INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Master's Thesis

Exploring the Potential of the Filtering Variational Quantum Eigensolver

Wanja Sajko

Supervisor:	Prof. Dr. Dieter Kranzlmüller
Advisors:	Korbinian Staudacher Xiao-Ting Michelle To Sven Prüfer (DLR)
Deadline:	26.10.2023

I hereby certify that I have written this Master's thesis thesis independently and that I have not used any sources other than those and aids other than those indicated.

Munich, October 26, 2023

. . . . . .
(Signature of the candidate)

Abstract

Job scheduling is a complex optimization problem with multiple variables, constraints, and goals. Solving such problems using classical computing can be challenging, since they are NP-complete and real-world instances can be quite large. Quantum computing is a promising solution, as it is theoretically faster than classical computing for certain types of problems.

In this thesis, we use the filtering variational quantum eigensolver (F-VQE), a parameterized quantum algorithm, to solve a simplified real-world scheduling problem. The F-VQE algorithm optimizes solutions by filtering out unpromising ones and using a classical optimization routine to refine the remaining solutions. Although the F-VQE algorithm is based on a paper by Amaro et al. [AMR+22], it has not yet been fully evaluated for solving scheduling problems. While VQEs have been successful in solving combinatorial optimization problems, we seek to assess the performance of F-VQE in solving scheduling problems.

We have two objectives in this research: firstly, to enhance and analyze the F-VQE algorithm, and secondly, to evaluate the potential of quantum computing in solving complex scheduling problems. To accomplish this, we will compare the performance of the F-VQE algorithm with other quantum and classical approaches for solving real-world scheduling problems. This will provide valuable insights into the effectiveness of quantum computing for solving these problems, as well as identify potential improvements to the F-VQE algorithm.

We delve deeper into the F-VQE algorithm to identify potential areas for improvement. We will examine various ansatz designs, different filtering strategies, and encoding techniques. Worthwhile additions are implemented and tested against.

To compare the F-VQE algorithm's performance with other variational quantum algorithms and an approach using Grover's algorithm, which is already implemented by the DLR. We evaluate the efficiency, scalability, and quality of solutions provided by each algorithm and discuss the potential benefits and drawbacks of the F-VQE for solving real-world scheduling problems.

Contents

1 Introduction

With the advent of real and usable quantum computers, old and new techniques and algorithms are coming into the spotlight. Although quantum computers are still plagued by problems such as limited size, vulnerability to errors, high costs, and limited supply, a big field for new innovations has opened. Especially with companies such as IBM, D-Wave, and Google making it easier than ever for common researchers to write their own algorithms and simulations by providing cloud-based access to quantum computers [Pow08, Mai20], with some even consisting of hundreds of qubits. Research spans a wide spectrum of use cases, encompassing familiar use cases like post-quantum security, molecular simulation, artificial intelligence, financial services, and the solving of combinatorial problems. [BBB$^+$21]. Advances would impact several important real-world applications, such as developing new drugs or materials, improving already impressive artificial intelligence products, making cryptography safer against quantum algorithms, and improving routing and scheduling tasks. Combinatorial problems hold particular significance in the realm of computer science. Three prominent examples are:

- Finding satisfying variable assignments of propositional formulae (SAT)
- Finding the shortest round trips in graphs (TSP)
- Finding the optimal schedule to complete a set of jobs (JSSP)

All three problems are NP-complete [Woe03] and are often, depending on the problem instance size, solved by heuristic approaches, resulting in approximations of optimal solutions. Combinatorial problems are used to model scheduling and pricing for airlines, to decide the routes of delivery trucks, or to make media and ad recommendations [Tre11]. It is yet unclear which or if any quantum algorithms will provide a real advantage for NP-complete problems over classical computing in real-life applications. It revolves around the distinction between bounded-error quantum polynomial time (BQP) and P complexity, highlighting that while quantum algorithms show theoretical promise, their practical applicability depends on factors like algorithm design, available quantum hardware, and the nature of the specific problem. BQP comprises problems that quantum computers can solve in polynomial time, and if P = BQP, the advantage of quantum algorithms diminishes.

Even with this in mind and faced with the limitations of today's Noisy-Intermediate Scale Quantum (NISQ) devices, multiple organizations and businesses are working on new methods to outperform classical computers in the near future [Gib19]. For a limited set of small problem instances, currently existing NISQ devices have already outperformed classical computers [RSR$^+$17, DBK$^+$22]. The number of operations in current quantum algorithms must be kept low, since with longer chains of operations, more and more errors are introduced into the system, resulting in inaccurate and unreliable outcomes. Some algorithms, which have theoretically been proven to outperform classical computers, such as Shor's algorithm [Sho94], are therefore not yet feasible. Given these hardware limitations, much of the

research effort has been directed toward hybrid algorithms using a combination of classical and quantum computation.

1.1 Contribution

In this work, we will evaluate and improve the Filtering-Variational Quantum Eingensolver (F-VQE) algorithm, which can be used to find optimal solutions to a given objective. The problem or objective is given by its objective function. The objective function encodes the problem and returns the quality of a given solution. In the range of all potential solutions, the ground state symbolizes the optimal solution, reflecting the lowest energy, cost, or highest quality. F-VQE's quantum nature allows it to navigate this extensive solution space more efficiently than classical algorithms, presenting numerous applications that might benefit from its usage. The F-VQE algorithm is based on a paper by Amaro et al. [AMR+22] and has not yet been fully evaluated for real-life use cases. One potential problem for which F-VQE could potentially provide an improvement is job scheduling. Here, one tries to find the best sequence of jobs that uses the least time or resources while not violating any given constraints. A comprehensive overview of this problem is given in 2.1.1. In this work, we aim to evaluate the suitability of the FVQE algorithm for job scheduling on the basis of a real problem at the German Space Operation Center. To evaluate the performance of F-VQE, we will examine different ansatz and filtering strategies, investigate encoding techniques like "binary encoding,", and use a hyperparameter optimization approach.

This work also compares the results of different quantum algorithms with each other. One solution uses Grover's algorithm to find all solutions that fulfill the given constraints. Grover's algorithm can find the solution for unstructured search problems where nothing is known (or no assumptions are used) about the structure of the solution space and the corresponding function. Afterwards, this set of solutions can be classically optimized for a set of goals. As with Shor's algorithm, Grover's algorithm requires a large number of operations and is therefore not optimally suited for NISQ devices. A deeper explanation of Grover's algorithm is given in 2.3.1.

The remaining quantum solutions, the Variational Quantum Eingensolver (VQE) and the Quantum Approximation Optimization Algorithm (QAOA), are approaches that have already been thoroughly researched and tested, making them great benchmarks. The algorithms along the F-VQE are part of a set of hybrid approaches in which a quantum computer serves as one component in the minimization of a set of parameters to ultimately find the minimum energy value of the objective.

1.2 Outline

Chapter 2 introduces the fundamentals of optimization, quantum computing, and the used quantum algorithms. Chapter 3 provides a concise overview of related work, while Chapter 4 explains the methods and concepts used in this work. In Chapter 5, we will talk about the specific problem and discuss the implementation of the final approach. These are evaluated through simulations and using quantum hardware in Chapter 6, along with a comparison with other quantum algorithms. Finally, Chapter 7 presents the conclusion and an outlook for future work.

2 Background

In this chapter, we will explain the underlying techniques and concepts on which the algorithms in the later parts build. We begin with an introduction to general optimization problems, explain the special case of job scheduling for the "job shop scheduling" problem, and introduce the most common methodology to find optimal solutions for a given objective. Then we explain the fundamental concepts of quantum computing and why there might be a potential advantage compared to classical computation. And lastly, we will talk about important quantum algorithms, some of which will be compared with the F-VQE algorithm.

2.1 Optimization

In the field of problem-solving, optimization is a fundamental discipline that empowers us to uncover the good solutions in a vast set of potential candidates for intricate challenges. Optimization, at its core, tries to approximate these solutions in an iterative fashion to converge to the final solution. Whether it is designing an efficient transportation network, maximizing profits in business, or fine-tuning a working schedule, optimization plays a pivotal role in finding optimal solutions from a myriad of possible options [LY16a]. The problem is defined by an objective function $f(\mathbf{x})$ considering a set of unknown real variables $\mathbf{x} = \{x_1,\ ...,\ x_n\}$, subject to a system of equalities $h(\mathbf{x}) = 0$ and inequalities $g(\mathbf{x}) \geq 0$ known as constraints. The aim is to calculate the extrema (maxima, minima, or stationary points) of this function. These extrema correspond to the optimal values for the variables $\mathbf{x}$ that satisfy these conditions and achieve the best possible outcome [LY16b].

$$\begin{aligned} &\textit{minimize } f(\mathbf{x}) \\ &\textit{subject to } h(\mathbf{x}) = 0,\ g(\mathbf{x}) \geq 0 \end{aligned}$$

While equality constraints can be put into practice, direct implementation of inequalities is not viable. Instead, they have to be converted to equalities first. This is achieved by introducing new variables—slack variables. Slack variables are additional non-negative variables in the solution vector that function as an offset with the size of inequality bounds. Each path or inequality needs its own slack variables. The encoding for slack variables can vary, but a very common one is the binary encoding. Here every slack variable is represented by the sum $\sum_l 2^l y_l$, where l is the bit index with a maximum bit length of L. The conversion from an inequality to an equality can be shown with a simple example:

$$\sum_{i=0}^{2} a_i \leq 2 \quad \equiv \quad \sum_{i=0}^{2} a_i + \sum_{l=0}^{1} 2^l y_l = 3 \tag{2.1}$$

To represent the bounds of the inequality, we need two bits, $l \in \{0, 1\}$. One can see that the inequality is fulfilled if an allocation of slack variables y_i for every state of the binary variables

a_i can be found, so that the equality is also fulfilled. Consider the state $a = \{1, 1, 0\}$; both slack variables y_0 and y_1 need to be zero to fulfill the equality. Naturally, there can be no allocation of slack variables for states that violate inequality to achieve equality. For the state $a = \{1, 1, 1\}$, the inequality is not met, and there is no allocation for y_0 and y_1 to fulfill the equality. The equality in 2.1 can be formulated as an objective function:

$$f(a, y) = (3 - \sum_{i=0}^{2} a_i + \sum_{l=0}^{1} 2^l y_l)^2 \tag{2.2}$$

which punishes variable allocations by how far they are over the given bounds of 3.

There are various specialized fields that address distinct challenges. For the purpose of this work, where we delve into combinatorial scheduling problems characterized by discrete decision variables and frequently involving non-linear objective functions, our primary focus centers on non-linear binary optimization with constraints.

This subset has two defining features. First, it encompasses problems where at least some of the constraints or the objective function exhibit nonlinearity; e.g., it contains terms for variable interactions: $f(\mathbf{x}) = ax_1 + \mathtt{bx_2x_3}$. Second, within this subset, all variables are binary, meaning they can only take on two values, typically 0 or 1 - $\mathbf{x} \in \{0, 1\}^n$. Combinatorial problems involve selecting a combination of elements from a set to achieve a desired outcome, which can be modeled through binary decision variables (0 or 1) to represent the inclusion or exclusion of elements. This makes binary optimization particularly well-suited for combinatorial problems. Combinatorial problems also typically involve specific requirements, limitations, and conditions that must be adhered to. In addition to the objective function, a set of constraints must be observed.

2.1.1 Job Shop Scheduling

We now want to introduce the concept of NP-complete combinatorial job scheduling problems via the well-known example "Job Shop Scheduling Problem" (JSSP) [Man60]. It is used to describe the workflow of machines completing jobs in manufacturing and production settings. For example, consider a company specializing in car manufacturing. This company operates a range of machines responsible for producing various components of their cars, such as the engine, chassis, body panels, and interior features. Each car's production involves a sequence of tasks that must be performed on specific machines. The company wants to find the order in which tasks for different cars should be executed on the available machines to minimize costs, use resources efficiently, and meet demand.

To formalize the JSSP, consider the following scenario: A set of jobs is denoted as $J = \{J_1, J_2, \ ..., \ J_n\}$, and a set of machines is represented as $M = \{M_1, \ M_2, \ ..., \ M_m\}$. Each job could be a step in the car manufacturing process of our imaginary company. We now try to find a schedule for all jobs and machines that minimizes, for example, the total time taken until every job is finished. Additionally, the following constraints have to be observed:

1. Each job J_i must be executed in a distinct order of machines. $S \in M^T$, where T is the number of machines needed for the job J_i.

2. Only one machine M_j can work on a job J_i simultaneously.

3. The time a machine M_j needs to complete its part of the job J_i can be different depending on the job and the machine. It is given by: $d(M_j, J_i)$.

4. Each machine can handle just one task at any given moment.

Imagine we have three different car types: a sports car, an off-road car, and a truck. Each type can be represented by a job: $\{J_1 \equiv \text{Sports Car}, J_2 \equiv \text{Off-road}, J_3 \equiv \text{Truck}\}$. To complete a car, all parts must be built on their respective machines. To simplify the process, we will only consider three parts: the engine, the chassis, and the wheels. $\{M_1 \equiv \text{Engine}, M_2 \equiv \text{Chassis}, M_3 \equiv \text{Wheels}\}$. The order in which each car has to be built is determined by its importance. A sports car needs to look dashing and be lightweight, so the chassis is most important. The truck needs a good engine to haul its cargo, and the off-road car needs good tires to master the terrain. In Table 2.1, the sequence of machines and the duration of production can be seen for each car.

Jobs	Sequence of machines with respective duration		
1	$M_2(3)$	$M_3(1)$	$M_1(2)$
2	$M_3(3)$	$M_1(2)$	$M_2(1)$
3	$M_1(4)$	$M_3(2)$	$M_2(2)$

Table 2.1: Sequence of machines and duration of production.

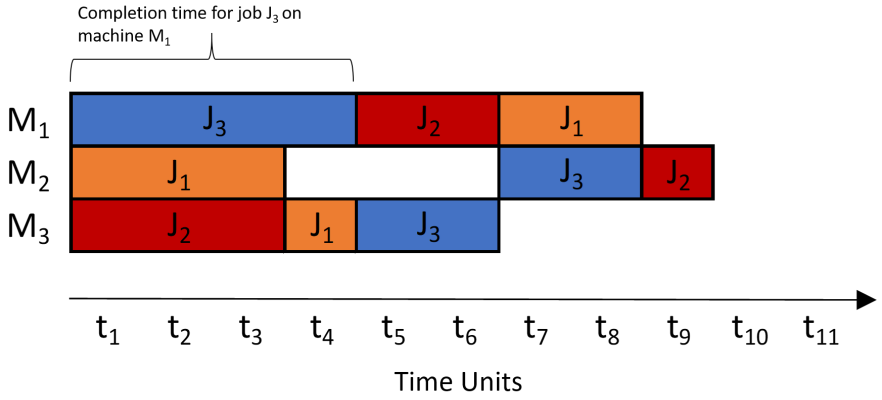


Figure 2.1: Gantt chart of the JSP with three jobs and three machines

Gantt charts often serve as a graphical depiction of the solution to the JSSP, providing a visual representation of the scheduling of individual jobs and machines spanning the entire required time frame. A chart showcases both the completion time of each job and the optimal scheduling of the machines. In Figure 2.1, the Gantt Chart shows a scheduling solution for our car manufacturing problem. According to this chart, the total completion time of the jobs is 9 time units.

Because the JSSP involves both discrete choices (task sequences) and resource constraints (machine availability), solving it optimally becomes an intricate computational task. Specialized optimization techniques, including heuristic and metaheuristic approaches, are often employed to efficiently navigate the complex solution space [CAS20].

2.1.2 Gradient Descent

We will now look at how one would find the extrema or an approximation of the extrema for a problem such as the JSSP. We will describe the basic technique of gradient descent, which is the standard of reference for all other advanced techniques. It is also used in our implementation for the VQE and F-VQE.

Gradient descent is a first-order method, which means the objective function, denoted as f, is required to have continuous first partial derivatives. It is then possible to compute the gradient $\nabla f(\mathbf{x})$ to iteratively update the variables $\mathbf{x}$. The updated variables, denoted as $\mathbf{x_{k+1}}$, are determined by the formula:

$$\mathbf{x_{k+1}} = \mathbf{x_k} - \alpha_k \nabla f(\mathbf{x_k})$$

with a step size or learning rate α_k, which governs the extent of movement in the direction opposite to the current gradient, leading towards a local minimum. The value of α is allowed to change at every iteration. Big step sizes reduce the number of steps but risk overshooting the minimum. Contrary to popular belief, a small step size is more precise, but the number of iterations to reach the minimum might also be higher. The process commences with an initial guess, $\mathbf{x_0}$, representing an initial cost of $f(\mathbf{x_0})$. As each subsequent value $\mathbf{x_1}, \mathbf{x_2}, \cdots$ follows the direction opposite to the gradient of f, a sequence is generated: $f(\mathbf{x_0}) \geq f(\mathbf{x_1}) \geq f(\mathbf{x_2}) \geq \cdots$. Through a sufficient number of iterations, the optimal solution $\mathbf{x}^*$ with $f(\mathbf{x}^*) \leq f(\mathbf{x_i})$, or an approximation to the best solution, can be reached.

In Figure 2.2, the gradient descent algorithm is presented for energy landscapes with only one (a, b) and with two variables x_0, x_1 (c). The energy landscape is computed via the objective function, where the x-axis represents the parameter values or the input of the objective function and the y-axis shows the corresponding energy value. For graph (c), an objective function with two variables is examined, resulting in a three-dimensional energy landscape. For the optimization process for one variable, small and large step sizes are compared. It can be seen that the step size has a significant impact on the end result. Also, the chosen starting point can influence the quality of the solution and the speed at which the algorithm converges. Even in the example with a small step size, the optimal solution could have been found if a better starting point was chosen. It is important to note that the optimal value for α is heavily dependent on the chosen algorithm and objective function. The example in Figure 2.2 shows an improvement for larger step sizes, but this is not always the case and has to be evaluated for the specific energy landscape. For the graph with two variables, only one step size is shown. It is important to see how the search for the minimum gets more and more complicated with the increasing number of variables. Typically, a lot more variables are employed, rendering the optimization of such a higher-dimensional landscape exceptionally challenging.

Gradient descent can also struggle with pronounced curvature or erratic gradients since we encounter multiple saddle points and have to deal with very jittery optimization steps. The gradient of the objective function at saddle points is negligible or zero, which in turn leads to only slim optimization steps. In response, Momentum [NBH20] extends the gradient descent optimization algorithm by facilitating the accumulation of momentum in a specific direction within the search space. This feature aids in overcoming oscillations resulting from noisy gradients or large learning rates and enables a smoother traversal of flat regions in the search space. The update rule for gradient descent with momentum ξ is given by:

$$\mathbf{x_{k+1}} = \mathbf{x_k} - \alpha_k \nabla f(\mathbf{x_k}) + \xi \nabla f(\mathbf{x_{k-1}})$$

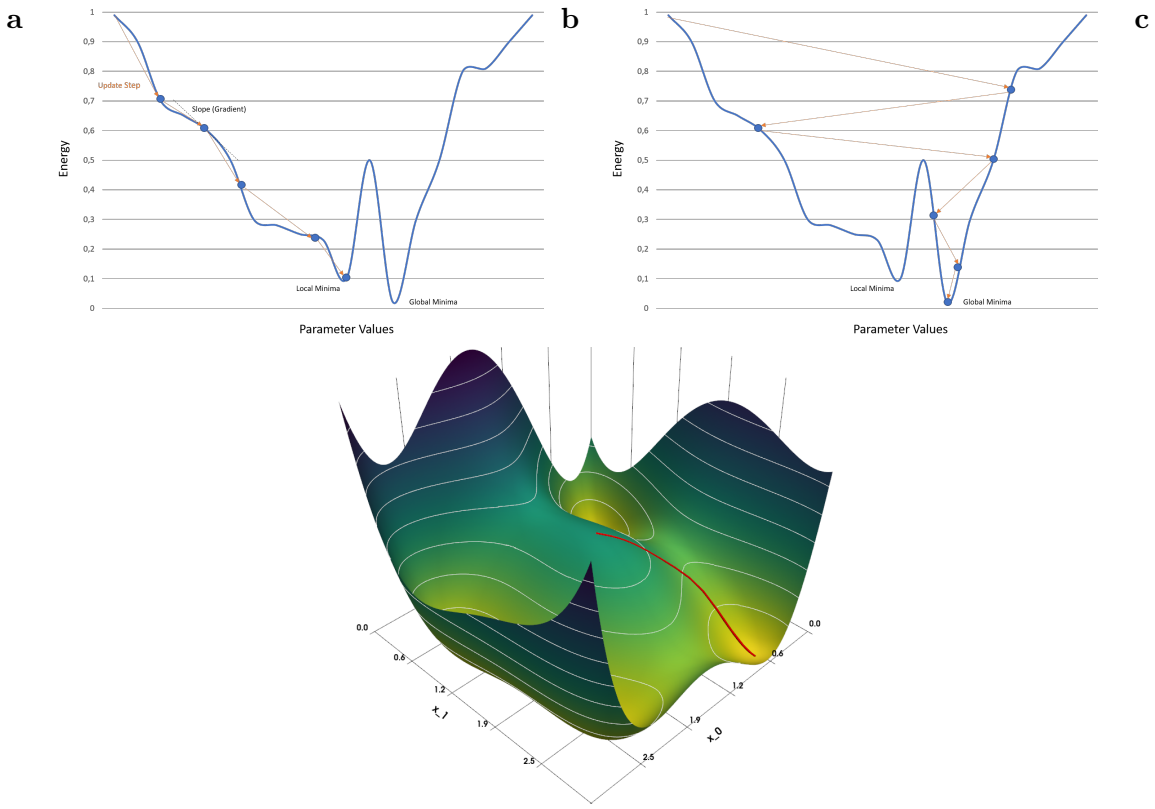


Figure 2.2: Example effect of step size on gradient descent. (a) Small step size. Energy gradually decreases but does not achieve the optimal point. (b) Large step size. Energy decreases but oscillates. Avoids local minima and achieves the optimal point. (c) Gradient descent examining two variables x_0 and x_1 on energy function simultaneously.

Here, $\xi \in [0, 1]$ is the ratio that governs the influence the previous gradient has on the current update step. In Figure 2.3, the effect of the momentum on oscillating gradients can be seen. Note that this is only an example and could also lead to an unoptimal result if the optimization runs into a local minima. It is crucial to find the right approach for the particular problem and to select good values for ξ.

It can also make a big difference in how the gradient $\nabla f(\mathbf{x})$ is computed. Two widely used approaches are mini batch [Rud17] and stochastic [Ket17] gradient descent. In mini-batch gradient descent, the average of multiple gradients is computed and used to update the parameters. This leads to smooth update steps but is computationally expensive for larger batch sizes. This is in contrast to stochastic gradient descent. Here, only one gradient is computed to update the parameters. Since just one example is considered at a time, the cost will fluctuate and will not necessarily decrease immediately. It is important to choose the best method for the concrete problem.

In our case, the approach is constrained by the nature of the quantum computer, where the quantity of data points is determined by the volume of measurements or samples. When dealing with a relatively small sample size, the optimization mirrors stochastic gradient descent, while with a larger number of samples, the optimization takes on the characteristics

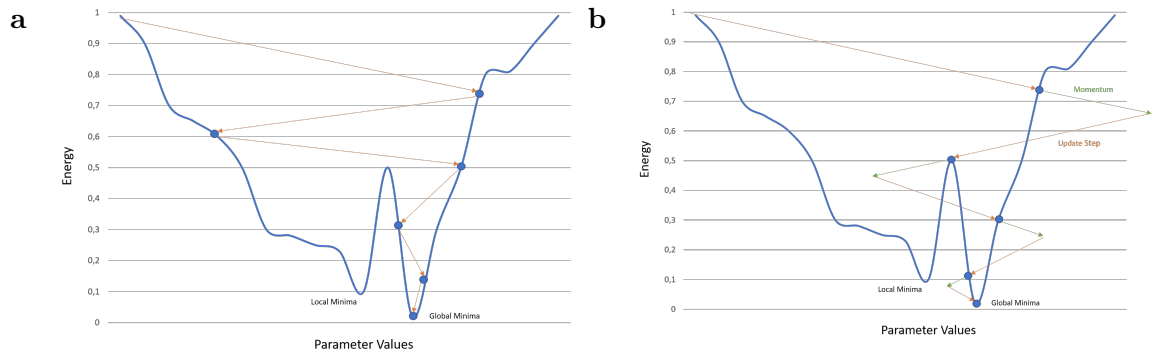


Figure 2.3: Example effect of momentum on gradient descent. (a) No momentum. No information from the last optimization step is used. (b) with momentum. The previous gradient informs the next optimization step.

of mini-batch gradient descent.

2.1.3 Other classical approaches

We will now look at other approaches, which we will later use for the implementation of QAOA.

Conjugate Gradient Descent

Nonlinear Conjugate Gradient Descent (CG) [CLR22] is a gradient descent method that combines local information (the gradient at the current point) with the previous direction to choose the direction of the following step:

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \alpha_k d_k$$

The set of directions $d = \{d_0, \cdots, d_{n-1}\}$ is given by:

$$d_k = \begin{cases} -\nabla f(\mathbf{x_k}) & \text{if } k = 0 \\ -\nabla f(\mathbf{x_{k-1}}) + \beta_k d_{k-1} & \text{otherwise} \end{cases}$$

The parameter β_k specifies the degree to which the previous direction influences the next direction. In the field of nonlinear CG, it is a big research topic, and there are various ways to compute the parameter. But regardless of the variant that is used, the produced directions d_k can lead to the computation of the following directions $d_{k+1} > 0$. In such a case, the algorithm can no longer guarantee a decrease in this direction. To circumvent this, the algorithm restarts the direction search by setting the new direction. $d_{k+1} = -\nabla f(\mathbf{x_k})$ every time $d_{k+1} > 0$.

Powell's algorithm

Powell's algorithm (Powell) [Pow64] is a numerical optimization algorithm that focuses on finding the optimal solution within a local region of the function's search space as opposed to seeking the global minimum or maximum across the entire search space.

Powell's algorithm initially starts with a random point within the function's search space x_0. Additionally, it maintains a set of search directions $d = \{d_0, \cdots, d_{n-1}\}$ similar to CG. The initial search directions are often aligned with the coordinate axes. The update step consists of minimizing $f(\mathbf{x}_k + \alpha_k \mathbf{d}_k)$ for each direction d_k.

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \sum_{i=0}^{n-1} \alpha_k d_k$$

where $\sum_{i=0}^{n-1} \alpha_k d_k$ is added to the set of direction vectors used in the next update step. On the other hand, the largest vector $\alpha_i d_i$ is removed from the set. These steps are repeated iteratively until no significant improvement is made.

Constrained Optimization by Linear Approximation

COBYLA [CLR$^+$20] is also a non-gradient-based optimization method. It instead uses a numerical optimization method for constrained problems where the derivative of the objective function is not known. It iteratively approximates the actual problem with a linear programming problem, which is then solved to obtain a candidate for the optimal solution. The candidate solution is evaluated using the original objective and constraint functions, yielding a new data point for the approximating linear programming problem. When the solution cannot be improved anymore, the step size is reduced, refining the search. When the step size becomes sufficiently small, the algorithm finishes. Due to the fact that linear approximations tend to be inefficient at a higher number of variables, this algorithm is suited mostly for small-dimension numbers. Even though our problem is suitable for COBYLA since it is also a constraint optimization problem, it can require a vast amount of variables, with scaling layers p reducing the efficiency.

Vanishing Gradients

For a lot of optimization problems and variational algorithms, it is a challenge to cope with barren plateaus and resulting vanishing gradients [TCC$^+$22]. This can have multiple causes, from choosing a poor objective function or starting initialization to alternative states with bad overlap. [TCC$^+$22]. The barren plateau is a result of the concentration of measure phenomenon [Led01] where functions with high dimensional variables and with small local oscillations are almost constant. The measured states are highly concentrated around a mean value, and the fraction of states that fall outside decreases exponentially in the number of variables. This is especially important for us since the number of variables increases with the number of qubits as well as with the depth of the circuit [Led01]. Gradients for observables within this flat plateau, where values concentrate around the mean, are exponentially small. Additionally, all observables can only be measured to a certain precision, making the distinction between positive and negative values for gradually vanishing gradients increasingly difficult. [TCC$^+$22]. Changing gradient signs leads to searches resembling random walks and will result in an exponentially small probability of exiting this barren plateau. It is essential to combat this phenomenon to guarantee a smooth optimization process.

2.2 Quantum Computing

As we will discuss various quantum algorithms going forward, we will give a concise introduction to quantum computing. Quantum computing is an interdisciplinary field that merges principles from quantum mechanics, computer science, and mathematics to explore a novel approach to computation. As of now, practical quantum computers remain in the early stages of development. And near-term quantum computers, with limited qubit counts, face challenges like noise, error rates, and qubit connectivity.

Computations in classical computers are based on bits, which can have a value of either 0 or 1 at any given time. These bits are manipulated using logic gates. Quantum computers, on the other hand, employ qubits for computations. The state of a qubit can be represented as an arbitrary linear combination of states $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,; \quad |0\rangle = \begin{bmatrix}1\\0\end{bmatrix},\ |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}; \quad \alpha^2 + \beta^2 = 1 \tag{2.3}$$

Here, α and β are the amplitudes for the respective states $|0\rangle$ or $|1\rangle$ and are complex numbers. For $\alpha, \beta \neq 0$, the state of the qubit is in between states $|0\rangle$ and $|1\rangle$. This phenomenon is known as superposition. The square of the amplitudes represents the probabilities for which the qubit exists in the states $|0\rangle$ or $|1\rangle$, respectively. This leads to another important property of qubits: the sum of the squared amplitudes is always equal to 1. Since parameters α and β are complex numbers, we can also write them in exponential form:

$$|\psi\rangle = |\alpha|e^{i\phi_\alpha}\,|0\rangle + |\beta|e^{i\phi_\beta}\,|1\rangle$$
$$|\psi\rangle = e^{i\phi_\alpha}(|\alpha|\,|0\rangle + |\beta|e^{i(\phi_\beta - \phi_\alpha)}\,|1\rangle)$$

Here ϕ_α is called the global phase and can be neglected since it has no further impact on the measurement or other operations. $\phi_\beta - \phi_\alpha$, on the other hand, is the relative phase. The relative phase is important for further quantum operations. In Figure 2.4, the relative phase can be seen as the rotation around the z-axis. While a rotation around this axis does not lead to a different measurement probability, it does influence future operations.

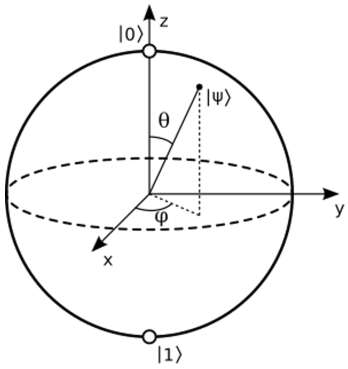


Figure 2.4: Bloch Sphere of one qubit [Com09]

In Figure 2.4, the state of a qubit is depicted on a bloch sphere. The state $|\psi\rangle$ is given as:

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

When measured, this superposition collapses, and the qubit ends up in state 0 with a probability of $|cos\frac{\theta}{2}|^2$ and in state 1 with a probability of $|e^{i\phi}sin\frac{\theta}{2}|^2$ [Hom22]. The superposition of qubits allows for a much larger and more complex set of possible combinations and computations compared to classical bits. For example, a system with two bits can only work with one combination at a time: 00, 01, 10, or 11. A quantum system with two qubits, on the other hand, can consist of a superposition over all four combinations: $|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$. So although we can only measure one of these four combinations in the end, a quantum algorithm can manipulate the quantum state and thus the amplitude of each combination.

