scales the energies to coincide with the target Hamiltonian, i.e., it makes the ground state have an energy of 0. Other than that, this changes nothing and the initial ground state will still be of the form of (3.4). This choice and the reason of the particular choice of the target Hamiltonian (3.22) should become more logical after reading Chapter 4, where we discuss the type of optimization problems we will be trying to solve in Chapter 5 and Chapter 6.

QA on the toy model

For now, we interpolate both the target and driving Hamiltonians with a linear schedule, $f(s) = s$ with $s = t/\tau$,

$$\tilde{H}(t) = \left(1 - \frac{t}{\tau}\right) H_x + \frac{t}{\tau} H_z, \tag{3.25}$$

and perform quantum annealing: we start from the ground state of (3.24) at $t = 0$, let the system evolve according to (3.25), and implement a Schrödinger evolution (3.8). To that end, we use the open-source QuTIP library [50], which we will be using all throughout this work [37]. A schematic representation of the annealing process is shown in Fig 3.1, and the instantaneous energy landscape for our system is represented in Fig 3.4-(a).

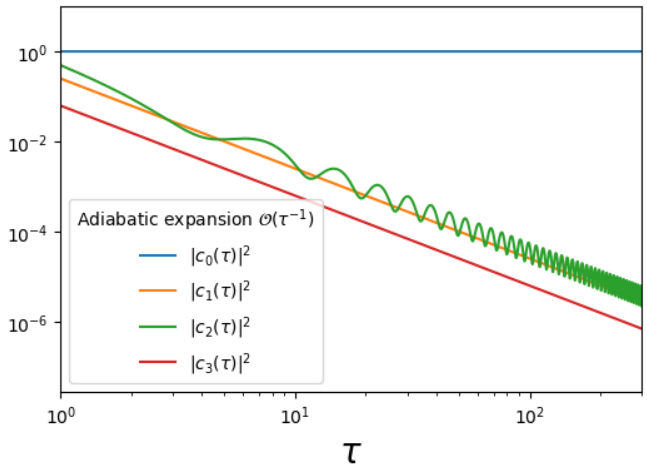


Figure 3.2: The coefficients of the instantaneous eigenstates after the evolution up to first order in the adiabatic approximation for the toy model (3.25), valid for τ large enough to be in the adiabatic regime. The modulus squared of the $j = 2$ coefficient is the only one that oscillates. The reason for this is that for all other coefficients, either $\mathcal{A}_j(0)$ or $\mathcal{A}_j(\tau)$ is zero, which makes them lose the oscillatory relative phase term when calculating the modulus squared. Instead, $c_2(s)$ has nonzero values for both cases, which gives rise to the oscillatory behavior proportional to $\cos(\tau \int_s \Delta_j)$. This makes the coefficient oscillate more and more as τ increases. The integral between $j = 2$ and the ground state $j = 0$ (see Fig 3.4-(a)) was calculated numerically to be 0.872. The c_0 term stays constant to this order.

Since this is a small system and we are using a linear schedule, which makes $\partial_s H(s) = H_z - H_x$ constant in time, the $\mathcal{O}(\tau^{-1})$ term from adiabatic perturbation theory, (3.19), can be computed explicitly for all c_i, with $i = 1, 2, 3$, since, as we saw, $c_0 \sim 1$ does not have a correction to this order, by computing the matrix elements of the derivative of the interpolated Hamiltonian. We plot these coefficients for different annealing times in Figure 3.2. Since we were able to calculate all of the coefficients, we can predict how the energy should approach the ground state energy (0, in our case) when τ is large enough such that the first order adiabatic expansion is valid:

$$\langle\psi(\tau)|H_z|\psi(\tau)\rangle = \sum_{j=1}^{3} \varepsilon_j |c_j|^2 \sim \frac{1}{\tau^2} \sum_{j=1}^{3} \left| \frac{\mathcal{A}_{j0}(0)}{\Delta_{j0}^2(0)} + \frac{\mathcal{A}_{j0}(\tau)}{\Delta_{j0}^2(\tau)} e^{i\tau \int_0^1 ds \Delta_{j0}(s)} \right|^2 . \tag{3.26}$$

This can be seen, dashed gray line, along with the results from QA in Figure 3.3. In that same figure, we see that the annealing results do follow the adiabatic behavior for high enough τ, and we even observe the oscillatory behavior of the residual energy in the QA results, as predicted by the adiabatic

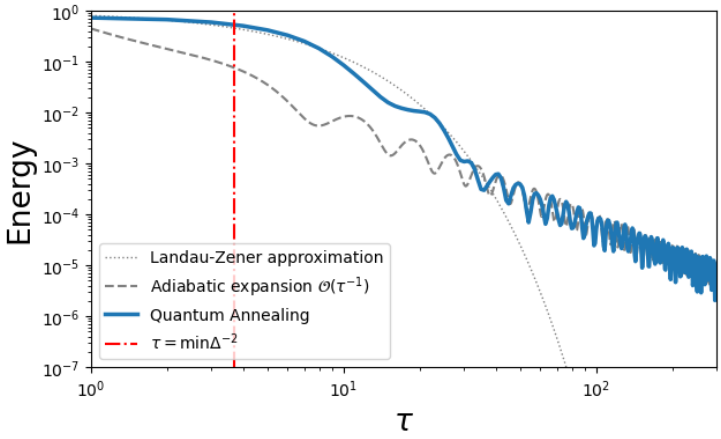


Figure 3.3: Results for QA with a linear schedule for the toy model (3.25). The dashed curve is (3.26), the first order from adiabatic perturbation theory. The dotted Landau-Zener curve follows (3.21) and simulates the probability of transitioning to the first excited level after the gap minimum crossing. This is an approximation that assumes that all the other possible transitions other than to the first excited state are negligible, due to the higher energy gap. The red vertical line marks the square of the inverse of the gap minimum, which helps indicate the locations of the different regimes. As expected, the LZ approximation is accurate around the vertical line, while the adiabatic expansion agrees with QA in the adiabatic regime.

expansion in Fig. 3.2. In the non-adiabatic regime we observe that the residual energy closely follows a Landau-Zener distribution, where the rate of change of the system is proportional to τ^{-1}, because of the linear annealing schedule.

We now move onto trying out MCTS for finding improved annealing paths for the simple toy model (3.22).

3.4 MCTS plays the toy model

As was discussed in Chapter 2, MCTS works on trees, which require a discrete search space by nature[7]. The way to proceed is then to express the schedule in Ansatz form, with a set of discrete parameters that can be adjusted to tune the path that the interpolated time-dependent system should follow. This will make the path optimization problem a variational minimization problem. The MCTS algorithm will then play a game where it has to choose said variational

[7]Interestingly, recent MCTS modifications exist that allow for continuous search space optimization for some simple problems by using a progressive widening technique for the MCTS algorithm [51].

parameters out of a discrete set at each turn, and return an annealing schedule after it is done.

Arguably, the most straightforward way of creating an Ansatz that can characterize an annealing schedule is what we will call the discretized QA schedule.

MCTS-guided discretized QA

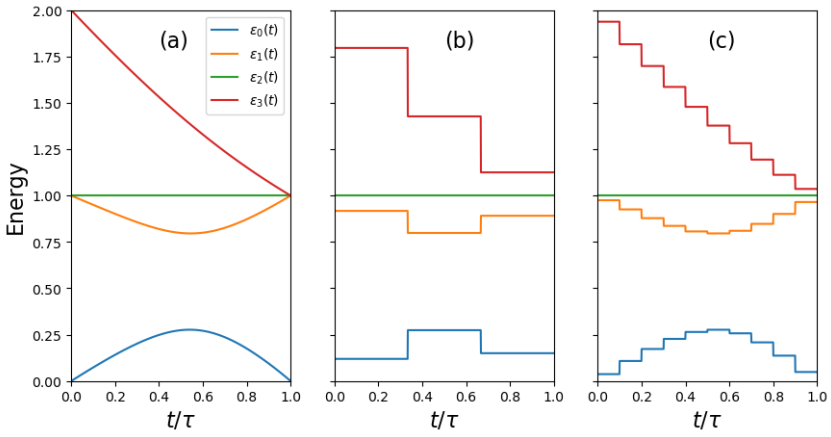


Figure 3.4: Instantaneous eigenspectra of the toy model with a linear annealing schedule for different discretizations of the schedule. The interpolated system has a minimum energy gap $\Delta = 0.52$ at $t = 0.53\tau$. (a) Continuous QA spectrum; (b) $P = 3$ discretized QA spectrum; (c) $P = 10$ discretized QA spectrum.

In discretized QA we divide the continuous annealing schedule into P segments, each with its corresponding length Δt_m and a constant value $s_m \in [0, 1]$. The resulting time evolution operator that such a schedule will induce will be the discretized version of the operator in (3.6),

$$U_{\text{step}}(\tau) = \prod_{m}^{\leftarrow P} e^{-iH(s_m)\Delta t_m}, \tag{3.27}$$

where the arrow signifies that the exponentials at earlier times should go on the right. This suffices to create a one-player board game for MCTS to play: a game with P turns, where the mth turn consists of choosing the value for s_m out of b possible values between 0 and 1. P will thus be the depth of the tree and b the branching factor, such that the tree representing the game will contain exactly b^P leaf nodes. We will select the choices for the schedule to be evenly spaced as

$$s_m \in \left\{ \frac{1}{2b}, \frac{3}{2b}, \frac{5}{2b}, ..., \frac{2b-1}{2b} \right\}, \tag{3.28}$$

where b determines the number of available choices. The choice of the values is such that it matches those of the discretized linear schedules in Figure 3.5, with $P = b$ steps.

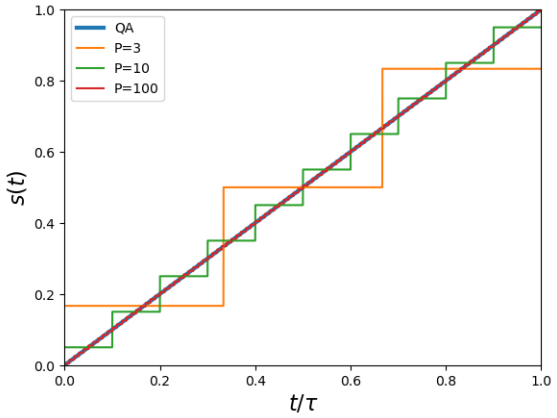
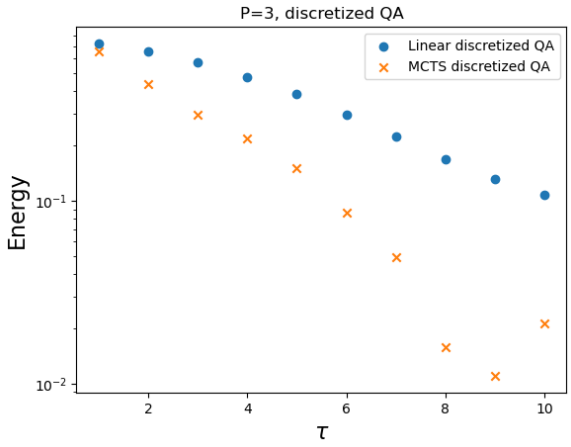


Figure 3.5: Discretized linear schedules for different P. The value of s is chosen to be the one at the midpoint of every time segment.

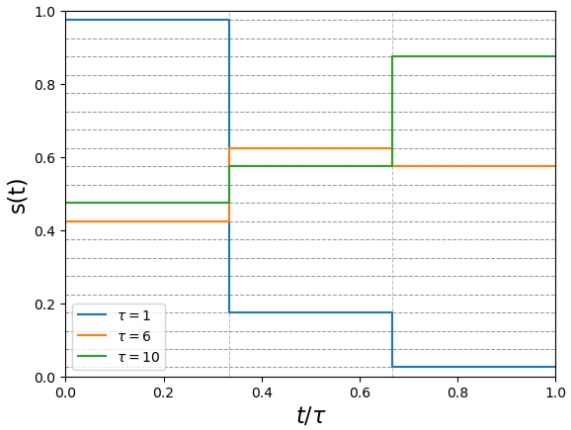
We will also choose to divide τ into segments with equal duration, $\Delta t_j = \tau/P$ for all $j = 1, ..., P$; and, with this, we are ready to start testing MCTS in the context of quantum optimization.

The results for $P = 3$ and $b = 20$ can be found in Figure 3.6, where the improvements of the MCTS path optimization can be seen to be meaningful. MCTS improves upon the linear on every single case, and it is curious to look at the schedules that it ends up choosing in Figure 3.6-(b): the higher the annealing time, the more it seems to prefer to utilize schedules that are closer to the linear solution, or that spend more time near the gap in an attempt to minimize excitations. Whereas, for low annealing times it seems to favor bang-bang type schedules. These can be intuitively understood as the system trying to rotate the spins around the z-axis first by applying a unitary operator with $s_m \sim 1$, and then bringing the system to the z-direction via a rotation around x, with $s_m \sim 0$. All in all, the MCTS improves the results and the choices it makes do make some physical sense.

We should note that we have no reason to believe that with just 200 MCTS simulations the results have converged to the global optimum in all cases. MCTS is based on randomness, so it is normal for it to just do worse sometimes, if it is unlucky and the random simulations guide it along a less-than-optimal path in the first turn. Even then, we know that it will give good paths in virtually all cases, as the paths it chooses will always have a high chance of leading to regions with good-scoring values of the schedule s_m. The $\tau = 10$ result is particularly interesting as it gives a worse energy than the $\tau = 8$ and $\tau = 9$ ones. This is likely because at that point, the error from the discretization starts to become significant, making the resulting energies



(a)



(b)

Figure 3.6: Results of MCTS schedule optimization on the toy model in the regime of $\tau < 10$, where the error from the discretization is mostly under control (see Sec. 3.4 and App. B.1). with $P = 3$, $b = 20$. MCTS ran for 200 cycles per turn. (a) MCTS and linear schedule energies of the $P = 3$ discretized schedule within the regime where the discretization error is not significant. Linear energies correspond to those of the blue line in Fig. 3.7. (b) Three example paths chosen by MCTS for different annealing times. The gray dashed grid on the background represents the possible choices the MCTS algorithm had available.

much more irregular, which makes MCTS struggle more than in other cases. These kinds of behaviors will be discussed in the next section and, in a broader sense, in later chapters.

These results allow us to make an extremely useful conclusion: even if the annealing time τ is not high enough to ensure adiabaticity in a given

regime, i.e., $\tau \lesssim \Delta^{-2}$, results can be significantly improved via annealing path optimization around the Landau-Zener regime. Meaning that not all hope is lost, even if the gap is very small, or closes very quickly with the system size as will be the case with hard optimization problems, and MCTS might be an appropriate tool for just that, which we will investigate in later chapters.

Putting a bound on the error

Before moving on, it is important to note a few things about the discrete QA schedule that we chose in the last section. We should note that, when discretizing the annealing schedule, the system cannot evolve smoothly any longer because the system Hamiltonian (3.25) will change in P bumps instead, as can be seen from its energy spectra in Figure 3.4-(b),(c). This means that the evolution can no longer ever be truly adiabatic, and that a possibly significant source of error has been introduced to the QA process. In this subsection we will see if and how this error can be kept under control.

It is already known from (3.27) that in the $P \to \infty$, $\Delta t_m \to 0$ limit we recover the continuous time evolution operator (3.6) by approximating the Δt_m-weighed sum in the exponents by a time integral. But it is also important[8] to see whether the size of the overlap

$$|\langle\psi(0)|U^\dagger(\tau)U_{\text{step}}(\tau)|\psi(0)\rangle|, \tag{3.29}$$

i.e., how much the discretized solution deviates from the continuous one, can be made arbitrarily close to 1 with a reasonably-scaling choice of the discretized mesh size. The full proof where we attempt to put bounds on the error has been relegated to Appendix B.1. Here we only very briefly summarize the procedure and present the main results.

The idea is to find upper bounds for the operator norm (defined in Appendix B.3) of the difference of the two time evolution operators. By following a similar, but more general, process to that in [52], we find that, for the case with a discretized linear schedule like in Figure 3.5, we have

$$||U(\tau) - U_{\text{step}}(\tau)|| \leq \sqrt{\text{poly}(n)\frac{2\tau}{P}}, \tag{3.30}$$

where poly(n) is a function that only contains terms which are polynomial in n, which happens to be $||H_z - H_x||$ in the case of a linear schedule. For the case of a system like the toy model, we quickly find that we can bound the quantity as $||H_z - H_x|| \leq 3n$, where in our specific case, the size is $n = 2$. The upper bound in (3.30) makes intuitive sense: as we increase the annealing time τ, each of the P pieces of duration Δt becomes larger, such that the approximation

[8] Furthermore, because of how related QAOA is to discretized QA, this discussion is also very relevant for Chapter 6, and it is even a first step to proving the universality of QAOA, as we will see.

becomes more inaccurate. Therefore, with discretized annealing schedules with P pieces, we get better results with increasing τ as expected, just like with continuous schedules, because the process is more and more adiabatic. This keeps happening until at one point, the error from the discretization becomes unmanageable and the results no longer improve with increased τ. We plot the results which confirm this for the toy model with different P in Fig. 3.7.

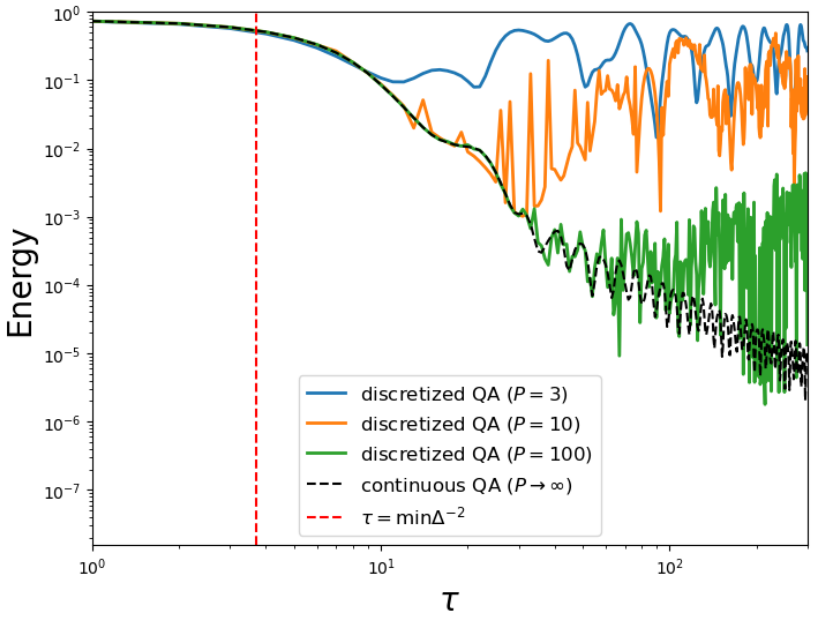


Figure 3.7: Residual energy after QA for the toy model with linear schedules with different number of discretization steps P. The higher the P, the longer it agrees with continuous QA until it breaks for a high enough annealing time.