2.2.1 Quantum Operations

To change the state of a qubit, a quantum gate is applied to the qubit. Quantum gates are represented by unitary matrices, which are square matrices with complex numbers as elements. Unitary matrices preserve the normalization of quantum states and ensure that the total probability of all possible outcomes remains 1. The application of a quantum gate to a qubit is equivalent to multiplying the state vector of the qubit by the corresponding unitary matrix. For example, one can look at the Hadamard gate (H gate). This gate operates on a single qubit and creates a superposition if applied to one of the basis states. It transforms the basis states $|0\rangle$ and $|1\rangle$ into equal superposition states, represented as:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \qquad H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

The Hadamard gate is often used at the beginning of quantum algorithms to prepare an initial superposition state.

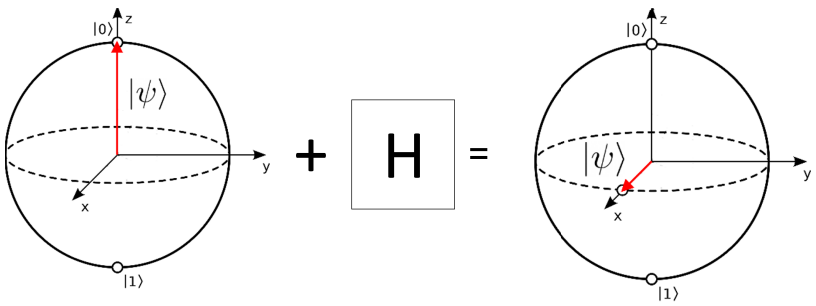


Figure 2.5: Effect of Hadamard gate on a qubit

One can also design gates that work on multiple qubits at the same time. One such example is the controlled-NOT gate (CNOT gate). The CNOT gate is a two-qubit gate that applies a NOT gate to the target qubit if the control qubit is in state $|1\rangle$. A two-qubit q_1, q_2 system is given by its state: $|\psi\rangle = |q_2 q_1\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$. The CNOT gate flips the state of the target qubit q_1 while leaving the control qubit q_2 unchanged. The concrete unitary and the application of the CNOT gate are shown in:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \qquad \begin{aligned} CNOT\,|00\rangle &= |00\rangle\,, \quad CNOT\,|01\rangle = |01\rangle \\ CNOT\,|10\rangle &= |11\rangle\,, \quad CNOT\,|11\rangle = |10\rangle \end{aligned}$$

Several other categories of gates are important for a variety of quantum algorithms. Often referred to as the NOT gate, the Pauli-X gate flips the state of a qubit from 0 to 1, and vice versa. Differently, the Pauli-Y and Pauli-Z gates introduce phase shifts and rotations around the Y and Z axes of the Bloch sphere, respectively. Rotation gates, which also play a major role in the quantum algorithms we will discuss later, allow for controlled rotations of the quantum state vector in the complex plane. There are three types of single-qubit rotation gates: the Rx, Ry, and Rz gates. Each of these gates rotates the state around the given axis by a parameter $\boldsymbol{\theta}$. For example, the two operations on the state $|0\rangle$: $H\,|0\rangle = Ry(\frac{\pi}{2})\,|0\rangle$ result in the same state. (Note, this is not necessarily true if applied to other states.) In figure 2.6, all three rotation gates are applied with an angle of $\frac{\pi}{2}$, starting with the Ry gate and ending with the Rx gate. The first Bloch sphere shows the initial state $|0\rangle$; in the second, the Ry gate is applied, which has the same effect as in 2.5. With the rotations Rz and Rx, the qubit is brought back to the state $|0\rangle$.

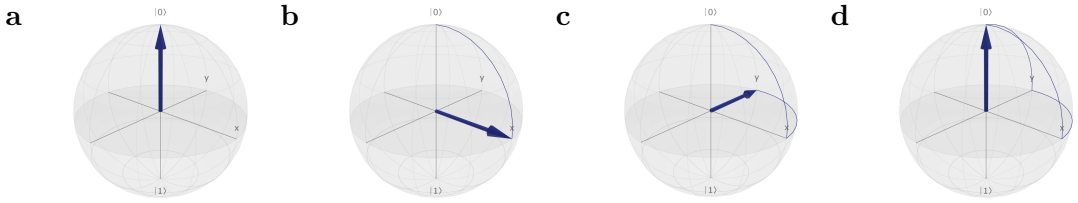


Figure 2.6: Bloch sphere after application of (a) no rotation. (b) $Ry(\frac{\pi}{2})$. (c) $Ry(\frac{\pi}{2})$, $Rz(\frac{\pi}{2})$. (d) $Ry(\frac{\pi}{2})$, $Rz(\frac{\pi}{2})$, $Rx(\frac{\pi}{2})$.

Quantum circuits are constructed by combining these gates in specific sequences to perform computations and implement quantum algorithms. The choice and arrangement of gates depend on the desired quantum computation or algorithm being executed. In figure 2.7, one such quantum circuit is depicted to showcase another important quantum effect: entanglement. The circuit consists of two qubits, $q[0]$ and $q[1]$. starting in state $|00\rangle$. The quantum operations are applied from left to right, beginning with the Hadamard gate followed by the CNOT gate. The control qubit is indicated by the small black dot and line. The NOT gate is depicted as $\bigoplus$. To receive the result of the circuit, the qubits $q[0]$, $q[1]$ need to be measured after state $|\psi_2\rangle$. Since this is an inherently statistical process, multiple measurements need to be taken to find the probability of all the resulting states.

When two qubits are entangled, their combined states cannot be factored into individual states. Every operation on one qubit immediately has an effect on the other as well. For example, one can look at the states $|\psi_0\rangle$, $|\psi_1\rangle$, and $|\psi_2\rangle$ in Figure 2.7:

$$|\psi_0\rangle = |00\rangle$$

The Hadamard-Gate is applied to the first qubit, resulting in $|\psi_1\rangle$:

$$|\psi_1\rangle = (H \otimes I)\,|00\rangle = H\,|0\rangle \otimes |0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,|0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$$

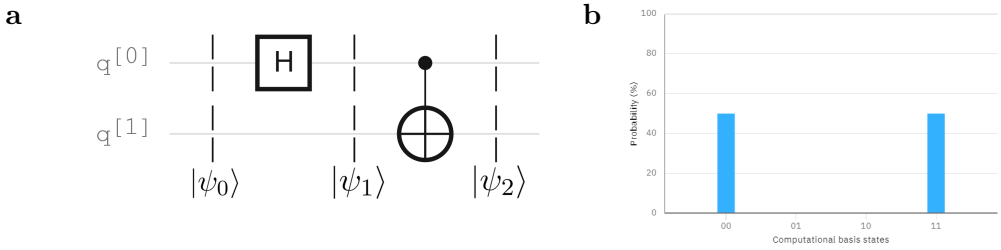


Figure 2.7: (a) Quantum Circuit with two qubits starting in state $|00\rangle$, creating a bell state. (b) Measurement probability for all input states

The CNOT gate is applied to both qubits. It flips the second qubit only if the first is in state $|1\rangle$:

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

$|\psi_2\rangle$ is a combination of both qubits being 0 and both qubits being 1. This state cannot be written as a simple product of the states of the individual qubits. If one of the qubits is measured, the state of the other one can be easily deducted. Were we measuring this circuit enough times, we would receive the probabilities: $P(|00\rangle) = (\frac{1}{\sqrt{2}})^2 = 50\% = P(|11\rangle)$.

2.3 Quantum Algorithms

Quantum algorithms are, as the name implies, algorithms that run on quantum computers. They use the techniques and phenomena described in the section above to achieve a speedup, or other efficiency improvement, over any possible classical algorithm.

2.3.1 Grover's Algorithm

One of the most famous quantum algorithms is the Grover algorithm. It was proposed by Lov Grover in 1996 and was used to solve unstructured search problems [Gro96]. It can find a marked entry or an entry that satisfies a constraint in an unsorted database. Today, it is also used as a subroutine for various other quantum algorithms [GWG21, GLRS16]. For our problem, it is used in its initial form. Grover's problem can be described more formally as follows: We have a system with $N = 2^n$ states, which we label $s_1, ..., s_N$. As in the previous definition, every state in our system can be seen as an entry in a database. We now want to find a state s_i that satisfies an arbitrary condition $C(s_i) = 1$. While for every other state $\bar{s_i}$, $C(\bar{s_i}) = 0$. Grover's algorithm aims to increase the amplitude of the state s_i and to decrease the amplitude of every other state $\bar{s_i}$.

The algorithm can be divided into three parts. In the beginning, an equal superposition over every qubit is created. This is easily accomplished by applying the Hadamard gate to each input qubit. The resulting input state of our system is described as:

$$|\psi_{init}\rangle = \frac{1}{\sqrt{N}} \sum_{s=1}^{N} |s\rangle$$

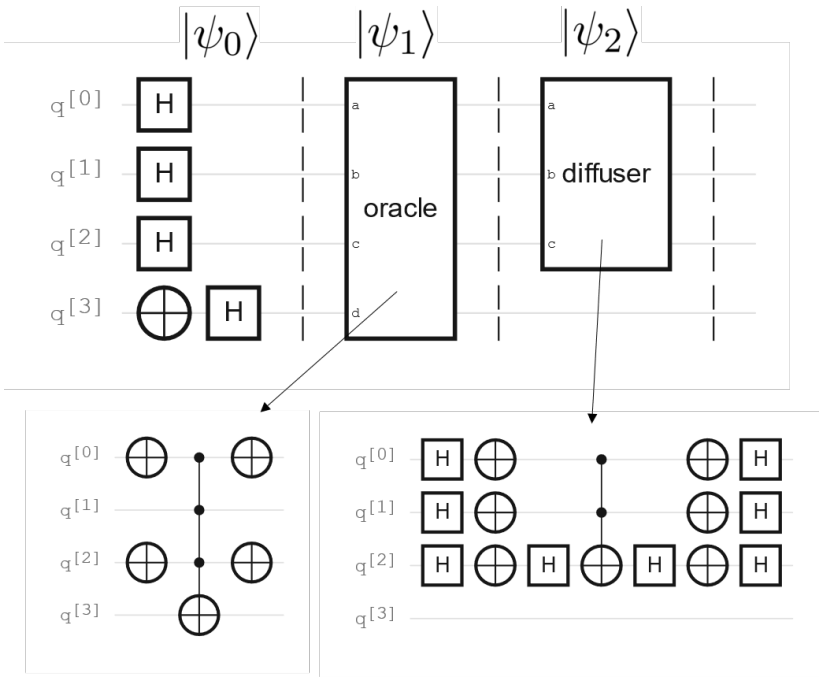


Figure 2.8: Quantum circuit and operations for Grover's algorithm for one iteration with an example oracle operator marking state $|010\rangle$.

Note that generally every qubit starts in the state $|0\rangle$; otherwise, one would not get the resulting state $|\psi_{init}\rangle$.

The next part encodes the condition $C(s)$ as a set of gates. This quantum oracle marks the input state s_i by flipping its amplitude. To achieve this, another qubit in the state $H|1\rangle$ is needed. In Figure 2.8, an example of such an oracle is given: if qubits $q[0]$, $q[1]$, and $q[2]$ are in state $|s_i\rangle = |010\rangle$, the state is marked by applying a bit flip to qubit $q[3]$. The state $|\psi_0\rangle$ after the oracle changes to $|\psi_1\rangle$:

$$|\psi_0\rangle = |\psi_{init}\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \rightarrow \quad \begin{aligned} |\psi_1\rangle &= |s\rangle \frac{1}{\sqrt{2}}(|C(s)\rangle - |1 \oplus C(s)\rangle) \\ &= |s\rangle (-1)^{C(s)} \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned}$$

Per the definition of C, only our marked input state (s_i) changes. Every other input state $\bar{s_i}$ stays unchanged: $|\bar{s_i}\rangle (-1)^{C(\bar{s_i})} = |\bar{s_i}\rangle$. The amplitude of our marked input state, however, does not change. If we were to measure this circuit, it would still result in an equal distribution. Therefore, in the last part of the algorithm, the diffuser operator, all amplitudes will be mirrored by their average. This will result in the amplification of our marked amplitude and a reduction of all other amplitudes. In Figure 2.9, the effect of the oracle and diffuser operators can be seen for the marked input state $|s_i\rangle = |010\rangle$ for the three input qubits $q[0]$, $q[1]$, and $q[2]$.

Parts two and three are repeated a specific number of times to increase the marked amplitude even further. Nevertheless, if these steps are repeated excessively, the marked amplitude can disproportionately influence the overall average. Since the marked amplitude is nega-

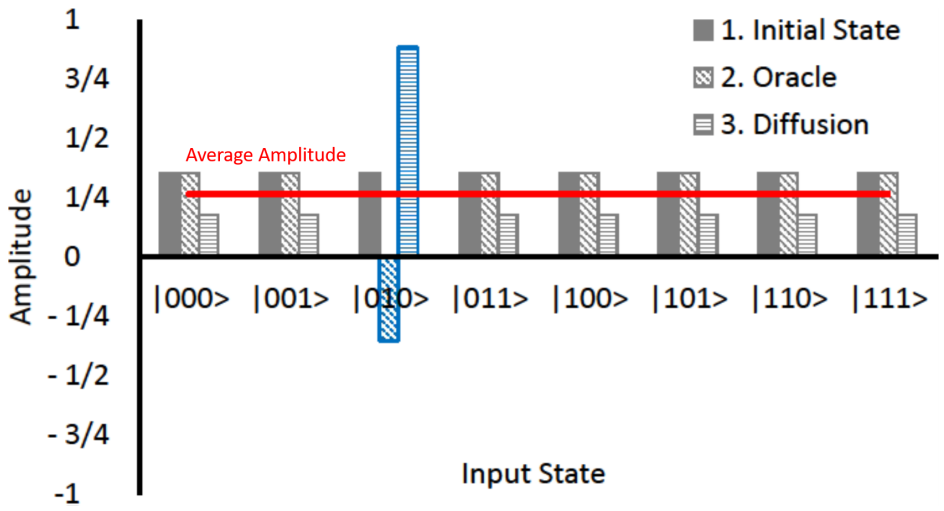


Figure 2.9: Amplitude distribution of all input states for $|\psi_{init}\rangle$, $|\psi_1\rangle$ and $|\psi_2\rangle$ as can be seen in [PSS+21, fig. 1]

tive, this could result in an equally negative average. The mirroring of the marked amplitude along this negative average can lead to a reduction in our marked amplitude.

2.3.2 Hamiltonian Formulation

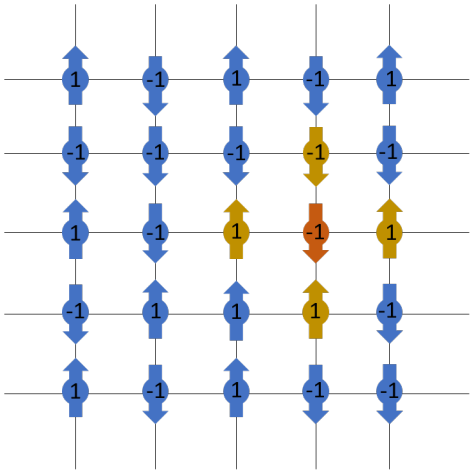
There is a wide variety of algorithms concerning Hamiltonians and their eigenvalues, eigenstates, and ground state [CAB+21, GTD19]. Since this is a crucial concept and is used throughout this work we will endeavor to explain it here. Hamiltonians are operators describing the properties of any physical system and are representations of this system's total energy spectrum, or eigenvalues [CMP18]. Measuring the system's total energy yields a spectrum of possible outcomes, represented by its eigenvalues. Each eigenvalue corresponds to a potential measurement for the system's total energy. There are multiple reasons why one would want to find the eigenvalues or the ground state of a Hamiltonian. For example, the ground state of molecules also determines their chemical reactivity, bonding, and stability, making it essential for understanding chemical reactions [KWPO+11]. And for combinatorial and optimization problems, finding the ground state corresponds to minimizing the objective function. Generally, the ground state represents the lowest possible energy state of the system and, therefore, the most stable configuration of the system.

One rather often used description is the spin-lattice Hamiltonian [SIR+21]. The model comprises discrete spin variables s_i, which can exist in either of two states, denoted as $+1$ or -1. These spins are organized in a lattice configuration, enabling them to interact with adjacent spins. For any two adjacent elements i, j on the lattice, there is an interaction that is described by a matrix entry $J_{i,j}$. In addition, every element i of the lattice has an external magnetic field (h_j).

$$H = -\sum_{i,j} J_{i,j} s_i s_j - \sum_i h_i s_i$$

In Figure 2.10, an example of a 2D spin-lattice is shown, where we look at the energy of a single spin and its nearest neighbors. In the example configuration, we can see that, for this

neighborhood, the system would flip the state of the red vertex since the energy decreases.



Energy for red vertex i with $J_{i,j} = 1$ and $h_i = 1$:

$H = -\sum_{i,j} J_{i,j} s_i s_j - \sum_i h_i s_i =$
$-1 \cdot [(-1 \cdot 1) + (-1 \cdot -1) + (-1 \cdot 1) +$
$(-1 \cdot 1)] - 1 \cdot [-1] = 3$

If vertex i changes to $+1$:
$H = 1 \cdot [(1 \cdot 1) + (1 \cdot -1) + (1 \cdot 1) +$
$(1 \cdot 1)] - 1 \cdot [1] = -3$

Figure 2.10: Hamiltonian as a 2D spin-lattice ising model

This is a reduced representation of Hamiltonians since it does not allow for more than quadratic terms. This is often enough since a lot of problems can be transformed into a quadratic one [CD22]. The model can also be extended for higher-order terms, allowing for cubic or polynomial interactions.

A good example of the Hamiltonian construction for a combinatorial problem is given in [WHJR18]. For a classical polynomial objective function with binary variables $x_1, ..., x_n$, the Hamiltonian can be constructed by substituting the objective function terms via $x_i = \frac{1-z_i}{2}, z_i \in \{-1, 1\}$. In the resulting polynomial $f(z) = \sum_{C \subset \{1,...n\}} \alpha_C \prod_{j \in C} z_j$, each occurrence of z_i can be replaced with the Pauli operator σ_i^z to finally obtain the Hamiltonian.

$$H = \sum_{C \subset \{1,...n\}} \alpha_C \otimes_{j \in C} \sigma_j^z$$

Where C is a subset of all variables or qubits, and the coefficient α_C is the coupling weight of the respective subset. [WHJR18] Every subset that is not needed will receive a coefficient of $\alpha_C = 0$.

Let us consider an easy example. We have a classical cost function $f(x) = 2x_1 + 8x_1x_2 + 32x_1x_2x_3$ and want to obtain the corresponding Hamiltonian. We replace the variables $x_i = \frac{1-z_i}{2}$ and receive the new polynomial: $f(z) = (1-z_1)+2(1-z_1)(1-z_2)+4(1-z_1)(1-z_2)(1-z_3)$ Finally, we use the Pauli operators instead of the variables z_i:

$$H = 7I - 7\sigma_1^z - 6\sigma_2^z - 4\sigma_3^z + 6\sigma_1^z\sigma_2^z + 4\sigma_1^z\sigma_3^z + 4\sigma_2^z\sigma_3^z - 4\sigma_1^z\sigma_2^z\sigma_3^z$$

It is important to note that we only consider Hamiltonians with the Pauli operator σ^z. In its more general form, the Hamiltonian can consist of all Pauli operators $\sigma^x, \sigma^y, \sigma^z$, making it much more complicated.

2.3.3 Quantum Phase Estimation

The Quantum Phase Estimation (QPE) algorithm was introduced by Kitaev, A. Y. [Kit95] and is used to estimate the phase corresponding to an eigenvalue of a given Hamiltonian.

However, such a method necessitates deep quantum circuits and therefore long coherence times [JEM+19a]. Nevertheless, we will take a look at their work and see how one could apply it to combinatorial problems, such as scheduling.

The first important step is to map the ground-state estimation to a phase estimation. This is done by constructing a unitary $\hat{U} = e^{iH\tau}$, which includes the Hamiltonian H. With the eigenvalue E of H mapped to the phase of its eigenvalue $e^{i2\pi\phi}$, the unitary changes to $\hat{U} = e^{i2\pi\phi\tau}$ with $E = \frac{2\pi\phi}{\tau}$. If it is then possible to estimate the phase ϕ of the unitary $\hat{U}$, one can also find the ground state of the Hamiltonian H. The phase-estimation algorithm (PEA) of Abrams and Lloyd [AL99] is doing exactly that. So let us look at the algorithm in more detail. It is comprised of two parts: 1. The wave function $|\psi\rangle$ represents the Hamiltonian. It is encoded on a qubit register S (state). 2. The register R (readout) is used for the phase estimation of the wave function $|\psi\rangle$.

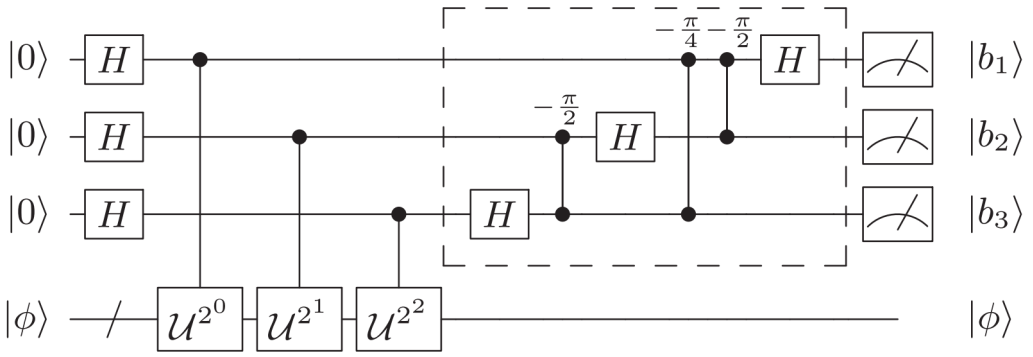


Figure 2.11: Quantum phase estimation as seen in [CCGFR20]

The wave function is encoded via the eigenstates of the Hamiltonian. However, these are, except for simple Hamiltonians, not known beforehand. Therefore, the input state is prepared with a good overlap with the ground state. It is then possible to evolve this state via the unitary U. The resulting state is given by:

$$|R\rangle \otimes |S\rangle = (\sum_n e^{(i2\pi\phi)n} |n\rangle) \otimes |\psi\rangle$$

with n enumerating the basis states of R. The inverse Fourier transform is then applied to the register R, resulting in an approximation of the phase ϕ, represented in binary. In Figure 2.11, this process is illustrated. The state is evolved by the unitaries U^{2^i}, and afterwards the inverse Fourier transform is applied in the dotted box.

One big problem with this approach is the number of qubits needed for the readout register R. To approximate a real number between 0 and 1 in binary for a chemically meaningful result, at least 20 qubits are required. This might not be a big factor for bigger problem instances, where the size of S will be substantially greater than for register R. However, for today's hardware, this does pose a significant problem since it also reduces the size of the problem instances one can examine.

2.3.4 Variatonal Quantum Algorithms

Variational Quantum Algorithms (VQA) are a subset of quantum algorithms that, in addition to their quantum part, make use of a classical subroutine. They are widely used for optimization problems, quantum chemistry, machine learning, or financial modeling [CAB+21]. They are also particularly well-suited for near-term quantum computers, where error rates and qubit connectivity limitations may make it challenging to implement more complex quantum algorithms.

The key idea behind variational quantum algorithms is to use a parameterized quantum circuit, also known as a variational quantum circuit, to represent a family of quantum states. In the classical subroutine, the parameters are adjusted based on an objective function. The structure of the variational circuit can vary. It is often chosen based on the specific problem being solved [FGG14, GBB+23] or just for maximal expressibility [LAS+22]. In figure 2.12, a rough outline of the workings of most VQAs can be seen.

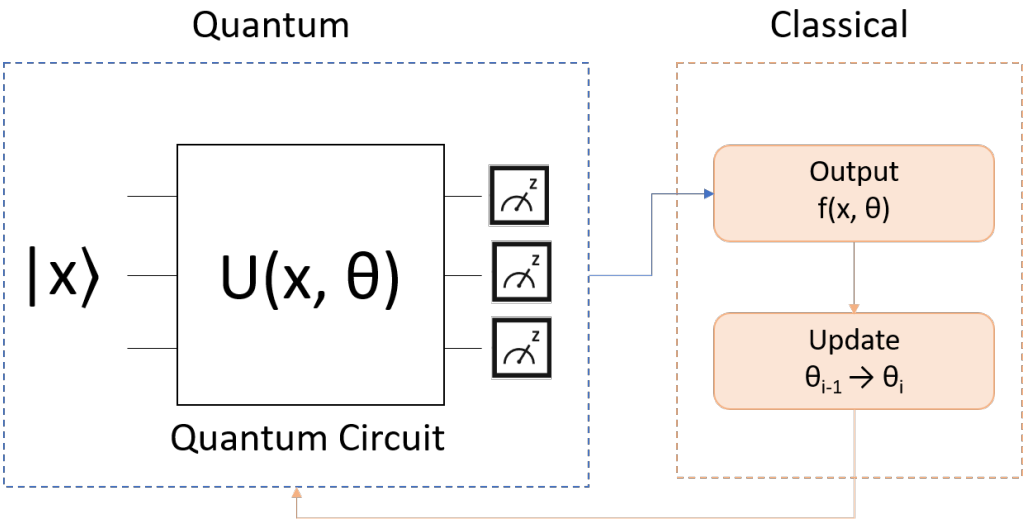


Figure 2.12: Schematic representation of the workflow of a VQA

In the following section, we will look at all components of a VQA in more detail.

The ansatz $U(x, \theta)$ is the parameterized quantum circuit. The tunable parameters $\boldsymbol{\theta}$ allow the circuit to represent a broad class of quantum states. The exact structure of the Ansatz circuit varies and is sometimes based on the algorithm, the problem, or a specific scheme to maximize expressibility [CAB+21]. Mostly, they consist of the same building blocks. Multiple rotation gates (which use the parameters $\boldsymbol{\theta}$ as input) and controlled operations, such as CNOT gates For more complex problems, larger and more intricate circuits with more parameters and gates may be required.

To apply VQAs to optimization problems, the problem is mapped onto a quantum system. This problem has no relation to the quantum system other than acting on the same space or a subset of qubits as the quantum system. An objective function $f(x, \theta)$ is defined based on the properties of the quantum state represented by the variational circuit [BMWV+23]. The objective function serves as a measure of how well the quantum state performs in solving the problem. In optimization problems, the goal is to minimize or maximize this objective function to find the best solution. For example, in machine learning tasks, the objective

function may involve the comparison of the quantum state's predictions with the true labels.

The circuit prepares a quantum state $|\psi(\boldsymbol{\theta})\rangle$ that depends on the specific values of the parameters $\boldsymbol{\theta}$. To extract information from the quantum state, it needs to be measured. The measurement outcomes are classical data points that are probabilistic, representing samples drawn from the probability distribution of the quantum state. These data points are then used to calculate the value of the objective function. For optimization problems, this step quantifies how well the current set of parameters performs in solving the problem.

The classical optimization process aims to find the values of the parameters $\boldsymbol{\theta}$ that lead to a minimum or maximum value of the objective function, thereby obtaining the best solution to the problem. Common classical optimization algorithms like gradient descent or Nelder-Mead are often used for this purpose [CAB+21]. The optimization process seeks to iteratively update the parameters to improve the performance of the variational circuit. In each iteration, the parameters of the variational circuit are updated based on the chosen classical optimization algorithm. The quantum circuit is evaluated, the objective function is calculated, and the optimization process continues until it converges to a satisfactory solution.

Parameter Shift Rule

For parameterized quantum circuits, it is also possible to directly compute the partial differential equations $\frac{\partial C_t(\boldsymbol{\theta})}{\partial \theta_j}$ of the cost function $C_t(\boldsymbol{\theta})$ via the parameter shift rule [Cro19].

The parameter-shift rule states that the derivative of a parameterized quantum circuit is proportional to the difference of two circuits with shifted parameters. Suppose we have an objective function $C(\boldsymbol{\theta})$ for our quantum circuit with parameterized gates: $U_G(\theta_j) = e^{-ia\theta_j G}$, where G is the Hermitian generator and a is a real constant. Note that it is, however, not necessary for the circuit to only consist of parameterized gates $U_G(\theta_j)$. The parameter shift approach is only applicable if the generator G of the gate $U_G(\theta_j)$, or its decomposed sub gates, have only two unique eigenvalues, e_0 and e_1. The derivative of a single parameter θ_j is given by:

$$\frac{\partial C(\boldsymbol{\theta})}{\partial \theta_j} = r(C(\boldsymbol{\theta} + \frac{\pi}{4r}\boldsymbol{\epsilon_j}) - C(\boldsymbol{\theta} - \frac{\pi}{4r}\boldsymbol{\epsilon_j})),$$

where $r = \frac{a}{2}(e_1 - e_0)$ is the shift constant and $\boldsymbol{\epsilon_j}$ is the unit vector for the parameter θ_j.

For example, we construct a parameterized circuit with a single qubit and a rotation $R_Y(\theta_0)$ gate. Our objective function is given as the Hamiltonian $H = 1 - \sigma_0^z$. Which is to say the optimal solution is reached for $\sigma_0^z = 1$. The cost function of our circuit is given by the expectation value of this Hamiltonian: $C_t(\theta) = \langle H \rangle_{|\psi(\theta)\rangle}$ for the quantum state $|\psi(\theta)\rangle$. For parameterized rotation gates $R_Y(\theta_j) = e^{-i\frac{1}{2}\theta_j Y}$, the shift constant is given as $r = \frac{1}{2}$. The derivative of our parameter θ_0 is then given by:

$$\frac{\partial C(\theta)}{\partial \theta_0} = \frac{1}{2}(C(\theta_0 + \frac{\pi}{2}) - C(\theta_0 - \frac{\pi}{2}) = \frac{1}{2}(\langle H \rangle_{|\psi(\theta_0 + \frac{\pi}{2})\rangle} - \langle H \rangle_{|\psi(\theta_0 - \frac{\pi}{2})\rangle})$$

We initialize our parameter $\theta_0 = \frac{\pi}{2}$ to create an equal superposition. The corresponding expectation value is $C_t(\theta_0) = \langle 1 - \sigma_0^z \rangle_{|\psi(\theta_0)\rangle} = 0.5$. For the two parameter shifts $\frac{\pi}{2}$ and $-\frac{\pi}{2}$, the expectation values are "0" and "1". The derivative of θ_0 is then given by:

$$\frac{\partial C(\theta)}{\partial \theta_0} = \frac{1}{2}(0 - 1) = -0.5$$

With the updated parameter $\bar{\theta_0} = \theta_0 - \alpha(-0.5) = 1$ for a simple learning rate $\alpha = 1$ the expectation value of H is given by:

$$C_t(\bar{\theta_0}) = \langle 1 - \sigma_0^z \rangle_{|\psi(\bar{\theta_0})\rangle} = 0$$

In the next section, we will look at two very important VQAs. The VQE and the Quantum Approximation Optimization Algorithm (QAOA) Both are later used as a comparison against the F-VQE algorithm.

VQE

In 2014, Peruzzo and McClean et al. introduced the Variational Quantum Eigensolver (VQE) as an innovative approach for addressing eigenvalue and optimization problems that remain beyond the reach of conventional classical computing methods [PMS+14].