As we increase the annealing time within regimes where the error dominates, we would expect some sort of pseudo-periodic behavior of the energy as the high size of the Δt in (3.27) introduces phases larger than 2π. In Appendix B.1 the procedure is outlined for more general schedules. We also check with the toy model that we should choose a P proportional to τ so as to properly approximate the time evolution of continuous QA (Fig B.1).

With this it should become clearer why the MCTS results in Fig 3.6a only went up to $\tau = 10$, as the error becomes as high as the residual energy after that point. The truth is that we will see later that the game that MCTS plays seems to become harder once we approach the edges of the Landau Zener regime, or when some other effects such as the discretization error come into play. This warrants a higher number of MCTS simulations for those cases or some other kinds of modifications, which will be mentioned and discussed in later chapters.

Chapter 4

The 3-Satisfiability problem

In this short chapter, we briefly discuss the combinatorial optimization problem we will be solving in subsequent chapters. Our problem of choice will be 3-SAT or 3-Satisfiability. In Section 4.1 we introduce the general k-SAT optimization problem and in Section 4.2 we explore a way to encode the problem into a quantum Hamiltonian.

4.1 Description of k-SAT

Before formulating the k-SAT problem, some definitions are in order. A boolean variable or a bit, which we represent with x, is an object that can only take two values, TRUE or FALSE. Such variables can be subjected to three kinds of basic operations: conjunction, represented by $\wedge$ (AND); disjunction, represented by $\vee$ (OR); negation, represented by $\neg$ (NOT). These three operations can be used as a basis to create more complicated operations[1]. A boolean statement or a logical statement is a function $\varphi(\mathbf{x})$ of n bits $\mathbf{x} = \{x_1, ..., x_n\}$, which combines them with the operations that were just mentioned. Thus, the boolean statement takes an n-bit-long string of bits and spits out a truth value. We will be looking for the particular assignment of bits $\mathbf{x}^*$ that satisfies the statement, such that $\varphi(\mathbf{x}^*) = TRUE$, but for some statements with a very specific form.

The k-Satisfiability or k-SAT problem deals with logical statements that are said to be in k Conjunctive Normal Form[2] (kCNF). A kCNF formula has a very particular structure: it is made up of m clauses separated by AND operations, where each clause is made up of k different bits separated by OR

[1] For example, an Exclusive OR operation between two bits x_1, x_2 can be written as $(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$.

[2] Also known as clausal normal form.

operations:

$$\varphi = \bigwedge_{i=1}^{m} C_i, \qquad C_i = l_{i_1} \vee l_{i_2} \vee ... \vee l_{i_k}, \tag{4.1}$$

where each of the l_{i_k} is a literal of a bit (either the bit itself or its negation, $l_i \in \{x_i, \neg x_i\}$) among the n bits in the problem. Every single logical statement can be rewritten as a CNF formula[3].

Note that it requires $\mathcal{O}(km)$ steps to check the validity of a solution, which means that k-SAT is in NP, as an answer can be checked in polynomial time in the problem size. The $k = 1$ case is trivial, as it suffices to check whether a bit and its negation are simultaneously in the logical statement to know that it is unsatisfiable, and it is satisfiable in all other cases, which means that 1-SAT is in P. The $k = 2$ case is a very well known case which also happens to be in P, but is harder to prove [20]. It is for $k > 2$ that the satisfiability problem starts to be very difficult and, from a computational viewpoint, interesting. In particular, 3-SAT is the first problem that was proven to be NP-complete, so that an algorithm that can solve 3-SAT in polynomial time would imply P = NP. In fact, it is often considered to be *the* NP-complete problem, and it is used to create most proofs for the NP-completeness of other problems in NP [53]. 3-SAT has an immense number of real-life applications as it is a type of problem that comes up in many planning problems, artificial intelligence, software-verification and much more [54, 53].

The difficulty of a 3-SAT problem can be roughly characterized by its $\alpha \equiv m/n$ clause to variable ratio. There exists a regime where, if one starts checking a high number of random 3-SAT problems, they start getting very difficult, which seems to happen for α in a small region around the critical value $\alpha_c \sim 4.2$. This is a well-known phase transition [55] represented in Figure 4.1.

3-SAT (and any k-SAT problem) has both an NP-complete decision version which asks whether there exists an assignment $\mathbf{x}^*$ that makes $\varphi(\mathbf{x}^*) = \text{TRUE}$ or not, where it is enough to find one assignment that satisfies the statement to solve the problem. And an NP-hard optimization version (known as MAX 3-SAT by computer scientists) which asks for the one assignment that violates the least amount of clauses, even when no assignment exists that satisfies φ. This is not really consequential for this work so we will not give too much importance to it; however, it is probably not harmful for us to know that we will technically be doing MAX 3-SAT instead of 3-SAT, which will be especially obvious in the case of QAOA, where, as we will see, the Ansatz can be surprisingly limiting for shallow depths.

[3] In a few cases, it is necessary to introduce new bits so that the size of the CNF statement with respect to the initial statement does not increase exponentially. Still, the equivalence of both statements is always kept, such that a solution for any one implies a solution for both.

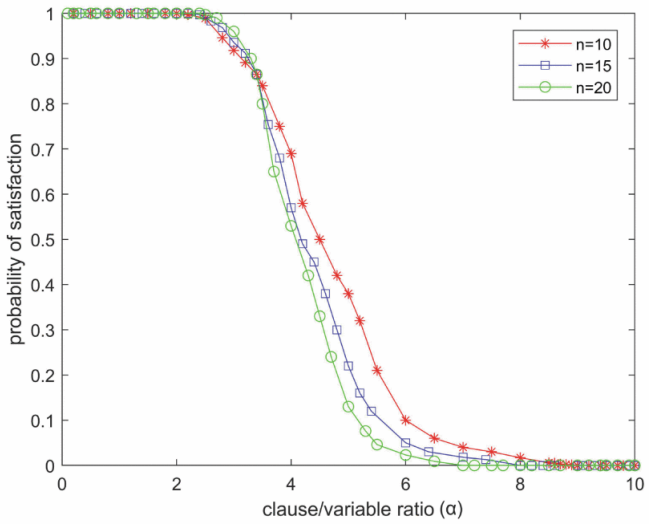


Figure 4.1: The phase transition for random 3-SAT instances. There exist three different regimes: for low α, before the phase transition, there are many variables n and very few clauses m, so that there is a high chance that most assignments satisfy the expression, making 3-SAT generally easy. For high α, after the transition, there are many clauses setting restrictions for a low number of bits, such that the problem instances have a very high chance of not containing a successful assignment, which also makes 3-SAT generally easy to solve. It is around the critical α_c that problems really become computationally hard. Figure taken from [56].

There is a surplus of other interesting properties of 3-SAT [56, 20], but they go way beyond the scope of this work. Suffice it to say that it is instances of such problems that we will be trying to solve in the subsequent chapters. However, we should mention that we will not be dealing with the 3-SAT problem in its logical statement form in most cases, but it will be converted into a quantum Hamiltonian, with the solution encoded in its ground state. We now briefly go over how we can go from a general satisfiability kCNF statement to its equivalent Hamiltonian.

4.2 Encoding a *k*-SAT instance into the target Hamiltonian

We mentioned in Chapter 1 that many NP-hard and NP-Complete combinatorial optimization problems can be formulated as Hamiltonians, where some of the most well known ones can be found in [22]. The satisfiability problem is unsurprisingly no exception.

In this section we see how a particular k-SAT problem statement can be

put into quantum Hamiltonian form, with the solution encoded in its ground state, as required by the quantum optimization methods that we are working with. The idea is for each energy eigenstate to represent a particular string of bits, where its energy tells us the quality of the answer, i.e., the more wrong an answer the higher its corresponding energy.

To build such a Hamiltonian, we first look at an individual boolean variable $x \in \{\text{TRUE}, \text{FALSE}\}$ and see how we could represent it as a quantum operator. We need an operator whose eigenvalues correspond to the two possible truth values of x. The most obvious candidate to build such an operator is of course σ^z, where the two spin orientations along the z axis represent the two truth values of x. If we chose $|\uparrow\rangle$ to be TRUE and $|\downarrow\rangle$ to be FALSE, we can re-scale the Pauli operator as

$$\hat{x} = \frac{1}{2}(\mathbb{1} - \sigma^z) \doteq \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \tag{4.2}$$

where the hat on top of $\hat{x}$ is there to signify that it is now a quantum operator. If the logical statement was the bit itself, $\varphi(x) = x$, (4.2) would represent the corresponding 1-SAT problem instance, i.e., an energy of 1 if the spin points down, which corresponds to x being FALSE, and 0 energy for the solution in which x is TRUE. If we instead had the negation of x, $\varphi(x) = \neg x$, we would need to swap the energies in the diagonal by changing the sign of σ^z,

$$\neg\hat{x} = \frac{1}{2}(\mathbb{1} + \sigma^z) \doteq \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \tag{4.3}$$

so that now the $x =$ FALSE solution is the lowest energy state. Now that we know how an individual bit can be represented, we move onto one entire disjunctive clause, such as the ones in (4.1). An individual clause in k-SAT is a disjunction of k literals, such that the clause evaluates to TRUE only if at least one of the literals in the clause does. The key insight here is that in our representations of bits (4.2), 0 energy represents a TRUE literal and an energy of 1 a FALSE one, which is the opposite of the values that one generally uses to represent truth values. Because of this, if 0 $\rightarrow$TRUE, 1 $\rightarrow$FALSE, the OR algebra between bits naturally arises from multiplication of operators of the form (4.2) and (4.3), scaled up to the required number of dimensions. Meaning that we can write a general individual clause C of k bits as a Hamiltonian by multiplying the corresponding literal operators

$$C = l_1 \vee l_2 \vee ... \vee l_k \rightarrow \hat{C} = \hat{l}_1 \vee \hat{l}_2 \vee ... \vee \hat{l}_k = \frac{1}{2^k}(\mathbb{1} \mp \sigma_1^z)(\mathbb{1} \mp \sigma_2^z)...(\mathbb{1} \mp \sigma_k^z), \tag{4.4}$$

where we choose the plus in cases where the particular literal involves a negation and the minus when it involves the bit itself. In this way, the resulting diagonal operator $\hat{C}$ from (4.4) will contain all of the possible bit configurations in its diagonal: the 0s in the diagonal will correspond to assignments that

satisfy at least one of the literals in the clause, and thus the clause, whereas the 1s in the diagonal will coincide with the assignments that fail to satisfy the clause as all literals within it evaluate to FALSE[4].

The last step to building a Hamiltonian for a Conjunctive Normal Form expression is to form the conjunction of all clauses in the expression. This can be done very naturally by simply summing over all clause operators (4.4). For a statement with n different bits in m clauses containing k bits each, the equivalent quantum Hamiltonian is

$$\varphi(\mathbf{x}) = \bigwedge_{i=1}^{m} C_i \rightarrow \hat{H} = \sum_{i=1}^{m} \hat{C}_i. \tag{4.5}$$

Each term in the sum is at most a k-body operator which acts trivially on the other $n-k$ bits. After performing the sum, the contributions from all clauses come together, such that the energy of each configuration indicates the number of clauses it violates. The ground state(s) is (are) the bit-string(s) that satisfy the whole expression; or alternatively, the assignments that satisfy the most amount of clauses. This makes this usable for both the NP-complete decision version of the satisfiability problem, and the NP-hard optimization version — unless, of course, we are dealing with 1-SAT, MAX 1-SAT, or 2-SAT, which are in P[5].

In practice, it is possible that, if one wants to solve 3-SAT instances on an actual quantum annealer, the k-body connectivity might not be available. In those cases, other more complicated ways are required in order to express the 3-SAT instance as the appropriate type of Hamiltonian [22].

Toy model Hamiltonian

The form of the target Hamiltonian (3.22) we chose for the simple toy model from the previous Chapter 3 was no coincidence. It was a Hamiltonian that was created following the procedure that was just explained for a satisfiability expression with $n = 2$. Let us take a look at the following 2-SAT expression:

$$\varphi(x_1, x_2) = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2). \tag{4.6}$$

This simple expression is satisfied by both bits being set to TRUE. By writing the operator of the first clause, we find $(x_1 \vee x_2) \rightarrow \mathrm{diag}[0,0,0,1]$ in the σ^z basis, which gives a nonzero energy for the assignment $x_1 = x_2 = \mathrm{FALSE}$, as required. Each of the other clauses will similarly introduce energy costs to the corresponding assignments, such that only the solution will be left with

[4] One might think that we have lost information in this process, as we do not keep track of the individual literal evaluations, but this is irrelevant for k-SAT as it is always sufficient to consider the evaluations at clause-level only.

[5] Interestingly, MAX 2-SAT is NP-hard.

0 energy as we saw in (3.23). We can easily check that the operator built according to (4.4) and (4.5),

$$H_z = \frac{1}{4}\left[(\mathbb{1} - \sigma_1^z)(\mathbb{1} - \sigma_2^z) + (\mathbb{1} + \sigma_1^z)(\mathbb{1} - \sigma_2^z) + (\mathbb{1} - \sigma_1^z)(\mathbb{1} + \sigma_2^z)\right], \quad (4.7)$$

evaluates to (3.22).

As a last note, this same procedure can also be used to create non quantum Hamiltonians. It suffices to use classical spin variables $s_i \in \{1, -1\}$ instead of σ_i^z operators. Building such energy functions could be a first step to solving k-SAT with classical energy minimization algorithms such as Simulated Annealing, which was mentioned in Chapter 1.1. This is also a nice way to appreciate the reason why all target Hamiltonians for classical combinatorial optimization can be made to be diagonal in the σ^z basis, as they always originate from classical Hamiltonians.

Chapter 5

MCTS-guided Quantum Annealing to solve 3-SAT

In this chapter we test MCTS as a tool to optimize QA schedules for hard combinatorial optimization problems. In Section 5.1 we take a look at the particular Ansatz we will be using for the annealing schedule, and thus, the game that MCTS will need to play. In Section 5.2 we see how the MCTS does for various instances of 3-SAT, and in Section 5.3 we look at some results in more detail. Lastly, we briefly explore the behavior of MCTS in noisy environments in Section 5.4.

5.1 Fourier gamemode

The discretized schedule (3.28), which we used for the toy model, is far from being the only (or the best) possible Ansatz for the annealing schedule.

Another interesting Ansatz for characterizing annealing schedules, proposed in [57], can be obtained by performing a Fourier series expansion around the linear schedule $f(s) = s$ as

$$f(t/\tau, \mathbf{B}) = \frac{t}{\tau} + \sum_{k=1}^{P} B_k \sin\left(\frac{k\pi t}{\tau}\right), \tag{5.1}$$

where this time the game will consist on selecting P Fourier coefficients $\mathbf{B} = \{B_1, ..., B_P\}$. Notice that if the same discretization for b choices is used for the Fourier coefficients as for the discretized schedule values s_m, the size of the tree will be the same as the discrete gamemode for an equivalent depth P. However, this Fourier gamemode has the advantage over the discretized schedule (3.28) that the schedule is continuous, which lets us forget about the discretization error. This means that with an schedule such as (5.1) we may increase τ as much as we would like without ever worrying about breaking the QA energy evaluations.

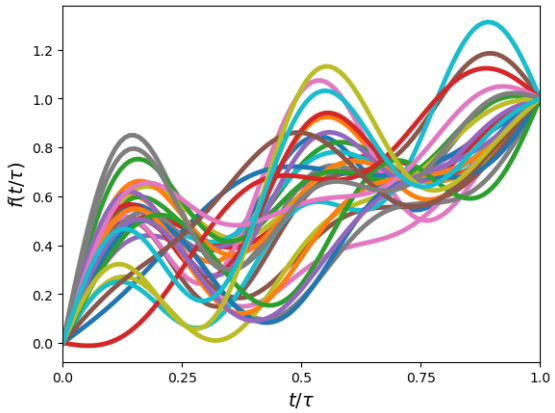


Figure 5.1: A number of annealing schedules $f(s)$ suggested by MCTS with $P = 5$ Fourier components.

Nevertheless, we should not conclude that an annealing path of the form (5.1) is always better than the discretized one. In fact, as we will see in subsequent sections of this chapter, we should make known that the Fourier coefficient optimization is probably not the absolute best way to approach annealing path optimization[1], and that is not to say that it is particularly terrible either. But this is not something that should worry us at all. The point of this chapter, and even that of this entire thesis, is not to provide some optimal control schedules for QA which are undoubtedly better than anything else. We merely want to see how the MCTS algorithm can hold itself with the particular problems that we confront it with. In the case of the Fourier gamemode, we let it adjust some Fourier coefficients around the linear solution and see how well it is able to do within the limitations of the Ansatz.

Notice that the time dependence still enters as the dimensionless time parameter $s = t/\tau$ for the Fourier Ansatz, meaning that the adiabatic condition still holds for Hamiltonians that are interpolated by these kinds of schedules.

5.2 Results on 3-SAT instances

We will run MCTS QA on 18 different problem instances of 11-bit 3-SAT with $\alpha = 3$, i.e., with 33 clauses per instance. All of the problems we will be dealing with are designed such that they all have one non-degenerate ground state with 0 energy, meaning each of them is solvable by the corresponding unique bit string that satisfies every single clause within the logical 3CNF statement.

[1] Likely an improved version of the Ansatz could be devised by basing it on the Chopped Random Basis (CRAB) Ansatz [58], in which the basis functions are randomized.

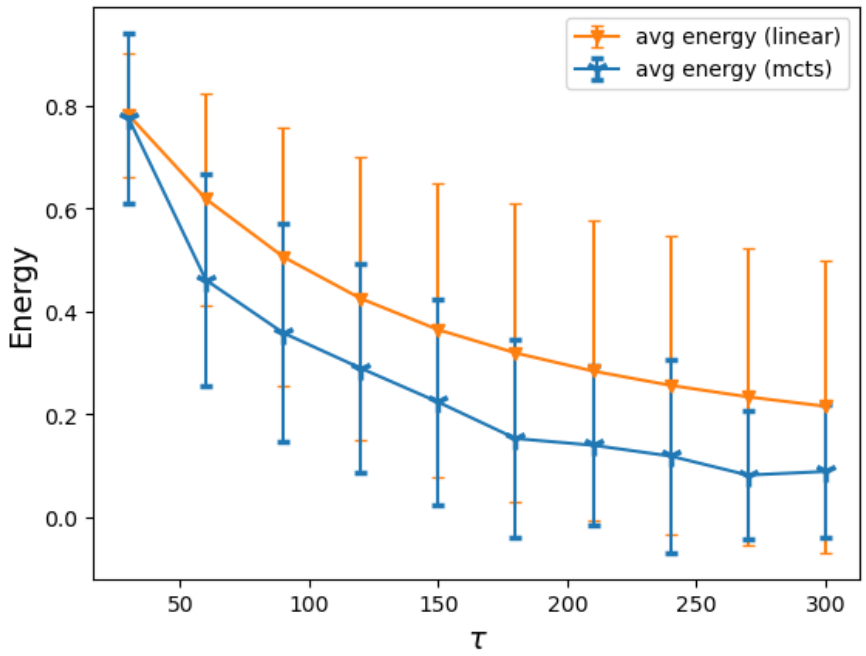


Figure 5.2: Energies of the final evolved state with a linear schedule (orange) and a MCTS-optimized schedule (blue) for 18 instances of $n = 11$ and $m = 33$ 3-SAT problems. The MCTS algorithm ran for 200 cycles per turn for a game with $P = 5$ Fourier components per instance. Each Fourier component had to be chosen from a set of $b = 40$ discretized values in [-0.2, 0.2]. The reason of choosing this particular range of values is to compare the performance of our algorithm to that of [57].

The main results can be found in Figure 5.2, where the performance of MCTS with a remarkably low number of simulation counts builds Fourier schedules of the type (5.1) with $P = 5$, and the energies are compared to the results obtained using a linear annealing schedule. The standard deviation of the energies across the instances is large because the 18 instances present very large differences in difficulty. This can be observed much more clearly in Figure 5.3, where all individual instance results are shown. The measured energy gap minima between the ground state and the first excited state range between 0.02 for instance 1, to 0.17 for instance 16. This makes the inverses of their squares significantly different. As such, the linear schedule struggles in the τ range under consideration only for some instances, which happen to be the ones where, in general, the MCTS improves the results the most. Because of the low number of simulations used, MCTS does occasionally fail to improve the results in some cases. But notice that we did not increase the number of simulations per turn from the 200 used for the model with 2 spins in Chapter 3.3, when now, apart from the problem being much harder, the

whole game tree has a much higher size of $40^5 \sim 10^8$ leaf nodes[2]. If one wanted the results to be better, that could be easily achieved by simply increasing the number of simulations. But there is also a lot of value in seeing that we can get very decent results for cases where a small number of calls to the annealing function were performed. Even so, our results are comparable to those in [57].

Figure 5.3: Individual linear (orange) and MCTS with 200 cycles per turn (blue) results for the 18 instances of 11-bit 3-SAT problems. Fig. 5.2 corresponds to the average of all problem instances of this Figure. This shows the clear disparity between different instances, some of them being considerably easier than others. These figures also display the randomness inherent to MCTS, as its results are much more irregular than the linear ones. We observed a similar behavior for the simpler model in Chapter 3.3.

[2]Our algorithm with $N_{\text{sim}} = 200$ MCTS cycles per turn will only visit $N_{\text{sim}}P - (N_{\text{sim}} - b) = 840$ leaf nodes for each instance, out of the $b^P \sim 10^8$ existing ones, and the number of unique leaf nodes is likely much lower, as many of them will repeat, especially in later turns!

5.3 What MCTS is doing: a closer look at some results

By looking at the energies of the final state after the QA evolution we do not really see how the MCTS algorithm is achieving the better results. In this section we look at a few examples to better illustrate what MCTS is doing.

We first focus on the instance with the smallest gap minimum out of the set of 18 problems, instance 1. Unsurprisingly, this is the instance in which the simple linear schedule struggles the most as can be seen in Figure 5.3. We can explicitly confirm that the unwanted excitations within the linear evolution are due to the gap crossing by diagonalizing the instantaneous Hamiltonian $H(s)$ at various times along the evolution path. We do so for two different annealing times in Figure 5.4-(a), where we see that until the gap minimum, the system was extremely close to the instantaneous ground state at all times, and it was precisely during the gap crossing that the unwanted deviation from the ground state happened. The lower annealing time achieves an even worse result as the gap is crossed at a higher rate of change, given by τ^{-1} for the linear schedule.

We then look at the two fundamentally different schedules that the MCTS algorithm comes up with to improve the results for instance 1, in Figure 5.4-(b). The schedule for $\tau = 300$ slows down at the gap minimum, which significantly reduces the excitations after the crossing as can be seen in Figure 5.4-(c), whereas, for a lower annealing time of $\tau = 60$, it chooses an entirely different schedule, which intentionally causes excitations and uses the excited levels as an alternative diabatic path. By crossing a gap minimum quickly, it makes sure that the probability for excitation is high, and it even utilizes the gap between the first and the second excited states to excite itself into the second excited level and thus partly avoid the gap between the ground state and the first excited state. By trying to make the first excited state as populated as possible before the very last quick crossing, it has some chance of ending in the ground state at the end. This agrees with the general behavior that was observed for the toy model schedules in Fig 3.6b, where it would choose non-adiabatic schedules for lower annealing times. We find that, in general, while slow and steady wins the race when we are closer to or slightly below the Landau-Zener regime $\tau \sim \Delta^{-2}$, fast and erratic schedules are preferable, for a lack of a better option, for regimes of lower τ.

Figure 5.5 shows another example of each type of annealing schedule, where the results of MCTS paths give particularly good energies and improvements over the linear schedule. As we increase the number of MCTS cycles, we expect the algorithm to find many more such paths.

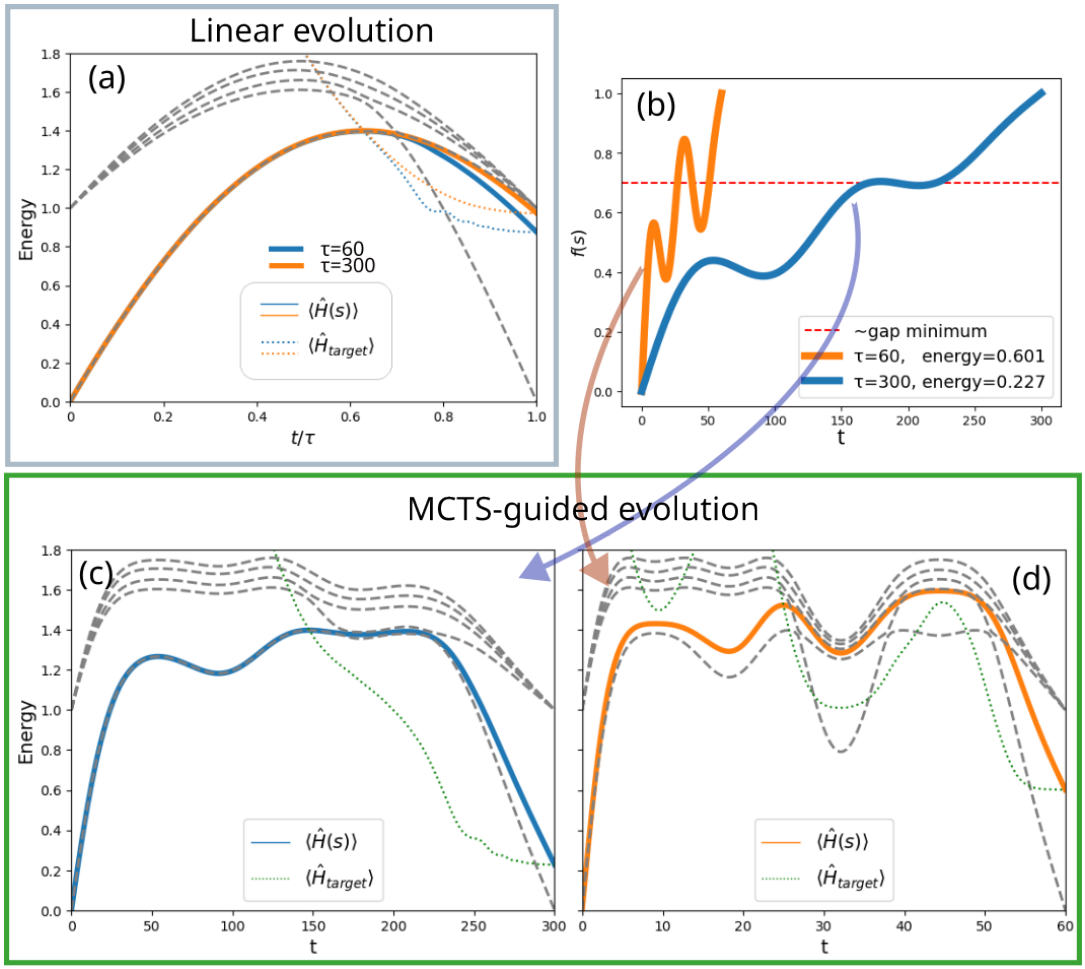


Figure 5.4: Results for the hardest 11-bit 3-SAT instance ($\Delta_{\min} = 0.028$) in more detail. In (a) we represent the instantaneous eigenvalues of the system with a linear interpolation of Hamiltonians by the dashed gray lines. We also represent the instantaneous energy of the system during the evolution for two different evolution times $\tau = 300$ (blue) and $\tau = 60$ (orange). As expected, the lower τ evolution has more losses due to excitations and achieves a worse energy. The dotted lines correspond to the expectation value of the target Hamiltonian H_z on the instantaneous states for the corresponding τ. (b) shows two choices of annealing schedules for solving instance 1 by MCTS for $\tau = 300$ (blue) and $\tau = 60$ (orange), and the dashed red line corresponds to the approximate location of the gap minimum. The blue path slows down near the gap, while the orange one crosses it various times without slowing down. These correspond to two different strategies that MCTS employs. (c) and (d) are equivalent to (a) but with the corresponding schedules from (b): the one in (c) tries to minimize loses by slowing down near the gap, and (d) employs a non-adiabatic path to get an approximate result.

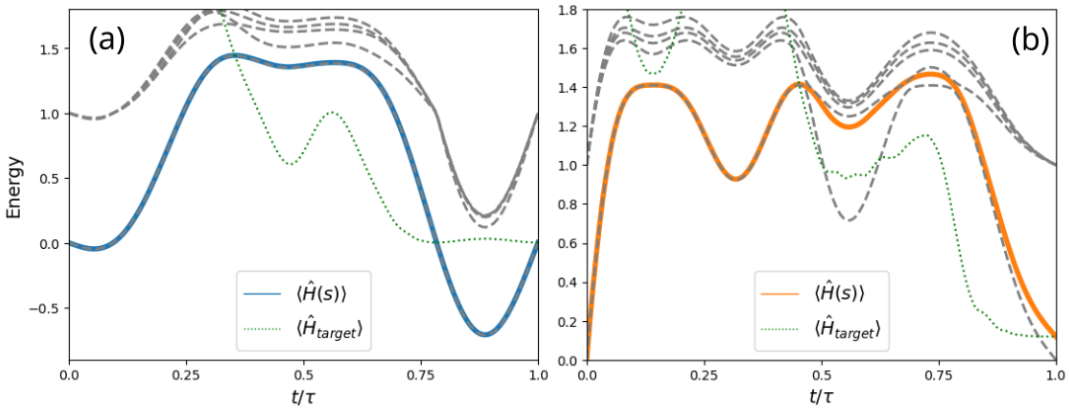


Figure 5.5: Two more examples of state evolutions along MCTS-designed annealing schedules. (a) MCTS took an already well scoring instance ($\langle H_z \rangle = 0.012$ at $t = \tau$ with a linear schedule) and further improved it to $\langle H_z \rangle = 0.002$ at $t = \tau$ by slowing down near the gap. Notice the eigenvalues take negative values, as MCTS chooses a schedule higher than 1 and lower than 0 near the endpoints, which allows it to slowdown for longer. (b) deals with one of the instances with a smaller gap, it scores an energy of $\langle H_z \rangle = 0.119$ at $t = \tau$ while the linear scores $\langle H_z \rangle = 0.651$ at $t = \tau$.

5.4 The noisy case

We conclude the analysis of Quantum Annealing with a brief look at one of the other postulated advantages of MCTS, especially relevant over gradient-based optimization algorithms: its ability to efficiently navigate noisy environments. For that we introduce a noisy component within the reward evaluations during the rollout stages of the MCTS cycle, such that the scores that the MCTS is dealing with are not the real energies of those paths. Every time the MCTS algorithm needs to evaluate a path $\boldsymbol{B}$, we add an extra Gaussian noise term, which serves as a way to simulate the various uncertainties that we would encounter in a real-life experiment [59].

In Figure 5.6 we see the very promising results for one example 7-bit 3-SAT instance with $m = 3n$ clauses in which MCTS is able to stay very close to the performance of the noiseless case, even for very strong noise. The two highest noise strengths are even larger than the gap minimum of the instance. We notice that in some cases, especially with high noise strengths and faster annealing times, MCTS can be fooled by the noise and fail to deliver the expected performance. We found that by restarting the MCTS search, we could find a game where it does well within 10 games in every case. In Appendix C we extend this noise-resilience analysis slightly by considering another type of noise.

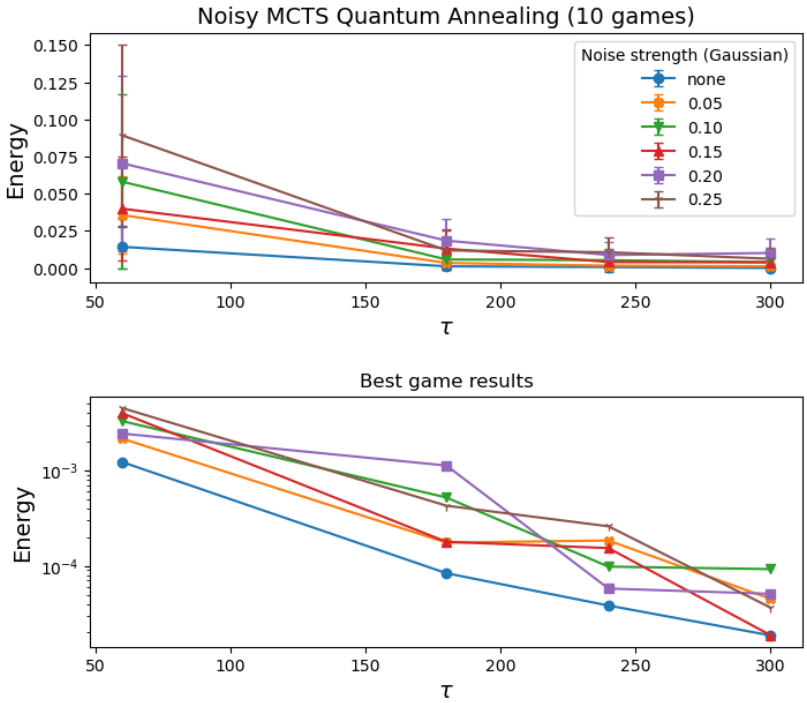


Figure 5.6: Resulting energies of MCTS in a noisy reward environment. (Top) average energies over 10 games with Gaussian noise for different annealing times and noise strengths. The annealing schedules that MCTS obtained came from information it obtained from the noisy landscape alone, and the energies in these figures are calculated with no noise with the paths that MCTS returned. (Bottom) minimum energies out of the 10 games. It is interesting that we could see that a gradient-based search could not converge even with the smallest noise strength considered in this figure. The energy gap is $\Delta = 0.18$ for the 3-SAT instance under consideration.

It would be very interesting to compare MCTS in a noisy environment with other gradient-free methods. This is left as a topic of future research.

Chapter 6

Quantum Approximate Optimization Algorithm (QAOA)

In this chapter we explore the remaining paradigm of near-term quantum algorithms by choosing the most relevant Variational Quantum Algorithm (VQA) for our use case. The Quantum Approximate Optimization Algorithm (QAOA), first proposed in [11], is a VQA that aims to provide approximate solutions to combinatorial optimization problems. It is one of the most promising algorithms to obtain some sort of quantum advantage on current quantum hardware [60]. We implement QAOA and see how our Monte Carlo Tree Search fares in this new context.

In Section 6.1 we go over what QAOA is and how it works. We then combine it with MCTS and see it in action for the toy model in Section 6.2. The remaining sections are devoted to doing the same for 3-SAT instances (Section 6.3), and the intriguing discussions and proposals that spring from those results (Section 6.4).

6.1 Theory

At its core, the Quantum Approximate Optimization Algorithm or QAOA is another way of obtaining the ground state of the target Hamiltonian H_z[1]. As we will see, it is curiously inspired by Quantum Annealing from Chapter 3.1; but, as a VQA, it is fundamentally a different kind of approach to the problem. Instead of adiabatically evolving the system according to some annealing path, QAOA is much more comparable to conventional optimization, where a cost function is minimized. In this case, a quantum device assists the optimizer

[1]We will keep using the same notation as before: H_z for the target Hamiltonian and H_x for the driving Hamiltonian. Keep in mind that in QAOA it is also usual to refer to them as the cost Hamiltonian H_C and the mixer Hamiltonian H_B, respectively.

with the evaluations of the cost function that are required in the minimization process.