The VQE optimizes an upper bound for the ground state λ_0 of the given Hamiltonian H and the quantum state $|\psi(\boldsymbol{\theta})\rangle$. It is given by the equation:

$$\lambda_0 \leq \langle \psi(\boldsymbol{\theta})| H |\psi(\boldsymbol{\theta})\rangle = \langle H \rangle(\boldsymbol{\theta})$$

It is now the goal to find the optimal parameterization of the ansatz circuit to minimize the expectation value of the Hamiltonian. In an ideal case, the expectation value should not only be an upper bound for the ground state but also be equal to it. The ansatz is most commonly initialized as the state $|0\rangle^N$ and can be expressed as a unitary gate $U(\boldsymbol{\theta})$ over all qubits N. The expectation value of the Hamiltonian can thus be rewritten as $\langle H \rangle(\boldsymbol{\theta}) = \langle 0|^N U^\dagger(\boldsymbol{\theta}) H U(\boldsymbol{\theta}) |0\rangle^N$. To compute the expectation value, one can write the Hamiltonian in a form that is directly measurable on a quantum computer, as we have done in the previous section. We can then minimize the energy of our problem by minimizing the sum of the expectation values of each Pauli string $\otimes_{j \in C} \sigma_j^z$.

$$E_{ground} = \min_{\boldsymbol{\theta}} \sum_{C \subset \{1,\dots n\}} \alpha_C \langle 0|^N U^\dagger(\boldsymbol{\theta}) \otimes_{j \in C} \sigma_j^z \; U(\boldsymbol{\theta}) |0\rangle^N$$

The energy for each Pauli sting is $E_C = \alpha_C \langle 0|^N U^\dagger(\boldsymbol{\theta}) \otimes_{j \in C} \sigma_j^z \; U(\boldsymbol{\theta}) |0\rangle^N$ is also the expectation value for that Pauli string and can be computed directly on the quantum device.

But one should note that it is not necessary to have this explicit Pauli string representation for the Hamiltonian. It is also possible to minimize the energy with an unknown objective function. The energy is given by $E(\boldsymbol{\theta}) = \sum_{i=0}^{N} p_i(\boldsymbol{\theta}) f(x_i)$. where x_i is the solution for the state $|i\rangle$ and $p_i(\boldsymbol{\theta})$ is the probability to measure the state $|i\rangle$ [ZMS+23]. The expectation value can then be computed with, for example, a simple average over a number of measurement samples (K).

$$\langle H \rangle(\boldsymbol{\theta}) \equiv \frac{1}{K} \sum_{i=0}^{K} E_i,$$

where E_i is the energy corresponding to the kth measured bit string. This can also be relevant if the objective function cannot easily be converted to a Hamiltonian or if the resulting Hamiltonian is not sparse enough. This could be the case if the objective function contains inequalities and slack variables are needed or if there are too many couplings between

qubits, making the Hamiltonian denser. In both cases, one might consider not formulating the Hamiltonian directly.

One of the VQE's key strengths is its compatibility with various quantum hardware, making it ideal for early quantum computing exploration. It optimizes performance by leveraging specific quantum architecture strengths and has an intrinsic ability to mitigate errors through variational techniques. [MRBAG16]

QAOA

As for the VQE The QAOA [FGG14] tries to find an upper bound for the ground state λ_0 of the given Hamiltonian H. However, the QAOA uses a different approach to achieve that. For one, it uses two sets of parameters for the quantum state $|\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle$. The ground-state energy is given by:

$$E_{ground} = \min_{\boldsymbol{\beta},\boldsymbol{\gamma}} \langle\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})| H_c |\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle$$

It also uses a fixed ansatz with a variational depth p. The ansatz circuit is dependent on the Hamiltonian of the objective function. It is inspired by the quantum annealing protocol, where a system is initialized in the easy-to-prepare ground state of a local or mixer Hamiltonian $H_x = \sum_i \sigma_i^x$, which is then slowly transformed to the problem Hamiltonian H_c [FT11]. The system Hamiltonian $H(t)$ is dependent on the time t. The time evolution of the system Hamiltonian $H(t)$ is given by: $H(t) \equiv tH_c + (1-t)H_x$. The QAOA, however, does not use a smooth transition from one Hamiltonian to the other. Instead, it utilizes a trotterized version where one can decompose the total time evolution into short-time operations during which the system Hamiltonian is approximately time-independent for each step. The trotterized ansatz state consists of alternately applying $e^{-i\gamma_i H_c}$ and $e^{-i\beta_i H_x}$:

$$|\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle = e^{-i\beta_p H_x} e^{-i\gamma_p H_c} \cdots e^{-i\beta_1 H_x} e^{-i\gamma_1 H_c} |+\rangle^N$$

The number of parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ is given by the depth p of the ansatz circuit. For each layer p, two parameters are needed. As the number of layers p increases, QAOA demonstrates a consistent improvement and eventually excels in the asymptotic $p \to \infty$ limit [FGG14]. At the same time, as the depth and number of parameters increases, local optimization methods become unreliable [SA19]. As the limitations of these techniques become more apparent, unlocking the potential of QAOA necessitates progress in the optimization of variational parameters.

3 Related Work

Solving combinatorial problems like the JSSP, Max-Cut [DGS18] or the nurse scheduling problem [LHDC09] is a challenging task for which researchers have tried a vast variety of solutions [ZDZ+19]. We will therefore explore some of the solutions to give the reader a more comprehensible impression of how one can go about solving this family of problem instances. An overview of the complexity of various scheduling variants and optimization strategies is provided in [ZDZ+19]. Furthermore, we will give a broad overview of different quantum variational algorithms and their peculiarities, advantages, and disadvantages. And lastly, we want to discuss a new branch of optimizations in which we move away from classical algorithms such as gradient descent and use a quantum computer directly to optimize our objective function.

3.1 Solving Scheduling Problems

First, we will look at different optimization strategies to solve a wide variety of scheduling problems. We will discuss both classical solutions and quantum ones.

On-Call Operator Scheduling for Satellites with Grover's Algorithm

3.1.1 Grover's Algorithm

One of the most important papers for this thesis is the work of the authors Antonius Scherer et al. at the German Space Operating Center (DLR) in their paper "OnCall Operator Scheduling for Satellites with Grover's Algorithm" [SGGC+21]. This implementation will also be used in this thesis as a comparison for the variational quantum algorithms we will present later on. In their paper, the application of Grover's quantum search algorithm for scheduling problems with constraints is introduced. Their specific problem is the creation of valid schedules for on-call spacecraft operators at the DLR. This is particularly interesting since we will also be working with the same problem later on. We will later explain the problem in more detail, but the idea is that the DLR is supervising multiple satellite missions that need to be attended to. The goal is to find a schedule over **d** days, **o** operators, and **p** positions so that every satellite mission (position) is attended to. At the same time, a set of constraints needs to be observed. In their paper, they only used three constraints as a proof of concept. TODO: correct number of constraints Later, we will increase the number of constraints to six and also use two goals, which are softer constraints. The three constraints the authors used were:

A Per tuple of day and position, at least one operator needs to be assigned.

B Out of three consecutive days, an operator is only allowed to work two.

C An operator can be assigned to at most one position a day.

The paper proposes a method to encode variables of the problem into a quantum state to use as the input for Grover's algorithm. This is done via a method called binary encoding as opposed to the more classical one-hot approach and can reduce the required number of qubits to encode the problem. One-Hot represents each category as a binary vector, where each category corresponds to a unique binary pattern, with a single "1" indicating the presence of a category and "0s" elsewhere. Let us look at a very simple example with four categories: "blue", "red", "yellow" and "green". To encode these in One-Hot, we create a binary vector with four entries, one for each category.

$$b = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \quad r = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \quad y = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \quad g = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

For binary encoding, each category is encoded as a binary string. This reduces the number of needed bits to log_2.

$$b = \begin{bmatrix}0\\0\end{bmatrix}, \quad r = \begin{bmatrix}0\\1\end{bmatrix}, \quad y = \begin{bmatrix}1\\0\end{bmatrix}, \quad g = \begin{bmatrix}1\\1\end{bmatrix}$$

However, in their problem, the authors have three sets of categories: days, operators, and positions, while binary encoding can only be applied to one of them. The biggest impact would be achieved if the set with the most categories was represented as binary. This, however, is a challenge since, with a binary encoding of the days, the size of the representable state space is reduced and valid schedules may be removed. This representation for 2^n days can be written as:

$$d^n_{o,p} = \begin{cases} 0..0 & \text{operator } \mathbf{o} \text{ is assigned to position } \mathbf{p} \text{ at day } 0 \\ \cdots & \\ 1..1 & \text{operator } \mathbf{o} \text{ is assigned to position } \mathbf{p} \text{ at day } 2^n \end{cases}$$

With the binary encoding of the days, every operator would be forced to work every position exactly once within the given number of days. This is an unwanted constraint for our problem definition and would lead to bad results. Instead, they use operators as a set of binary categories. This also results in an encoding that intrinsically adheres to constraint A. It thus only reduces the state space by removing invalid solutions. For 2^n operators, the binary representation yields:

$$o^n_{d,p} = \begin{cases} 0..0 & \text{operator } 0 \text{ is assigned to position } \mathbf{p} \text{ at day } \mathbf{d} \\ \cdots & \\ 1..1 & \text{operator } 2^n \text{ is assigned to position } \mathbf{p} \text{ at day } \mathbf{d} \end{cases}$$

The main part of the algorithm is the oracle, in which the constraints are encoded. This is accomplished by incrementing a counter register every time a constraint is violated. Only for states with zero conflicts is the amplitude negated, resulting in an amplification of these states by the diffuser operator. We will now exemplarily look at how the authors implemented constraint C in their oracle. In this constraint, for any day and any operator, we want that the operator not be scheduled for two or more positions. If this is violated, the counter register is incremented.

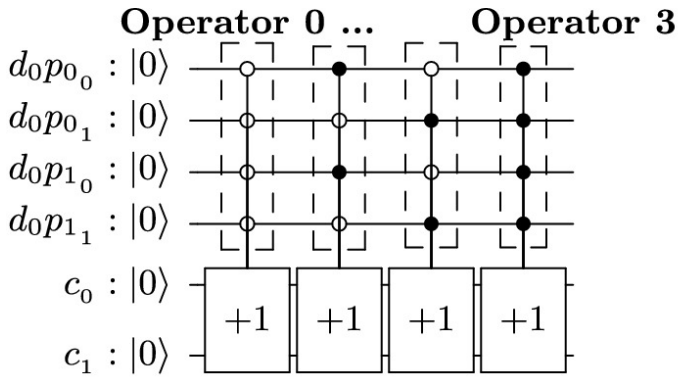


Figure 3.1: Quantum oracle for constraint C as seen in [SGGC+21]

Figure 3.1 depicts an example oracle constraint C with one day, four operators, and two positions. Each block checks if one operator works on both positions p_0 and p_1. The respective positions are encoded on the first two and on the last two qubits. If the qubit allocation for the positions is the same, the operator with that particular representation is working on both positions on the same day. Resulting in an increase in the counter register. Note that even though four "+1" gates are depicted in the circuit, only one can be activated simultaneously.

One of the main challenges in this work is that it is not possible to make an accurate guess as to how many valid solutions there are. This is a major drawback for Grover's algorithm since it needs the number of valid solutions a priori to calculate the number of iterations. If it is unknown, a vast effort has to be extended to approximate the right number of iterations. Another difficulty is finding the upper bound of constraint violations. This is needed to initiate the counter register in the right size and to prevent overflows, which would lead to the wrong classification of correct and incorrect solutions. In their work, the minimal number of counter qubits and Grover iterations was empirically evaluated for a few small problem cases. In figure 3.2, four different problem instances are given. For each case, the success rate is evaluated for one to six counterqubits.

In their follow-up paper, "Evolving Spacecraft Quantum On-Call Scheduling" [PSPS23], the authors addressed the problem of the unknown number of Grover iterations. They implemented the algorithm introduced in [BBHT98] to increment the number of iterations until an adequate solution is found. They also introduced a number of improvements to a variety of parts, resulting in a cost reduction ranging from 26% up to 90% depending on the specific problem and used constraints. In figure 3.3, three major improvements are compared against the work in the previous paper. In the first graph, the authors improved the implementation of Constraint C using a different operation to detect violations. In the second diagram, the results of a new method for Constraint B are presented. Although the method uses more qubits, the sharp reduction in gate costs is a reasonable trade-off in the eyes of the authors. And in the last chart, the gate cost for a quantum fourier transformation-based counter register is shown.

In both papers, their methodology is validated using Qiskit on smaller problem instances (up to 30 qubits) suitable for existing quantum simulators. The authors successfully showcased the process of finding a valid schedule for the "on-call spacecraft operator scheduling" problem through a quantum search algorithm.

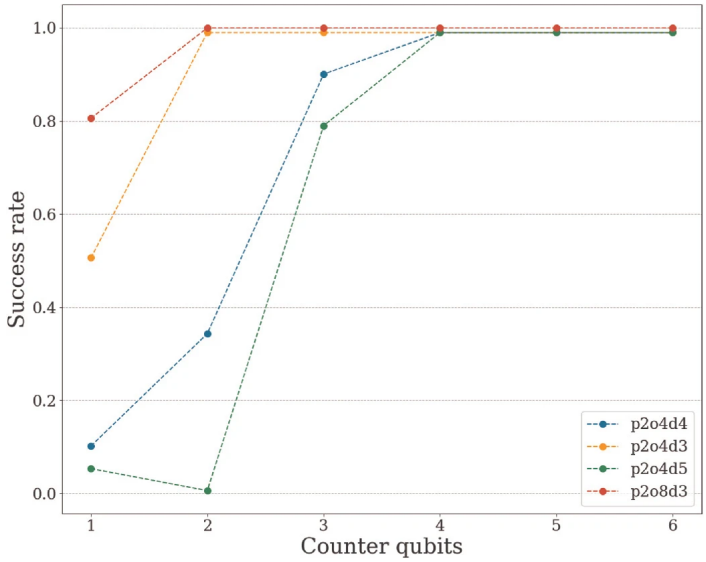


Figure 3.2: Success rate for different number of counter register qubits as seen in [SGGC+21]

3.1.2 QAOA

Another directly linked paper, "Quantum Approximation for Wireless Scheduling" by Choi, Jaeho et al. [COK20], uses the QAOA to solve scheduling problems. Since we will also compare our solution with an implementation of QAOA, we will now look at how this paper addresses a wireless scheduling problem for device-to-device (D2D) wireless networks [KCM16].

D2D networks are special cases of peer-to-peer (P2P) networks where the connectivity of their devices is determined by a shared medium with interference. Furthermore, connectivity is dependent on the distance between devices. A link between them only exists if they are in the same transmission range. In contrast, in a P2P network, communication can be implemented via a common IP connection. In a D2D network, direct connections between pairs of user devices can be established without the need to pass through a central base station, unlike an IP connection. In particular, each device caches popular files independently, according to a certain optimal distribution. When a user needs a file not already present in its own cache, it obtains it from one of its neighbors through a short-range D2D link. As user density increases, the aggregate storage capacity of the D2D network increases linearly with the number of users, while the average communication distance decreases.

Suppose such a D2D wireless network consists of a set of one-hop links, where each consists of one transmitter and its associated receiver. Each link has a queue of data that needs to be sent. Naturally, the queue backlog should be kept to a minimum for optimal workflow. If some links are sending data at the same time, it might lead to interference and corrupt the transmission. This behavior can be represented as a conflict graph, where each link is a vertex. If two links are connected by an edge, interference would occur if both were active.

The conflict graph can be formulated by its adjacency matrix $E_{i,j}$ for links l_i, l_j:

$$E_{i,j} = \begin{cases} 1 & \text{if } l_i \text{ and } l_j \text{ interfere} \\ 0 & \text{otherwise} \end{cases}$$

The goal is to find a schedule of links where no interference occurs. This means no two

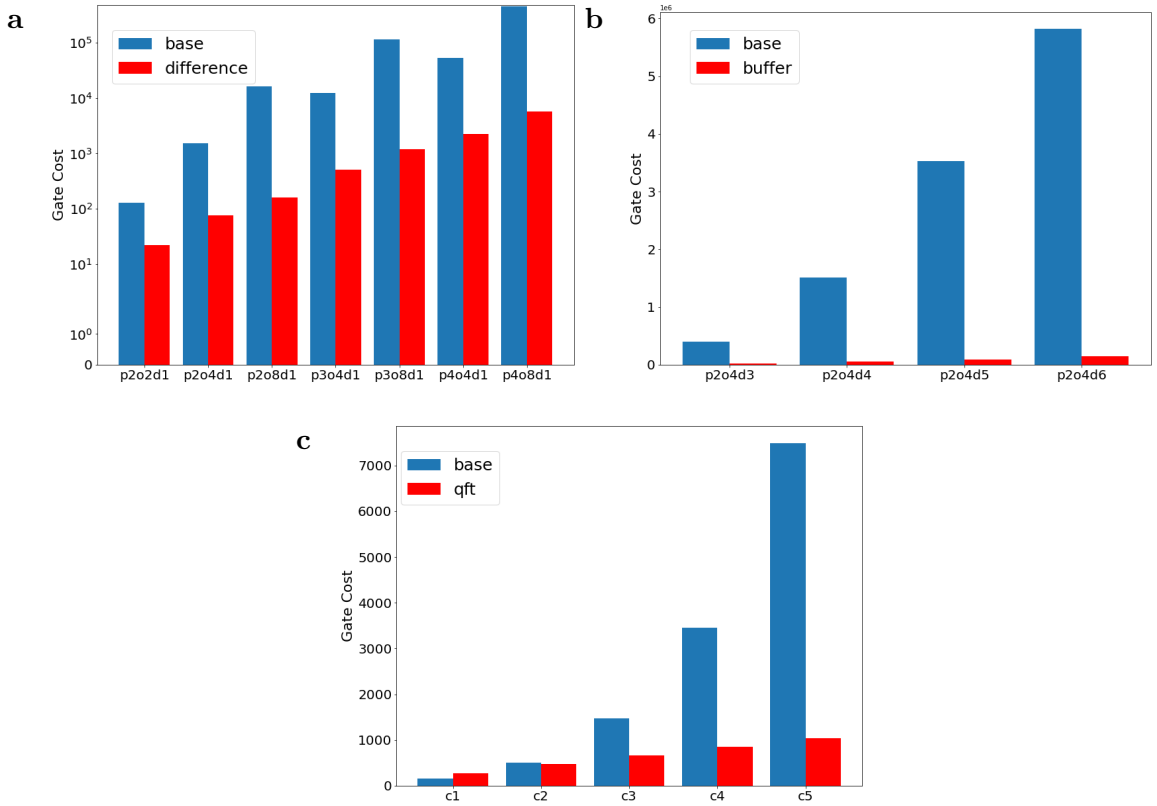


Figure 3.3: Gate cost for three major improvements as seen in [PSPS23]. (a) Improved implementation of Constraint C The scale is logarithmic, so we see a reduction of a factor of roughly 100 in some basic cases. (b) Improved implementation of Constraint B. Scale is 100.000 steps. (c) Improved implementation of the counter register Comparison of different counter register sizes with a constant control qubit amount of 3.

connected links can be present in this schedule. This can now be formulated as a max-weight independent set (MWIS) problem [Bas01], where the weights w_i of each link depend on their queue backlog:

$$\begin{aligned} \text{maximize} \quad & \sum_i w_i x_i \\ \text{subject to} \quad & x_i + x_j + E_{i,j} \leq 2 \end{aligned}$$

where $x_i = 1$ if l_i is scheduled and 0 otherwise. The above formulation ensures that conflicting links are not scheduled simultaneously: if $E_{i,j} = 0$, then $x_i + x_j + E_{i,j} \leq 2$ no matter what link is scheduled. In contrast, if $E_{i,j} = 1$ at most, one of the two links can be scheduled.

To solve this problem with the QAOA, the objective function and constraint need to be encoded in a Hamiltonian. The authors use two different Hamiltonians, H_o and H_c, for the objective and constraint, respectively. The objective Hamiltonian H_o is rather straight forward; the main difference to chapter 2.3.2 is the transformation from maximization to

minimization. It is given by:

$$H_o = \sum_i \frac{1}{2} w_i \sigma_i^z$$

The constraint Hamiltonian H_c, on the other hand, is rather complex. In a naive implementation, one would need to encode the set of inequalities, which is very costly. The authors instead look at three separate cases:

A A set of all link pairs (l_i, l_j) where both are not scheduled

B A set of all link pairs (l_i, l_j) where only one is scheduled

C A set of all link pairs (l_i, l_j) where both are scheduled

As we saw earlier in cases A and B, the inequality is already fulfilled. It is therefore enough to look at case C, where both adjacent nodes of the conflict graph are scheduled and an interference is present:

$$C = \sum_i \sum_j (x_i \wedge x_j)(w_i + w_j)$$

where $x_i \wedge x_j$ can be written as $1 - x_i - x_j + x_i x_j$. This can then be converted to the Hamiltonian H_c, where we yet again have to transform C to a minimization:

$$H_c = \sum_i \sum_j -\frac{1}{4}(\sigma_i^z + \sigma_j^z - \sigma_i^z \sigma_j^z)(w_i + w_j)$$

Based on the definitions of H_o and H_c, the problem Hamiltonian H_p can be defined as $H_p = H_o + \rho H_c$, where $\rho \geq 1$ is the penalty rate. With this, the Hamiltonian H_p is used for QAOA, as described in 2.3.4.

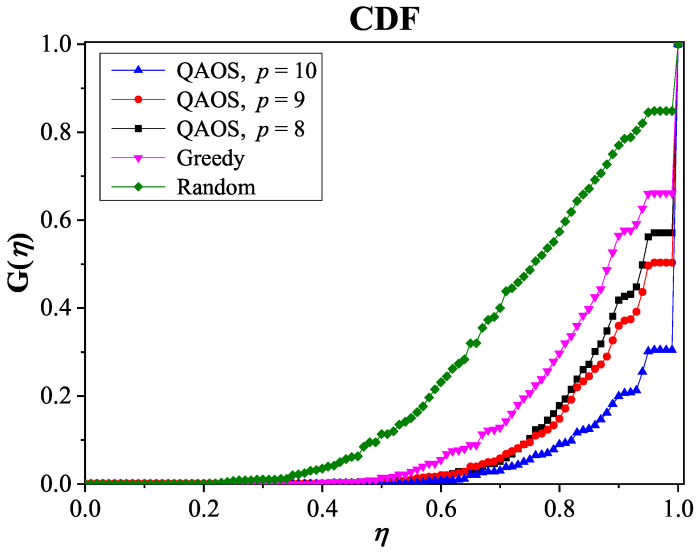


Figure 3.4: CDF $G(\eta)$ of approximations ratio η for different algorithms as seen in [COK20]

The performance of their implementation was evaluated and compared to random search and greedy search algorithms. The evaluations were conducted on randomly generated conflict graphs with 10 nodes. As can be seen in figure 3.4, the implementation consistently outperformed the other methods across various conflict graph instances. The performance is quantitatively measured with the approximation ration $\eta = \frac{a}{b}$, where a is the best solution of the algorithm and b is the true best solution. In the graph, the cumulative distribution function (CDF) $G(\eta)$ for each algorithm is shown. The CDF sums up the total likelihood of η up to the current point, making solutions with an increase for lower values of η worse than ones with a steep increase for higher values of η. They achieved optimal solutions in a significant proportion of cases, indicating the implementation's effectiveness in accurately solving the wireless scheduling problem.

3.1.3 VQE

There have also been solutions to a variety of scheduling problems using the VQE algorithm. In the paper "Variational Algorithms for Workflow Scheduling Problems in Gate-Based Quantum Devices," the authors Julia Plewa et al. [PSR21] solve the workflow scheduling problem [TPMR20], which is related to the earlier JSSP. They use a VQE and QAOA implementation to solve this problem and compare the results. In addition, they use different encoding schemes to represent the problem. Together with the two encoding schemes One-Hot and Binary we already know, they used a third one: Domain Wall Encoding [Cha19].

Let us first look at the difference between workflow scheduling and the JSSP. In contrast to the JSSP, jobs have intrinsic dependencies on other jobs, which have to be completed before the job can be begun. The set of job dependencies can be decomposed into a set of R paths that lead from the starting job to the final job. A binary matrix $P(i, k)$ defines if a job J_i is on the path k or not. Jobs are also not restricted to a specific machine. Instead, each machine M_j has an associated cost $C(i, j)$ for running a job J_i. The time $T(i, j)$ a machine M_j needs to complete a job J_i is analogous to the JSSP formulation. Additionally, the authors introduced a new deadline constraint, d, where all jobs must be completed before the deadline is reached. To satisfy this constraint, every path in R has to have an execution time smaller than d. This means the goal of this problem is no longer the shortest completion time but rather the lowest total cost while staying within the deadline. Let us consider a simple example of six jobs J_1, J_2, J_3, J_4, J_5, and J_6 and two machines M_1, M_2. We will ignore the deadline constraint in this example and just show the different encoding schemes.

Jobs	Machines with respective costs for the job	
J_0	$M_0(2)$	$M_1(3)$
J_1	$M_0(1)$	$M_1(2)$
J_2	$M_0(2)$	$M_1(4)$
J_3	$M_0(1)$	$M_1(2)$
J_4	$M_0(2)$	$M_1(3)$
J_5	$M_0(4)$	$M_1(2)$

Table 3.1: Machines with respective cost for job.

The solution to our example workflow scheduling problem can be seen as a directed acyclic graph (DAG) in figure 3.5.

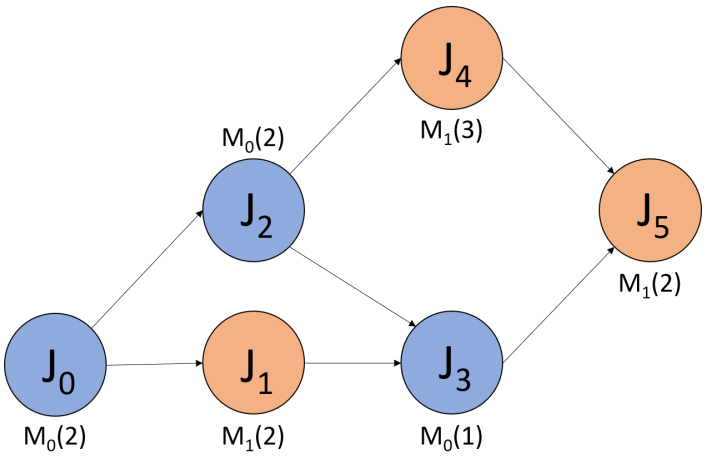


Figure 3.5: DAG for workflow scheduling with six jobs and two machines.

Next, we will take a look at the formulation of the objective function. We begin with the simple minimization problem:

$$\text{minimize} \quad \sum_{i}^{J} \sum_{j}^{M} c_{i,j} x_{i,j}$$

where the variable $x_{i,j}$ is 1 if job J_i is scheduled on machine M_j. Other than in the formulation in Section 3.1.2, the constraints are directly part of the objective function and do not construct separate constraint Hamiltonians. This is achieved by introducing penalty terms, which increase the cost if the constraint is not met. The first constraint is the deadline constraint.

$$\sum_{i}^{J} \sum_{j}^{M} t_{i,j} x_{i,j} \leq d$$

As we discussed in the previous section, it is not trivial to implement such an inequality. In this case, we have to deal with one inequality for each path k in R. The set of inequalities is given by:

$$\sum_{k}^{R} (\sum_{i}^{J} \sum_{j}^{M} p_{i,k} t_{i,j} x_{i,j} \leq d) \tag{3.1}$$

To represent an inequality in our objective function, we first have to convert it to an equality. For that, we define the set of slack variables $y_{k,l}$ according to 2.1, where l is the bit index and k is the path index. Then we just have to apply 2.1 and 2.2:

$$O_{time}(X, Y) = \sum_{k}^{R} (d - \sum_{i}^{J} \sum_{j}^{M} p_{i,k} t_{i,j} x_{i,j} + \sum_{l} 2^l y_{k,l})^2$$

Before we explain the last constraint, we will take a look at the different encoding schemes. For One-Hot, this is rather straight forward: every entry in X is a variable o in the input vector.

$$X_{i,j} = o_{i,j}, \text{ for job } J_i \text{ and machine } M_j$$

To define X for binary encoding is, on the other hand, not as easy. For the machines M is a number of variables $m_{w,i}$ with $w \in \{0, ..., W\}$, where W is the maximal bit length to represent all machines. M, as binary numbers, and i, the index of the job, are needed for each job-machine pair.

$$X_{i,j} = \prod_{w}^{W} (1 - (m_{w,i} - b_{j,w})^2), \text{ for job } J_i \text{ and machine } M_j$$

where $b_{j,w}$ is the bit-wise allocation for machine M_j. With this, we only need the W number of variables for each job, J_i, instead of $|M|$. Finally, domain wall encoding is dependent on pairs of neighboring bits rather than the bit values directly, which allows for the saving of a single bit. The information is stored via the position of a bit transition $d_{i-1} = 1$ to $d_i = 0$, where only one such transition is allowed in the sequence. Additionally, two virtual bits with fixed values: $d_{-1} = 1$ and $d_{|M|} = 0$ can be imagined to account for the first and last bit transition.

Machines	Domain Wall Encoding
0	10000
1	11000
2	11100
3	11110

Table 3.2: Domain Wall Encoding for Four Machines

The matrix X is then given by the difference of subsequent bits:

$$X_{i,j} = \begin{cases} 1 - d_{i,j} & \text{, for job } J_i \text{ and machine } M_0 \\ d_{i,j-1} - d_{i,j} & \text{, for job } J_i \text{ and machines } M_m,\ 1 \leq m \leq |M| - 2 \\ d_{i,j-1} & \text{, for job } J_i \text{ and machine } M_{|M|-1} \end{cases}$$

However, for an invalid vector with more than one bit of transition, X might also contain the value -1. This can be avoided by either squaring the value or by excluding invalid vectors via another constraint.

The final constraint makes sure all invalid bit configurations are punished. This is encoding-specific, but not that complicated. For one-hot encoding, a valid bitstring can only contain a single 1 for each machine-job pair.

$$\sum_{i}^{J} (1 - \sum_{j}^{M} x_{i,j})^2$$

For binary encoding, every dummy machine created for a machine count $|M| \mod 2 \neq 0$ must be punished.

$$\sum_{i}^{J} \sum_{u}^{U} \prod_{w}^{W} (1 - (m_{w,i} - b_{u,w})^2)$$

where U is the set of dummy machines. And for the domain wall encoding, there is only one bit of transition allowed. We can simply count all transitions:

$$\sum_{i}^{J}((1-d_{i,0})^2+\sum_{j=1}^{|M|-2}(d_{i,j-1}-d_{i,j})^2+d_{i,|M|-1}^2)$$

In their research, the authors also introduce mixer operators for the different encoding schemes for QAOA. These mixer operators are chosen to ensure that, during transitions between different configurations, the optimization process stays within the set of valid solutions, known as the feasible subspace. The default mixer operator used in QAOA is given as H_x in section 2.3.4, which essentially acts as a bit flip operator. However, while H_x is a straightforward choice and allows for transitions between states, it has limitations. Specifically, it can sometimes generate invalid states or states that violate problem constraints, leading to incorrect solutions. In their paper, the custom mixer operators are especially beneficial when applied to one-hot encoding and domain wall encoding. For their comparison, the authors evaluated the performance of the VQE and QAOA while considering various factors, including the encoding methods, classical optimization algorithms, and the selection of parameters for quadratic unconstrained binary optimization (QUBO). In their experiments, the results indicate that for smaller problem instances, VQE tends to outperform QAOA in terms of finding optimal solutions. In figure 3.6, the percentage of incorrect solutions for VQE and QAOA is compared for the domain wall and binary encoding schemes and for different classical optimizers. While the two algorithms perform quite similarly for the optimizers "L-BFGS-B" and "NELDER-MEAD", VQE outperforms QAOA in the other two.