Like in all VQAs, in QAOA we have a cost function that depends on some variational parameters $\boldsymbol{\theta}$. For QAOA the cost function corresponds to the expectation value of the target Hamiltonian that encodes our combinatorial optimization problem,

$$C(\boldsymbol{\theta}) = \langle \psi_0 | U^\dagger(\boldsymbol{\theta}) H_z U(\boldsymbol{\theta}) | \psi_0 \rangle \,, \tag{6.1}$$

with $|\psi_0\rangle$ being the initial state we defined in (3.4). The unitary Ansatz $U(\boldsymbol{\theta})$ is the parameterized quantum circuit that the classical subroutine needs to optimize by by adjusting its variational parameters. We now look at said unitary Ansatz for the case of QAOA.

The QAOA Ansatz

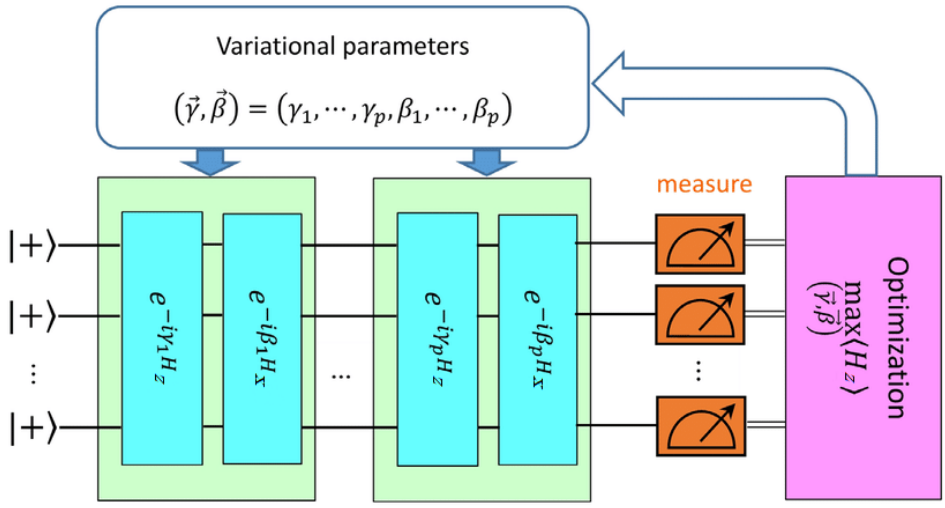


Figure 6.1: Schematic representation of QAOA. Figure adapted from [61].

The reason we are discussing QAOA only after Adiabatic Quantum Computing is that QAOA can be formulated in a way that is much more straightforward to grasp once one understands Quantum Annealing (QA). This is because the QAOA Ansatz $U(\boldsymbol{\theta})$ is heavily inspired by QA. The starting point to derive the QAOA Ansatz is the discretized QA time evolution operator (3.27), which was obtained by dividing the time into P segments wherein the time-dependent QA Hamiltonian (3.3) was taken to be constant. We rewrite the product of discretized unitaries here:

$$U_{\text{step}} = \prod_{m=1}^{\leftarrow P} e^{-iH(s_m)\Delta t_m}. \tag{6.2}$$

The only required further simplification to get to the QAOA unitary is to split the exponential into two as

$$e^{-iH(s_m)\Delta t_m} \to e^{-i(1-s_m)H_x\Delta t_m} e^{-is_m H_z \Delta t_m}, \tag{6.3}$$

where the error after this operation, known as Trotter-splitting, is of the order $\mathcal{O}(\Delta t_m^2)$ for each m [62]. In Appendix B we more carefully study the errors of going from the QA time evolution operator to the QAOA one. We may now define the so called QAOA parameters as

$$\beta_m = (1 - s_m)\Delta t_m, \qquad \gamma_m = s_m \Delta t_m, \tag{6.4}$$

such that we have two separate kinds of unitaries: one which contains the combinatorial optimization problem to be solved, which we adjust with the real parameter γ and the other which entangles all the spins, tuned by the other real parameter β. Thus, QAOA is nothing but the generalized version of Trotterized discretized QA, where we have more freedom in both parameters in the exponentials, as they are not bound to each other by the annealing path like in the case of QA. The unitary ansatz for a depth P takes the form

$$U(\boldsymbol{\gamma}, \boldsymbol{\beta}) = e^{-i\beta_P H_x} e^{-i\gamma_P H_z} ... e^{-i\beta_1 H_x} e^{-i\gamma_1 H_z}, \tag{6.5}$$

which lets us construct the parameterized state as

$$|\psi_P(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = U(\boldsymbol{\gamma}, \boldsymbol{\beta}) |\psi_0\rangle , \tag{6.6}$$

where $|\psi_0\rangle$ corresponds to the ground state of H_x, (3.4). With this we can already define the cost function for a depth P with respect to the $2P$ variational parameters $C(\boldsymbol{\theta}) = E_P(\boldsymbol{\gamma}, \boldsymbol{\beta})$ as

$$E_P(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \psi_P(\boldsymbol{\gamma}, \boldsymbol{\beta}) | H_z | \psi_P(\boldsymbol{\gamma}, \boldsymbol{\beta}) \rangle . \tag{6.7}$$

This Ansatz is known to be universal [60]. The goal in QAOA is to find the optimal set of QAOA parameters

$$(\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*) = \underset{(\boldsymbol{\gamma}, \boldsymbol{\beta})}{\arg\min}\, E_P(\boldsymbol{\gamma}, \boldsymbol{\beta}), \tag{6.8}$$

which is done with the help of a classical optimizer as depicted in Figure 6.1. As was mentioned in the introductory chapter, the real potential of this quantum/classical hybrid framework lies in the fact that the classical optimizer only deals with the optimization of $2P$ parameters, while only the quantum circuit deals with the immense 2^n dimensional Hilbert space. Nevertheless, the minimization in the $2P$-dimensional parameter space is a highly non-trivial task [8]. The existence of a large number of sub-optimal local minima and barren plateaus can make them very arduous to navigate [24]. For the most common gradient-based minimization approaches, the former makes it

extremely important to be able to find good initial parameters close to the global minimum [63, 64], so as not to get trapped in local minima, while the latter further hinders the job of the optimizer as it encounters flat regions in which the vanishing gradient is unable to guide the search.

Finding ways to traverse these complicated landscapes is thus a topic of great interest [65, 8, 5]. In the following section we test how the MCTS algorithm does in this context to see its strengths and shortcomings, and whether it could be a helpful tool to be used in this context.

6.2 MCTS-optimized QAOA on the toy model

We first very quickly test MCTS on the toy model (3.22) presented in Chapter 3.3, so as to get an idea of how the QAOA game functions. The task of MCTS is now to choose the QAOA parameters. The way we formulate the game for QAOA is the following: the game has $2P$ turns, at each odd turn the player must choose a value for the corresponding γ parameter, and the same happens for the parameter β in the even turns. Each parameter is chosen out of a selection of b discretized values. Every time the MCTS algorithm gets to the rollout stage (see Chapter 2.2), it chooses a set of parameters, at which point the initial state can be evolved according to them and the cost function evaluated. The reward obtained in the rollout can be related to the energy of the final state (6.7) by possibly manipulating it in one of the ways outlined in Appendix A.3. The search space will contain exactly b^{2P} leaf-nodes in this case. The scaling is worse than in the case of discretized QA, because now for every step we need to choose two parameters instead of one, which

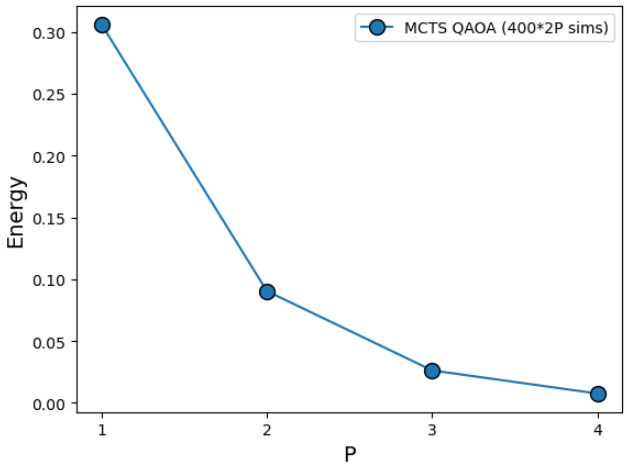


Figure 6.2: QAOA results on the toy model up to $P = 4$. With $b = 20$ values in $[0, 2\pi]$ for the parameters, for a similar search space to the one for QA in Chapter 3.3. With 400 MCTS cycles per turn for $P > 1$, and 50 for $P = 1$.

is the price to pay for the extra freedom of QAOA. In Figure 6.2 we can see how the MCTS algorithm did for some shallow depths with a less-than-ideal discretization[2].

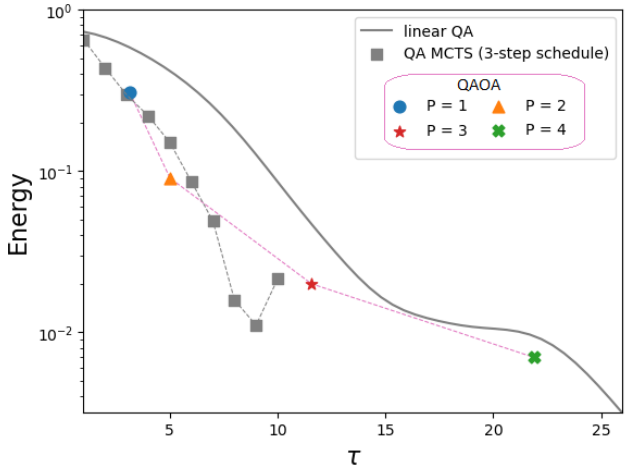


Figure 6.3: MCTS-optimized QAOA and QA results for the toy model. The equivalent discretized annealing times have been calculated with (6.9). This way we can compare the QAOA results to the equivalent τ regime for discretized QA for the same system. The solid gray line corresponds to the linear continuous annealing energy, whereas the gray squares are the MCTS-optimized discretized QA results, i.e., the ones in Fig. 3.6a. The $P = 1$ and $P = 2$ solutions seem to be on par with the QA results, as they are at or below the MCTS line. The other two, although we do not have MCTS QA data for those τ values (because the discretization error forbade us from exploring them), are likely not at the global minimum, as even if they are below the continuous annealing line, it seems that the energies could be lower if the QA results follow their trend. We explore the reason for this in later sections.

Since QA and QAOA are so closely related, there are ways to compare the results to those of QA. From the definition of the QAOA parameters (6.4) it follows that

$$\sum_{m=1}^{P}(\gamma_m + \beta_m) = \sum_{m=1}^{P} \Delta t_m = \tau \tag{6.9}$$

lets us define an equivalent annealing time[3] for a set of parameters $(\boldsymbol{\gamma}, \boldsymbol{\beta})$.

[2]Some values of the parameters are probably redundant for such a simple system because of symmetries, which we could exploit to eliminate the degeneracies in energy and have a finer discretization [64]. But we are simply using this problem to see that MCTS is able to play it before moving on to the case with 3-SAT, which is the one we care about.

[3]We could also define the corresponding discretized annealing schedule values: each step with a duration $\Delta t_m = \gamma_m + \beta_m$ and a value $s_m = \frac{\gamma_m}{\gamma_m + \beta_m}$. Of course, this will be true up to the Trotter error.

And in Figure 6.3 we plot the discretized QA and QAOA results for the toy model together. Those results show how much more complicated the game of QAOA gets with increasing P.

With these results on the toy model, we move on to applying our algorithm to 3-SAT problem instances.

6.3 MCTS-optimized QAOA for 3-SAT

Before showing the results for 3-SAT, there is an important note that we should be aware of: when dealing with 3-SAT instances, QAOA presents some reachability deficits [66]. What this means is that unless we choose an unreasonably high depth P, there will not exist a set of parameters that will evolve the system close to the ground state[4]. Specifically, with our particular clause densities, according to [66], we would need a QAOA circuit depth of $P \sim 45$ to be able to achieve an overlap $|\langle\Psi_0|\psi_P(\boldsymbol{\gamma},\boldsymbol{\beta})\rangle|^2 > 0.95$ in our 3-SAT problems. We will, naturally, not be going to such high depths. This is, however, not something we should be bothered by, because for hard optimization problems there is a lot of value to being able to efficiently find approximate solutions, such as a bit string that violates only a small number of clauses. In QAOA with a shallow depth, we will likely only be able to look for such solutions, and we will simply be testing how the MCTS algorithm works within the limitations of the algorithm for the case of 3-SAT.

Results

With that out of the way, we move onto the results of running MCTS as a way to optimize the parameters. We increase the branching factor from the last section to $b = 40$ like we did for the case of QA. We will also be comparing the MCTS results to a more traditional gradient-descent optimization method, specifically the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [67].

The results are shown in Figure 6.4. At first sight, the MCTS results seem quite unfortunate: while the $P = 1$ results converge to the optimal parameters, for $P = 2$ (where the size of the tree is comparable to that of the QA case) MCTS struggles to converge in some instances, and for anything beyond that, it is not able to find good results in the vast majority of cases because of the increase in difficulty.

In this section we display the results and comment on why they are so unfortunate, and in the next ones we explore the reasons on why this happened, which will let us provide some concrete alternatives and improvements to make MCTS work for QAOA.

[4] This is also the case with 2-SAT, because we are technically dealing with the optimization version of the k-satisfiability problems, which are NP-hard even for the case of $k = 2$.

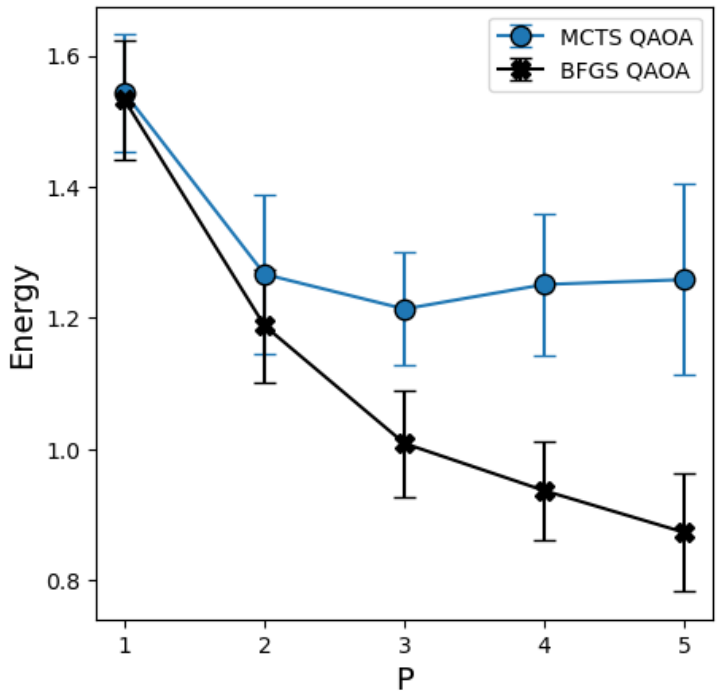
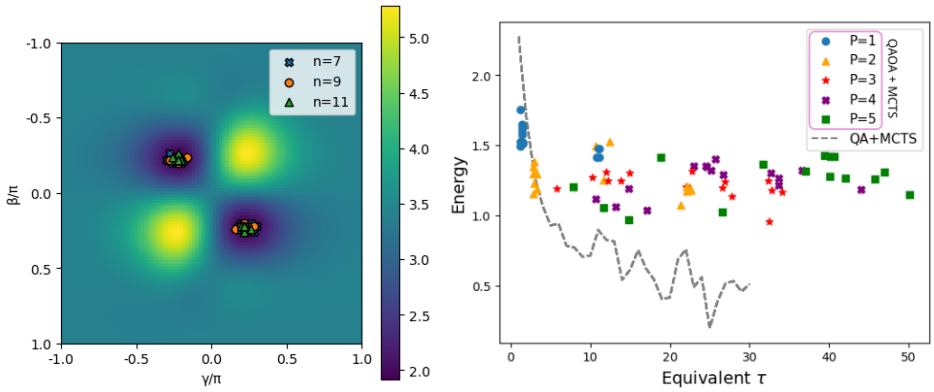


Figure 6.4: QAOA energies for MCTS-optimized QAOA (blue circles) and BFGS (black crosses) over 15 3-SAT instances. The MCTS results are averaged over 15 7-bit 3-SAT instances by choosing the best out of 10 MCTS games ($b = 40$, $2P$ turns) for each, while the BFGS search was run 1000 times from random initial parameters. The latter represents the probable global minima, especially for low depths, and works as a benchmark for MCTS, to see how close it gets to them. The size of the MCTS tree can be cut down in half because of the inversion symmetry of the QAOA parameters, which follows from the QAOA Ansatz. The number of MCTS cycles used is the same as for the toy model. The effects of the reachability problems for 3-SAT that we mentioned are apparent from the values of the energies we are working with.

The $P = 1$ MCTS results are good, but that is not surprising, as the tree size is small for $P = 1$. Furthermore, the $P = 1$ energy landscapes happen to be fairly simple, even for 3-SAT instances of increased size, as can be seen in Figure 6.5a; and, on top of that, the landscape does not change much from instance to instance, meaning that the task of finding the minimum should be very viable for most optimizers, as the initial parameters could be very straightforwardly selected. The case of $P \geq 2$, which is when the problem starts to get more interesting, is the point at which MCTS struggles. For one,



(a) (b)

Figure 6.5: (a) Average P=1 energy landscape for 45 3-SAT QAOA instances of 7, 9 and 11 bits. The markers represent the minima of each individual instance with the corresponding number of bits, with which the average was calculated. All of the minima are concentrated around the same area. The inversion symmetry is very clear from the figure too. (b) Comparison of the MCTS-optimized QA and QAOA results for 3-SAT. The equivalent annealing time for QAOA has been calculated using (6.9). The $P > 2$ results as well as some $P = 2$ ones can be seen to probably be stuck in local minima.

the size of the search-space increases exponentially[5], but most importantly, the energy landscape becomes increasingly complicated to navigate for the version of MCTS we are using. This can be confirmed from Figure 6.5b, where the $P = 1$ and some $P = 2$ solutions are seen to be probably at the global minimum, while the other ones are not.

This happens to the point that the MCTS energy gets worse as we increase the depth P, as can be seen in Figure 6.4. We know this should never be the case, as the results for P should be at least as good as the ones for $P - 1$ in every case[6].

One could think that the discretization of the parameters might be the source of the issue. The MCTS does in fact only work with a discrete set of the parameters, instead of the full continuous space like the BFGS algorithm. We found this not to be the case for the problems we are considering: a set of discretized parameters very close in energy to the BFGS results exists in every case. It does exists, but the MCTS is almost never able to find it for $P > 2$.

The most interesting part is that the MCTS results are still similar, with

[5]QAOA-MCTS scales worse than chess! With an average branching factor of 40 vs 35 for chess. See Chapter 2.1.

[6]This is because for some P, we can reach the previous $P - 1$ parameters by choosing $\gamma_p = \beta_p = 0$, making the P depth results always at least as good as the ones for $P - 1$.

only minimal improvements, as we increase the number of MCTS cycles beyond the number that we used to obtain the results. We explore why all of this happens and propose new methods to overcome the problems in the next section.

6.4 Proposed solutions

From seeing the ill-fated results of the last section, we should not just conclude that MCTS is not an appropriate tool to optimize the QAOA parameters. In this section we propose two ways in which MCTS can be used in QAOA, as well as trying to understand why it fails. Doing the latter successfully will also allow us to propose a set of concrete and powerful modifications to the MCTS algorithm which could very conceivably improve the results for QAOA, as well as the ones for QA, as they would make it much better suited to play quantum optimization games.

MCTS-initialized QAOA

Finding good initial parameters for a gradient-based search is very valuable in QAOA, as they can significantly reduce the number of required gradient searches before arriving to the global minimum. One of the ways of doing that is employing the parameters of the equivalent schedule from Trotterized linear QA with equal time steps, where one can vary the time step Δt to find different promising starting points [63]. Alternatively, one can obtain the initial parameters by interpolating the previous depth solutions for some problem instances [64].

MCTS, although it was not able to obtain good results by itself in the last section, has the clear advantage that it does not require any initialization, as the algorithm chooses what parts of the tree are most appropriate to explore by itself. It might therefore have some value as a tool to find said initial parameters. We have seen that very often, although MCTS obtains bad energies, the algorithm is not necessarily choosing parameters that are far from the optimal ones. Instead, they often seem to lie on the slope to the global minimum. This is especially true for $P = 2$. An example of this can be seen in Figure 6.6, where we used the parameters that the MCTS algorithm found as starting points for BFGS searches. These good results indicate that many MCTS results which seem bad might not actually be stuck in local minima.

Thus, although we saw that MCTS is not able to find the global minimum in most cases, because of how the algorithm functions, we can be assured that it did inevitably take the search to regions where the averages scores were good, so that good solutions exist; and, as such, it might very well have some

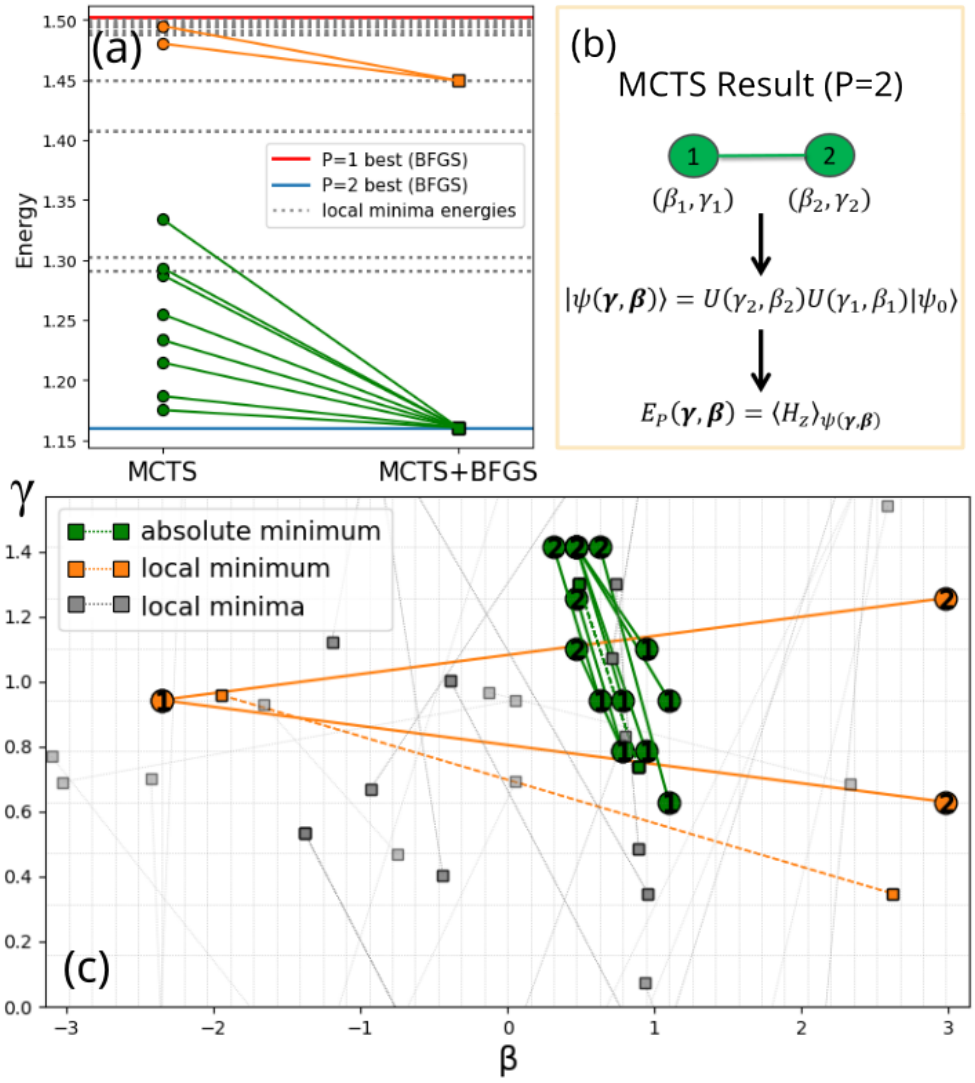


Figure 6.6: Example of MCTS-initialized BFGS to find the global minimum for $P = 2$. (a) shows the energies given by MCTS (left) and the energies given by one run of BFGS with the MCTS results as initial parameters (right). Despite the fact that some of the MCTS results can be seen to be closer to other local minima in energy (dotted gray lines), 8 of the 10 games converged to the global minimum, meaning that they were in its vicinity instead. (c) is a representation of part of the parameter-space for the example 3-SAT instance: each pair of dots is a result given by MCTS as explained in (b), which lie on one of the vertices of the grid on the background, which corresponds to the discretization of the parameters on the space of (c). The pairs of squares are the various minima found by the randomly initialized BFGS searches, which helps see the complexity of the landscape. All of the green pairs of dots in (c) converged to the global minimum after the BFGS search.

value as a way of finding initial parameters close to the optimal solution for a BFGS search.

MCTS with iterative search space updates

The second and most notable and promising way to use our MCTS algorithm for QAOA is to exploit the regularity of the QAOA optimal parameters. It has been discovered, for example in [64], that for a large number of combinatorial optimization problem instances, the parameters exhibited a remarkably predictable behavior: as the depth P was increased, the optimal γ_k, $k \in [1, P]$, increased monotonically with increasing k, while the β_k decreased in the same way. They also always occupied roughly the same range of values, even as P increased. Making use of this, in [64] the QAOA parameters were interpolated from the previous depth parameters according to said rules to create excellent sets of initial parameters for BFGS, which were already very close to the global minimum, so that the gradient search would just need to slightly shift the parameters from the initial values to find the minimum. This regularity was already visible in $P = 2$ and would persist up to fairly high P values.

Instead of using this to find good initial parameters for a gradient-based search, we can instead devise a way in which we utilize this to aid the MCTS search. With this in mind, we can propose the following adjustments: suppose we have the optimal parameters $\{\gamma_1, \beta_1, ..., \gamma_P, \beta_P\}$ for a depth P, and that we are working with a regular problem instance that behaves like the ones in [64], we can let MCTS search for the new $P+1$ parameters $\{\gamma'_1, \beta'_1, ..., \gamma'_P, \beta'_P\}$ in a mesh of b discrete values within

$$\gamma'_k \in [\gamma_{k-1}(1-\delta), \gamma_k(1+\delta)], \qquad \beta'_k \in [\beta_{k-1}(1-\delta), \beta_k(1+\delta)], \tag{6.10}$$

where $k \in [0, P+1]$. The values for β_0, γ_0, β_{P+1} and γ_{P+1} are some heuristically chosen values by looking at the known optimal parameters that limit the range of the new parameters, and δ is a constant that lets MCTS search beyond the bounds of the restricted regions.

By looking at the BFGS optimal-parameters, we were able to confirm that this regularity also seems to be present for many 3-SAT instances. We show the potential of this method in Figure 6.7, where we test it for an example instance which exhibits the aforementioned regularity of solutions, and we compare it to the BFGS and normal MCTS results. The new search-space-restricted MCTS is able to find good solutions with ease, that match and sometimes even improve upon the BFGS ones.

We also show explicitly the process of going from the MCTS+BFGS minimum of $P = 2$ to the result for $P = 3$ in Figure 6.8.

It is true that we might argue that using MCTS in this way does break one of the MCTS selling points that were mentioned in Chapter 2: its versatility. We are somewhat adding heuristics by only letting it look in some subspace of the space of parameters, instead of letting it freely explore the full search

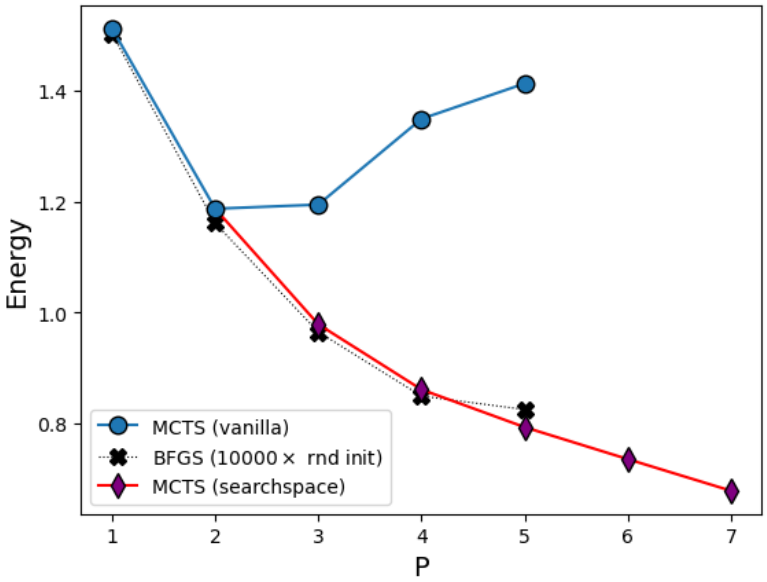


Figure 6.7: Comparison between ordinary MCTS (blue circles), BFGS with many random initializations (black crosses) and search space tuned MCTS (purple diamonds) for a 3-SAT instance with regular solutions. Starting from the $P = 2$ solution, each subsequent search space MCTS result is built with the previous one according to (6.10) with $\delta = 0.05$. The regularity of solutions holds remarkably well, and MCTS is able to navigate the new landscapes with ease, even improving upon the BFGS result for $P = 5$. The number of MCTS cycles only increased linearly with P, as we kept them fixed at 1000 per turn.

space. This means that within this searchspace restriction method MCTS will be very limited to attempt strategies that do not take advantage of the regularity. Although, since the regularity of QAOA is a very well-established phenomenon, it could be a very useful way of making use of it in cases where we know it will be there.

Even then, with the restricted searchspace MCTS we still have a tree with the same large branching factor and exponential growth b^{2P}, but now we are able to avoid many parts of the complicated $2P$-dimensional landscape and thus reduce the number of local minima. This has the added benefit that the resolution of the MCTS algorithm increases at every P, since the $b \times b$ values of each $\{\gamma_m, \beta_m\}$ are sampled in a progressively smaller region of the parameter space. Therefore, its accuracy should get increasingly closer to a continuous-space optimization algorithm at each consecutive deepening, which makes any discretization-related errors become less and less relevant as we iteratively increase the depth.

This confirms that the complexity of the landscape was the main issue which prevented MCTS from working for QAOA without restricting the

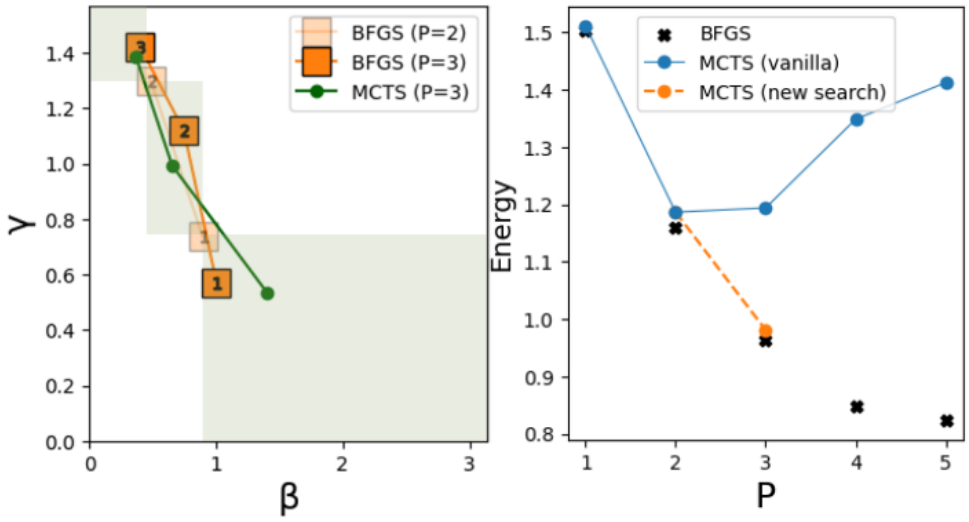


Figure 6.8: Example of searchspace-restricted MCTS. The parameter-space is on the left and the energies on the right. As it starts from the $P = 2$ minimum (transparent orange squares on the left), the algorithm looks within the shaded green region, which corresponds to the new reduced ($\beta_0 = \pi$, $\gamma_0 = \beta_3 = 0$, $\gamma_3 = \pi/2$, $\delta = 0$) search-space for $P = 3$. As can be seen on the right, the MCTS is able to escape the local minima and find a good energy this way.

searchspace, as opposed to the large size of the tree, which MCTS is known to be able to deal with appropriately.

Discussion on an improved MCTS algorithm

As we mentioned in Chapter 2, an MCTS algorithm is characterized by its search and playout policies, as well as other details related to the final choice criterion and to the treatment of the tree from previous turns. This makes it very convenient to come up with new versions of MCTS that are specifically designed for the task at hand.

We already found two possibly interesting uses for our algorithm in the context of QAOA, with no modifications[7]. However, if we are able understand the mechanism that impedes MCTS from choosing the right path down the tree, we could propose modified versions of MCTS which might be more apt to tackle QAOA with no search space restrictions or gradient-based corrections.

By testing MCTS with custom smaller versions of QAOA trees, we were able to find what is most likely the issue that MCTS encounters. It has to do with the combination of having a complex landscape and it being a single-

[7]We do not consider restricting the search space as a modification to the algorithm, as its policies and selection criteria remain unchanged. It is the exact same algorithm, the only difference being that we show it only some parts of the parameter space, and hide others.

player game. The former is a fairly self-explanatory reason for MCTS to struggle, but the latter implies a very important distinction: in a single-player game we are looking for the absolutely best path down the tree, while, for a multi-player game, we are looking for the path that is as good as possible for us, but as bad as possible for the opponents, no matter what their chosen actions may be. Now, if we consider a case with a complicated reward landscape, where a very well scoring path is hidden among many bad paths, in the case of a multi-player game, that path is likely resulting from one or more players making very foolish moves. But in the single-player game there is no opponent, and that very path could be the one we are looking for. Because of the averaging of the rewards, those kinds of paths can be obscured from the MCTS algorithm, making it choose paths that lead to more mediocre, but on average better, regions of the parameter space than the hidden path. While on the case of a multi-player game, if the average obscures that path, it often ends up being beneficial for the algorithm instead.

In view of this, we propose three modifications to the algorithm, specifically to its selection policy, tree recycling and the final choice criterion. These are summarized in Appendix A.5. This is still something that we are working on, so we currently have no definite answer on how much this can improve the algorithm for quantum optimization. As some preliminary tests, we show the results of MCTS modified with modifications 1 and 2 from Appendix A.5 on the toy model: the modified MCTS (*"MCTSv2"* in the figures) optimizes discretized annealing paths with $P = 3$ steps in Figure 6.9, and the QAOA parameters in Figure 6.10. We compare the results to our previous MCTS algorithm, and, in both cases, the improvements are substantial.

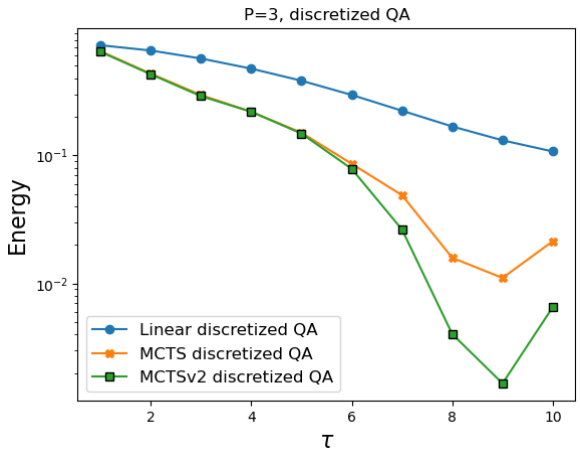


Figure 6.9: QA on the toy model, with partly modified MCTS (Modifications 1,2 in Appendix A.5). It is able to improve the energy over standard MCTS in every single case. Again, at $\tau = 10$ the error from the discretization makes it so that the energy is worse than the previous case. Compare with Fig. 3.6a.