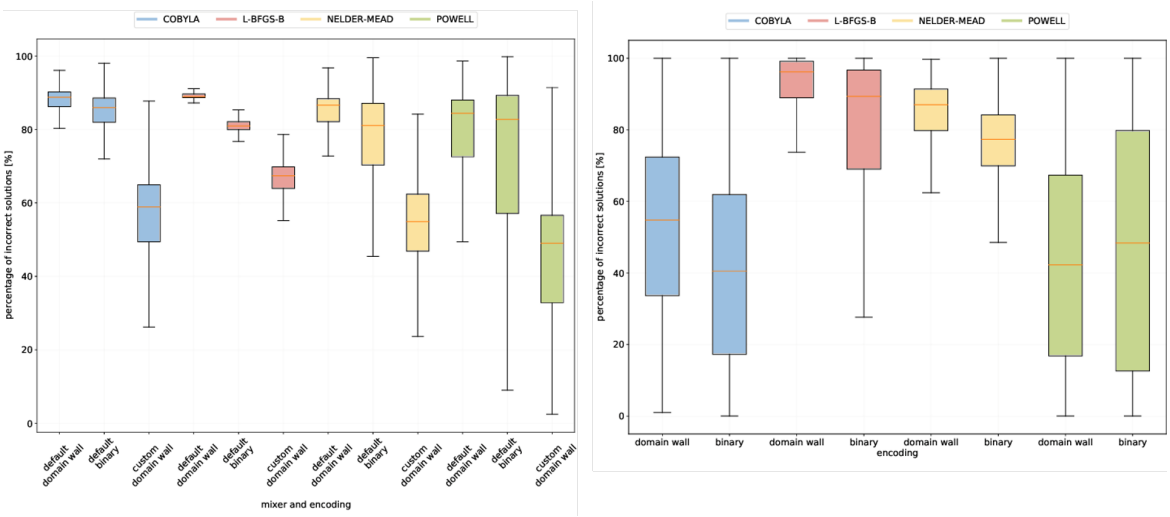


Figure 3.6: Percentage of incorrect solutions as seen in [PSR21]. (left) QAOA; (right) VQE

And while increasing the QAOA parameter p improves results up to $p = 2$, it does not for $p \geq 3$. Moreover, the choice of classical optimizer plays a substantial role in influencing the performance of VQE. Different classical optimization algorithms yield varying results when combined with VQE. It's worth noting that the benefits of custom mixers in QAOA may not translate as effectively into real quantum computers due to the presence of noise. Lastly, the

authors emphasize the critical role of selecting appropriate weights for the objective function and constraints.

3.2 Variational Quantum Algorithms

Other variational algorithms have been introduced that can simulate the real or imaginary time dynamics of quantum systems [LDG+21]. While there has been extensive research on finding the ground states of physical systems, the exploration of excited states has not received as much attention [JEM+19b]. Hamiltonian time evolution can be used to sequentially calculate its energy levels and provide a larger use case than only ground state evaluation.

The Hamiltonian simulation problem [Llo96] represents the standard approach for simulating the time evolution of a quantum system. In this problem, we assume the quantum system whose time evolution we wish to simulate consists of n qubits, and we want to simulate its time evolution for time t, in the sense that we are provided with the initial state $|\psi_0\rangle$ and we want to compute the state of the system at time t, $|\psi_t\rangle$. The goal of an efficient simulation is to solve the problem in polynomial time for n and t. The relation between the output state at time t and the initial state at time 0 is given by the Schrödinger equation for time-independent Hamiltonians:

$$|\psi_t\rangle = e^{-iHt} |\psi_0\rangle$$

By encoding quantum states with a parameterized circuit $|\psi(\boldsymbol{\theta}_t)\rangle$, the evolution of the state can be mapped to the evolution of the parameters $\boldsymbol{\theta}_t$) controlling the circuit. The time evolution for a small time-step γt is then given by:

$$|\psi_{t+\gamma t}\rangle = e^{-iH\gamma t} |\psi(\boldsymbol{\theta}_t)\rangle$$

The evolution can be approximated by a sequence of short-time evolutions using the well-known Trotter product formula. With N time steps of size $\tau = \gamma t/N$, one obtains $e^{-iH\gamma t} \approx U(\tau)^N$, with $U(\tau) = e^{-ih_k H_k \tau} \cdots e^{-ih_1 H_1 \tau}$, where H_k are tensor products of Pauli operators and h_k are real numbers. The accuracy of this approximation can be improved using higher-order Trotter product formulas. The authors Barison et al. [BVC21] aim to maximize the overlap between the evolved parameterized state $|\psi_{t+\gamma t}\rangle$ and the state $|\psi(\boldsymbol{\theta}_t + \bar{\boldsymbol{\theta}}_t)\rangle$:

$$L(\bar{\boldsymbol{\theta}}_t, \gamma t) = \frac{1 - |\langle \psi(\boldsymbol{\theta}_t)| e^{iH\gamma t} |\psi(\boldsymbol{\theta}_t + \bar{\boldsymbol{\theta}}_t)\rangle|^2}{\gamma t^2}$$

where $\bar{\boldsymbol{\theta}}_t$ is a vector of parameter variations. The optimal optimization step $\bar{\boldsymbol{\theta}}_t$ is determined by the gradient $\frac{\partial}{\partial \theta_j} L(\bar{\boldsymbol{\theta}}_t, \gamma t)$. They use a quantum circuit of the form:

$$|\psi\rangle = V_p U_p(\theta_p) \cdots V_1 U_1(\theta_1)$$

where only the gates $U_j(\theta_j)$ depend on a parameter θ_j. The parameterized gates $U_j(\theta_j)$ are single qubit rotation gates, which allow the authors to compute the gradient via the parameter shift rule, as outlined in Section 2.3.4. To find the optimal parameter perturbation $\bar{\boldsymbol{\theta}}_t$, the authors use the gradient descent update rule:

$$\bar{\boldsymbol{\theta}}_t = \bar{\boldsymbol{\theta}}_{t-1} - \eta \nabla L(\bar{\boldsymbol{\theta}}_{t-1}, \gamma t)$$

until $L(\bar{\boldsymbol{\theta}}_{t-1}, \gamma t)$ converges to a desired threshold v. After convergence, the new parameters are given by:

$$\boldsymbol{\theta}_{t+\gamma t} = \boldsymbol{\theta}_t + \bar{\boldsymbol{\theta}}_t$$

They also show that their approach does not suffer from barren plateaus. This is despite using a global objective function, which is shown to be especially vulnerable to vanishing gradients [CSV+21]. Their approach is based on the fidelity between the two states $|\psi_{t+\gamma t}\rangle$ and $|\psi(\boldsymbol{\theta}_t + \bar{\boldsymbol{\theta}}_t)\rangle$. It can be demonstrated that when the two states undergo an infinitesimal transformation, the gradient's variance retains a non-zero lower bound [HK21]. This is the case since at every step, the infidelity between the states is optimized and, therefore, transformed. They also demonstrate an approach in which they replace the global objective function with a local one with the same minimum. Since local objective functions do not suffer from the same barren plateaus as global ones do [CSV+21], this should avoid barren plateaus even if the transformation for each optimization step is not big enough. The authors implemented both approaches and compared them later on. The used circuit design consists of d blocks, where one block is equivalent to the Trotter-Suzuki approximation $U(\tau)$.

$$|\psi\rangle = \prod_{l=1}^{d} c_l(\theta^{(l)}) = \prod_{l=1}^{d} \left[\prod_{i=1}^{N} R_\alpha^{(i)}(\theta_i^{(l)}) \right] \left[\prod_{j=1}^{N-1} e^{-i\theta_j^{(l)} \sigma_j^z \sigma_{j+1}^z} \right]$$

Every block $c_l(\theta^{(l)})$ has unique parameters. $\theta^{(l)}$ and consists of single qubit rotations $R_\alpha^{(i)}(\theta_i^{(l)})$ and entangling two-qubit gates $e^{-i\theta_j^{(l)} \sigma_j^z \sigma_{j+1}^z}$. For $\alpha = x$, a block c_l is equivalent to $U(\tau)$, but a more general formulation is also possible for $\alpha = \{x, y\}$.

In their comparative analysis between their own algorithm (p-VQD) and the TDVA (Time-Dependent Variational Algorithm), the authors observed that, for a fixed number of samples, p-VQD exhibited a significant advantage.

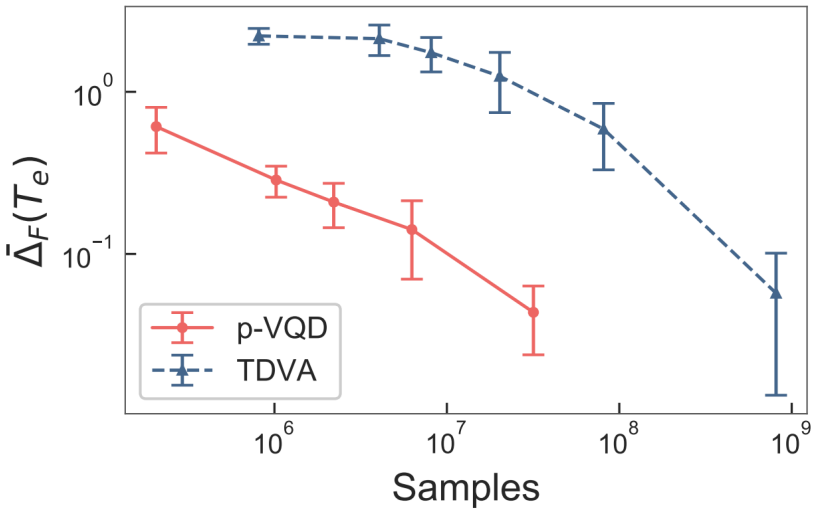


Figure 3.7: Integrated infidelity (ΔF) for both algorithms as seen in [BVC21]. With total time of evolution $T_e = 3$, 60 time steps, 3 variables, 10 iterations, and depth $d = 3$ for both p-VQD and TDVA.

P-VQD shows comparable results while using one order of magnitude fewer shots. In 3.7, the integrated infidelity (ΔF) was observed to be up to an order of magnitude lower when compared to TDVA, even with an equivalent number of time steps. The authors also showed a linear scaling behavior (O(p)) with respect to the number of parameters for their algorithm, as can be seen in 3.8. This is in stark contrast to TDVA's quadratic scaling.

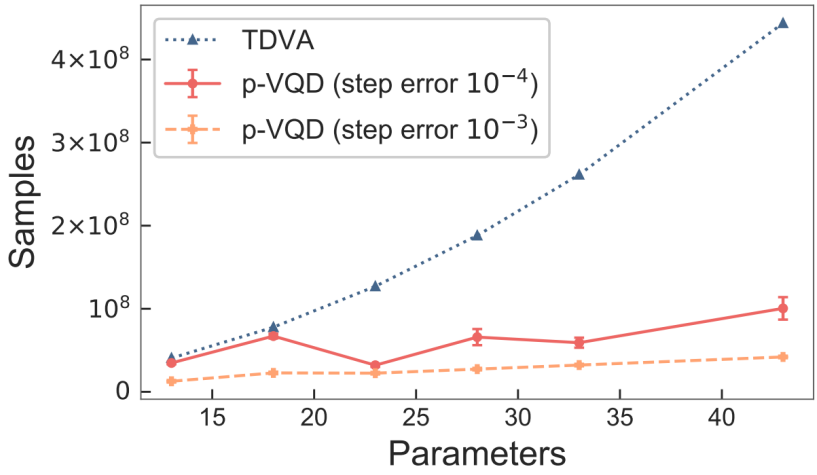


Figure 3.8: Measurements needed for increasing parameters as seen in [BVC21]. With 3 variables and 8000 shots per circuit evaluation, Depth d varies from $d = 2$ to $d = 8$ according to the number of parameters. For p-VQE, the number of measurements depends on the optimization of the parameter variation, while it is fixed for TDVA.

Finally, they compared the minimization of the global (O_G) and local (O_L) objective functions for different time steps and an increasing number of variables. The two objective functions show similar performances in terms of the number of iterations required. However, in general, barren plateaus may not show up until even larger systems are considered, making this comparison not conclusive.

4 F-VQE

In this section, we will examine the extension of VQE presented in the paper "Filtering variational quantum algorithms for combinatorial optimization" [AMR+22]. This is the groundwork for this thesis and will therefore be discussed in more detail. There is limited work for this algorithm, and it is not yet clear if it provides a good approach to our scheduling problem. In the following chapter, all development steps and ideas considered in this work will be addressed. In particular, all concepts used in the final implementation will be presented.

With the algorithm of Amaro et al., the first design decision was already made when the work began. The initial concept was to implement the F-VQE algorithm as outlined in the paper, along with some other approaches for a later comparison. The new challenge was to extend F-VQE to our scheduling problem and optimize meta parameters for better results. This and the implementation of the other approaches are discussed in Section 5. Here, we will begin by going over all the differences between F-VQE and VQE and how one would implement this approach via the weighted Max-Cut problem.

The weighted Max-Cut problem is a widely explored dilemma with a strong presence in the literature. It's a familiar challenge in which the objective is to partition the vertices of a graph into two sets in such a way that the total weight of the edges connecting these sets is maximized. Here we look at a weighted simple undirected and connected graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{W})$ where $\mathcal{V} = \{1, 2, ..., N\}$ is the set of vertices, $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of edges between different vertices, and $\mathcal{W} = \{w_e \in [0, 1] : e \in \mathcal{E}\}$ is the set of random weights uniformly distributed in the range [0, 1] for all edges.

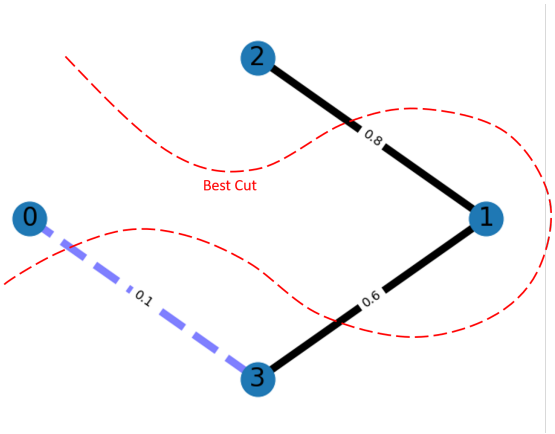


Figure 4.1: Max cut example graph with $|\mathcal{V}| = 4$ nodes. Red line shows optimal cut. Large weights are shown as black edges. Small weights are shown as purple edges.

An example of such a graph can be seen in figure 4.1. The Max Cut problem is equivalent to minimizing the Hamiltonian of the Ising model we discussed in Section 2.3.2. Here, each vertex i of the graph is represented by a variable $z_i = \{+1, -1\}$. A configuration for variables

z_i partitions $\mathcal{V}$ into two sets: $\mathcal{V}^+$ and $\mathcal{V}^-$. The objective function C for a configuration is given by:

$$C = \sum_{i,j \in \mathcal{E}} \frac{(1 - z_i)(1 + z_j)}{4} \mathcal{W}_{i,j}$$

This function adds the weight at index i, j to the cost if $z_i \neq z_j$. Minimizing this energy is equivalent to the min-cut problem. By subtracting from an upper bound a of the cost function, one receives the max-cut problem:

$$H = a - bC$$

4.1 Filtering Operator

F-VQE and VQE work in a similar pattern, but instead of approximating the expectation value of the Hamiltonian, F-VQE instead approximates the action of a filtering operator (F) on the parameterized quantum circuit. This filtering operator is dependent on the Hamiltonian and thus also leads to the minimal expectation value. The filtering operator cannot be directly observed, but the filter value for a configuration can be efficiently computed via a real-valued function $f(E, \tau)$ of the energy E and a parameter $\tau > 0$. An example of a filtering operator is the inverse of the Hamiltonian in regards to τ. For the Max-Cut Hamiltonian, this can be written as:

$$f(H, \tau) = (a - bC)^{-\tau}$$

For a given energy E of the Hamiltonian, this can be efficiently computed.

By repeatedly applying the filtering operator F to a quantum state, we achieve two significant outcomes: the elimination of high-energy eigenstates, which correspond to suboptimal solutions for the combinatorial optimization problem, and an increase in alignment with the ground state. In figure 4.2, the distribution is given for the max cut example after applying the inverse filter 0, 6, 9, and 12 times. As can be seen, the overlap with the ground state "1100" increases with every successive filter application. This corresponds to a cut between every edge: $\mathcal{V}_0 = 0, \mathcal{V}_1 = 0, \mathcal{V}_2 = 1$, and $\mathcal{V}_3 = 1$ with an energy of $E = 1.5$. In our implementation, we decided to establish an inherent direction for the graph by assigning distinct weights based on the order of indices on each edge. We set one weight value as illustrated in Figure 4.1, while the weight for the other direction is set to zero. This choice was made to minimize the number of optimal states, as it ensures that "1100" is the optimal state rather than "0011." Nevertheless, the measurement probability also increases for the state "0100". This is due to the low weight associated between the vertices $\mathcal{V}_0$ and $\mathcal{V}_3$, which makes it difficult for the optimizer to find a difference in the two expectation values.

To explain this phenomenon, we first have to take a look at the default distribution of eigenbases or pure states $|\lambda_x\rangle$, where $x = \{0, \cdots, 2^{n-1}\}$ for a quantum state $|\psi\rangle$. The distribution for an observable corresponding to a self-adjoint operator A is given by the Born rule [Lan09]. For our case, the observable A is the quantum state $|\psi\rangle$. This means that A has an orthonormal basis of eigenvectors $|\lambda_x\rangle$ with corresponding eigenvalues λ_x. The Born rule states that the measured result of A will be one of its eigenvalues, λ_x. When we want to find the likelihood of measuring the specific eigenvalue λ_x for an operator A in our

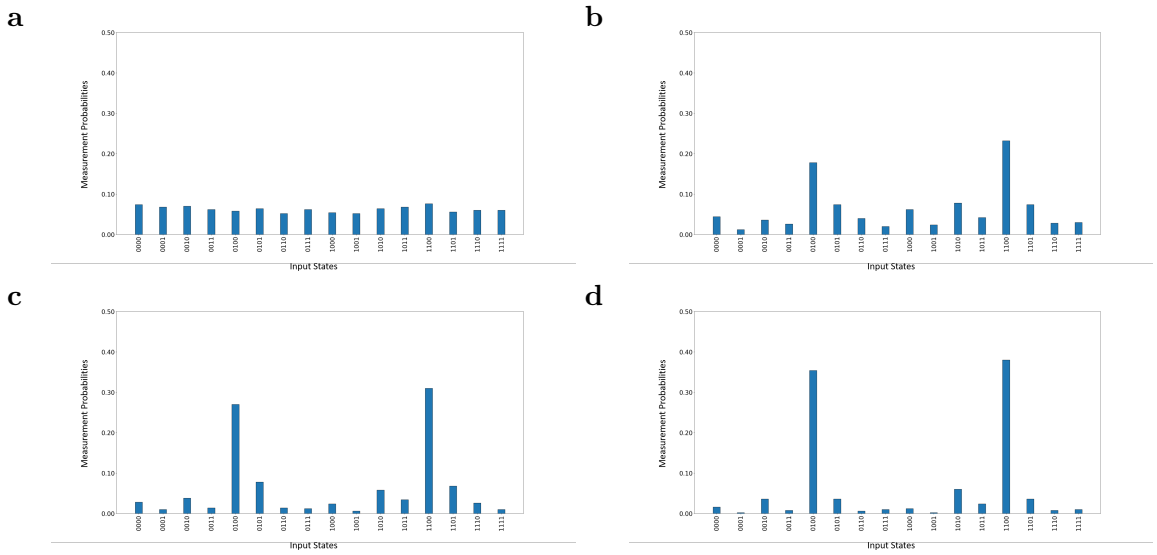


Figure 4.2: Probability distribution for Max Cut example with four qubits after applying the inverse filter (a) 0, (b) 6, (c) 9, and (d) 12 times

quantum system $|\psi\rangle$, we can calculate it using the formula: $P(A = \lambda_x \big| |\psi\rangle) = \langle\psi| P_x |\psi\rangle$. Here, P_x represents the projection of λ_x onto the eigenspace of the operator A. If our quantum system is in a pure state $|\lambda_x\rangle$, the probability of detecting the eigenvalue λ_x when we measure operator A is:

$$P(A = \lambda_x \big| |\psi\rangle) = |\langle\psi|\lambda_x\rangle|^2$$

This default distribution changes with the application of the filtering operator F on the quantum state $|\psi\rangle$. The quantum state resulting in the application of the filter operator $|F\psi\rangle$ is given by the application of the filtering operator onto the state $|\psi\rangle$ and division by the square root of the expectation value of the squared filter operator to preserve the normalization of the quantum state: $|F\psi\rangle = F|\psi\rangle / \sqrt{\langle F^2\rangle_\psi}$. We demand that our filtering operator be strictly positive in the range $[E_{min}, E_{max}]$. The probability distribution for the state resulting from the application of such a filtering operator $|F\psi\rangle$ depends on the energy E_x of each eigenstate $|\lambda_x\rangle$:

$$P(A = \lambda_x \big| |F\psi\rangle) = |\langle F\psi|\lambda_x\rangle|^2 = \left| f(E_x, \tau)/\sqrt{\langle F^2\rangle_\psi} \cdot \langle\psi|\lambda_x\rangle \right|^2 = \frac{f^2(E_x, \tau)}{\langle F^2\rangle_\psi} P(A = \lambda_x \big| |\psi\rangle)$$

If we now choose our filtering operator F in such a way that $f^2(E_x, \tau)$ is strictly decreasing on the interval given by the complete spectrum of the Hamiltonian, we can see that it indeed increases the overlap with the ground state. For $|\lambda_x\rangle$ with energy $E_x \approx E_{min}$, the squared filtering value $f^2(E_x, \tau)$ will reach its maximum and therefore be larger than the squared expectation value of the filtering operator $\langle F^2\rangle_\psi$. This results in $\frac{f^2(E_x,\tau)}{\langle F^2\rangle_\psi} > 1$ and will increase the overall measuring probability. Vice versa for $|\lambda_x\rangle$ with energy $E_x \approx E_{max}$. This effect becomes more pronounced with each application of the filtering operator, and after a significant number of iterations, it leads to the emergence of the ground state.

But since we cannot produce the state $|F\psi\rangle$ directly, we have to approximate the application of the filtering operator. In order to do that, we use a parameterized quantum

circuit as we would for VQE. But instead of searching for the parameters that minimize the expectation value of the Hamiltonian H, we search for parameters that minimize the Euclidean distance between the parameterized quantum state $|\psi(\boldsymbol{\theta}_t)\rangle$ and the state that would be produced if we could apply the filtering operator directly $|F_t\psi(\boldsymbol{\theta}_{t-1})\rangle$. Where t denotes the number of filter applications or optimization steps. By iteratively reducing the squared euclidean distance between the two states, the approximation becomes more distinct. The time-dependent cost function $C_t(\boldsymbol{\theta})$ is given by:

$$C_t(\boldsymbol{\theta}) = \frac{1}{2}\Big|\Big| \, |\psi(\boldsymbol{\theta}_t)\rangle - |F_t\psi(\boldsymbol{\theta}_{t-1})\rangle \, \Big|\Big|^2 = 1 - \frac{Re\langle\psi(\boldsymbol{\theta}_{t-1})|F_t|\psi(\boldsymbol{\theta}_t)\rangle}{\sqrt{\langle F_t^2\rangle_{\psi(\boldsymbol{\theta}_{t-1})}}}$$

This formulation no longer requires the state $|F\psi\rangle$ and instead uses the expectation value of F in regards to the quantum state before and after the optimization step: $\langle\psi(\boldsymbol{\theta}_{t-1})|F_t|\psi(\boldsymbol{\theta}_t)\rangle$. From now on, we will abbreviate $\psi(\boldsymbol{\theta}_t)$ with just ψ_t.

4.2 Optimizing the Cost Function

Optimizing the cost function $C_t(\boldsymbol{\theta})$ can be approached in two ways. The first method involves implementing the cost function directly and utilizing its output, such as by approximating a gradient. The second approach is to compute the gradient directly. In the first case, it's feasible to approximate the expectation value $\langle F_t^2\rangle_{\psi_{t-1}}$, as we will discuss later. However, estimating $Re\langle\psi_{t-1}|F_t|\psi_t\rangle$ is not a straightforward task. To approximate this part, one can use the Hadamard test [MF19].

To avoid the overhead of the Hadamard test, it is also possible to just compute the partial differential equations $\frac{\partial C_t(\boldsymbol{\theta})}{\partial\theta_j}$ via the parameter shift rule 2.3.4. For VQE, the parameter-shift rule can be used with the objective function $C(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle$. The analytical gradient of the objective function with parameters θ_j for rotation R_y gates is given as:

$$\frac{\partial\langle\psi|H|\psi\rangle}{\partial\theta_j}\bigg|_{\boldsymbol{\theta}_{t-1}} = \frac{\langle\psi_{t-1}^{j+}|H|\psi_{t-1}^{j+}\rangle - \langle\psi_{t-1}^{j-}|H|\psi_{t-1}^{j-}\rangle}{2}$$

In this context, the circuits $|\psi_{t-1}^{j\pm}\rangle$, which correspond to states $|\psi(\boldsymbol{\theta}_{t-1} \pm \frac{\pi}{2}\epsilon_j)\rangle$, are realized by adjusting the parameter θ_j by either adding or subtracting $\frac{\pi}{2}$. For the F-VQE, the partial derivative of the parameter θ_j for the parameter vector $\boldsymbol{\theta}_{t-1}$ is similarly given as:

$$\frac{\partial C_t(\boldsymbol{\theta})}{\partial\theta_j}\bigg|_{\boldsymbol{\theta}_{t-1}} = -\frac{\langle\psi_{t-1}^{j+}|F_t|\psi_{t-1}^{j+}\rangle - \langle\psi_{t-1}^{j-}|F_t|\psi_{t-1}^{j-}\rangle}{4\sqrt{\langle\psi_{t-1}|F_t^2|\psi_{t-1}\rangle}}$$

Note that the denominator's expected value remains consistent across all partial derivatives (θ_j) during a fixed optimization step (t). The F-VQE algorithm therefore needs only $2\dim(\boldsymbol{\theta}) + 1$ circuit evaluations per optimization step t. Only one more than the VQE with a similar gradient computation. The expectation value of F_t or F_t^2 in regards to the circuits $|\psi_{t-1}^{j\pm}\rangle$ and $|\psi_{t-1}\rangle$ can be efficiently approximated via the Monte Carlo estimator [LMB+20] of the function $f(E,\tau)$:

$$\langle F_t\rangle_\psi \approx \frac{1}{M}\sum_x M_x f(E_x,\tau)$$

This can be computed by sampling the quantum state M times in the Hamiltonian eigenbasis. For the eigenstate $|\lambda_x\rangle$, M_x and $f(E_x, \tau)$ are the corresponding sample count and filter value. For the VQE, it is also possible to directly observe the Hamiltonian operator, as we have seen in Section 2.3.4. The parameters are updated according to the gradient descent rule we discussed in Section 2.1.2. For optimization step t, the new parameter vector $\boldsymbol{\theta}_t$ is given by:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha \nabla C(\boldsymbol{\theta}_{t-1}, \tau), \text{ with: } \nabla C(\boldsymbol{\theta}_{t-1}, \tau) = \sum_j \frac{\partial C_t(\boldsymbol{\theta})}{\partial \theta_j}\bigg|_{\boldsymbol{\theta}_{t-1}} \boldsymbol{\epsilon}_j$$

4.3 Vanishing Gradients

While there are different ways to combat the occurrence of barren plateaus, we use the parameter τ to influence the gradient norm to stay in the vicinity of some desired large and fixed value g_c. This value is a fixed parameter and can vary depending on the objective function and problem instance. To dynamically adapt τ, we use a heuristic approach in which we try to find a new τ_t with $|| \nabla C_t(\tau_t)|_{\boldsymbol{\theta}_{t-1}} ||^2 = g_c$ for the threshold $g_c > 0$. For our implementation, we chose the threshold $g_c = 0.1$. For large values of τ, the gradient norm saturates at a finite value that is determined by the overlap of the gradient circuits with the ground state. We incrementally increase τ_t until:

1. we find a value τ_u with $|| \nabla C_t(\tau_u)|_{\boldsymbol{\theta}_{t-1}} ||^2 > g_c$, or
2. $|| \nabla C_t(\tau_t)|_{\boldsymbol{\theta}_{t-1}} ||^2$ converges to a constant.

We then select the new τ_t for which the gradient norm was closest below the threshold g_c.

This heuristic leads to each partial derivative changing non-trivially as a function of τ. It is therefore not a normalization or a constant re-scaling of the gradient. In Figure 4.3, the workflow of the F-VQE algorithm is illustrated. We begin with the initial measurement of the quantum circuit to get a first value for the expectation value of the filtering operator $\langle F_t \rangle_\psi$. Then we update the parameter vector $\boldsymbol{\theta_t}$ by using the parameter shift rule. And lastly, we update the parameter τ_t to guarantee non-vanishing gradients.

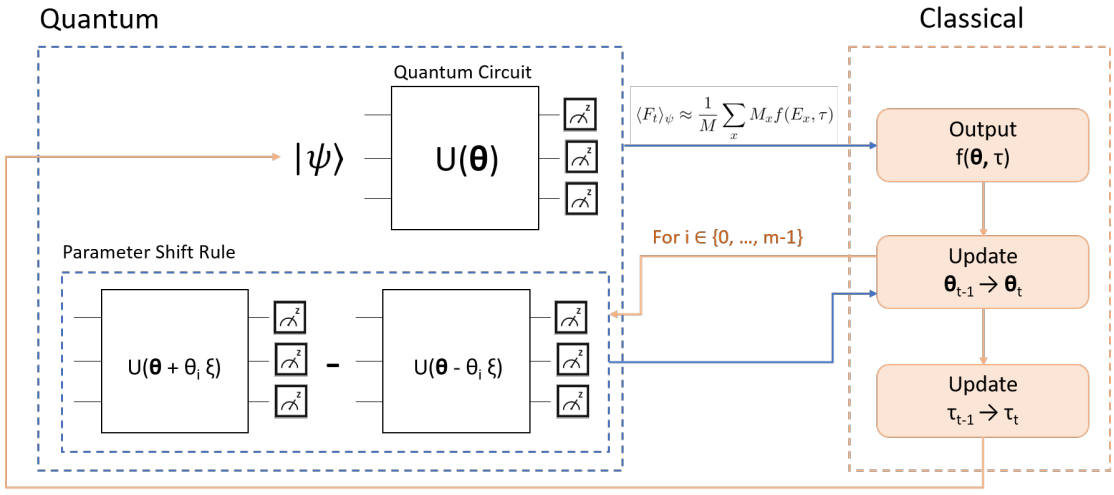


Figure 4.3: Workflow of the F-VQE algorithm separated into quantum and classical parts

5 Methodolegy

Next, we will introduce the concrete problem at the DLR and its objective. We will discuss the optimization metrics and the constraints to be followed. However, at least in this work, not all constraints are implemented or tested. Furthermore, we will look at the implementation of all solvers, their respective advantages and disadvantages, and how they differ from each other.

5.1 Problem Definition

The DLR operates a range of spacecraft missions through its German Space Operations Center (GSOC) division. These missions, as illustrated in Figure 5.1, often necessitate the continuous presence of one or more operators for specific subsystems, referred to as positions, over specified time periods. To facilitate this, the operators follow an on-call spacecraft operator scheduling process, which undergoes periodic updates throughout the year to account for changes in operator availability or new personnel assignments [SGGC+21].