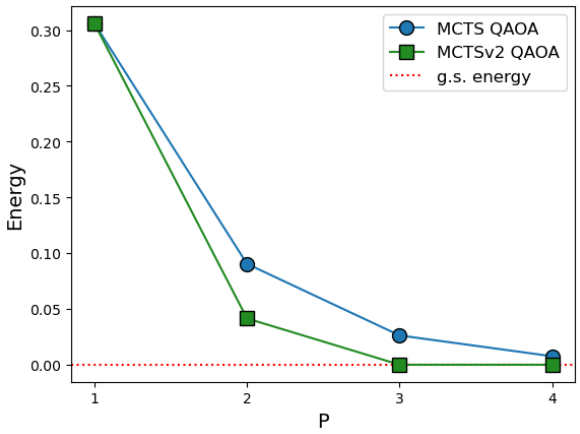


Figure 6.10: QAOA on the toy model, with partly modified MCTS (Modifications 1,2 in Appendix A.5). It is able to find the exact ground state of the system for $P > 2$, in spite of the rough discretization used. Compare with Fig. 6.2.

This is not to say that, because we knew our games would be single-player games, we should have expected our MCTS to not work from the beginning. In fact, equivalent MCTS algorithms to our unmodified one have been found to work very well in some single-player games too [68, 69]. And we even saw that it worked well for the case of QA in Chapter 5.2. This behavior of MCTS is not something that has been extensively studied. That is certainly the case in physics, where, to the best of our knowledge, both papers where MCTS is used similarly in the context of quantum optimization, [57, 59], utilize the standard MCTS algorithm for their single-player games. It is in the field of computer science that one can find a handful of publications where the authors have run into similar problems with their algorithms for single-player card games and puzzle games, and suggested solutions similar to ours [70, 71]. The promising prospect of fully implementing our proposed modifications and determining how much they improve the performance of the algorithm is left as a topic of future research.

Chapter 7

Conclusion and future directions

In this thesis we have explored two of the main existing computational frameworks to solve hard combinatorial optimization problems on near-term quantum hardware. In view of the difficulties that hinder these kinds of algorithms, the main task of the thesis was to investigate the validity of the Monte Carlo Tree Search (MCTS) algorithm, a decision making algorithm especially distinguished in the world of board game artificial-intelligence creation with many recent accomplishments, in the context of quantum optimization.

We focused on both the adiabatic quantum computational paradigm with quantum annealing (QA), and the hybrid quantum/classical variational one with the Quantum Approximate Optimization Algorithm (QAOA). The goal of both algorithms is to find the ground state of a target Hamiltonian, in which the solution of the classical problem instance is encoded. To that end, we formulated the optimization task as a single-player board game, in which the MCTS player is tasked with finding a set of parameters that maximizes the score: in the case of QA, the game consisted in building annealing schedules $f(s)$, functions that dictate the evolution of the interpolated system; whereas, in the case of QAOA, the game involved finding the QAOA parameters $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ with which to build the parameterized unitary Ansatz.

For the case of QA, we found that MCTS was able to very skillfully construct annealing schedules for solving hard 3-SAT problem instances. Remarkably, MCTS was able to find annealing schedules that either slowed down near the delicate gap minimum, or opt for non-adiabatic schedules and surf the excited states in search of an approximately good result. All of this was done with no knowledge of the location of the minimum gap or the form of the instantaneous energy eigenvalues. Furthermore, MCTS showed signs of being particularly well suited for noisy environments and could find good results in places where a more traditional gradient-based searcher would be completely helpless. We were able to achieve acceptable results with a fairly low num-

ber of 200 MCTS cycles per turn, which with our implementation of MCTS corresponds to exactly 200 function evaluations.

The remaining sections of this work were dedicated to the study of MCTS applied to QAOA. MCTS had never been used before in QAOA to directly optimize the parameters[1]. This case turned out to be significantly more complicated for MCTS than QA: with increasing depth P, our implementation of MCTS would struggle tremendously to optimally navigate the increasingly complicated landscape and would appear to only be viable in the most shallow depths. But it was precisely in trying to understand and looking for the sources of these complications that the most compelling findings of this work emerged. We were able to propose two ways in which MCTS can help with QAOA. Firstly, we saw that it might have some potential as a method of obtaining the initial parameters for a subsequent gradient-based search, as many of the results of MCTS with not-so-good energies were in close proximity to the global minimum in the parameters themselves. Secondly, we found a way to make use of the well-established regularity of the optimal parameters in QAOA, which we were able to observe in many 3-SAT instances, to iteratively build solutions for increasing depths by restricting the search of the MCTS algorithm to the relevant regions in the space of the parameters, with finer and finer discretization. Our preliminary tests on this purely MCTS iterative restricting of the search space gave very good results and warrant further study.

We went further and tried to understand the reasons and mechanisms that make MCTS fail in the case of QAOA. By testing the algorithm in custom trees where the particular landscape complexity from QAOA was present, we were able to come up with a set of concrete modifications to our MCTS which might very well allow it to optimize QAOA parameters for non-regular problem instances and possibly even significantly improve the results for the regular ones, as well as the ones from QA, especially in cases where the traditional implementation of MCTS struggled the most. For this, it was especially helpful to read the works of researchers working in fields other than physics, especially computer scientists, who have struggled with MCTS for single-player search spaces for much longer, and the problems of whom could have some striking similarities to our own, even though their uses are completely different to ours. In general, great benefits could be obtained from the interplay with computer scientists, and an algorithm that someone has coded for an obscure card game could happen to be a truly useful source of inspiration to improve our algorithms in physics-related contexts.

[1] In [59], which was published while the work on this thesis was ongoing, MCTS is used in the context of QAOA. However, it is used for choosing the Hamiltonians from a discrete set to build custom generalized QAOA Ansatze, while the parameter optimization is still done with gradient-based methods. This makes both approaches fundamentally different.

Apart from those proposals, there are other very natural ways to continue or expand upon this work. For one, MCTS is known to function very well together with neural networks, which could improve our results substantially [57]. The noise resistance of the algorithm for both computational paradigms is likewise something that would be interesting to study further. Furthermore, to see how our algorithm does in a real setting, testing MCTS-based optimization on actual quantum hardware could also be an attractive thing to do, as well as trying it out for more algorithms and more kinds of hard problems.

It is entirely possible that NISQ era algorithms will not be able to give us real practical advantages over classical devices. It could very well be that one day it will be proven that no general quantum advantage can ever be achieved within the limitations of current quantum devices. However, even if that turns out to be the case, having spent time with this would not be a waste of time. It is still deeply meaningful to try and make advances in improving NISQ era algorithms, as well as improving existing hardware and finding problems which could be appropriate for solving in current quantum devices, as the advances we make in these areas will give us a better understanding of QC in general and even be useful in later fault-tolerant eras.

Appendix A

Deeper dive into MCTS

A.1 More tic-tac-toe benchmarks

Figure A.1 shows the remaining tic-tac-toe results for MCTS that complement the discussion of Chapter 2.3. To see that our MCTS [37] is playing tic-tac-toe appropriately, apart from looking at the results it obtains in games, we also want to look at the particular decisions it makes in different types of positions. Specifically, these are tests where MCTS chooses the move it would play in a given state of the game for varying numbers of cycles. These are different kind of positions, where the player needs to either defend against threats or execute its own threats.

We found that, overall, these results are very much comparable to other implementations of MCTS.

A.2 About the final choice

Once the MCTS has run for the desired amount of cycles, we are left with a tree in memory that contains the nodes with some score and visits values. It is at that point that the MCTS must choose one move to make on the actual game board.

There are many ways to choose such moves [34]. It turns out that choosing the child that maximizes the UCB score is not a good choice. This makes sense as the UCB score tells us which move would minimize our regret when exploring, not necessarily the best move. Throughout the work we have been using the *max_child* criterion to choose said definitive move, as it turned out to be the optimal one as can be seen in Figures A.2 and A.3. The *max_child* criterion consists on choosing the child that maximizes the first term, i.e., the average reward of the UCB formula.

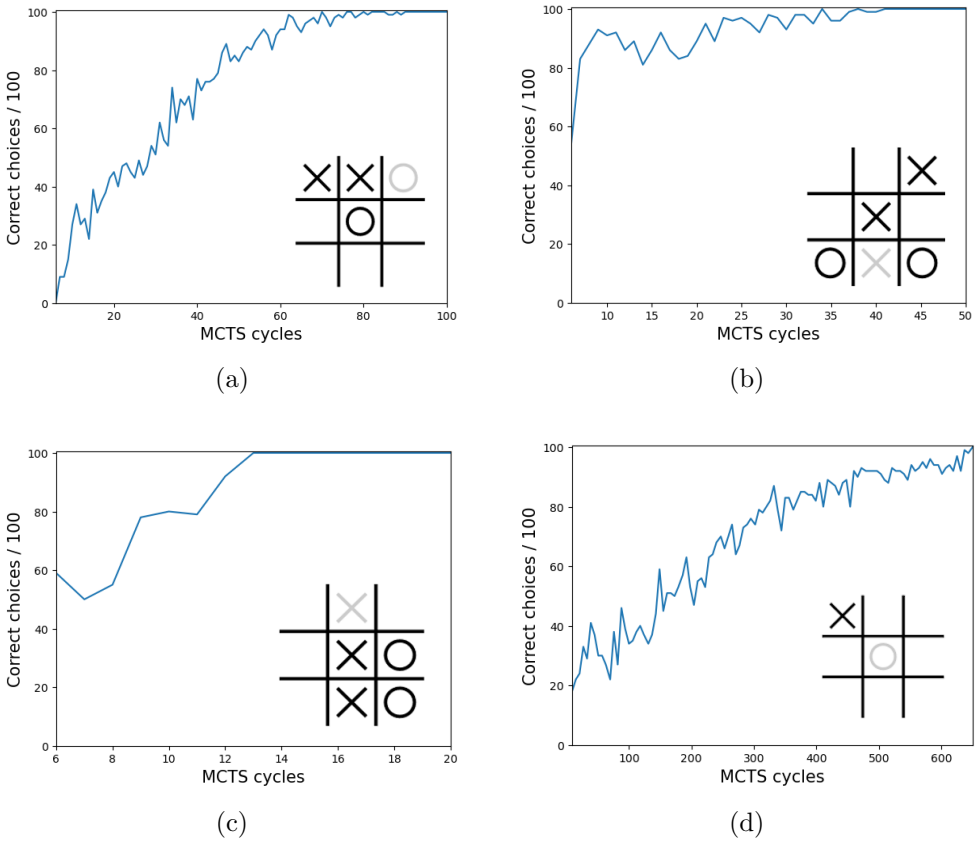


Figure A.1: MCTS chooses a move from given tic-tac-toe positions. We give MCTS a board state (see the diagram in each figure) and see if it as able to find the only correct move in the position (the gray move in the corresponding diagram). In (a) and (b) MCTS needs to defend against the opponent's threat of winning. Since, in theory, the game has many ways it can continue even after MCTS fails these tests, especially if playing against an imperfect opponent, this could be a position that MCTS might have had some difficulty evaluating correctly, but it is eventually always able to find the best move consistently. (c) shows that MCTS is quickly able to identify and play a winning move if there is one in the position, as it is able to consistently choose that move with a very low number of simulations. This makes sense because as soon as the winning move is expanded into the tree in memory, it will always have a score at least as good as every one of its sibling nodes. The most challenging test comes in (d): when the first player plays the initial move in a corner, there is only one move that does not lose by force against perfect play. MCTS, as expected, needs a higher number of cycles to identify this. We see that, in general, the more moves there are in the position, i.e., the larger the tree is below the root node, the more cycles MCTS requires to achieve optimal play. $C = \sqrt{2}$ and the *max_child* last choice were used for all tests.

There is another very used criterion along with *max_child* known as *robust_child*, which chooses the child that received the maximum number of visits during the simulation. This, although not as intuitive as *max_child* is supposed to be the often preferred criterion, although *max_child* always seemed to do better in the case of quantum optimization.

There may, however, be some value in exploring other last choice criteria as they could be more suitable for one player games. One such criterion could be choosing the child from which the best random simulation was run. This would not make sense for two-player games, but might be specially relevant for one-player games that we are dealing with where there is no opponent and the best scoring path might be very hidden for particularly hard problem instances and set ups.

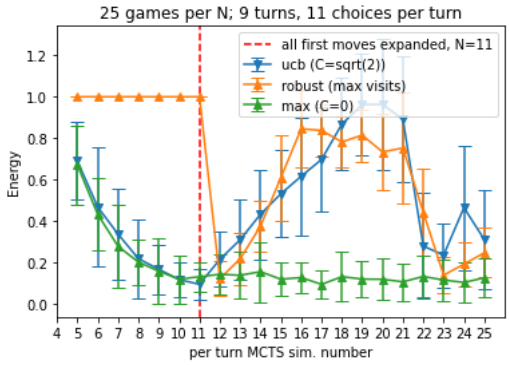


Figure A.2: Comparison between three different criteria for the last choices: *ucb* chooses the child as it would in the selection stage, *robust* chooses the child which has been visited the most, *max* chooses the child with the maximum average score, i.e., the first term in (2.1). The dashed vertical line represents the simulation at which every legal move from the root position has been added to the tree in memory. Before that, not all legal moves from the root node were visible to the MCTS algorithm.

A.3 Rewards in quantum optimization games

The rewards for tic-tac-toe in Chapter 2.3 were simple. Just win, tie or lose. However, that is not so with quantum optimization games, what should we do when we have a a score, an energy ε, as the reward? We thought of several options:

- Since 0 is the best energy, the most straightforward way is just to give $\omega = -\varepsilon$ as the reward.

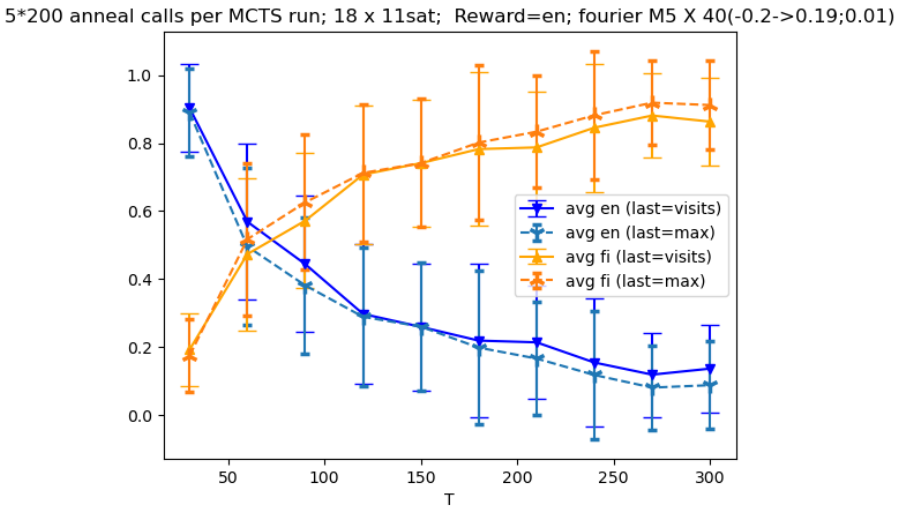


Figure A.3: Comparison between different last choices for MCTS-QA averaged over the 18 instances of 11 bit 3-SAT problems vs annealing time. The max child always seems to do at least as good as robust child.

- Normalize it between 0 and 1. This is not strictly necessary, but we do it in most cases because it gives a guarantee on the bound of the regret, and the theoretical value of $C = \sqrt{2}$ [36].
  - We can do this linearly: we know that for 3-SAT the maximum eigenvalue of H_z, i.e., the worst possible score one can get is the number of clauses αn. We can then simply map the scores linearly as $[\alpha n, 0] \to [0, 1]$ by dividing every score by αn, and then subtracting it from 1.
  - We can also do it with another non linear mapping, such as $\omega = e^{-a\varepsilon}$. This has the added benefit that, by adjusting a, MCTS will be able to understand that not all rewards have the same value. An improvement among high rewards is more rewarded than one among bad ones.

In most cases we will use the third option throughout this work with $a = 2$, which we decided on after trying many different values.

A.4 Tree recycling: why and why not

Another very pertinent question is deciding what to do with the tree once the last choice has been made and the turn has finished. One might initially think that saving and reusing the tree is the best solution. This can be done by pruning all other branches and making the chosen move the new root node.

This way, all of the simulation information of the memory tree below the chosen node remains. However, this could very well be detrimental; having more information is not always good. The fact is that starting from a tree in which simulations have already been run makes it more difficult to change MCTS's mind into choosing a different path than the previously expanded tree might suggest. The simulations in the new turn will have more value, as they start from a deeper level and thus more accurately simulate the scores of the nodes. The problem is that simulations from previous turns make the algorithm much more confident on the values that it has started with, and the new important simulations will not as easily change this, as lower quality simulations are clouding its judgment. Therefore, especially if the first turn choice was not optimal, the MCTS will do worse than if the tree was thrown away entirely. In our cases, only a minimal difference was noticed between recycling and throwing away the tree. In general, we found that the average score over a high number of MCTS games would be slightly higher when throwing away the tree, but so would be the standard deviation of the scores.

It would be highly interesting, and maybe even a possible topic for a Bachelor's thesis in computer science, to explore solutions to this issue. One idea could be making simulations from later turns have a higher weight in calculating the average: an MCTS backpropagation in turn m would increase the visits along the route of ancestor nodes by m instead of 1, similarly counting the obtained reward m times.