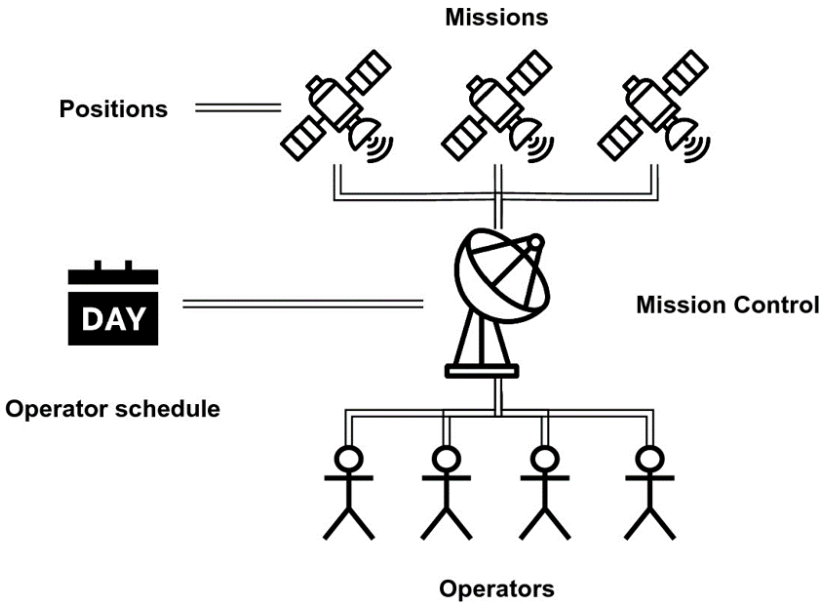


Figure 5.1: Visualization of On-Call Scheduling Problem with Different Operators, Positions, Satellite Missions, and a Set Number of Days as Seen in [SGGC+21]

A typical scheduling task involves assigning roughly 50 operators to 20 different positions over a 180-day time frame [SGGC+21]. This schedule must remain adaptable, as adjustments may be required if assumptions change over time. Each on-call shift encompasses an entire day, and an operator's ability to cover certain positions depends on their training and responsibilities.

Creating a valid schedule entails gathering input from both the spacecraft missions and the operators themselves. This schedule must adhere to a set of constraints, including but not limited to:

(i) Operators can only work on some given positions.

(ii) Per day position pair, at least one operator needs to be assigned.

(iii) Per day position pair, at most one operator is allowed to be assigned.

(iv) An operator can be assigned to at most one position a day.

(v) Operators can specify days in advance when they are unavailable.

(vi) A partially filled on-call schedule may be supplied and needs to be obeyed.

(vii) Operators can work at most two out of any three consecutive weeks.

(viii) Operators can work at most 35 days out of any 105 consecutive days.

It's important to note that not all constraints are always applicable. Depending on the specific problem requirements, certain constraints may or may not be used. For instance, constraints (ii) and (iii) can be employed selectively. If a scenario dictates that certain positions must be continuously supervised and can even be assigned to multiple operators, only constraint (ii) is necessary. Conversely, if the situation involves positions that are only managed by a single operator, are non-critical, and do not require constant manning, constraint (iii) becomes relevant. For our specific problem, both constraints are always used in tandem. Furthermore, not all valid schedules are equally suited to a variety of demands. Therefore, we also include multiple goals for which the algorithm should optimize its results. These goals function as soft constraints, allowing some degree of flexibility in achieving them. In some cases, it may be acceptable for certain goals to be partially met, depending on the extent of deviation. We will optimize for the following goals:

(a) Operators shall work, preferably whole weeks.

(b) All operators shall work a similar number of days.

Currently, at GSOC, a heuristic search algorithm that utilizes backtracking, powered by the Plato library [LWM+12], is employed as the problem-solving approach.

5.1.1 Variable Encoding

In this section, we delve into the specifics of how operators are encoded in our implementation for scheduling. We consider two encoding techniques: one-hot encoding and binary encoding, both of which were introduced in Section 3.1.3. For the binary encoded scheduling, we will use the same approach outlined in Section 3.1.1 and use the operators as the binary category.

One-Hot Encoding

One-Hot Encoding is a method where we represent operators for each pair of days and positions (day-positions) using a vector of binary variables. To represent N operators, we need a binary vector with N variables, where each variable corresponds to one operator. These binary vectors are structured as follows:

$$d_i p_j = \begin{cases} d_i p_j^{(0)} \\ d_i p_j^{(1)} \\ \vdots \\ d_i p_j^{(N-1)} \end{cases}$$

Therefore, we require a total of $P \cdot N \cdot D$ binary variables denoted as $x_{i,j}^{(k)}$ for operators $o^{(k)}$, for $i \in \{0, \cdots, D-1\}, j \in \{0, \cdots, P-1\}$, and $k \in \{0, \cdots, N-1\}$. The decision variables $o_{i,j}^{(k)}$ are then mapped directly to the binary variables $x_{i,j}^{(k)}$:

$$o_{i,j}^{(k)} = x_{i,j}^{(k)}$$

These binary variables are used to represent the assignment of operators to specific day positions. The abstract decision variables are given by:

$$o_{i,j}^{(k)} = \begin{cases} 1, \text{if operator k is working on position } p_J \text{ and day } d_i \\ 0, \text{otherwise} \end{cases}$$

Binary Encoding

Binary encoding is an alternative approach for encoding operators in the scheduling problem. For each pair of day positions ($d_i p_j$), we utilize a binary representation for the operators. Specifically, for $N = 2^n$ operators, we require n bits. For $N = 4$ operators, we would only need binary vectors of length two, where:

$$o^{(0)} = [0, 0] \qquad o^{(1)} = [0, 1]$$
$$o^{(2)} = [1, 0] \qquad o^{(3)} = [1, 1]$$

$$o^{(0)} = [0, 0, 0, 1] \qquad o^{(1)} = [0, 0, 1, 0]$$
$$o^{(2)} = [0, 1, 0, 0] \qquad o^{(3)} = [1, 0, 0, 0]$$

The general bit representation is structured as follows:

$$d_i p_j = \begin{cases} d_i p_j^{(0)} \\ d_i p_j^{(1)} \\ \vdots \\ d_i p_j^{(n-1)} \end{cases}$$

where a bit allocation of variables $d_i p_j^{(l)}$ corresponds to an operator $o^{(l)}$. In this case, we only need $P \cdot n \cdot D$ binary variables denoted as $x_{i,j}^{(w)}$, where w ranges from 0 to $n-1$. These variables are calculated using the following formula:

$$o_{i,j}^{(k)} = \prod_{w=0}^{n-1} \left(1 - \left(x_{i,j}^{(w)} - b_{k,w}\right)^2\right), \quad k \in \{0, \cdots, N-1\}$$

Here, $b_{k,w}$ represents the bit-wise allocation for operator $o^{(k)}$. These binary variables are used to represent the assignment of operators to specific day positions, similar to one-hot encoding.

5.1.2 Constraint and Goal Formulation

We will now look at all constraints and goals in more detail and explain how one would implement them. It is important to note that each constraint or goal alters the energy landscape of the problem, making it more difficult to compute but not necessarily more complex to find the global minima. In the following section, we will look at a simple example with 2 operators, 3 days, and 2 positions for which we compute the energy landscape for each constraint and goal. For this example, we define the operator position capabilities in such a way that operator $o^{(0)}$ is able to work only on position p_0 while operator $o^{(1)}$ can work on both positions. Likewise with the definition of the outages, operator $o^{(0)}$ has an outage on day d_0 while operator $o^{(1)}$ can work on all days. In tables 5.1, a visual representation of the position capabilities and outages is given.

Operators	p_0	p_1
$o^{(0)}$	✓	×
$o^{(1)}$	✓	✓

Operators	d_0	d_1	d_2
$o^{(0)}$	×	✓	✓
$o^{(1)}$	✓	✓	✓

Table 5.1: Left: Example of operator position capabilities. Right: Example of operator outages

For constraint (vi) a partially filled schedule is needed to compute the corresponding energy. The used partial schedule in the example in Table 5.2 has 50% of all shifts already occupied: operator $o^{(1)}$ is working on day-positions d_0p_1, d_1p_1, and d_2p_0.

Day d_0		Day d_1		Day d_2	
p_0	p_1	p_0	p_1	p_0	p_1
-	$o^{(1)}$	-	$o^{(1)}$	$o^{(1)}$	-

Table 5.2: Example for constraint (vi): partial schedule filled to 50% of all shifts

As we can see in Tables 5.2 and 5.1, a correct schedule is no longer possible since operator $o^{(0)}$, even though they are able to work on position p_0, has an outage on day d_0 and operator $o^{(1)}$ already works on position p_1 this day. This can also be seen in Figure 5.2, where the combined energy landscape of all goals and constraints is shown. Even at the global minimum at result "010101", which corresponds to a schedule where operator $o^{(0)}$ is scheduled for all positions p_0 and operator $o^{(1)}$ for all positions p_1, the energy is not zero.

Note that if one compares the overall energy landscape with the individual energy landscapes, it can sometimes be easier to find a global minimum if you add more constraints or goals. For example, in Figure 5.4, because of its steep peaks and valleys, it would be rather easy to run into a local minimum. By adding the rest of the constraints and goals, the slopes can be reduced, making it easier to escape these local minima.

In the following subsections, we will only exemplarily show the different energy landscapes for single constraints and goals.

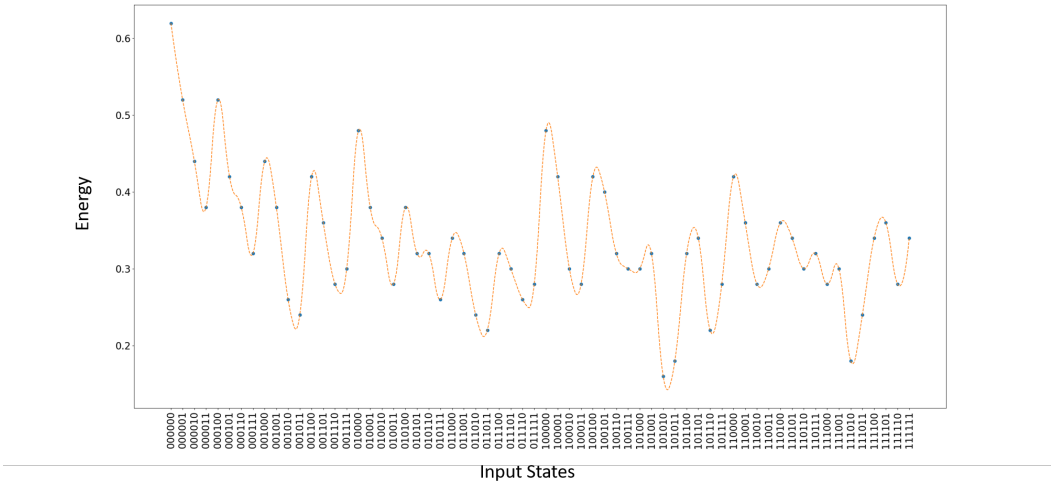


Figure 5.2: Energy landscape for all binary encoded goals and constraints for 2 operators, 3 days, and 2 positions

Constraint (i): Operators can only work some positions

For the first (i) constraint, every operator is defined with a set of available positions.

$$o^{(k)} : p_{o^{(k)}}, \text{ where } p_{o^{(k)}} \subseteq P$$

If the operator is scheduled for a different position on any day of the on-call plan, the constraint is violated and the plan is not valid. This can be written as a matrix $y_j^{(k)}$, where an entry is zero if the operator $o^{(k)}$ can work on position p_j:

$$y_j^{(k)} = \begin{cases} 0, & \text{if } p_j \in p_o(k) \\ 1, & \text{otherwise} \end{cases}$$

The penalty $C_{(i)}$ is given by the sum over all operators and day-position pairs with the matrix $y_j^{(k)}$:

$$C_{(i)} = \sum_{k}^{N} \sum_{i}^{D} \sum_{j}^{P} y_j^{(k)} \cdot o_{i,j}^{(k)}$$

This penalty is zero if all operators N can work on all positions P or if every operator $o^{(k)}$ only works on positions $p_j \in p_{o^{(k)}}$.

Constraint (ii) and (iii): Per day position pair, at least or at most one operator needs to be assigned

Constraints (ii) and (iii) are dependent on the chosen encoding. For binary encoding, the two constraints are already achieved through the encoding scheme, since only exactly **one** operators can be scheduled for a day-position pair $d_i p_j$. For the one-hot encoding, however, this is not the case.

For constraint (ii), the sum over all operators for the day-position pair has to be 1 or more. The penalty $C_{(ii)}(d_i, p_j)$ for a given day d_i and position p_j is given by the sum over all pairs of two operators $o^{(k)}$ and $o^{(l)}$:

$$C_{(ii)}(d_i, p_j) = \sum_{k,l}^{\binom{N}{2}} (1 - o_{i,j}^{(k)} - o_{i,j}^{(l)} + o_{i,j}^{(k)} \cdot o_{i,j}^{(l)})$$

This penalty is zero if one or more operators are working on the day-position pair. The penalty $C_{(ii)}$ for constraint (ii) can therefore be given by a simple summation of all day-position pair penalties:

$$C_{(ii)} = \sum_{i}^{D} \sum_{j}^{P} C_{(ii)}(d_i, p_j)$$

The penalty $C_{(iii)}(d_i, p_j)$, on the other hand, is the exact opposite:

$$C_{(iii)}(d_i, p_j) = \sum_{k,l}^{\binom{N}{2}} o_{i,j}^{(k)} \cdot o_{i,j}^{(l)}$$

This penalty is zero if no operator is working on the day-position pair. Just as with constraint (ii), the penalty $C_{(iii)}$ is given by a simple summation of all day-position pair penalties:

$$C_{(iii)} = \sum_{i}^{D} \sum_{j}^{P} C_{(iii)}(d_i, p_j)$$

With the summation of $C_{(ii)}(d_i, p_j)$ and $C_{(iii)}(d_i, p_j)$, the penalties for constraint (ii) and (iii) are dependent on the number of day-position pairs that violate the constraint. The more pairs violate the constraints, the higher the penalty term. This might not be wanted in some cases, but for our problem, this might help in combating barren plateaus since it is easier to find a better solution by simply satisfying one more day-position pair.

Constraint (iv): An operator can be assigned to at most one position a day

This constraint is working akin to constraint (iii), but where constraint (iii) compared two operators for each day position, this constraint compares two positions for every operator-day pair. This results in increased energy for operators who work more than one position per day. The penalty $C_{(iv)}$ is given by:

$$C_{(iv)} = \sum_{k}^{N} \sum_{i}^{D} \sum_{j,l}^{\binom{P}{2}} o_{i,j}^{(k)} \cdot o_{i,l}^{(k)}$$

As said before, with every combination of two positions where the operator is working, the penalty increases. Only if we cannot find a pair of two positions where the operator is scheduled will the penalty be zero. The energy landscape for specifically constraint (iv) as a binary encoded problem is given in Figure 5.3, where we can see the minimization of the energy at eight distinct points: "010101", "010110", "011001", "011010", and their inverted counterparts. In every bit string, we can see that for each day-position pair: "$p_1d_2, p_0d_2/p_1d_1, p_0d_1/p_1d_0, p_0d_0$," no two identical operators $o^{(k)}$ are assigned.

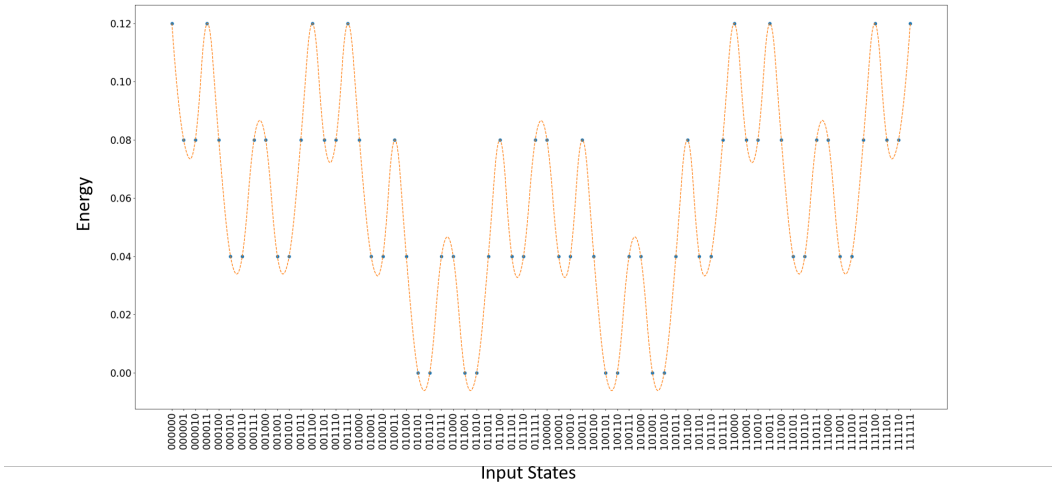


Figure 5.3: Energy landscape for constraint (iv) as a binary encoded problem for 2 operators, 3 days, and 2 positions

Constraint (v): Operators can specify days in advance when they are unavailable

Constraint (v) is rather similar to constraint (i). Here, every operator is defined by a set of unavailable days or outages.

$$o^{(k)} : d_{o^{(k)}}, \text{ where } d_{o^{(k)}} \subseteq D$$

If the operator is scheduled for a day in subset d_o for any position, the constraint is violated and the on-call plan is not valid. This can also be written as a binary matrix $\chi_i^{(k)}$, where an entry is zero if the operator $o^{(k)}$ can work on day d_i:

$$\chi_i^{(k)} = \begin{cases} 0, \text{ if } d_i \notin d_o(k) \\ 1, \text{ otherwise} \end{cases}$$

Similar to constraint (i), the penalty $C_{(i)}$ is given by the sum over all operators and day-position pairs with the binary matrix $\chi_i^{(k)}$:

$$C_{(v)} = \sum_{k}^{N} \sum_{i}^{D} \sum_{j}^{P} \chi_i^{(k)} \cdot o_{i,j}^{(k)}$$

This penalty is zero if no operator has an outage on any day or if every operator $o^{(k)}$ only works on days $d_i \notin d_{o^{(k)}}$.

Constraint (vi): A partially filled on-call schedule may be supplied and needs to be obeyed

Constraint (vi) is an extension of constraints (i) and (v). But rather than limiting the operator's capabilities to just days or positions, this constraint combines both depending on the partial on-call schedule. A partial on-call schedule defines already scheduled day-position pairs $(d, p) \subseteq d_i \times p_j$, which is the same as pairs not being available for the rest of the operators. A matrix $(xy)_{i,j}$ defines these day-position pairs for all operators $o^{(k)}$, positions p_j, and days d_i.

$$(xy)_{i,j} = \begin{cases} 0, \text{ if } d_i p_j \notin (d, p) \\ 1, \text{ otherwise} \end{cases}$$

We do not need to specify different values for different operators since an already scheduled day position is unavailable for all operators. The penalty $C_{(vi)}$ is given just as for constraints (i) and (v):

$$C_{(vi)} = \sum_{k}^{N} \sum_{i}^{D} \sum_{j}^{P} (xy)_{i,j} \cdot o_{i,j}^{(k)}$$

The penalty is zero if the partial On-Call schedule is empty or if no operator $o^{(k)}$ is scheduled for $d_i p_j \in (d, p)$.

Constraint (vii) and (viii): Operators can work at most n out of any m consecutive weeks

The final two constraints are somewhat more intricate compared to the others, as they involve inequalities that must be taken into account. Since we only work with small problem instances, we will only use constraint (vii), but both constraints work similarly, just with different inequalities. For constraint (vii), an operator is only allowed to work two days in a row, no matter what position, for a sliding window of length three days. The constraint is violated if an operator is assigned to a position for all three consecutive days during a sliding window. The constraint is given by a set of inequalities for each operator $o^{(k)}$ and W sliding windows $w_m \in \{0, \cdots, W-1\}$:

$$\left(\sum_{i}^{D_m} \left(\sum_{j}^{P} o_{i,j}^{(k)} - \sum_{j}^{P-1} o_{i,j}^{(k)} \cdot o_{i,j+1}^{(k)} \right) \right) \leq 2$$

where D_m are all days in the sliding window w_m. The term $\sum_j^P o_{i,j}^{(k)} - \sum_j^{P-1} o_{i,j}^{(k)} \cdot o_{i,j+1}^{(k)}$ defines if the operator $o^{(k)}$ is working on any position on day d_i. It is zero if the operator does not work on any position and one otherwise. It is important to choose a term that is exactly 0 or 1 so we do not accidentally violate the inequality.

As we have seen in Section 2.1 for one sliding window w_m and one operator $o^{(k)}$, the inequality can be expressed with two slack variables s_0, s_1:

$$\left(\sum_{i}^{D_m} \left(\sum_{j}^{P} o_{i,j}^{(k)} - \sum_{j}^{P-1} o_{i,j}^{(k)} \cdot o_{i,j+1}^{(k)} \right) + s_0 + s_1 - 2 \right)^2$$

Unfortunately, one needs two new slack variables for every sliding window and every operator:

$$C_{(vii)} = \sum_{k}^{N}\sum_{m}^{W}\left(\sum_{i}^{D_m}\left(\sum_{j}^{P} o_{i,j}^{(k)} - \sum_{j}^{P-1} o_{i,j}^{(k)} \cdot o_{i,j+1}^{(k)}\right) + s_0^{(k,m)} + s_1^{(k,m)} - 2\right)^2$$

For 4 operators and 2 sliding windows, $4 \cdot 2 \cdot 2 = 16$ slack variables would be needed. This is not possible for the current quantum hardware since every new slack variable needs one qubit. For approaches using this Hamiltonian formulation, we decided to exclude this constraint.

Goal (a): Operators shall work preferably whole weeks

To implement punishing terms for goals, one has to define a function measuring conformity with the goal. The further the schedule is from the goal, the higher the punishing factor has to be. For goal (a), every operator should roughly work the same number of shifts for a given number of days. We achieve this by defining the average number of working shifts every operator should work.

$$\omega = \left\lceil \frac{D \cdot P}{N} \right\rceil$$

The deviation from ω is the punishing factor for each operator:

$$C_{(a)} = \sum_{k}^{N}\left(\omega - \sum_{i}^{D}\sum_{j}^{P} o_{i,j}^{(k)}\right)^2$$

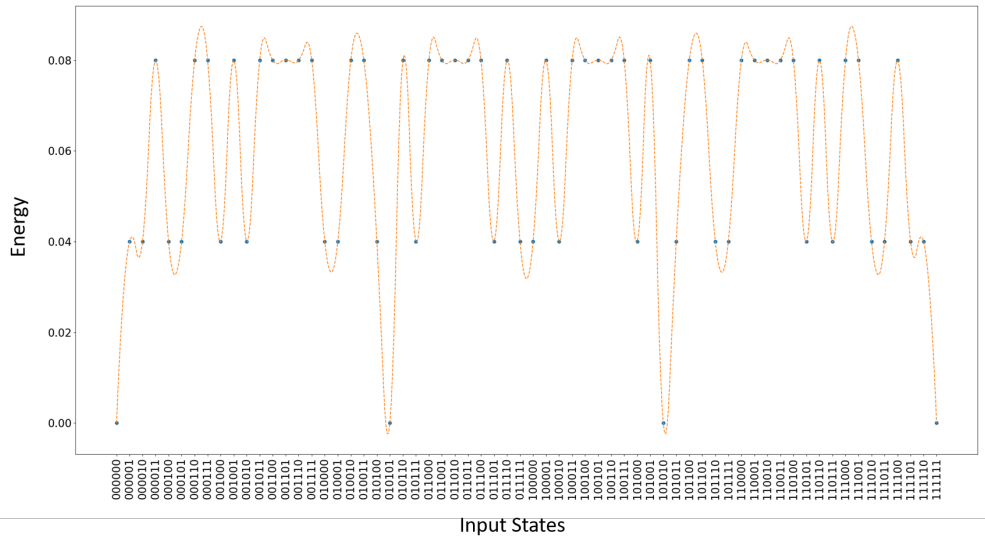


Figure 5.4: Energy landscape for binary encoded problem with 2 operators, 3 days, and 2 positions for goal (b)

Goal (b): All operators shall work a similar amount of days

For this goal, the objective is for every operator to work or to be free for a full week. To test this goal for smaller problem instances with $D < 7$, we will define D as the full week. To measure the conformity with this goal, we want to calculate the deviation from the two extrema 0 and D or 7 if $D > 7$:

$$C_{(b)} = \sum_{k}^{N} \sum_{j}^{P} \left(D - \sum_{i}^{D} o_{i,j}^{(k)} \right) \cdot \sum_{i}^{D} o_{i,j}^{(k)}$$

The energy landscape in Figure 5.4 for this goal is very hard to traverse and find the global minimum. With its steep peaks and multiple local minima, it is easy to miss the optimal solution.

In the following two sections, we will discuss the three different VQAs, beginning with the QAOA solver, which utilizes the Hamiltonian formulation of the objective function and therefore excludes constraint (vii). Afterwards, we will look at the methods used for both the VQE and F-VQE.

5.2 Hamiltonian Solver

In this section, we will look at the implementation of the QAOA solver using the Hamiltonian defined by the constraints and goals in 5.1.2. This implementation uses the procedure outlined in 2.3.4, where we divide the Hamiltonian into not commuting Pauli strings $\otimes_{j\in\rho}\sigma_j^z$, where ρ is the set of indices for a given Pauli string. Let us imagine a system with $\mathcal{N} = 5$ qubits, and we want to define the Pauli string for the first and second qubits. The set ρ is therefore defined by just those two indices: $\rho = \{0, 1\}$, and the resulting Pauli string is: $\sigma_0^z \otimes \sigma_1^z$. The energy or the expectation value of this Pauli string $E_{\{0,1\}} = \alpha_{\{0,1\}} \langle 0|^{\mathcal{N}} U^\dagger(\boldsymbol{\theta})\ \sigma_0^z \otimes \sigma_1^z\ U(\boldsymbol{\theta})\, |0\rangle^{\mathcal{N}}$ can then be directly measured on a quantum computer.

To generate all necessary Pauli strings, we first have to simplify the objective function, adding the constraints $c \in \{(i), (ii), (iii), (iv), (v), (vi)\}$ and goals $g \in \{(a), (b)\}$:

$$f = \alpha \sum_{c} C_c + \beta \sum_{g} C_g \tag{5.1}$$

The simplified terms are given by:

$$f = \sum_{\rho \subset \{0,\cdots,\mathcal{N}-1\}} \alpha_\rho \prod_{i\in\rho} x_i \tag{5.2}$$

The product $\prod_{i\in\rho} x_i$ is taken over all variables x_i belonging to the subset ρ and represents the variable interaction terms inherent to the objective. The values α_ρ represent the coefficients associated with each subset. If a subset is not part of the objective, the corresponding coefficient is zero. The variables x_i correspond to the qubits of the system and can be expressed as $o_{i,j}^{(k)}$ or $o_{i,j}^{(w)}$, as seen in Subsection 5.1.1. It is possible to find such a representation for every objective f since all variables x_i are binary and we can reduce quadratic or higher-order terms to linear combinations. The term $(x_0 x_1)^2$, for example, is equivalent to $x_0^2 x_1^2 = x_0 x_1$. With all terms ρ, we use the substitution from Subsection 2.3.2 $x_i = \frac{1-z_i}{2}$

to generate the Pauli strings $\otimes_{j\in\rho}\sigma_j^z$. Note that we exclude the constraints vii and viii in our objective f since both require a vast amount of slack variables and are therefore not usable with current quantum hardware.

Let us look at a simple example: We want to solve a problem with two operators, positions and days, while using constraints i, iv, and v and both goals. Operator $o^{(0)}$ has an outage on day d_0 and can only work on positions p_0. Operator $o^{(1)}$, on the other hand, has no outage and can work both positions. For this example, we will use binary encoding, so we only need four qubits or variables. $x_0 \equiv d_0p_0, x_1 \equiv d_0p_1, x_2 \equiv d_1p_0, x_3 \equiv d_1p_1$ for the day-position pairs. The terms for the different goals and constraints are given by Equation 5.5.

$$
\begin{aligned}
C_{(a)} &= -6x_0 - 6x_1 - 6x_2 - 6x_3 + 4x_0x_1 + 4x_0x_2 + 4x_0x_3 + 4x_1x_2 + 4x_1x_3 + 4x_2x_3 + 8 \\
C_{(b)} &= +2x_0 + 2x_1 + 2x_2 + 2x_3 - 4x_0x_2 - 4x_1x_3 \\
\hline
C_{(i)} &= -2x_1 - 2x_3 + 4 \\
C_{(iv)} &= -2x_0 - 2x_1 - 2x_2 - 2x_3 + 4x_0x_1 + 4x_2x_3 + 4 \\
C_{(v)} &= -2x_0 - 2x_1 + 4
\end{aligned}
$$

Figure 5.5: Punishment terms for 2 operators, 3 days, and 2 positions with goals (a), (b), and constraints (i), (iv), and (v)

For α and β both set to 1, we can simplify the objective function in Equation 5.1 to the form in Equation 5.2:

$$f = -8x_0 - 10x_1 - 6x_2 - 8x_3 + 8x_0x_1 + 4x_0x_3 + 4x_1x_2 + 8x_2x_3 + 20$$

The objective function can then be mapped to a quantum Hamiltonian via variable substitution. The Hamiltonian, denoted as H_C, is given by the Pauli strings ρ. In this case, the Hamiltonian H_C is represented as:

$$H_C = \sigma_0^z + 2\sigma_1^z + \sigma_3^z + 2\sigma_0^z\sigma_1^z + \sigma_0^z\sigma_3^z + \sigma_1^z\sigma_2^z + 2\sigma_2^z\sigma_3^z + 10$$

Here, σ_i^z represents a Pauli-Z operator acting on qubit i. The terms in the Hamiltonian capture the interactions and constraints inherent in the original optimization problem.

For QAOA, the ansatz circuit is given by the Hamiltonian H_C and the mixer Hamiltonian H_x, where we use the simple bit flip mixer as defined in Subsection 2.3.4. The gates for the circuit are given by:

$$U_{H_C}(\gamma) = e^{-i\gamma H_C} \qquad U_{H_x}(\beta) = \prod_j e^{-i\beta X}$$

A circuit design for the QAOA ansatz with four qubits can be seen in Figure 5.6. The number of repeated applications of the unitaries is given as the parameter p and can be seen as the gate block inside the big brackets. As both the unitary for H_C and H_x need one parameter each for every layer p, $2p$ parameters $\boldsymbol{\gamma} = (\gamma_1, \cdots, \gamma_p)$ and $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_p)$ are required for a circuit execution. In Figure 5.7, the implementation of the problem Hamiltonian unitary $U_{H_C}(\gamma)$ is shown for the first layer. In the example, we show a circuit for 2 operators, 2 days,

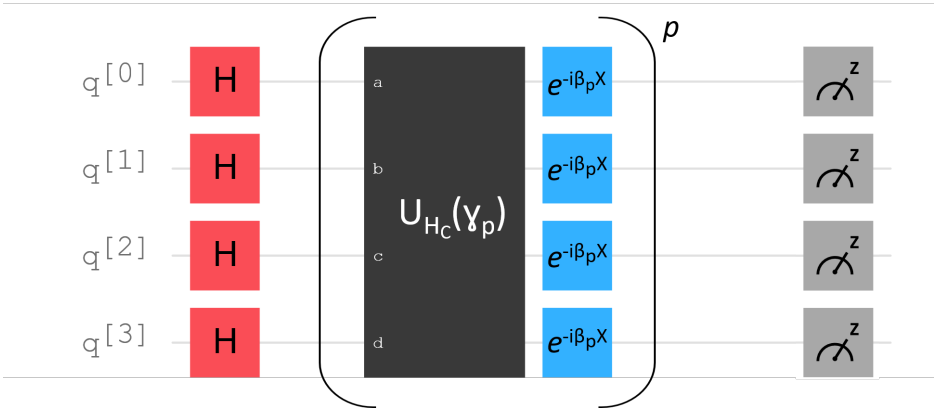


Figure 5.6: QAOA ansatz circuit for four qubits. The circuit uses p layers, where each layer uses different values for β and γ. Black: The unitary for the problem Hamiltonian H_C. Blue: The unitary for the mixer Hamiltonian.

and 2 positions while trying to optimize for the constraints (i) (dark blue), (iv) (light blue), and goals a (dark red) and b (light red). We omitted the constraint (v) for the construction of $U_{H_C}(\gamma)$ since it works comparable to constraint (i) and would lead to a similar gate design.

In our example input, the only restriction to operator position capabilities is for operator $o^{(0)}$, where they are not able to work on position p_0. The dark blue gate block for constraint (i) is constructed with that in mind and places two single rotation gates on the day-position pairs d_0p_1 and d_0p_1. With the right value for γ, it is possible to increase the measurement probability of operator $o^{(1)}$ and decrease the probability of operator $o^{(0)}$ for both day-position pairs, leading to a better overlap with the constraint. For constraint (iv), we need to compare the two different positions for each day to check if the operator is the same. And just as with constraint (i), for the right values of γ, the probability of measuring the same operator for both positions can be minimized. The gate design for goals (a) and (b) is a bit more complicated but can be understood by looking at the problem formulation in Equation 5.5. For both goals, the variable interactions can be seen in the multivariable terms. The punishment term $C_{(a)}$ of goal (a) has every two-variable interaction term, so the gate design is constructed accordingly. For this small example, this is by far the most complicated gate design, but because of the large amount of variable interactions, it is possible to construct a vast amount of output states. Through the classical optimization of the parameters γ, it is then possible to increase the measurement probability of all states that achieve the goal. Likewise for goal (b), where we have the interaction terms between x_0x_2 and x_1x_3. The resulting gate design is therefore much smaller but retains the property of achieving an overlap with all optimal states.

The classical optimization part is then done by three optimizers: conjugate gradient descent (CG) 2.1.3, Powell's algorithm (Powell) 2.1.3, and constrained optimization by linear approximation (COBYLA) 2.1.3. In chapter 6, a comparison of the results for each of these solvers is given.

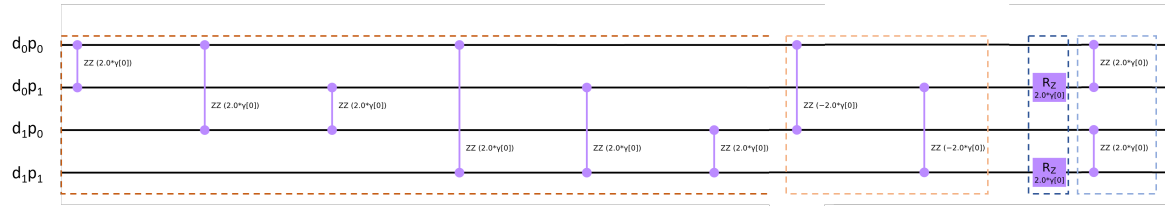


Figure 5.7: $U_{H_C}(\gamma)$ gate design for 2 operators, 2 days, 2 positions, constraints i (Dark blue), iv (Light blue), and goals a (Dark red) and b (Light red).

5.3 Black Box Solver

In this section, we will look into the implementation of F-VQE and VQE using a black box objective function. A black-box objective function f' does not need to be in a Hamiltonian form and generally does not need to conform to any specifications. The output of such a black box function, however, does construct an implicit diagonal Hamiltonian:

$$H = \begin{bmatrix} f'(\mathbf{x}_0) & & \\ & \ddots & \\ & & f'(\mathbf{x}_{\mathcal{N}-1}) \end{bmatrix}$$

This Hamiltonian, however, cannot be efficiently computed since it would require evaluating all $\mathcal{N}$ inputs, essentially solving the problem directly. The advantage of such a black-box objective function is that it is not constrained by the same limitations as a regular Hamiltonian formulation. For example, to convert an objective function with inequalities to a Hamiltonian necessitates a transformation into equalities, which, as we have seen in 2.1, often times requires multiple slack variables, which also results in a large number of additional qubits. Black-box functions, on the other hand, do not impose these constraints, offering greater flexibility in handling different types of objective functions. If we look at constraint (vii), an easy solution would be to define the energy output as:

$$C'_{(vii)}(k,m) = \begin{cases} \sum_i^{D_m} \left(\sum_j^{P} o_{i,j}^{(k)} - \sum_j^{P-1} o_{i,j}^{(k)} \cdot o_{i,j+1}^{(k)} \right) - 2 & , \text{ if } > 0 \\ 0 & , \text{ otherwise} \end{cases}$$

for operator $o^{(k)}$ and sliding window w_m. And since we compute the expectation value of $\langle F_t \rangle_\psi$ and $\langle H \rangle_\psi$ via an approximation in which we sample single data points and compute the corresponding energy, this approach is perfectly suitable. Additionally, checking for constraint violations becomes considerably more straightforward. Rather than attempting to formulate the objective function for all possible variable occupations, we work with a fixed-bit string configuration in which we know the exact assignment of each variable. For instance, when examining constraint (iv), where an operator is not allowed to simultaneously be assigned to more than one position in a day, we no longer need to exhaustively evaluate all combinations of position pairs. Instead, with a known schedule, as with all NP-complete problems, it becomes a trivial process to identify if any operator is working on two or more positions at the same time. This simplification leads to a substantial reduction in the energy computation for a given set of bit strings. Consequently, the new energy value for a set of bit strings is given by the number of constraint violations along with the overlap for each goal.

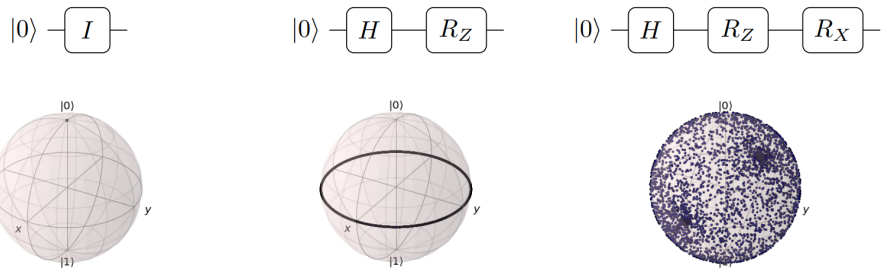


Figure 5.8: Potentially achievable states for single qubit parameterized circuits as seen in [SJAG19, Fig. 1]

Having examined how the new energy values for a black-box objective function are determined, let us now shift our focus to the crucial aspect of selecting the right approach for variational quantum algorithms. This selection is particularly significant, especially in the case of VQE when dealing with multiple Pauli operators such as σ_x, σ_y, and σ_z, where the solution space is notably expansive. But even if only one Pauli operator is used, it is often not trivial to find a good alternative circuit. For a good approach, it is essential to maximize its span in parts of the input space that contain the solution. This span of possible states an ansatz can reach is referred to as its expressibility. For a circuit with just one qubit, this measures its capability to explore the Bloch sphere and generate its states. Figure 5.8 illustrates the states that are potentially achievable for three distinct circuits, each consisting of a single qubit. It is, however, often intractable to construct high-explicity algorithms due to the higher number of parameters and increasing circuit depths. A good ansatz must find a balance between efficiency and expressibility. The expressibility of a given ansatz can be measured through the distribution of unitaries the ansatz can create. This score is a value between 0 and 1, and reaches maximum expressibility at value 0[SJAG19].

Additionally, it is also possible to measure the entangling capability of an ansatz. Circuits with a high entangling capacity can generate highly entangled states even at low depths. Generating highly entangled states helps with efficiently representing the solution space for tasks such as ground state preparation [SJAG19]. Strongly entangled circuits can be realized by layered architectures, including two-qubit gates such as CNOT gates. The entangling capability is also a measure between 0 and 1, but reaches its maximum at value 1 [SJAG19].

The two chosen ansatz circuits for F-VQE and VQE use a combination of parameterized rotation (R_Y) gates and either CNOT gates or parameterized controlled rotation (R_X) gates. In Figure 5.9, the specific design for a circuit with four qubits can be seen. Circuit (a_1) is the same circuit used in the first F-VQE paper and will be used as a baseline. Circuit (a_2) is an ansatz introduced in [SJAG19] and was one of the best performing once in their comparison.

The number of rotation gates and parameters for circuit (a_1) is given by: $q + p(4\left\lfloor \frac{q}{2} \right\rfloor - 2)$, and the number of CNOT gates by: $p(2\left\lfloor \frac{q}{2} \right\rfloor - 1)$, for q qubits and p layers. For circuit (a_2), the number of parameters is given by the total number of gates: $2d(2q)$, where we need $p(2q)$ rotation R_Y and controlled rotation R_X gates. In Figure 5.10, expressibility scores and entangling capabilities are shown for both ansatz circuits with four qubits and varying numbers of layers p indicated by the different colors. The expressibility score is on a logarithmic scale for better clarity between individual scores.

Expressibility scores for well-constructed ansatz circuits for problems with multiple Pauli

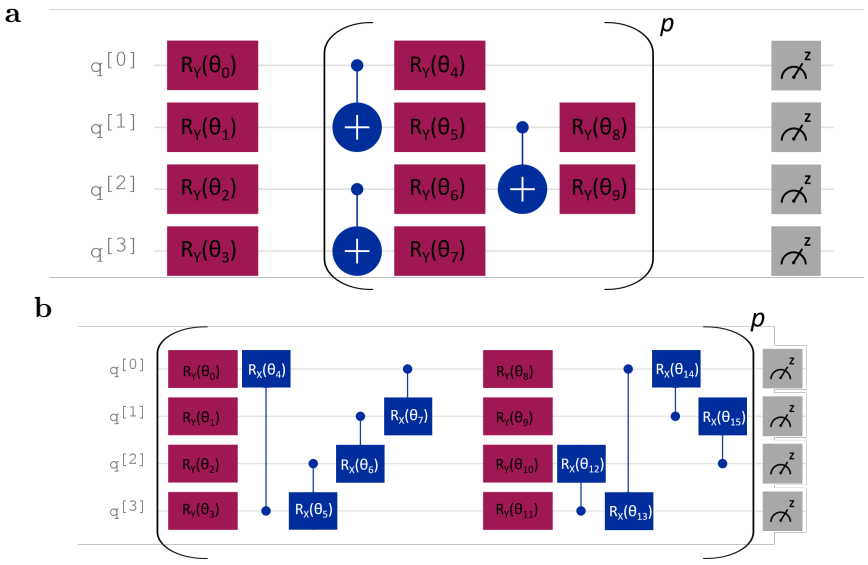
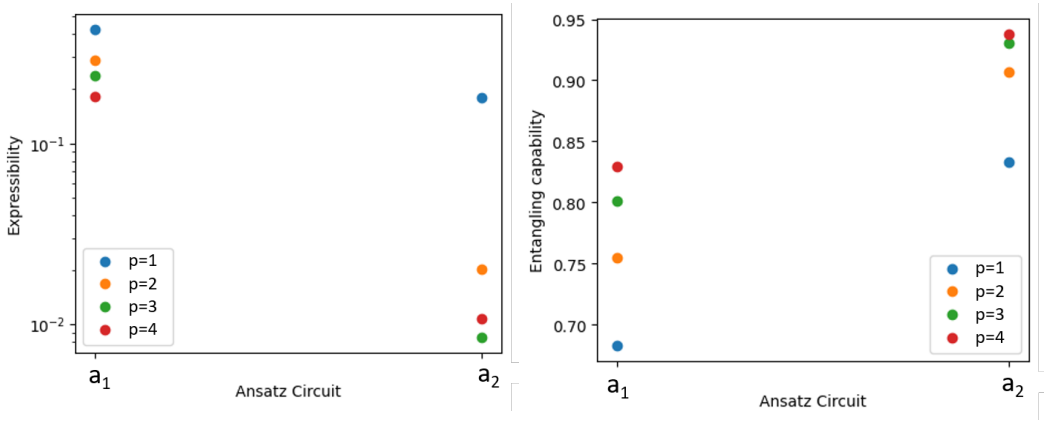


Figure 5.9: Ansatz circuits for four qubits each for F-VQE and VQE. Both circuits use p layers, where each layer uses additional values for θ. (a) Circuit a_1: Only rotation R_Y gates use angles θ_i. Controlled NOT gates are static. (b) Circuit a_2: Both rotation R_Y and controlled rotation R_X gates use parameters θ_i.

operators typically fall within the range of 0 to 0.2 [SJAG19]. We only investigate problems with Pauli σ^z operators, allowing for worse expressibility scores since not all states need to be achievable. Still, it is important to investigate if the better expressibility score for circuit (a_1) will also result in improved results or if even less-than-optimal expressibility scores are adequate. It is also noteworthy that there is an apparent enhancement in expressibility as the number of layers p increases. Circuit (a_2) also exhibits lower expressibility scores, even for fewer layers. Likewise, the entangling capability shows an increase with more layers, and again, the second circuit shows an improvement compared to the first ansatz. Interestingly, the second ansatz reaches its peak expressibility at $p = 3$ and not at the maximal tested number of layers. This is an indication that the circuit has achieved its maximum expressiveness, and differences between $p = 3$ and $p = 4$ can be attributed to the inherent stochastic and noisy nature of measurements. It remains to be determined whether the added complexity and parameter count justify this improvement.

Just as it is important to choose the right alternative circuit, it is equally essential to find a suitable initial parameter assignment. While the ansatz, amongst others, tries to provide a vast solution space with good ground state overlap, a good initial parameter vector $\boldsymbol{\theta}$ specifies where one starts to search for the optimal solution. If an initial input state is chosen in the neighborhood or inside the space that contains the solution, it is less likely to run into local minima, and fewer optimization steps are needed to reach a solution. There is extensive research on how one can choose such an initial state, ranging from multi-start methods, local pre-optimization, and classically computed solutions for a relaxed version of the given problem. In our implementation, we tried two different methods:

1. Uniform distribution over all input states $|+\rangle = R_Y(\pi/2)\,|0\rangle^{\mathcal{N}}$
2. Random initialization for $\boldsymbol{\theta}, \boldsymbol{\theta}_i \in \{-\pi, \pi\}$



	Circuit layers for ansatz (a_1)			
	1	2	3	4
Expressibility	0.427022	0.288892	0.236862	0.180237
Entangling capability	0.683144	0.755244	0.801661	0.829320

	Circuit layers for ansatz (a_2)			
	1	2	3	4
Expressibility	0.179573	0.020272	0.008531	0.010727
Entangling capability	0.832683	0.906962	0.930052	0.938004

Figure 5.10: Comparison of expressibility and entangling capability for both circuits with four qubits and varying number of layers p

In Section 6, an overview of the results for each approach is given.

Next, we want to discuss how we compute the effective step size $\alpha_k \nabla f(\mathbf{x_k})$ for our approach. The gradient descent algorithm slowly decreases its effective step size near stationary points, resulting in no further improvement once such a point is reached. While we try to combat this in part with the adaptation of τ as seen in Section4.3, we want to incorporate an additional approach: normalized gradient descent [SYRY21]. Its core idea is to ignore the magnitude of the gradient $\nabla f(\mathbf{x_k})$ and introduce a consistent effective step size:

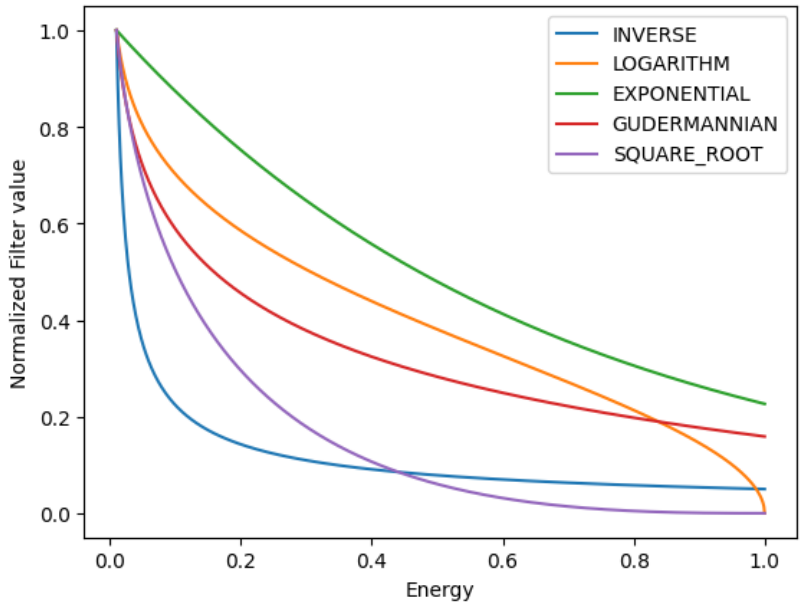
$$\mathbf{x_{k+1}} = \mathbf{x_k} - \alpha_k \frac{\nabla f(\mathbf{x_k})}{||\nabla f(\mathbf{x_k})||}$$

Through the normalization of the gradient, the step size α_k becomes the effective step size. This should work well with our adaptive τ, since this approach only changes the optimization step, while τ influences the actual gradient computation.

5.3.1 Additions for F-VQE

F-VQE requires additional considerations regarding the filter operators and the setting of bounds to keep $E_{min}, E_{max} \in (0, 1]$. We begin by looking at the different filter operators we used in our comparison. An overview of all the filtering function definitions is given in Figure 5.11. The first three filter functions are taken from the previous work on F-VQE,

while the others are derived from sigmoid functions. Sigmoid functions are a good fit for a filter operator since they are both real valued and strictly monotone on the interval [0, 1].



Filter	$f(H;\tau)$	τ
Inverse	$H^{-\tau}$	0.65
Logarithm	$(-log(H))^{\tau}$	0.51
Exponential	$e^{-\tau H}$	1.5
1 - Gudermannian	$1 - \frac{3}{2} \cdot tan^{-1}(tanh((H/2)^{\tau}))$	0.2
1 - Square root division	$(1 - (H/\sqrt{H}))^{\tau}$	2.5

Figure 5.11: Compared filtering operators. The normalized filter value is computed by dividing all values by their maximum. Filter values are computed for energies in range $[E_0, 1]$ where $E_0 = 0.001$ is the ground state. For each filter operator, the used value for τ for this plot is given in the table. For the computation of the filter value, a problem instance with six qubits was used.

Before we apply filtering operators to our objective function, we need to re-scale the energy range to [0, 1]. Otherwise some filter operators are no longer well defined or strictly decreasing. This can be done via the parameters α and β in Equation 5.1. To find suitable parameters, we first initiate the algorithm with unscaled values for α and β and start a calibration run to get an even distribution over the energy spectrum $\vec{E}$. The lower $l_{f'}$ and upper $u_{f'}$ bounds of the objective function are then given by $l_{f'} = 0.001$ and $u_{f'} = max(\vec{E})$. The lower bound is arbitrary and chosen to be larger than 1. To ensure the minimal energy E_{min} does not decrease beyond this point, the lower bound $l_{f'}$ is added to the objective

function. Afterwards, the new parameters $\bar{\alpha}$ and $\bar{\beta}$ are given by:

$$\bar{\alpha} = \frac{\epsilon}{u_{f'}}\alpha; \quad \bar{\beta} = \frac{\epsilon}{u_{f'}}\beta$$

where $\epsilon \in (l_{f'}, 1]$ is the new maximal energy of the scaled objective function. Since it is not probable that we will find the actual maximal energy in the calibration run, it is prudent to set ϵ to a value not too close to 1. We choose $\epsilon = 0.8$, which results in almost no energy values larger than 1 for the following iterations. This approach works well since measurement probabilities for high-energy states are reduced with each iteration. It is thus not likely to measure higher maximal energies than in the first calibration run. We also only consider one parameter $\gamma = \frac{\alpha}{\beta}$ for our objective function since we are mostly interested in the difference of the goal and constraint energy for constraints $c \in \{(i), (ii), (iii), (iv), (v), (vi), (vii)\}$ and goals $g \in \{(a), (b)\}$:

$$f' = l_{f'} + \bar{\alpha}\sum_c C'_c + \bar{\beta}\sum_g C'_g \equiv l_{f'} + \frac{\epsilon}{u_{f'}}\left(\gamma\sum_c C'_c + \sum_g C'_g\right)$$

6 Evaluation

In this section, we present a comprehensive evaluation of the methodologies in Chapter 5 and algorithms in Chapter 4 and Subsection 2.3.4 presented earlier. Our primary objective is to gain a deeper understanding of the F-VQE algorithm and to find suitable approaches to optimize its performance. We will analyze and compare the different strategies and solvers we introduced previously. In addition to delving into the details of the three primary solvers, F-VQE, VQE, and QAOA, we will also provide a comparative analysis with the Grover solver (discussed in Section 3.1.1), which has been developed by the DLR. Through this evaluation, we aim to provide valuable insights into the practical applicability and limitations of the F-VQE algorithm for real-world problems.

Section 6.1 introduces both the simulator and quantum hardware on which all tests were run. Afterwards, section 6.2 will present an outline of the chosen test cases. In section 6.3, we will delve into the heart of our findings. Subsection 6.3.1 will offer an overview of the hyperparameters employed in the F-VQE and VQE approaches and present the optimization process and the final values. Lastly, we will conclude our evaluation in Section 6.4, highlighting the key takeaways. We aim to showcase the capabilities and limitations of the F-VQE approach for real-life combinatorial scheduling problems and evaluate its suitability for such applications.

6.1 Hardware

Due to time and implementation constraints, test cases could not be run on the same hardware. For the QAOA implementation, we relied on the "ibm_qasm_simulator," which introduced delays due to the need for constant data transfer between our local system and the Qiskit backend.

Both F-VQE and VQE were implemented to support parallelization for multiple circuit executions. It was thus able to spread the $2m + 1$ circuit evaluations per iteration over multiple processes. To this end, we used the High-Performance Data Analytics Platform (HPDA)[Cen23] developed by the German Aerospace Center (DLR) and the Leibniz Supercomputing Centre (LRZ). The system relies on Lenovo's ThinkSystem SD650-N V2 servers and DSS-G memory. It consists of 61 CPU nodes, each equipped with two 40-core Intel Xeon Platinum processors. The nodes are interconnected using Infiniband HDR technology, allowing the platform to process data at speeds of 320 GB/s. Furthermore, we were able to run two test cases for F-VQE on actual quantum hardware. To achieve this, we also utilized the Qiskit backend, employing two quantum processors: "ibm_nairobi" [IBM23] with seven qubits and "ibm_brisbane" [IBM23] with 127 qubits.

Given that Grover's algorithm necessitates only one circuit evaluation, we couldn't employ the same parallelization approach used for F-VQE and VQE. Regrettably, we faced challenges with the Qiskit backend for the deep circuits Grover requires, preventing us from executing our simulations there. Consequently, we had to conduct the simulations on our local system, imposing severe constraints on the problem instance sizes.

6.2 Test Cases

In our evaluation, we consider a set of test cases that represent small combinatorial scheduling problems. These test cases vary in terms of the number of positions, operators, and days, making them suitable for small-scale testing of the F-VQE and VQE algorithms. Specifically, our test cases encompass scheduling scenarios involving:

Positions: Test cases are designed to handle between 2 and 3 positions. This variability allows us to assess the performance of our algorithms for different constraints, such as (i) or (iv).

Operators: The number of operators considered in our test cases ranges from 2 to 6 for binary-encoded operators and 2 to 3 for one-hot encoding.

Days: Test cases involve scheduling activities over a span of 3 to 5 days. The consideration of different time frames is crucial for constraint (vii) to evaluate multiple sliding windows.

Each operator's capabilities and work outage periods are randomly determined, reflecting the realistic and dynamic nature of scheduling challenges. Additionally, partially filled schedules vary from 10By including a diverse set of small-scale test cases, we aim to assess the suitability of the F-VQE algorithm for tackling real-life combinatorial scheduling problems with a wide spectrum of constraints. In Table 6.1, a single example with three days, four operators, and three positions can be seen.

Day d_0			Day d_1			Day d_2		
p_0	p_1	p_2	p_0	p_1	p_2	p_0	p_1	p_2
$o^{(1)}$	-	-	-	-	-	-	-	$o^{(2)}$

Operators	p_0	p_1	p_2
$o^{(0)}$	√	√	√
$o^{(1)}$	√	×	×
$o^{(2)}$	√	√	√
$o^{(3)}$	√	√	×

d_0	d_1	d_2
×	√	√
√	√	√
√	√	√
√	√	×

Table 6.1: Test case with 3 positions, 4 operators, and 3 days. Top: Partial schedule with operator $o^{(1)}$ at shift p_0d_0 and operator $o^{(2)}$ at shift p_2d_2. Left: Operator position capabilities. Operator $o^{(1)}$ can only work on position p_0; operator $o^{(3)}$ cannot work on position p_2. Right: operator outages. Operator $o^{(0)}$ has an outage on day d_0 and operator $o^{(3)}$ on day d_2.

6.3 Results

In this section, we will look at the results of our evaluation. We will begin by explaining how we found and optimized various hyperparameters, and then we will show the results for the previously defined test cases for F-VQE, VQE, QAOA, and the Grover solver. And lastly, we will show the results obtained from an actual quantum computer.

6.3.1 Hyperparameters

Both F-VQE and VQE utilize multiple variables to enhance the optimization process. It is important to choose good values for these hyperparameters since, otherwise, no solution can be found. To that end, we used the Bayesian optimization SMAC3 algorithm [LEF+22], which is a global optimization strategy for black-box functions. It is usually employed to optimize costly functions. Importantly, the algorithm is able to optimize for multiple objectives over multiple problem instances. This ensures that found hyperparameter configurations do not only perform for a single problem instance or objective, but for many. To make the optimization meaningful, care was taken in the implementation to choose a consistent seed to ensure reproducibility. Unfortunately, due to time constraints, we were not able to conduct additional hyperparamter optimizations for different initialization strategies, ansatz states, filter operators, or the QAOA and Grover solvers. This should be kept in mind for the later comparisons. The following hyperparameters were used in the implementation and are sorted by relevance:

Learning Rate:
consistent effective step size for each iteration. It should not affect the computation time but will most likely require more iterations for lower values to find a good solution.

Number of Iterations:
indicates how many training iterations are run. It directly affects the required computation time. This hyperparameter was taken into consideration to find potential dependencies between the number of required iterations and other hyperparameters.

Quantum Circuit Evaluations:
regulates how many samples are taken per iteration. It has significant effects on the accuracy of the computed gradient. It has to be seen if greater accuracy leads to better results or if the introduced noise allows the optimization to escape local minima.

Ansatz depth:
defines how many layers the ansatz circuit has. Affects computation time for simulations. It also introduces more parameters θ to the objective function, making it more difficult to find the global minima. It is important to find a minimal alternative design that is expressible enough to have ground-state overlap.

Momentum:
level of influence from previous gradient 2.1.2. As with the learning rate, this hyperparameter can significantly influence the speed and accuracy of the optimization process. It is, however, not clear if our objective function is suitable for a momentum-based approach. If this is not the case, we will expect values near zero.

Initial value for τ:
The parameter τ influences the effective step size, making it important to choose a good initial point. Bad initial values could make the beginning of the optimization process and the updating of subsequent values for τ harder.

Within this work, all six hyperparameters were optimized. We did not, however, use the same test cases we introduced in 6.2 but rather took random samples where no care was taken that they were actually solvable. This should ensure that optimized hyperparameter values are not only suitable for our test cases. To evaluate the hyperparameter configuration and the associated model quality, we used three objectives:

Mean energy:
average energy of all samples after the final iteration. However, it is not sufficient, as it is not clear if the actual ground state was found.

Success rate:
indicates how many of the final set of samples are valid schedules. Together with the average energy, it is very likely that the ground state was found. Still, there is no guarantee that the schedule with optimal goal overlap will be found.

Computation time:
ensures that a trade-off between solution quality and required time is reached. Otherwise, higher values for training iterations or circuit layers would always be preferred.

Due to the limited computation time on quantum computers, hyperparameter optimization was performed as a simulation on a classical computer. Since we optimized the parameters for F-VQE and VQE, we were able to run the optimization process on the LRZ HPDA cluster. In Figure 6.1, the full optimization process can be seen. We compare the two objectives "mean energy" and "success rate" for all found hyperparameter configurations. Each blue dot is one hyperparameter configuration, and its "mean energy" and "success rate" are the averages over multiple trials where different test case samples were used to compute a solution. The incumbents are the best configurations for a particular iteration of the hyperparameter optimizer. One iteration encompasses multiple trials but is not always consistent in the number of trials. The number of trials within one iteration depends on the ease with which the optimizer finds a better solution with which it can start a new iteration. We displayed the optimal configurations of the first and last three iterations, recognizable by the different colors. The red dotted lines show the improvement from one iteration to the next, and the black dotted lines show the advancement within one iteration. It can be seen that the optimizer was stuck inside a local minima for most of its iterations and only managed to break out and find the global optimum in its last iterations.

This can also be seen in Figure 6.2 (a), which illustrates the success rate across all trials alongside their respective learning rates. During the initial and mid-phase trials, a learning rate close to one was usually chosen, leading to suboptimal success rates of $> 90\%$ with high variance. In the final trials, the optimizer converged to configurations with learning rates around $\alpha \approx 0.4$, significantly enhancing the success rate to nearly 100% with low variance. On the other hand, 6.2 (b) sees no improvement for different ansatz circuit depths. Instead, it shows that later trials lean toward circuit depths $d = 1$ since they require less computation time and still yield the same results. This is likely the case since even the circuit design

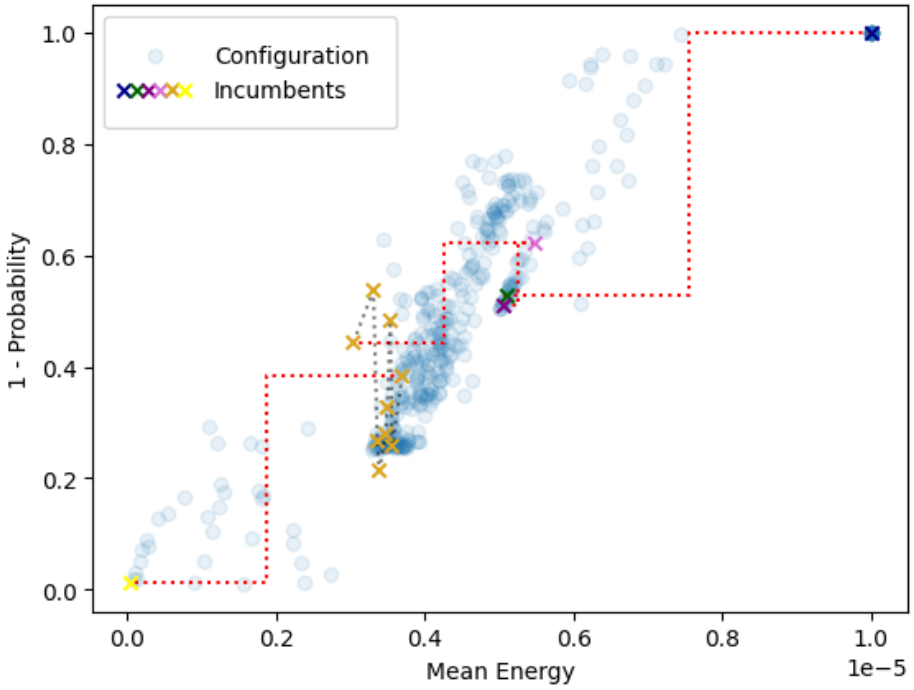


Figure 6.1: Hyperparameter optimization over all configurations

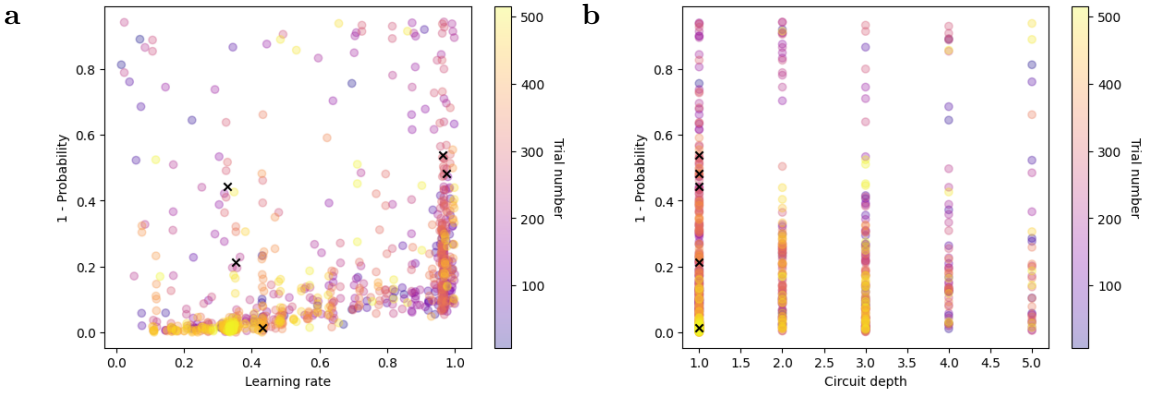


Figure 6.2: Success rate for all trials. Black crosses show the incumbents of the last two iterations (a) vs. learning rate. Eventual convergence to $\alpha \approx 0.4$ in final trials (b) vs. ansatz circuit depth. Different amounts of circuit layers $d \in \{1, \cdots, 5\}$ for ansatz 5.9 (a) No improvement for deeper circuits can be seen.

with one layer is expressible enough to have sufficient overlap with the ground state. It is interesting to see if later experiments show the same outcome.