A.5 On possible MCTS modifications

The problems that the MCTS algorithm encounters with QAOA have to do with the single-player nature of the game combined with the highly deep global minima, surrounded by significantly worse scoring nodes. To combat this, we propose a set of single-player oriented modifications:

- Use the maximum simulation reward as the criterion of choosing the definitive move to play, instead of the more common *max_child* or *robust_child* criteria, which we will call *max_reward*.
- We do not want a random playout that finds the best path to just be used to calculate an average and go to waste. Because of this, we also fully expand the path that the maximum simulation took, which makes sure the path will never be forgotten.
- By changing the final choice to *max_reward*, we do not change how the MCTS cycle itself plays out. This means that the algorithm will think that paths which give good results on average are the most promising to do random playouts from. We can add an extra term $\sqrt{\sigma_i^2 + D/n_i}$, where $\sigma_i^2 = \frac{\sum w^2 - \bar{w}_i n_i}{n_i}$ and D is a large constant, so that, during the

selection policy, the algorithm might understand that paths which have been seen to deviate highly from the mean are also to be considered. This is because we are looking for the one best path in a one player game, and we want the algorithm to also weigh in that consideration in the selection stage. The form of the new term comes from the one proposed in [70], but there are potentially other terms that we could use, even ones that utilize the current maximum reward.

The effectiveness of these modifications is still something that we are studying. But they work very well in the preliminary tests that we have run, and there is reason to believe that with these modifications the QA results will also improve. The $\tau = 10$ game in Fig. 3.6a, for example, for which the MCTS struggled, can be easily overcome by an MCTS algorithm with just the first two modifications on this list to find an improved energy. These modifications also give us the absolute certainty that the increasing of MCTS cycles will always improve the choice of MCTS, with no exceptions.

Appendix B

QA to QAOA: what is lost on the way

B.1 Discretizing the annealing schedule

The procedure to set bounds on the errors is adapted from the one in [52]. The differences are that here we make the case more general which allows us to analyze non-linear schedules, and we provide a much more complete proof of **Lemma 1** of their paper.

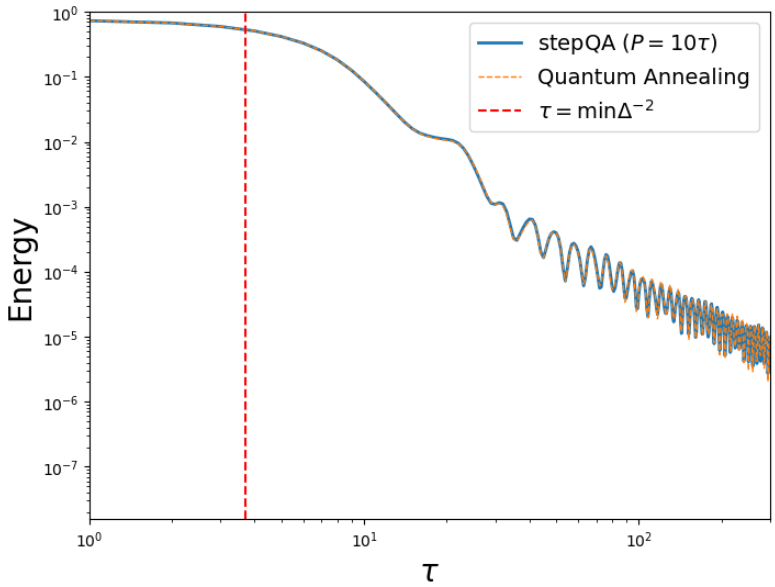


Figure B.1: Linear QA schedule and a properly discretized linear QA schedule such that the error is kept under control at all times.

General proof

Suppose we have two Hamiltonians $H_1(t)$ and $H_2(t)$, with their corresponding time evolution operators $U_1(t)$ and $U_2(t)$. If we evolve an initial state $|\psi(0)\rangle$ with each operator, for a total time τ, we define

$$|\psi_1(\tau)\rangle = U_1(\tau)\,|\psi(0)\rangle\,, \qquad |\psi_2(\tau)\rangle = U_2(\tau)\,|\psi(0)\rangle\,. \tag{B.1}$$

We will assume that the operator norm (B.22) of the difference between the Hamiltonians is at all times t at most as big as a function $\delta(t)$

$$||H_1(t) - H_2(t)|| \leq \delta(t). \tag{B.2}$$

The idea is to see how different both states can become after the evolution. Let us look at the overlap $\langle\psi_1(t)|\psi_2(t)\rangle$, by differentiating it and making use of the corresponding Schrödinger equations we find

$$\frac{d}{dt}\,\langle\psi_2(t)|\psi_1(t)\rangle = -i\,\langle\psi_2(t)|(H_1(t) - H_2(t))|\psi_1(t)\rangle\,. \tag{B.3}$$

Defining $\dot{u}(t) \equiv \frac{d}{dt}\,\langle\psi_2(t)|\psi_1(t)\rangle$, $\Delta H(t) \equiv H_1(t) - H_2(t)$, we integrate both sides from an initial time $t = 0$, where the states where the same $\langle\psi(0)|\psi(0)\rangle = 1$, to the total evolution time τ, and take the modulus on both sides:

$$|u(\tau) - 1| = \left|\int_0^\tau dt\,\langle\psi_2(t)|\Delta H(t)|\psi_1(t)\rangle\right|. \tag{B.4}$$

We look at the right-hand side first: we may bound it from above like so

$$\left|\int_0^\tau dt\,\langle\psi_2(t)|\Delta H(t)|\psi_1(t)\rangle\right| \leq \int_0^\tau dt\,|\langle\psi_2(t)|\Delta H(t)|\psi_1(t)\rangle| \leq \int_0^\tau dt\delta(t), \tag{B.5}$$

where we have made use of the fact that the integrand can be at most as big as $\delta(t)$ at all times. We now bound the left-hand side from below as

$$|u(\tau) - 1| \geq ||u(\tau)| - |1|| = 1 - |u(\tau)|. \tag{B.6}$$

The last step follows from the fact that $|u(\tau)| \leq 1$ for all times by definition. We have thus arrived at the very useful expression

$$|\,\langle\psi_2(\tau)|\psi_1(\tau)\rangle\,| \geq 1 - \int_0^\tau dt\delta(t), \tag{B.7}$$

which limits how much the modulus of the overlap between the two states can decrease by. From this last expression it follows that a bound exists for $||\,|\psi_1(\tau) - \psi_2(\tau)\rangle\,||$. By using (B.1) we find that the square of the operator norm is

$$||U_1(\tau) - U_2(\tau)||^2 \equiv \langle\psi(0)|(U_1^\dagger(\tau) - U_2^\dagger(\tau))(U_1(\tau) - U_2(\tau))|\psi(0)\rangle\,, \tag{B.8}$$

which after making use of the unitarity of the time evolution operators, the fact that the initial state is normalized, and inequality (B.7), brings us to

$$||U_1(\tau) - U_2(\tau)|| \leq \sqrt{2 \int_0^\tau dt\delta(t)}. \tag{B.9}$$

This way we have found that if we can limit the operator norm of $\Delta H(t)$, we can set an upper bound on how much the evolved states can deviate from each other.

The case of QA

We do this for the case where $H_1(t) = H(t) = (1 - s(t))H_x + s(t)H_z$ is the interpolated Hamiltonian from continuous QA, and $H_2(t) = H_{\text{step}}(t) = (1 - s(t_i)))\, H_x + s(t_i)H_z$, where, $t_i = \lceil \frac{t}{\Delta t} \rceil \Delta t$ its discretized version[1]. We choose that the highest value of the schedule is used as s_m, so that the difference between the Hamiltonians gives the worst case scenario. We now try to set a bound on the size of their difference, in order to use the result proven in the last section.

$$H(t) - H_{\text{step}}(t) = \left(s\left(\left\lceil \frac{t}{\Delta t} \right\rceil \Delta t \right) - s(t) \right) (H_x - H_z), \tag{B.10}$$

which, as long as $s(t)$ does not vary too much within Δt, we can approximate by its first order expansion around the point $\lceil \frac{t}{\Delta t} \rceil = \frac{t_i}{\Delta t} = \frac{t}{\Delta t}$, as

$$s\left(\left\lceil \frac{t}{\Delta t} \right\rceil \Delta t \right) \sim s(t) + \dot{s}(t) \left(\left\lceil \frac{t}{\Delta t} \right\rceil - \frac{t}{\Delta t} \right) \Delta t, \tag{B.11}$$

which, after inserting into (B.10), gives

$$H(t) - H_{\text{step}}(t) = \dot{s}(t) \left(\left\lceil \frac{t}{\Delta t} \right\rceil - \frac{t}{\Delta t} \right) \Delta t (H_x - H_z) \leq \dot{s}(t) \Delta t (H_x - H_z), \tag{B.12}$$

because the ceiling of a number and the number can differ at most by 1. The operator norm of the difference is

$$||(H(t) - H_{\text{step}})|| \leq \delta(t) = \dot{s}(t)\Delta t ||(H_x - H_z)|| = \dot{s}(t)\Delta t \times \text{poly}(n). \tag{B.13}$$

We made the very reasonable assumption that $||(H_x - H_z)||$ is a polynomial function on its size $\text{poly}(n)$, which always holds by construction.

At this point we may bound the quantity $||U(\tau) - U_{\text{step}}(\tau)||$ as

$$||U(\tau) - U_{\text{step}}(\tau)|| \leq \sqrt{2 \int_0^\tau dt \dot{s}(t) \Delta t\, \text{poly}(n)}. \tag{B.14}$$

[1]The ceiling function is defined as $\lceil x \rceil = \min\{m \in \mathbb{Z} | m \geq x\}$

If all time steps are of equal size, $\Delta t = \tau/P$ and $s(0) = 0$, $s(\tau) = 1$,

$$||U(\tau) - U_{\text{step}}(\tau)|| \leq \sqrt{\frac{2\tau}{P}\,\text{poly}(n)}. \tag{B.15}$$

This is the exact same answer as if we had considered a linear schedule to begin with, since in that case, we would have

$$H_{\text{step}}(t) = \left(1 - \frac{1}{P}\left\lceil\frac{Pt}{\tau}\right\rceil\right) H_x + \frac{1}{P}\left\lceil\frac{Pt}{\tau}\right\rceil H_z, \tag{B.16}$$

so that, $H(t) - H_{\text{step}}(t) \leq \frac{1}{M}(H_x - H_z)$, and the integration would give the extra missing τ factor, agreeing with [52]. The difference is that in this case s was already linear in time so that we did not need to approximate it.

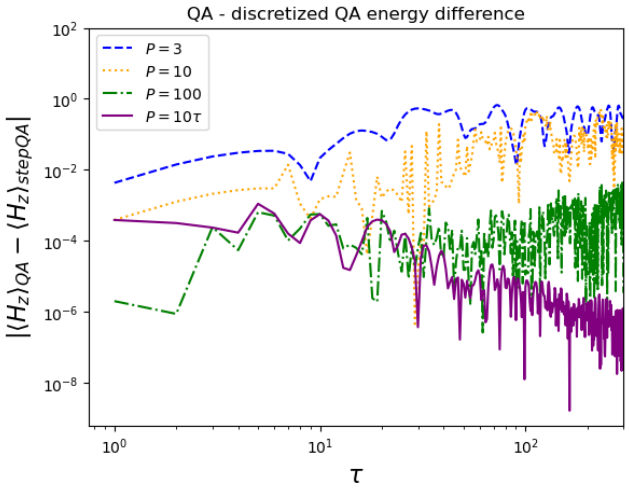


Figure B.2: Calculated error for the linear schedule for the toy model. The orange and purple agree at $\tau = 1$, and the green and purple at $\tau = 10$, as expected. The purple line goes down as it needs to be better than the others once it has a higher P, which could be derived from our upper bound estimate.

A look at $||H_x - H_z||$

By the triangle inequality we have

$$||H_x - H_z|| \leq ||H_x|| + ||H_z||. \tag{B.17}$$

For the case of 3-SAT, the maximum eigenvalue of H_z is the number of clauses αn (because that is the energy of the worst possible bit assignment), whereas for the toy problem we have that it is exactly 1. The maximum eigenvalue of H_x is instead bounded by n, which can be seen by applying the triangle inequality n times.

Therefore, we find that $||U(\tau) - U_{\text{step}}(\tau)|| \leq \sqrt{\frac{2\tau\alpha(n+1)}{P}}$ for a general 3-SAT instance; and, for the toy model with $n = 2$, $||U(\tau) - U_{\text{step}}(\tau)|| \leq \sqrt{\frac{6\tau}{P}}$, although we can reduce the upper bound by a factor $\frac{1}{\sqrt{2}}$ as we know the exact upper bound for $||H_z||$ to be 1.

Figures B.1, B.2 where created with these results in mind. We see the confirmation that the upper bounds are valid, and that we can keep the error under control at all times by scaling P as $\tau\,\text{poly}(n)$.

B.2 Trotter splitting the time evolution operator

To arrive at the QAOA time evolution operator $U_{\text{Trotter}}(\gamma, \beta)$, it is sufficient to Trotter-split the individual exponentials in $U_{\text{step}}(t)$ that we dealt with in the last section. We look at the Trotterization of the mth exponential: $U^m_{\text{step}} = e^{-i(1-s_m)\Delta t_m H_x - i s_m \Delta t_m H_z} \rightarrow U^m_{\text{Trotter}} = e^{-i(1-s_m)\Delta t_m H_x} e^{-i s_m \Delta t_m H_z}$.

The size of the error will be given to leading order by the Baker-Campbell-Hausdorff formula [72] to be

$$||U'_j - U''_j|| = \frac{\Delta t_m^2}{2} s_j(1 - s_j)||[H_i, H_f]|| + \mathcal{O}(\Delta t)^2. \tag{B.18}$$

Since the operator norm satisfies the triangle inequality, we can say that

$$||[H_i, H_f]|| \leq ||H_i H_f|| + ||H_f H_i|| \leq 2||H_i|| \cdot ||H_f|| = 2\alpha n^2 \tag{B.19}$$

gives us an upper bound on the norm. With the other factors, assuming a linear schedule we arrive at the conclusion that the norm of the difference must be

$$||U'(\tau) - U''(\tau)|| \in \mathcal{O}\left(\frac{2\alpha n^2 \tau^2}{P^2}\right). \tag{B.20}$$

By comparing it directly to the continuous QA evolution [52]:

$$\left|\left|U(\tau) - \prod_{m=1}^{P} U^m_{\text{Trotter}}(\tau)\right|\right| \in \mathcal{O}\left(\frac{\tau^2\,\text{poly(n)}}{P}\right). \tag{B.21}$$

Since this tells us that we need a circuit depth that increases polynomially with the system size and annealing time, this is a way of proving the universality of QAOA, by starting from the universality of QA [39].

B.3 Other proofs

Definition of operator norm

Let A be an operator. We represent its operator norm as $||A||$,

$$||A|| \equiv \sup_{|\psi\rangle \neq 0} \frac{||A\,|\psi\rangle\,||}{||\,|\psi\rangle\,||}, \tag{B.22}$$

where $||v||$, with v a vector (such as $|\psi\rangle$), is the usual norm $||\,|\psi\rangle\,||^2 \equiv \langle\psi|\psi\rangle$.

Proof operator norm is submultiplicative

$$\begin{aligned} ||AB|| &= \sup_{|\psi\rangle \neq 0} \frac{||AB\,|\psi\rangle\,||}{||\,|\psi\rangle\,||} = \sup_{|\psi\rangle \neq 0} \frac{||AB\,|\psi\rangle\,||}{||B\,|\psi\rangle\,||} \frac{||B\,|\psi\rangle\,||}{||\,|\psi\rangle\,||} \\ &\leq \sup_{|B\psi\rangle \neq 0} \frac{||AB\,|\psi\rangle\,||}{||B\,|\psi\rangle\,||} \sup_{|\psi\rangle \neq 0} \frac{||B\,|\psi\rangle\,||}{||\,|\psi\rangle\,||} = ||A|| \cdot ||B||. \end{aligned} \tag{B.23}$$

Proof operator norm sets a bound on eigenvalues

As we saw in equation (B.22), we can define the equivalent to the $L2$ norm for vectors for an operator A in the Hilbert space $\mathcal{H}$ as

$$||A|| = \sup_{|\psi\rangle \neq 0} \frac{||A\,|\psi\rangle\,||}{||\,|\psi\rangle\,||} = \sup_{|\psi\rangle \neq 0} \frac{\langle\psi|A^\dagger A|\psi\rangle}{\langle\psi|\psi\rangle} \tag{B.24}$$

if this quantity exists, intuitively, it is saying is that A will scale a vector $|\psi\rangle$ by at most said quantity. ie $||A\,|\psi\rangle\,|| \leq ||A||\,|\psi\rangle\,, \forall\,|\psi\rangle \in \mathcal{H}$.

In our case, the operator A was the difference between two Hermitian operators, which is Hermitian. Therefore, there exists an orthonormal basis $\{|a\rangle\}$ in which A is diagonal, so that

$$||A|| = \sup_{|\psi\rangle \neq 0} \frac{\sum_{a,a'} \langle\psi|a'\rangle\,\langle a'|\,A^\dagger A\,|a\rangle\,\langle a|\psi\rangle}{\langle\psi|\psi\rangle} = \sup_{|\psi\rangle \neq 0} \frac{\sum_a |a|^2|\,\langle\psi|a\rangle\,|^2}{\langle\psi|\psi\rangle}. \tag{B.25}$$

We can now set an upper bound on this quantity. If $a_{\max}$ is the maximum eigenvalue of A, we have that

$$|a_{\max}|^2 \geq \sup_{|\psi\rangle \neq 0} \sum_a |a|^2|\,\langle\psi|a\rangle\,|^2, \tag{B.26}$$

because $|\,\langle\psi|a\rangle\,|^2 \leq 1$.

Appendix C

More noisy results

Here we very briefly show the results of testing a different kind of noisy environment than the simple Gaussian of Chapter 5.4 on the same example instance for MCTS-guided QA. Inspired by [59], we model the quantum measurement noise. This too samples the noise from a Gaussian distribution, but the standard deviation of the distribution is given by

$$\Delta H = \sqrt{\langle\psi(\tau)|H_z^2|\psi(\tau)\rangle - \langle\psi(\tau)|H_z|\psi(\tau)\rangle^2}. \tag{C.1}$$

This quantity vanishes when the state $|\psi(\tau)\rangle$ is an eigenstate of H_z, particularly the ground state, and it increases as it gets further away from it as can be seen in Figure C.1.