Figure 6.3 illustrates the relationship between momentum and learning rate. The configurations are differentiated by their spatial density, revealing two distinct clusters at $\alpha \approx 0.3, m \approx 0.2$ and $\alpha \approx 0.9, m \approx 0.9$. These clusters correspond to the local minima, in which the optimizer stayed most of its iterations ($\alpha \approx 0.9, m \approx 0.9$) and the global minima with lower overall cost ($\alpha \approx 0.3, m \approx 0.2$). This can also be seen by the higher

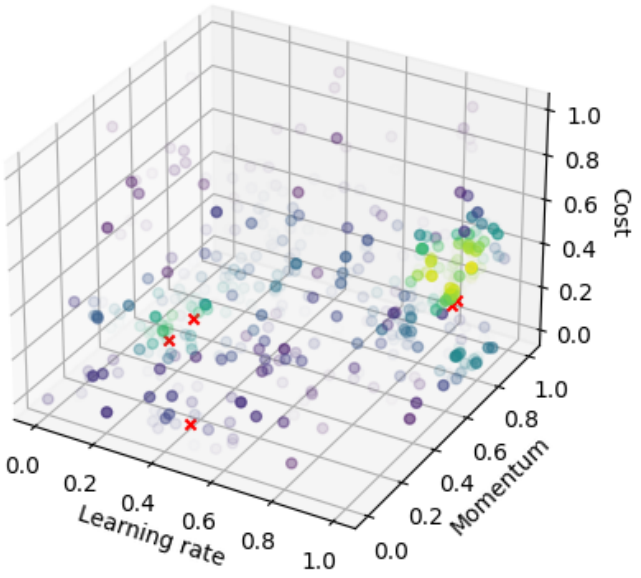


Figure 6.3: Mean Energy as Cost for all configurations with learning rate and momentum. Red crosses show the incumbents of the last two iterations. Configurations are density-color-coded showing two distinct clusters.

concentration of configurations and higher average cost in cluster 1. Interestingly, for higher values of both α and m, it appears that we take excessively large steps, causing us to overshoot the global minima. In contrast, using more precise step sizes enables us to locate the global minimum effectively.

Finally, Figure 6.4 presents the relationship between the number of training iterations and the quantity of samples taken in each iteration. Each configuration is depicted alongside its corresponding cost, which, in this case, is represented by the mean energy. This graph shows that there might also be two distinct local minima alongside the global minima. The first cluster involves configurations with a low number of iterations and samples, while the second cluster comprises configurations with a higher number of iterations and a still low sample count. Notably, both clusters feature two incumbents from the second-to-last iteration. However, the cluster with more training iterations exhibits an overall lower mean energy. The global minimum, on the other hand, is in cluster three and is characterized by a low number of iterations but a high number of samples. This cluster is notably smaller, as the optimizer only managed to discover it in the final iteration. By looking at the last two clusters, both of which exhibit a lower mean energy than the first cluster, it seems that more iterations are needed when fewer samples are used, and vice versa.

6.3.2 Results on a Simulator

In our evaluation, we initially assessed the convergence of all algorithms using a 6-qubit instance of a scheduling problem with two positions, three days, and two operators. Then we used the same problem with one-hot encoding, which resulted in an instance with 12 qubits, to compare the performance of F-VQE, VQE, and QAOA. We did not include comparisons with classical solvers in this work since all instances under consideration can be solved

exactly, rendering heuristic approaches unnecessary. Our primary focus lies in analyzing the convergence and scalability of the VQAs. All quantum processors and simulators were accessed through the Qiskit framework [Cro18].

We conducted tests for both encodings, evaluating the performance of the F-VQE, VQE, and QAOA algorithms. Furthermore, we examined QAOA with three classical optimizers, which we discussed in Section 2.1. For VQE and F-VQE, we explored two different ansatz circuits, each initialized from different points. The Grover solver by [SGGC+21] was also used to compute a solution for one small and one midsized instance. The algorithm is currently only constructed for binary-encoded problem inputs, so we only ran one experiment per problem instance. Each VQA experiment ran for 55 iterations and used 2400 measurements per iteration. The Grover test cases were tested for Grover iterations between 1 and 10 and also used 2400 shots. To make the comparison between different approaches easier, we computed the minimum energy of each test case and used it to normalize the results to the range [0, 1].

$$E^* = \frac{\bar{E} - E_{min}}{E_{max} - E_{min}} \in [0, 1]$$

where E_{min} and E_{max} are the minimum and maximum energy, respectively, for a given problem instance, and $\bar{E}$ is the weighted average of the unfiltered energy of the given algorithm per iteration:

$$\bar{E} = \frac{1}{M} \sum_{i=0}^{M} M_i \cdot E_i$$

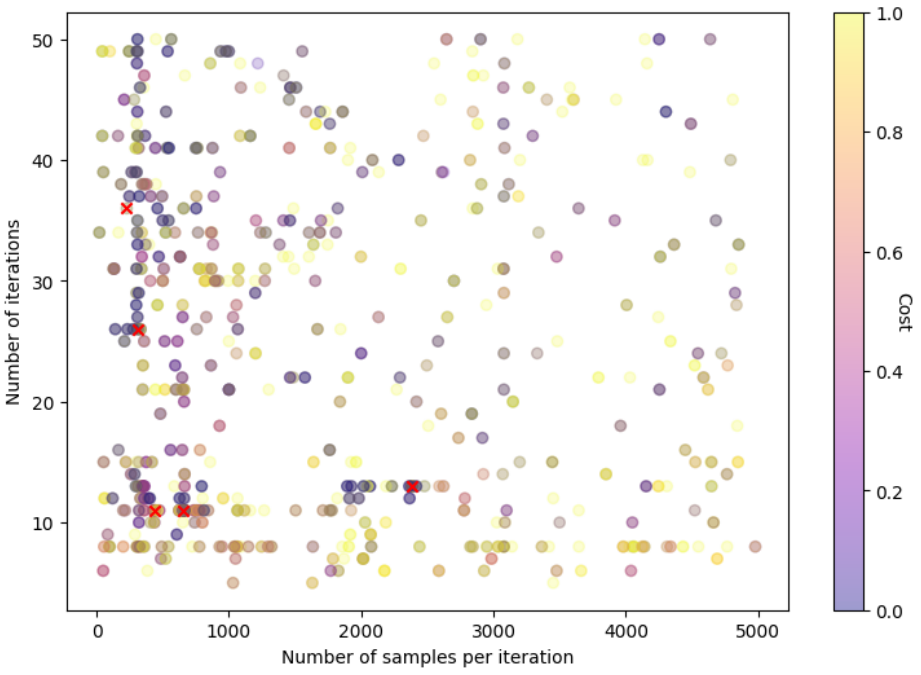


Figure 6.4: Mean energy cost for 1-55 training iterations and 1-5000 samples per iteration. Formation of three low-cost clusters for iterations and samples: (35, 500), (10, 500), and (10, 2000)

for M samples per iteration, where M_i is the amount of measurements for energy E_i. If multiple approaches are compared in a single plot, the minimal and maximal energy are the global values for all given approaches. This also means that, while in a plot with a single problem instance, the scaled energy E^* should approach the ground state $E_{min} = 0$, the ground state energy for a set of problem instances is not necessarily 0 for all of them. For a good solution, we want the scaled energy E^* to approach the minimal energy E_{min}.

We also computed the probability for measuring the ground state $P_f(E_{min})$ energy over each iteration and the probability of finding a schedule $P_s(C)$ that complies with all constraints. Ideally, we would like to find the ground state with a frequency of $P_f(E_{min}) \approx 1$ and measure the correct solutions with a probability of $P_s(C) \approx 1$.

Performance on 2p2o3d Schedule

We used a small initial example to test the viability of all tested algorithms since it requires sufficiently few qubits, can be simulated on any hardware, and can also be run by essentially every quantum processor and simulator. To make the results comparable, we use the same number of samples, ansatz circuits, and initial points. The number of circuit layers will vary since QAOA requires more to reach decent results. In Table 6.2, the used test case can be seen. We decided not to include any outages since this would make constraint (iv), where every operator is only allowed to work in one position per day, unachievable. For this small example, the sliding window constraint is also not applicable since both operators have to work three out of three days. This makes it a good example to compare Hamiltonian and black box solvers since both approaches can implement the same constraints. For bigger examples, we would need to artificially constrict the constraint set for black box solvers to make a meaningful comparison. We therefore do not use QAOA in our comparisons for larger problem instances.

Day d_0		Day d_1		Day d_2	
p_0	p_1	p_0	p_1	p_0	p_1
-	-	-	-	$o^{(1)}$	-

Operators	p_0	p_1
$o^{(0)}$	√	√
$o^{(1)}$	√	×

d_0	d_1	d_2
√	√	√
√	√	√

Table 6.2: Test case with 2 positions, 2 operators, and 3 days. Top: Partial schedule with operator $o^{(1)}$ at shift p_0d_2. Left: Operator position capabilities. Operator $o^{(1)}$ can only work on position p_0. Right: operator outages. No outages for any operator

For assessing the quality of a solution, a good metric is necessary to achieve comparable results. To this end, we compare the scaled weighted averages E^*, the success probability $P_s(C)$, the percentage amount of measurements for the current best energy $P_m(M)$, and the ground state frequency $P_f(E_{min})$. In each comparison, all other hyperparameters are kept

constant. But before we compare the performance of each algorithm, we first want to find the best-performing alternative circuits, initial points, and classical optimizers.

In Figure 6.5, we compare three different classical optimizers for the QAOA algorithm on our small problem instance. Each algorithm is also shown with three different circuit layers, ranging from one to three. Additionally, in Table 6.3, the corresponding execution times can be found.

We can see in both that with increasing layers, the performance and execution time increase. For all three optimizers, we find higher ground state frequency, lower average energy, and a higher success rate and measurement percentage for increasing circuit layers. We can also see that, especially for the optimizer "Conjugate Gradient (CG)", the algorithm did not run the full 55 training iterations. This is always the case when the optimizer is unable to find a good next optimization step. This could either be because the algorithm found the global optimum, or because the optimizer is stuck at other extrema, the gradient vanishes, or only a local minima is found. Since no optimizer had remarkable ground state frequency, it can be said to be one of the later two. In this small test case, both the "Powel" and the "CG" optimizers achieved the best performances. But since the last optimizer, "Coblya," produces almost similar results and takes the least amount of time to compute all 55 iterations, we choose it going forward. Interestingly, it is important to note that while the execution times of "CG" and "Coblya" seem similar, one has to consider that "CG" only ran for a maximum of five iterations. Even with just five iterations, it found better solutions than "Coblya".

	Layers 1	Layers 2	Layers 3
Coblya	03:28	05:48	08:21
Powell	09:33	35:31	39:33
CG	05:14	09:22	14:19

Table 6.3: Execution times (in minutes) for each classical optimizer and circuit layers $p \in [1, 3]$ Coblya is the fastest, even though CG only runs for a maximum of 5 iterations.

Next, we look at the two different encoding schemes. It is important to keep in mind that while both present the same problem instance, one-hot encoding requires $\frac{log_2 N}{N}$ more qubits (where N is the number of operators). In Figure 6.6, both encoding schemes are shown for F-VQE with the ansatz circuit (a_1) and (a_2), as well as for QAOA with the optimizers "Coblya" and "Powell".

For QAOA, we chose the best-performing number of circuit layers $p = 3$. Interestingly, we observe the opposite for both algorithms: while F-VQE performs better in terms of success rate $P_s(C)$ and average energy E^* with binary encoding, QAOA performs better with one-hot encoding. It also does not make a difference what optimizer method or what alternative circuit was chosen. Notably, F-VQE with one-hot encoding fails to discover the ground state altogether, suggesting convergence to a local minimum. This is further confirmed by the nearly 100% measurement percentage $P_m(M)$, signifying that all samples originate from this local minimum, consequently resulting in a 0% ground state frequency $P_f(E_{min})$. This is also the case for QAOA, where, even with lower average energy and a higher success rate $P_s(C)$, the ground state frequency is 0% for both optimizers.

While we did not look at the different ansatz circuits, initial point selection, or different

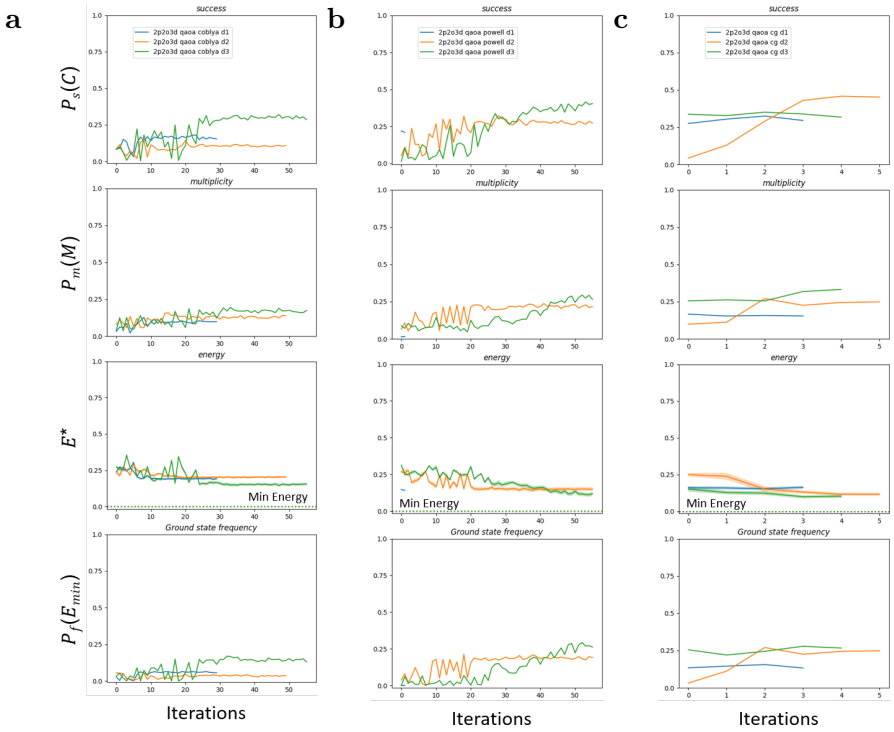


Figure 6.5: $P_s(C)$, $P_m(M)$, E^*, $P_f(E_{min})$ are measured for all three QAOA classical solvers using varying circuit depths from 1-3. (a) Coblya is a classical solver. Not gradient-based. (b) Powell is a classical solver. Not gradient-based. (c) conjugate gradient. Gradient based.

filter operators, we want to show the preliminary results for all variational solvers for our small problem instance. For this comparison, we opted for ansatz circuit (a_1), the uniform initial point, and the "Inverse" filter, as these choices have minimal impact on small-scale problem instances. Even further, the initial point selection does not result in comprehensive results. This is most likely due to the fact that a random initialization is inherently noisy and only achieves better performance on certain occasions. In Figure 6.7, the performance of the F-VQE, VQE, and QAOA algorithms is shown. For this comparison, we chose to use only one circuit layer for F-VQE and VQE while using three for QAOA. The number of layers is gathered from the previous hyperparameter optimization. F-VQE and VQE use the same remaining hyperparameters, and both are employing the parameter shift rule to update the parameters θ. For the small-problem instance, we see consistently better performances in each metric. While VQE still achieves comparable results, QAOA only reaches half the success rate $P_s(C)$ and a quarter of the ground state probability $P_f(E_{min})$. For VQE, we can also observe a noisier optimization process. It finds a successful yet not optimal solution in the early iterations but manages to escape these local minima, resulting in a decrease in the success rate. Because of this initial local minima, VQE takes more iterations to find the actual optimal result. Still, even after reaching the global optimum, the algorithm remains noisy and almost escapes the global minima.

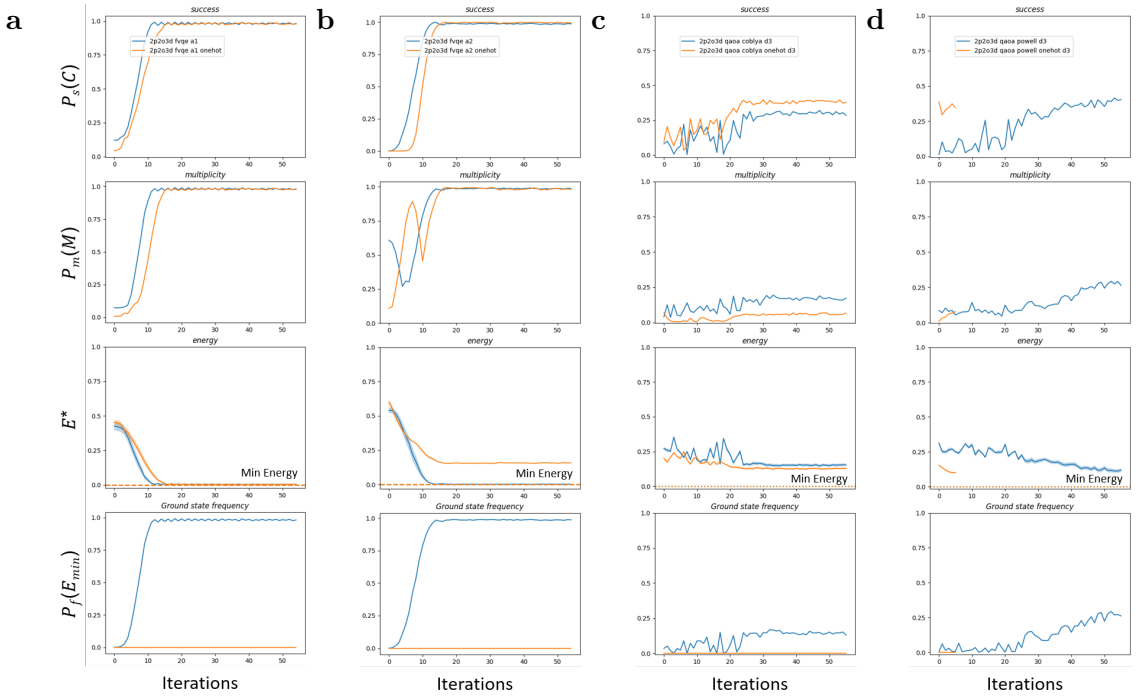


Figure 6.6: $P_s(C)$, $P_m(M)$, E^*, $P_f(E_{min})$ are compared for both encoding schemes. (a) F-VQE with ansatz (a_1). One-Hot encoding runs in local minima at $E \approx 0.174$, very close to global minima. $E \approx 0.165$ (b) F-VQE with ansatz (a_2). One-Hot also fails to find a global optimum but runs into worse local minima. (c) QAOA with Coblya and depth 3. One-Hot performs better in $P_s(C)$, $P_m(M)$, and E^* but fails to obtain the ground state. (d) QAOA with Powell at depth 3. One-Hot is unable to find update steps after iteration 6.

Performance on larger schedules

For the larger problem instances, we chose three examples to examine in more detail. The first two examples use two or three positions, six operators, and three days. Both require 18 qubits for a binary-encoded problem. The third example consists of three positions, five operators, and three days and uses 27 qubits. Bigger problem instances are increasingly difficult to simulate, which is why we did not choose even larger instances.

In Figure 6.8, we compare both ansatz circuits, each with one layer for the last two problem instances. We can see almost equal performance in both examples and ansatz circuits, with only a small difference in mean energy for the example with three positions. Still, both approaches are not able to find the ground state and run into local minima. For the second example, both approaches are able to find the ground state with almost 100% accuracy. Circuit (a_2) converges slightly faster to the optimum. After finding the global minima, it also predominately finds a single solution until it discovers a second global minima, which results in an even split in the measurement probability. Circuit (a_1), on the other hand, finds both minima directly.

Since both alternative circuits seem to perform similarly, we will choose the first one for further test cases since it requires fewer quantum gates and parameters. In the next test

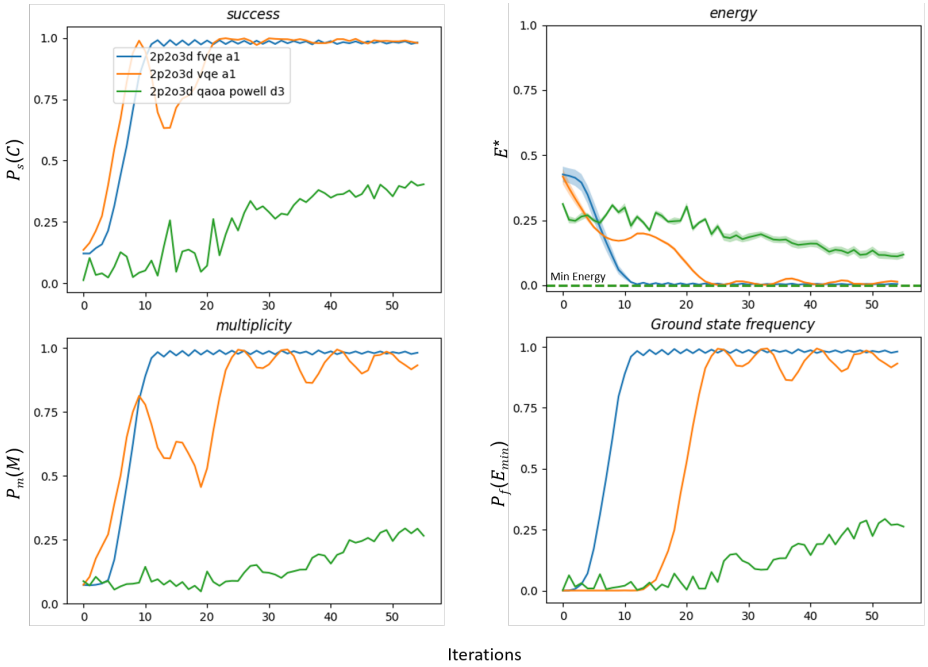


Figure 6.7: Performance comparison of F-VQE, VQE, and QAOA for small problem instances of 2 positions, 2 operators, and 3 days. F-VQE outperforms both other algorithms in every metric.

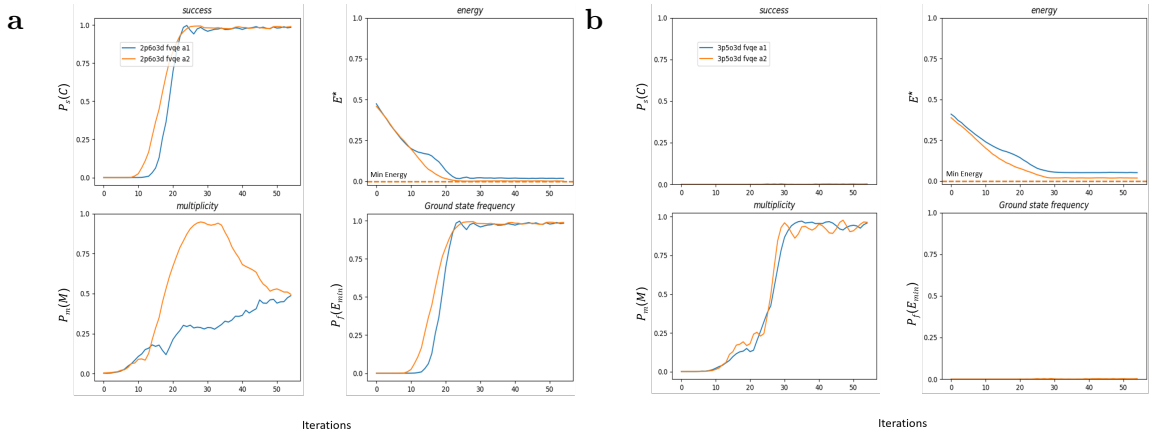


Figure 6.8: Performance analysis for ansatz (a_1) and (a_2). Both ansatz circuits show similar results. (a) Problem instance with 2 positions, 6 operators, and 3 days (b) Problem instance with 3 positions, 5 operators, and 3 days.

case, we want to compare the different filtering operators with each other. In Figure 6.9, all five filter operators and the VQE algorithm (used as a benchmark) are compared using the example of two positions, five operators, and three days. With the exception of the "inverse" filter, all filter operators exhibit inferior performance compared to VQE. While they manage to find the ground state, they struggle to minimize the mean energy and achieve high success rates. Next to the "Inverse" filter, only the "Squared Root" filter attains a 100% success rate

	2p6o3d		3p5o3d	
	Ansatz (a_1)	Ansatz (a_2)	Ansatz (a_1)	Ansatz (a_2)
Time	17:19	18:10	2:33:36	8:15:36
Evaluations	5670	7830	8586	11718
parameters	52	72	81	108

Table 6.4: Execution time (in hours and minutes), number of circuit evaluations, and number of optimized parameters for both circuit ansätze and both problem instances.

and a mean energy close to E_{min}, but it requires an additional 10 iterations compared to VQE. While each filtering operator shows improvement in the final iterations, they are still slower to converge than VQE, or the "inverse" filter. These results are unexpected, especially when compared to the findings in previous work [AMR+22], where both the "Exponential" and "Logarithm" filters outperformed the VQE implementation. Due to the performance of the other filtering operators, we will only use the "inverse" filter in the following experiments.

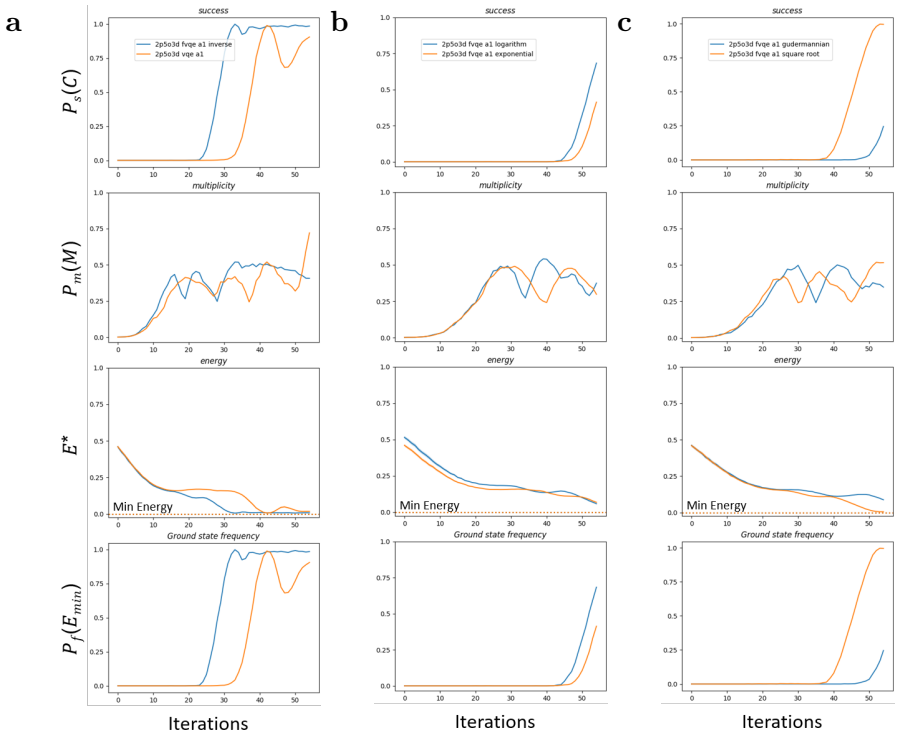


Figure 6.9: Comparison of $P_s(C)$, $P_m(M)$, E^*, and $P_f(E_{min})$ for all five filtering operators and the VQE algorithm. (a) "Inverse" filter and VQE. As with the small problem instance, F-VQE out performs VQE (b) "logarithm" and "exponential" filter operators. Both filter operators show similar results, with the "Logarithm" filter having a slight edge in convergence speed. (c) "Gudermannian" and "square root" filters The "square root" filter achieves a success rate and ground state probability of close to 100%.

In the next test case, we aim to assess the performance of F-VQE and VQE for larger problem instances and validate our earlier hypothesis. In Figure 6.10, the results for all

three examples are used for the comparison between F-VQE and VQE, both using a single layer approach (a_1). And just as with the small example, F-VQE consistently performs better even in larger instances. VQE takes more iteration to find the ground state and even escapes the already-found global optima in the first two examples. The performance for the largest problem is equally bad for both algorithms since neither is able to find the ground state or a successful solution. Still, both are able to significantly reduce the mean energy but stagnate at around 30 iterations. Along with the peaking measurement probability, this indicates that both algorithms found a local minimum and are unable to find optimization steps that would let them escape.

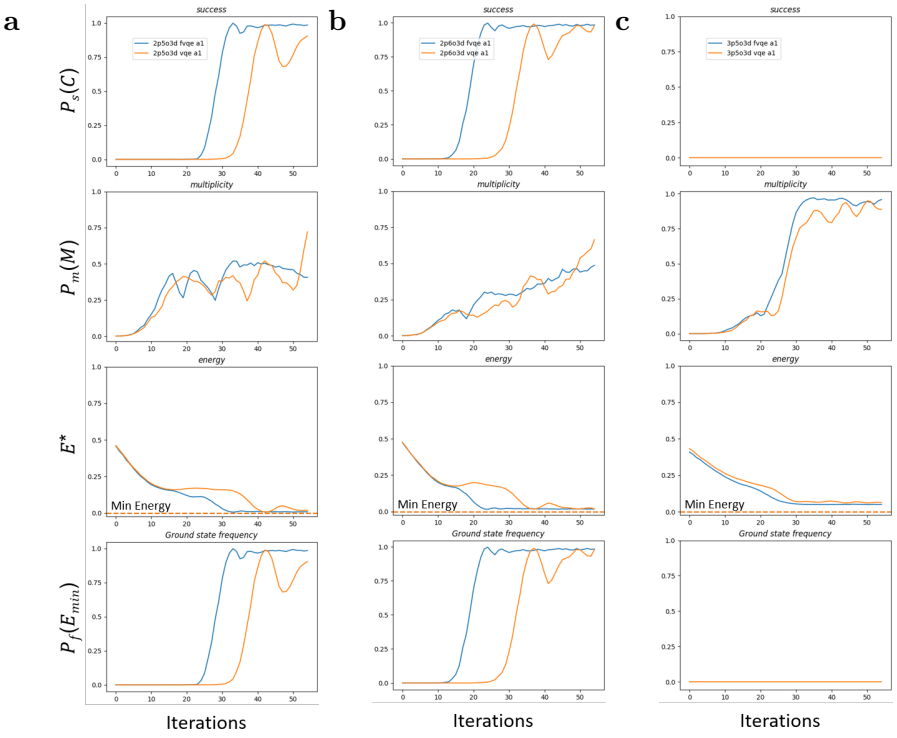


Figure 6.10: Performance comparison between the F-VQE and VQE algorithms. Both algorithms use ansatz (a_1) with one layer and 2400 samples per iteration. (a) Problem instance with 2 positions, 5 operators, and 3 days (b) Problem instance with 2 positions, 6 operators, and 3 days (c) Problem instance with 3 positions, 5 operators, and 3 days.