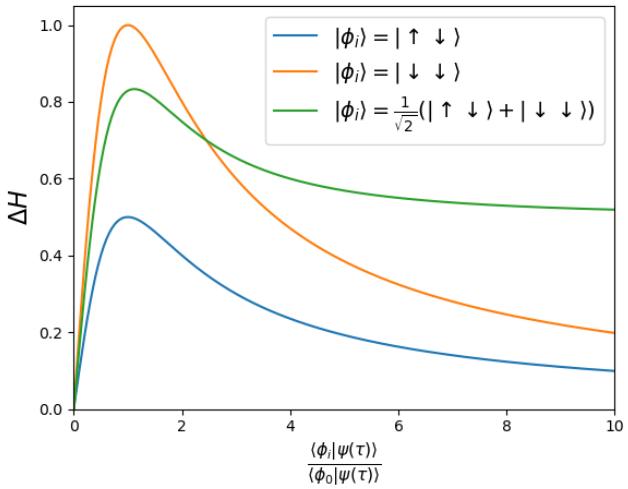


Figure C.1: Quantum measurement error strength as a function of the ratio between the overlap of the evolved system $|\psi(\tau)\rangle$ with the ground state $|\phi_0\rangle = |\uparrow\uparrow\rangle$ and other eigenstates $|\phi_i\rangle$. The system is $H_z = \mathrm{diag}(0, 1, 1, 2)$, so that the state $|\downarrow\downarrow\rangle$ creates bigger fluctuations than the other eigenstates.

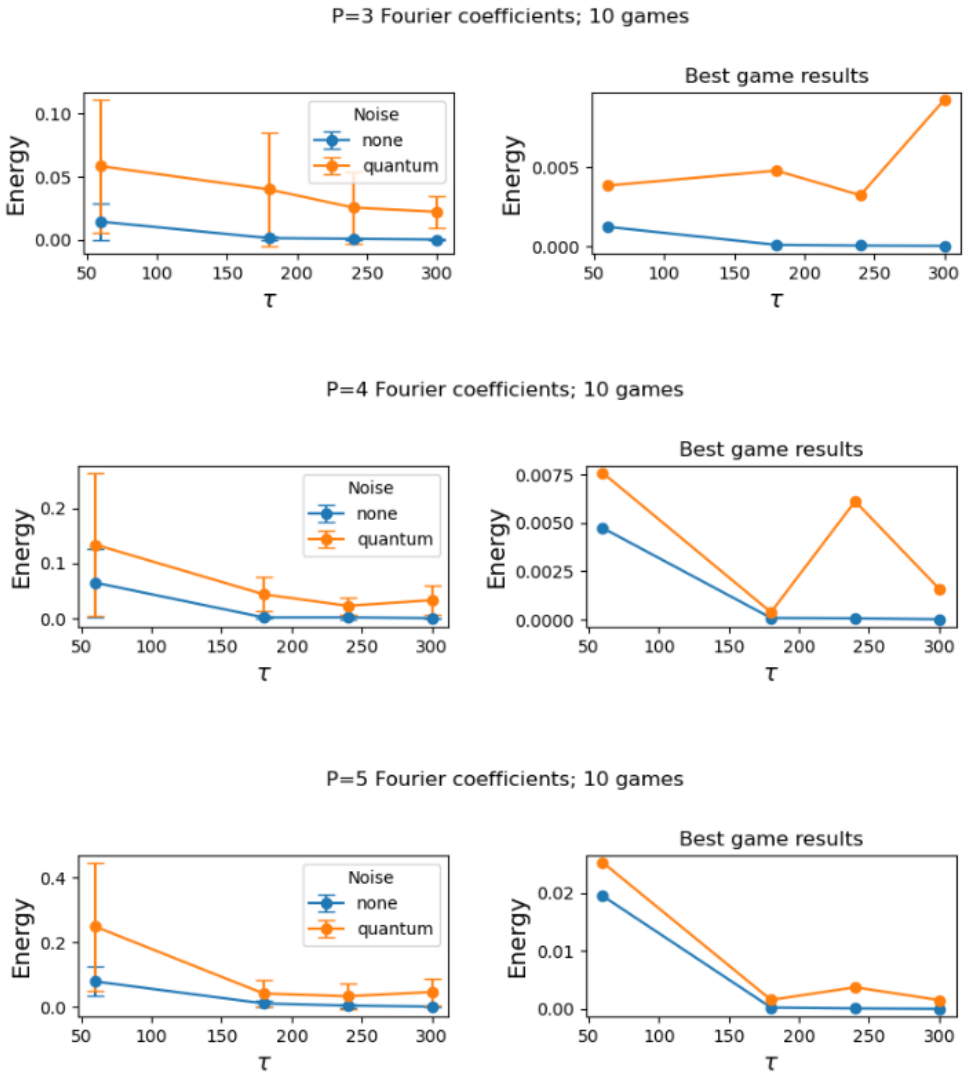


Figure C.2: Resulting energies of MCTS in a noisy reward environment with quantum measurement noise. We test it for the case of $P = 3, 4, 5$ Fourier components. This is a more challenging noise for MCTS, and it was not able to converge close to the noiseless case in 10 games for a few cases, so that it would require more than 10 games for proper convergence.

Bibliography

[1] P. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in: Proceedings 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 124–134. doi:10.1109/SFCS.1994.365700.

[2] A. W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations, Physical Review Letters 103 (15) (oct 2009). doi:10.1103/physrevlett.103.150502.

[3] L. K. Grover, A fast quantum mechanical algorithm for database search (1996). arXiv:quant-ph/9605043.

[4] J. Roffe, Quantum error correction: an introductory guide, Contemporary Physics 60 (3) (2019) 226–245. doi:10.1080/00107514.2019.1667078.

[5] K. Bharti, et al., Noisy intermediate-scale quantum algorithms, Reviews of Modern Physics 94 (1) (feb 2022). doi:10.1103/revmodphys.94.015004.

[6] A. Frank, et al., Quantum supremacy using a programmable superconducting processor, Nature 574 (7779) (2019) 505–510. doi:10.1038/s41586-019-1666-5.

[7] Z. Han-Sen, et al., Quantum computational advantage using photons, Science 370 (6523) (2020) 1460–1463. doi:10.1126/science.abe8770.

[8] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, P. J. Coles, Variational quantum algorithms, Nature Reviews Physics 3 (9) (2021) 625–644. doi:10.1038/s42254-021-00348-9.

[9] T. Yamazaki, S. Matsuura, A. Narimani, A. Saidmuradov, A. Zaribafiyan, Towards the practical application of near-term quantum computers in quantum chemistry simulations: A problem decomposition approach (2018).

[10] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, J. L. O'Brien, A variational eigenvalue solver on a photonic quantum processor, Nature Communications 5 (1) (jul 2014). doi:10.1038/ncomms5213.

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

[12] A. Finnila, M. Gomez, C. Sebenik, C. Stenson, J. Doll, Quantum annealing: A new method for minimizing multidimensional functions, Chemical Physics Letters 219 (5) (1994) 343–348. doi:https://doi.org/10.1016/0009-2614(94)00117-0.

[13] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680. doi:10.1126/science.220.4598.671.

[14] M. F. Ashby, D. R. Jones, Chapter 6 - driving force for structural change, in: Engineering Materials 2 (Fourth Edition), International Series on Materials Science and Technology, Butterworth-Heinemann, 2013, pp. 109–123. doi:https://doi.org/10.1016/B978-0-08-096668-7.00006-1.

[15] E. Farhi, J. Goldstone, S. Gutmann, Quantum adiabatic evolution algorithms versus simulated annealing (2002). arXiv:quant-ph/0201031.

[16] A. Ichiki, M. Ohzeki, Upper bound inequality for calculation time in simulated annealing analogous to adiabatic theorem in quantum systems (2021). arXiv:2107.01792.

[17] S. Suzuki, A comparison of classical and quantum annealing dynamics, Journal of Physics: Conference Series 143 (1) (2009) 012002. doi:10.1088/1742-6596/143/1/012002.

[18] M. M. Wauters, R. Fazio, H. Nishimori, G. E. Santoro, Direct comparison of quantum and simulated annealing on a fully connected ising ferromagnet, Phys. Rev. A 96 (2017) 022326. doi:10.1103/PhysRevA.96.022326.

[19] E. Kapit, V. Oganesyan, Noise-tolerant quantum speedups in quantum annealing without fine tuning, Quantum Science and Technology 6 (2) (2021) 025013. doi:10.1088/2058-9565/abd59a.

[20] S. Arora, B. Barak, Computational Complexity: A Modern Approach, 1st Edition, Cambridge University Press, USA, 2009.

[21] D. Gosset, D. Nagaj, Quantum 3-SAT is QMA1-complete, in: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, IEEE, 2013. doi:10.1109/focs.2013.86.

[22] A. Lucas, Ising formulations of many NP problems, Frontiers in Physics 2 (2014). doi:10.3389/fphy.2014.00005.

[23] G. E. Crooks, Performance of the quantum approximate optimization algorithm on the maximum cut problem (2018). arXiv:1811.08419.

[24] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, H. Neven, Barren plateaus in quantum neural network training landscapes, Nature Communications 9 (1) (nov 2018). doi:10.1038/s41467-018-07090-4.

[25] A. Braida, S. Martiel, I. Todinca, Anti-crossings occurrence as exponentially closing gaps in quantum annealing (2023). arXiv:2304.12872.

[26] S. Morita, Faster annealing schedules for quantum annealing, Journal of the Physical Society of Japan 76 (10) (2007) 104001. doi:10.1143/JPSJ.76.104001.

[27] L. Kocsis, C. Szepesvári, Bandit based monte-carlo planning, Vol. 2006, 2006, pp. 282–293. doi:10.1007/11871842_29.

[28] M. Świechowski, K. Godlewski, B. Sawicki, J. Mańdziuk, Monte carlo tree search: a review of recent modifications and applications, Artificial Intelligence Review 56 (3) (2022) 2497–2562. doi:10.1007/s10462-022-10228-y.

[29] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 3rd Edition, Prentice Hall, 2010.

[30] S. Thrun, Learning to play the game of chess, Advances in neural information processing systems 7 (1994).

[31] S. Gelly, D. Silver, Monte-carlo tree search and rapid action value estimation in computer go, Artificial Intelligence 175 (2011) 1856–1875.

[32] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, D. Hassabis, Mastering chess and shogi by self-play with a general reinforcement learning algorithm (2017). arXiv:1712.01815.

[33] B. Sheppard, World-championship-caliber scrabble, Artificial Intelligence 12 (2002) 241–275. doi:10.1016/S0004-3702(01)00166-7.

[34] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, S. Colton, A survey of monte carlo tree search methods, IEEE Transactions on Computational Intelligence and AI in Games 4 (1) (2012) 1–43. doi:10.1109/TCIAIG.2012.2186810.

[35] C. Mansley, A. Weinstein, M. Littman, Sample-based planning for continuous action markov decision processes., 2011.

[36] P. Auer, N. Cesa-Bianchi, P. Fischer, Finite-time analysis of the multiarmed bandit problem, Machine Learning 47 (2002) 235–256. doi:10.1023/A:1013689704352.

[37] The code developed for this work will be publicly available in, A. Agirre, Code repository, https://github.com/agiandoni/quantumMCTS (2022).

[38] S. Bravyi, D. P. DiVincenzo, R. I. Oliveira, B. M. Terhal, The complexity of stoquastic local hamiltonian problems (2007). arXiv:quant-ph/0606140.

[39] T. Albash, D. A. Lidar, Adiabatic quantum computation, Reviews of Modern Physics 90 (1) (jan 2018). doi:10.1103/revmodphys.90.015002.

[40] A. Messiah, Quantum mechanics (1976).

[41] K.-P. Marzlin, B. C. Sanders, Inconsistency in the application of the adiabatic theorem, Phys. Rev. Lett. 93 (2004) 160408. doi:10.1103/PhysRevLett.93.160408.

[42] L. D. Landau, Zur theorie der energieubertragung. ii, Physikalische Zeitschrift der Sowjetunion 2 (46) (1932).

[43] C. Zener, R. H. Fowler, 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) (1932) 696–702. doi:10.1098/rspa.1932.0165.

[44] A. V. Shytov, Landau-zener transitions in a multilevel system: An exact result, Phys. Rev. A 70 (2004) 052708. doi:10.1103/PhysRevA.70.052708.

[45] N. V. Vitanov, B. M. Garraway, Landau-zener model: Effects of finite coupling duration, Phys. Rev. A 53 (1996) 4288–4304. doi:10.1103/PhysRevA.53.4288.

[46] V. Mehta, F. Jin, H. D. Raedt, K. Michielsen, Quantum annealing for hard 2-satisfiability problems: Distribution and scaling of minimum energy gap and success probability, Physical Review A 105 (6) (jun 2022). doi:10.1103/physreva.105.062406.

[47] A. Soriani, P. Nazé , M. V. S. Bonança, B. Gardas, S. Deffner, Three phases of quantum annealing: Fast, slow, and very slow, Physical Review A 105 (4) (apr 2022). doi:10.1103/physreva.105.042423.

[48] W. H. Zurek, Cosmological experiments in superfluid helium? (Oct. 1985). doi:10.1038/317505a0.

[49] T. Caneva, R. Fazio, G. E. Santoro, Adiabatic quantum dynamics of a random ising chain across its quantum critical point, Physical Review B 76 (14) (oct 2007). doi:10.1103/physrevb.76.144427.

[50] J. Johansson, P. Nation, F. Nori, Qutip 2: A python framework for the dynamics of open quantum systems, Computer Physics Communications 184 (4) (2013) 1234–1240. doi:https://doi.org/10.1016/j.cpc.2012.11.019.

[51] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker, A0c: Alpha zero in continuous action space (2018). arXiv:1805.09613.

[52] W. van Dam, M. Mosca, U. Vazirani, How powerful is adiabatic quantum computation?, in: Proceedings 42nd IEEE Symposium on Foundations of Computer Science, IEEE, 2001. doi:10.1109/sfcs.2001.959902.

[53] S. Zielinski, J. Nüßlein, J. Stein, T. Gabor, C. Linnhoff-Popien, S. Feld, Pattern qubos: Algorithmic construction of 3sat-to-qubo transformations (2023). arXiv:2305.02659.

[54] J. Marques-Silva, Practical applications of boolean satisfiability, in: 2008 9th International Workshop on Discrete Event Systems, 2008, pp. 74–80. doi:10.1109/WODES.2008.4605925.

[55] P. Cheeseman, B. Kanefsky, W. M. Taylor, Where the really hard problems are, in: Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1, IJCAI'91, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991, p. 331–337.

[56] Q. Lin, X. Wang, J. Niu, A new method for 3-satisfiability problem phase transition on structural entropy, IEEE Access 9 (2021) 2093–2099. doi:10.1109/ACCESS.2020.3047930.

[57] Y.-Q. Chen, Y. Chen, C.-K. Lee, S. Zhang, C.-Y. Hsieh, Optimizing quantum annealing schedules with monte carlo tree search enhanced with neural networks (2022). arXiv:2004.02836.

[58] T. Caneva, T. Calarco, S. Montangero, Chopped random-basis quantum optimization, Physical Review A 84 (2) (aug 2011). doi:10.1103/physreva.84.022326.

[59] J. Yao, H. Li, M. Bukov, L. Lin, L. Ying, Monte carlo tree search based hybrid optimization of variational quantum circuits (2022). arXiv:2203.16707.

[60] M. E. S. Morales, J. D. Biamonte, Z. Zimborás, On the universality of the quantum approximate optimization algorithm, Quantum Information Processing 19 (9) (aug 2020). doi:10.1007/s11128-020-02748-9.

[61] Y. Ruan, S. Marsh, X. Xue, Z. Liu, J. Wang, The quantum approximate algorithm for solving traveling salesman problem, Computers, Materials, Continua 63 (2020) 1237–1247. doi:10.32604/cmc.2020.010001.

[62] G. Mbeng, R. Fazio, G. Santoro, Quantum annealing: a journey through digitalitalization, control, and hybrid quantum variational schemes (06 2019).

[63] S. H. Sack, M. Serbyn, Quantum annealing initialization of the quantum approximate optimization algorithm, Quantum 5 (2021) 491. doi:10.22331/q-2021-07-01-491.

[64] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, M. D. Lukin, Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices, Physical Review X 10 (2) (jun 2020). doi:10.1103/physrevx.10.021067.

[65] M. M. Wauters, E. Panizon, G. B. Mbeng, G. E. Santoro, Reinforcement-learning-assisted quantum optimization, Physical Review Research 2 (3) (sep 2020). doi:10.1103/physrevresearch.2.033446.

[66] V. Akshay, H. Philathong, M. E. S. Morales, J. D. Biamonte, Reachability deficits in quantum approximate optimization, Phys. Rev. Lett. 124 (2020) 090504. doi:10.1103/PhysRevLett.124.090504.

[67] R. Fletcher, Practical Methods of Optimization, John Wiley & Sons, New York, NY, USA, 1987.

[68] K. Godlewski, B. Sawicki, MCTS based agents for multistage single-player card game, in: 2020 IEEE 21st International Conference on Computational Problems of Electrical Engineering (CPEE), IEEE, 2020. doi:10.1109/cpee50798.2020.9238707.

[69] R. Bjarnason, A. Fern, P. Tadepalli, Lower bounding klondike solitaire with monte-carlo planning, Proceedings of the International Conference on Automated Planning and Scheduling 19 (1) (2009) 26–33. doi:10.1609/icaps.v19i1.13363.

[70] M. Schadd, M. Winands, H. Herik, G. Chaslot, J. Uiterwijk, Single-player monte-carlo tree search, 2008, pp. 1–12. doi:10.1007/978-3-540-87608-3_1.

[71] Y. Björnsson, H. Finnsson, Cadiaplayer: A simulation-based general game player, Computational Intelligence and AI in Games, IEEE Transactions on 1 (2009) 4 – 15. doi:10.1109/TCIAIG.2009.2018702.

[72] S. Klarsfeld, J. A. Oteo, The baker-campbell-hausdorff formula and the convergence of the magnus expansion (Nov 1989). doi:10.1088/0305-4470/22/21/018.