Finally, we will take a separate look at the Grover implementation since it employs no variational approach, currently does not yet consider goals, and singularly optimizes for the constraint set. This also means that the ground state frequency is no longer significant since every successful solution is simultaneously the ground state. The success rate and minimization of the average energy, however, can still be compared with the variational approaches. Grover's algorithm is typically only applied once with a given number of *Grover iterations*. These iterations differ in the sense that they are not continuous executions of the same circuit with varying parameters but are inherited parameters of the circuit design. For multiple Grover iterations, a gate block inside the circuit is repeatedly applied to achieve an amplification of the ground state measurement probability. Nevertheless, we will use

this parameter in the same way as we did for the variational approaches to illustrate the amplification process. In Figure 6.11, we show the overall performance of Grover's algorithm over an interval of 1–10 iterations.

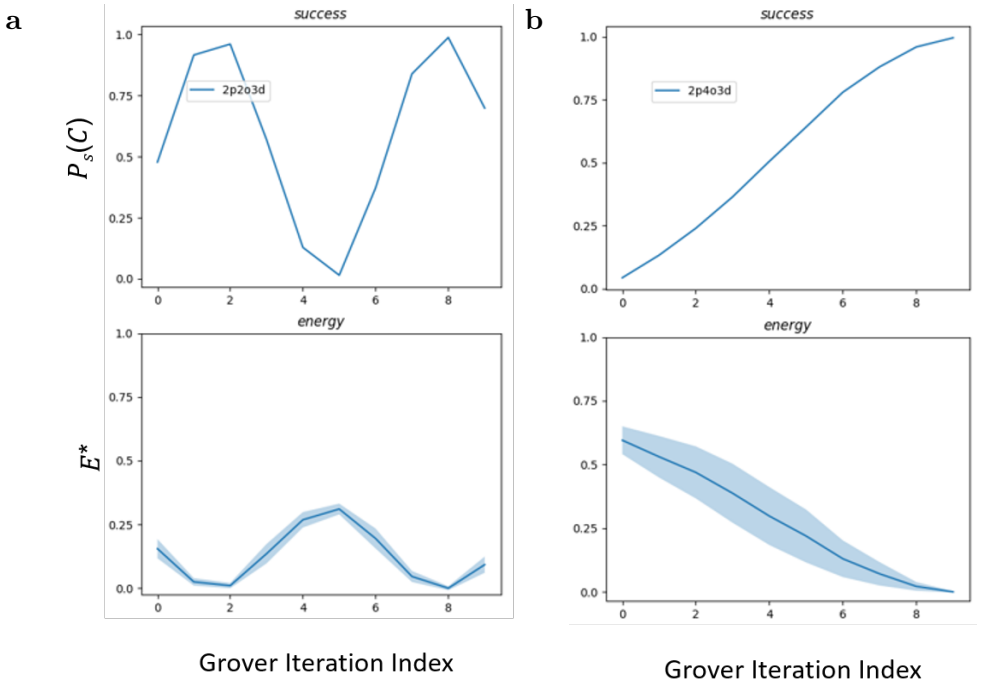


Figure 6.11: Performance of the Grover solver for different numbers of iterations. In contrast to the VQAs, each iteration is a separate full execution of the algorithm. (a) Problem instance with 2 positions, 2 operators, and 3 days (b) Problem instance with 2 positions, 4 operators, and 3 days.

For the smaller problem instance, we observe a peak in the success rate for a circuit with three and nine Grover iterations. This behavior is to be expected from a Grover implementation since the success rate periodically increases and decreases with the number of iterations used. The wavelength is determined by the ratio of successful solutions within the entire solution space. For the larger example, we only found the first peak at 10 iterations. For further iterations, one would expect a slow decline until the success rate reaches 0%. In contrast to the variational algorithms, Grover's algorithm finds multiple global minima, as can be seen by the percentage measurement amount. This is mostly due to the fact that no goals were considered, resulting in an overall greater number of global minima. In Table 6.5, the execution time, gate cost, circuit depth, and number of required qubits are shown for the respective best-performing iteration number. The gate cost is a measure to compare different circuit designs and get an idea of how the algorithm would perform on a true quantum computer. It is given by the gates in the longest path of the circuit:

$$cost = ||S*|| + 10 \cdot ||CNOT||$$

Where S* is the set of single qubit gates used in the longest path. The longest path in a circuit is given by the qubit, which contains the most gates. To compute this metric, one has to transpile the circuit into a basic gate set only containing a single qubit and CNOT gates.

The circuit depth is a similar metric, but it does not take the type of gate into account. It also shows the longest path for the untransposed circuit. Both metrics are important since long circuits with many multi-qubit gates pose a limit to current quantum hardware with low coherence times. To get an idea of the dimension of these values for the second problem instance, F-VQE and QAOA only require a gate cost of 23 and 1369 and a circuit depth of 6 and 7, respectively. The table also highlights an interesting observation: Grover's algorithm demands a larger number of qubits for solving the same problem instances compared to the variational approaches. For instance, in the case of the larger example, F-VQE encodes the problem with 12 qubits, while Grover's algorithm requires 20 qubits to address the same instance. This is due to the fact that Grover utilizes additional quantum registers to detect or count the number of constraint violations.

	Execution Time	Gate Cost	Circuit Depth	Qubits
2p2o3d	00:02	25003	92	12
2p4o3d	00:08	2382061	102	20

Table 6.5: Execution time, gate cost, circuit depth and number of qubits for Grover's algorithm.

6.3.3 Results on Quantum Hardware

For our test runs on real quantum hardware, we used the Qiskit backend. We had access to a seven- and 127-qubit computer. For the smaller quantum processor, we were limited to our smallest problem instance, which requires six qubits. For the larger one, we tried to test a more extensive example. Unfortunately, our allocated processing time ran out before we achieved noteworthy results. We will therefore only show the results of the smaller quantum circuit.

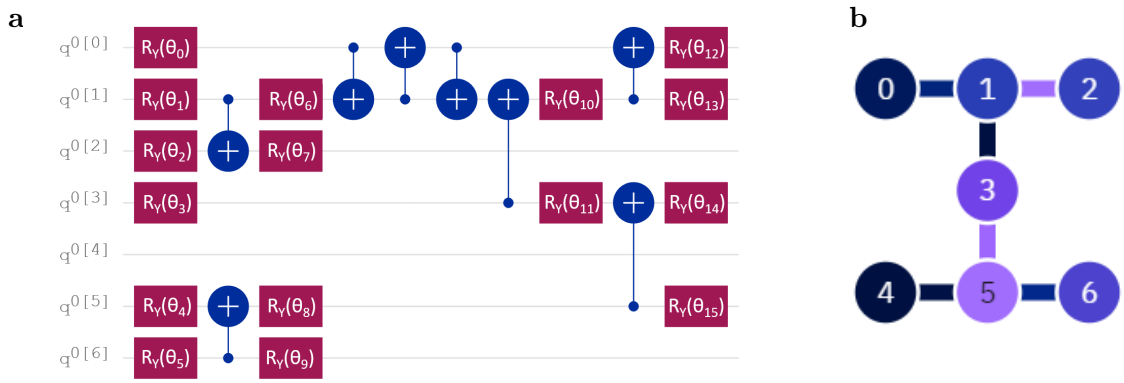


Figure 6.12: (a) Circuit qubit and gate mapping to quantum hardware. (b) Circuit Connectivity Map for the ibm_nairobi quantum processor: A seven-qubit layout was used for the six-qubit F-VQE problem instance. Each circle represents a physical qubit, and the lines illustrate their physical connectivity. The color gradient indicates the error rate for each connection and qubit, with darker colors signifying lower error rates.

Since IBM has to rebuild the circuit to fit on the actual hardware, all results included bit string encodings with seven instead of six bits. To get the true bit string, one can find the qubit that has no instructions in the circuit and remove the bit at the corresponding index. In Figure 6.12, the rebuild circuit and the quantum processor architecture can be seen.

The circuit now utilizes all seven qubits, but notably, qubit $q0[4]$ remains unused throughout. The arrangement of quantum gates is also tailored to the specific hardware layout. For example, in ansatz (a_1), a CNOT gate is placed between the second and third qubits. However, due to the hardware limitations, such a gate is not possible, so the circuit layout must be adjusted. This adaptation also helps in optimizing the routing of quantum gates to qubits with lower error rates. It's important to emphasize that the error rates (represented by the color gradient) indicate that qubit $q0[4]$ exhibits the lowest error. However, it's essential to clarify that this doesn't necessarily imply that qubit $q0[4]$ is always the least error-prone, as qubits undergo regular recalibrations. Additionally, it's worth noting that the two figures were not generated simultaneously, so it is likely that qubit $q0[4]$ may have had the highest error during the circuit mapping.

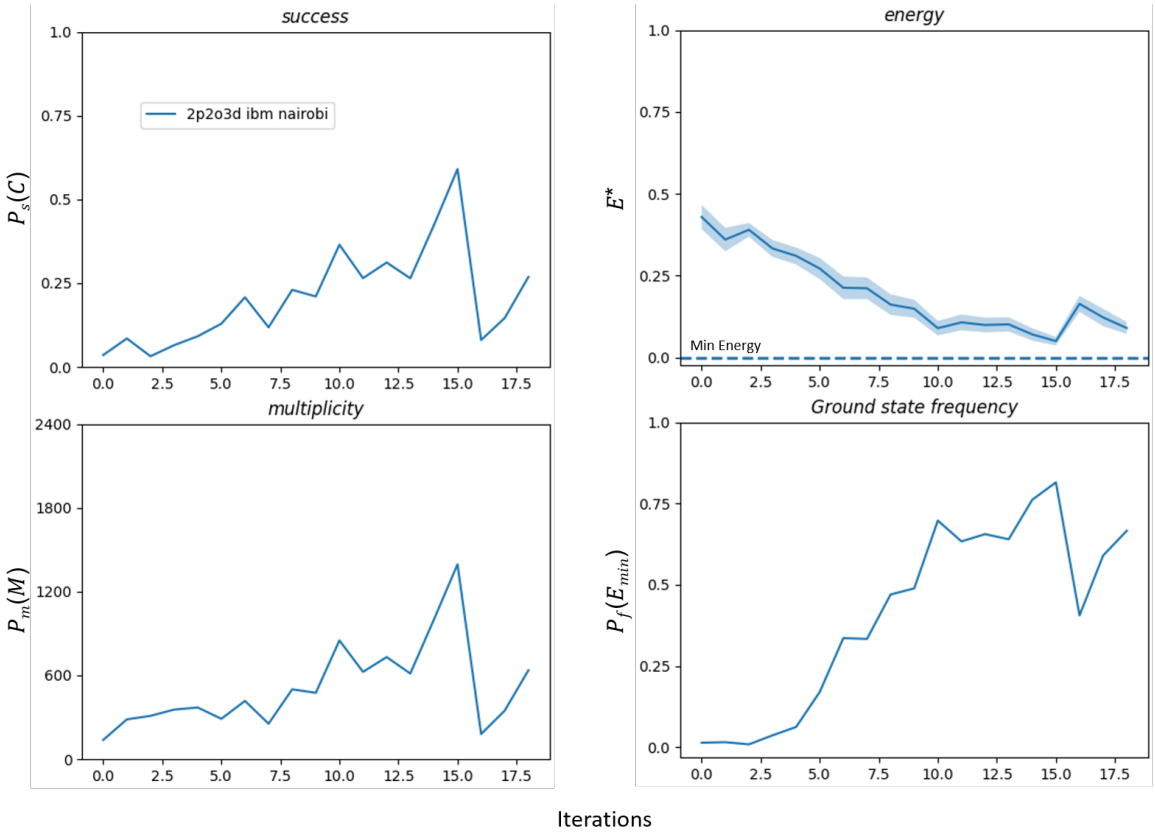


Figure 6.13: F-VQE $P_s(C)$, $P_m(M)$, E^*, and $P_f(E_{min})$ for the DLR OnCall scheduling problem instance using 6 qubits and 2400 shots on the IBM quantum processors ibm_nairobi.

As it was expected, Figure 6.12 illustrates a higher variance in both 'in energy' and 'mean energy' in the data from the quantum processor when compared to the simulator. Still, even with increased noise, the solver is able to find the ground state and achieves close to 80%

overlap after only 15 iterations. This is not significantly larger than the 10 iterations the F-VQE algorithm requires to reach its own peak at almost 100% overlap. Interestingly, the distribution for the success rate and multiplicity is almost identical throughout all iterations. This means the algorithm only finds a single valid solution at every iteration step (excluding iteration 2). Every other schedule found violates at least one constraint and is therefore invalid. We can observe the same phenomenon for the simulator run, but only for the same configuration. If one-hot encoding, a circuit (a_2), or another algorithm is used, this is not the case.

6.4 Discussion

We examined the convergence of all algorithms using a small 6-qubit instance of a scheduling problem with two positions, three days, and two operators. We then extended our investigation to include one-hot encoding, which expanded the problem to 12 qubits, enabling us to compare the performance of both encodings for F-VQE, VQE, and QAOA. It's important to note that the choice of encoding methods, such as binary encoding, can significantly impact the algorithm's capacity to tackle larger problem instances. For instance, F-VQE was capable of efficiently handling a problem involving 18 qubits, which would have been impractical using the one-hot encoding approach, as it would have required 36 qubits. Binary encoding also leads to a more amendable energy landscape since One-Hot encoding introduces a large number of inherent infeasible configurations, resulting in a lot more local optima.

We did not undertake comparisons with classical solvers since all problem instances can be solved exactly, making heuristic approaches redundant. Our primary objective was to analyze the convergence and scalability of F-VQE versus other quantum algorithms. In that regard, it is important to note that not only was F-VQE able to outperform QAOA, it was also able to implement all constraints since it was not forced to implement the required inequalities via slack variables. F-VQE also has faster convergence speeds and less noise when compared to VQE.

In our evaluation, we considered several key aspects for each algorithm, including the number of circuit layers, initial points, and classical optimizers. It is important to determine which combination of these factors works best for a given problem. We also found that for combinatorial problems, the expressibility of the circuit arrangement does not seem to play a big role when a baseline is met. Both ansatz circuits performed similarly, and further layers in either did not increase the performance. It is yet to be seen if better expressibility is needed for larger problem instances.

We also evaluated the performance of six different filtering operators. Interestingly, only the initially used "inverse" filter performed better than VQE. This is even more surprising if one considers the results obtained by Amaro et al., where both the "logarithm" and "exponential" filters performed similarly to the "inverse" filter. In our results, however, both filter operators perform the worst out of the five. Exploring the source of the disparities in the results would be of interest, especially considering that the parameters used should be consistent.

Additionally, we conducted a separate analysis of Grover's algorithm. Unlike VQAs, Grover's algorithm does not rely on variational approaches and primarily targets constraint satisfaction. We evaluated Grover's algorithm's performance in terms of success rates and execution times and observed how the number of iterations affects the algorithm's results.

For small problem instances, Grover's algorithm is able to achieve results comparable to F-VQE within a shorter time frame. However, Grover's algorithm necessitates the use of deeper circuits, prior knowledge of the number of iterations, or the application of heuristic methods. Moreover, it utilizes a larger number of qubits and requires further optimization to find solutions that also align with the given set of goals.

Our test on actual quantum hardware using a seven-qubit processor provided insights into the practical applicability of the F-VQE algorithm. Notably, the quantum hardware introduced additional noise into the results compared to the ideal simulator. Despite this noise, the solver achieved impressive results and found the ground state within a few iterations.

In light of the results, it is difficult to say if the F-VQE algorithm is applicable to scheduling problems. It certainly does perform the best out of all the examined approaches. It is, however, concerning to see the algorithm struggle with even small problem instances of 27 qubits. While this instance represents the most substantial case in our study, it pales in comparison to real-world applications involving approximately 50 operators, 20 distinct positions, and spanning over 180 days. In the broader context, the used examples remain relatively modest, even if they are of moderate size for quantum algorithms. This does not lend credibility to the scalability of this application, even if quantum computers increase in size, become less noisy, and find more constant error mitigation approaches. This is especially the case since the algorithm easily finds solutions for smaller instances but encounters challenges as the problem size grows. However, the algorithm demonstrates an ability to converge toward local minima, often in proximity to the global optima. It also might be able to escape these local minima and reach a good solution if the right techniques are applied. Overall, F-VQE exhibited superior performance among all the algorithms examined, highlighting a promising avenue for further advancements in variational quantum algorithms.

Our evaluation provides an introduction to the performance of quantum algorithms in solving scheduling problems. It is important to note that the selected problems in this work are small in comparison to real-world scheduling applications. These preliminary findings serve as a foundation for further exploration of the potential and challenges of quantum computing in addressing complex scheduling problems.

In our results, F-VQE showed the most promise among the algorithms in solving scheduling problems, particularly for smaller instances. Filtering operators yielded mixed results compared to previous research. Test cases for real quantum hardware introduced noise but still delivered promising results. However, scalability for larger instances remains a challenge.

7 Conclusion

In the analysis and evaluation of the F-VQE algorithm and the question of its applicability for real-life scheduling problems, we implemented several test cases and rival optimization algorithms. Our main objective was to evaluate the suitability of the F-VQE algorithm for job scheduling on the basis of a real problem at the German Space Operation Center, and through the comparison of different test cases and algorithms, we have found various answers to this question. We tested F-VQE over multiple test cases, with instances ranging up to 27 qubits. We also tested the algorithm on real quantum hardware, but since our access and computing time were limited, we were only able to use a small test case with six qubits.

While all algorithms displayed robust performance on smaller instances, they encountered challenges as the problem complexity increased. We observed that Grover's algorithm, though quicker in finding solutions, demanded a substantial increase in quantum resources and precision, making it less resource-efficient than variational algorithms. F-VQE consistently demonstrated the best performance across the test cases, making it the leading candidate for solving scheduling problems with quantum computing. The results underscore the potential for further improvements in variational quantum algorithms, especially when applied to scheduling problems.

As we showed in our results, one of the many challenges for VQA optimization is solving larger and more complex problem instances. Either because of noisier energy landscapes, more parameters, deeper circuits, or more qubits It will be crucial to improve the convergence of F-VQE and other VQAs, even for larger problem instances. Our experiments suggest that F-VQE performs favorably compared to classical VQE, QAOA, and Grover's algorithms. This could be a step in the right direction and possibly lead to even better results with improved approaches.

This is also the case since, just as VQE, F-VQE is hardware independent and can construct alternative circuits that fit the architecture. Even with uncertainty about which quantum hardware will prevail, the F-VQE can be applied, tested, studied, and improved. It is also not restricted by a rigid Hamiltonian formulation and can instead employ a black-box objective function. This compares very favorably with Hamiltonian solvers since it allows for greater flexibility in the objective function.

7.1 Future Work

In the pursuit of advancing variational quantum algorithms, especially the F-VQE for combinatorial scheduling problems, several promising directions for future research are open.

Firstly, exploring adaptive circuit designs tailored to the unique characteristics of specific scheduling problems or circuits that adapt according to the current optimization process [TSB+21] holds significant potential. Adaptive quantum circuits may enhance algorithm efficiency and solution quality and help combat barren plateaus. Mitigating the issue of barren plateaus in variational quantum algorithms is another essential focus for future work.

Implementing strategies to alleviate the challenges posed by barren plateaus, such as circuit designs less susceptible to this problem [HSCC22] or local objective functions [CSV+21], can significantly impact the algorithm's stability. Moreover, the investigation of the multi-basis encoding [PKAY22] method beyond binary and one-hot encoding offers a rich area for exploration. Utilizing this encoding scheme will allow us to explore the potential of F-VQE for even larger problem instances. Warm starting techniques [TBB+23], such as the relaxation of the underlying objective function, which involve initializing quantum algorithms closer to optimal solutions based on previous experiences, present an avenue for improving algorithm convergence. The incorporation of quantum gradient descent methods [SIKC20], which are tailored for quantum-aware optimization, is a growing field. Future research should investigate the integration of quantum gradient descent techniques into variational quantum algorithms to enhance their performance on large scheduling instances.

Additionally, a more comprehensive test would shed more light on the scalability of the F-VQE algorithm. Since we were only able to perform single tests for each examined method and approach, it would be interesting to see if our results are confirmed if tested over multiple executions. Further hyperparameter optimization for different ansatz circuits and filtering operators would potentially also allow us to find more optimal parameters. Finally, an extensive evaluation of the F-VQE algorithm on real quantum hardware is a promising next step considering the already good results in this work.

Bibliography

[AL99] Abrams, Daniel S. ; Lloyd, Seth: Quantum Algorithm Providing Exponential Speed Increase for Finding Eigenvalues and Eigenvectors. In: *Phys. Rev. Lett.* 83 (1999), Dec, 5162–5165. http://dx.doi.org/10.1103/PhysRevLett.83.5162. – DOI 10.1103/PhysRevLett.83.5162

[AMR+22] Amaro, David ; Modica, Carlo ; Rosenkranz, Matthias ; Fiorentini, Mattia ; Benedetti, Marcello ; Lubasch, Michael: Filtering variational quantum algorithms for combinatorial optimization. In: *Quantum Science and Technology* 7 (2022), jan, Nr. 1, 015021. http://dx.doi.org/10.1088/2058-9565/ac3e54. – DOI 10.1088/2058–9565/ac3e54

[Bas01] Basagni, Stefano: Finding a Maximal Weighted Independent Set in Wireless Networks. In: *Telecommun. Syst.* 18 (2001), sep, Nr. 1–3, 155–168. http://dx.doi.org/10.1023/A:1016747704458. – DOI 10.1023/A:1016747704458. – ISSN 1018–4864

[BBB+21] Bayerstadler, Andreas ; Becquin, Guillaume ; Binder, Julia ; Botter, Thierry ; Ehm, Hans ; Ehmer, Thomas ; Erdmann, Marvin ; Gaus, Norbert ; Harbach, Philipp ; Hess, Maximilian u. a.: Industry quantum computing applications. In: *EPJ Quantum Technology* 8 (2021), Nr. 1, S. 25

[BBHT98] Boyer, Michel ; Brassard, Gilles ; Høyer, Peter ; Tapp, Alain: Tight Bounds on Quantum Searching. In: *Fortschritte der Physik* 46 (1998), jun, Nr. 4-5, 493–505. http://dx.doi.org/10.1002/(sici)1521-3978(199806)46:4/5<493::aid-prop493>3.0.co;2-p. – DOI 10.1002/(sici)1521–3978(199806)46:4/5¡493::aid–prop493¿3.0.co;2–p

[BMWV+23] Bonet-Monroig, Xavier ; Wang, Hao ; Vermetten, Diederick ; Senjean, Bruno ; Moussa, Charles ; Bäck, Thomas ; Dunjko, Vedran ; O'Brien, Thomas E.: Performance comparison of optimization methods on variational quantum algorithms. In: *Physical Review A* 107 (2023), Nr. 3, S. 032407

[BVC21] Barison, Stefano ; Vicentini, Filippo ; Carleo, Giuseppe: An efficient quantum algorithm for the time evolution of parameterized circuits. In: *Quantum* 5 (2021), jul, 512. http://dx.doi.org/10.22331/q-2021-07-28-512. – DOI 10.22331/q–2021–07–28–512

[CAB+21] Cerezo, Marco ; Arrasmith, Andrew ; Babbush, Ryan ; Benjamin, Simon C. ; Endo, Suguru ; Fujii, Keisuke ; McClean, Jarrod R. ; Mitarai, Kosuke ; Yuan, Xiao ; Cincio, Lukasz u. a.: Variational quantum algorithms. In: *Nature Reviews Physics* 3 (2021), Nr. 9, S. 625–644

[CAS20] Cebi, Ceren ; Atac, Enes ; Sahingoz, Ozgur K.: Job Shop Scheduling Problem and Solution Algorithms: A Review. In: *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2020, S. 1–7

[CCGFR20] Cruz, Pedro M. ; Catarina, Gonçalo ; Gautier, Ronan ; Fernández-Rossier, Joaquín: Optimizing quantum phase estimation for the simulation of Hamiltonian eigenstates. In: *Quantum Science and Technology* 5 (2020), Nr. 4, S. 044005

[CD22] Campbell, Colin ; Dahl, Edward: QAOA of the Highest Order. In: *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*, 2022, S. 141–146

[Cen23] Centre, Leibniz S.: *Innovative platform for exploring the Earth: terrabyte*. `https://www.lrz.de/presse/ereignisse/2023-06-14-terrabyte-start-eng/`. Version: 2023

[Cha19] Chancellor, Nicholas: Domain wall encoding of discrete variables for quantum annealing and QAOA. In: *Quantum Science and Technology* 4 (2019), aug, Nr. 4, 045004. `http://dx.doi.org/10.1088/2058-9565/ab33c2`. – DOI 10.1088/2058–9565/ab33c2

[CLR+20] Churkin, Alexander ; Lewkiewicz, Stephanie ; Reinharz, Vladimir ; Dahari, Harel ; Barash, Danny: Efficient Methods for Parameter Estimation of Ordinary and Partial Differential Equation Models of Viral Hepatitis Kinetics. In: *Mathematics* 8 (2020), Nr. 9. `http://dx.doi.org/10.3390/math8091483`. – DOI 10.3390/math8091483. – ISSN 2227–7390

[CLR22] Chan–Renous-Legoubin, Rémi ; Royer, Clément W.: A nonlinear conjugate gradient method with complexity guarantees and its application to nonconvex regression. In: *EURO Journal on Computational Optimization* 10 (2022), 100044. `http://dx.doi.org/https://doi.org/10.1016/j.ejco.2022.100044`. – DOI https://doi.org/10.1016/j.ejco.2022.100044. – ISSN 2192–4406

[CMP18] Cubitt, Toby S. ; Montanaro, Ashley ; Piddock, Stephen: Universal quantum Hamiltonians. In: *Proceedings of the National Academy of Sciences* 115 (2018), Nr. 38, 9497-9502. `http://dx.doi.org/10.1073/pnas.1804949115`. – DOI 10.1073/pnas.1804949115

[COK20] Choi, Jaeho ; Oh, Seunghyeok ; Kim, Joongheon: Quantum approximation for wireless scheduling. In: *Applied Sciences* 10 (2020), Nr. 20, S. 7116

[Com09] Commons, Wikimedia: *Bloch sphere*. `https://commons.wikimedia.org/wiki/File:Bloch_sphere.svg`. Version: 2009

[Cro18] Cross, Andrew: The IBM Q experience and QISKit open-source quantum computing software. In: *APS March meeting abstracts* Bd. 2018, 2018, S. L58–003

[Cro19] Crooks, Gavin E.: *Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition.* 2019

[CSV+21] Cerezo, M. ; Sone, Akira ; Volkoff, Tyler ; Cincio, Lukasz ; Coles, Patrick J.: Cost function dependent barren plateaus in shallow parametrized quantum circuits. In: *Nature Communications* 12 (2021), mar, Nr. 1. http://dx.doi.org/10.1038/s41467-021-21728-w. – DOI 10.1038/s41467–021–21728–w

[DBK+22] Daley, Andrew J. ; Bloch, Immanuel ; Kokail, Christian ; Flannigan, Stuart ; Pearson, Natalie ; Troyer, Matthias ; Zoller, Peter: Practical quantum advantage in quantum simulation. In: *Nature* 607 (2022), Jul, Nr. 7920, 667-676. http://dx.doi.org/10.1038/s41586-022-04940-6. – DOI 10.1038/s41586–022–04940–6. – ISSN 1476–4687

[DGS18] Dunning, Iain ; Gupta, Swati ; Silberholz, John: What works best when? A systematic evaluation of heuristics for Max-Cut and QUBO. In: *INFORMS Journal on Computing* 30 (2018), Nr. 3, S. 608–624

[FGG14] Farhi, Edward ; Goldstone, Jeffrey ; Gutmann, Sam: *A Quantum Approximate Optimization Algorithm.* 2014

[FT11] Falco, Diego de ; Tamascelli, Dario: An introduction to quantum annealing. In: *RAIRO - Theoretical Informatics and Applications* 45 (2011), jan, Nr. 1, 99–116. http://dx.doi.org/10.1051/ita/2011013. – DOI 10.1051/ita/2011013

[GBB+23] Grimsley, Harper R. ; Barron, George S. ; Barnes, Edwin ; Economou, Sophia E. ; Mayhall, Nicholas J.: Adaptive, problem-tailored variational quantum eigensolver mitigates rough parameter landscapes and barren plateaus. In: *npj Quantum Information* 9 (2023), Nr. 1, S. 19

[Gib19] Gibney, Elizabeth: Hello Quantum World! google publishes landmark quantum supremacy claim. In: *Nature* 574 (2019), Nr. 7779, S. 461–462. http://dx.doi.org/10.1038/d41586-019-03213-z. – DOI 10.1038/d41586–019–03213–z

[GLRS16] Grassl, Markus ; Langenberg, Brandon ; Roetteler, Martin ; Steinwandt, Rainer: Applying Grover's Algorithm to AES: Quantum Resource Estimates. In: Takagi, Tsuyoshi (Hrsg.): *Post-Quantum Cryptography.* Cham : Springer International Publishing, 2016. – ISBN 978–3–319–29360–8, S. 29–43

[Gro96] Grover, Lov K.: A fast quantum mechanical algorithm for database search. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, S. 212–219

[GTD19] Goto, Hayato ; Tatsumura, Kosuke ; Dixon, Alexander R.: Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems. In: *Science Advances* 5 (2019), Nr. 4, eaav2372. http://dx.doi.org/10.1126/sciadv.aav2372. – DOI 10.1126/sciadv.aav2372

[GWG21] Gilliam, Austin ; Woerner, Stefan ; Gonciulea, Constantin: Grover Adaptive Search for Constrained Polynomial Binary Optimization. In: *Quantum* 5 (2021), apr, 428. http://dx.doi.org/10.22331/q-2021-04-08-428. – DOI 10.22331/q–2021–04–08–428

[HK21] Haug, Tobias ; Kim, M. S.: *Optimal training of variational quantum algorithms without barren plateaus.* 2021

[Hom22] Homeister, Matthias: *Quantum Computing verstehen.* Springer Vieweg, Wiesbaden, 2022. – 1–7, 20–21 S. https://link.springer.com/book/10.1007/978-3-658-36434-2. – ISBN 978–3–658–36433–5

[HSCC22] Holmes, Zoë ; Sharma, Kunal ; Cerezo, M. ; Coles, Patrick J.: Connecting Ansatz Expressibility to Gradient Magnitudes and Barren Plateaus. In: *PRX Quantum* 3 (2022), Jan, 010313. http://dx.doi.org/10.1103/PRXQuantum.3.010313. – DOI 10.1103/PRXQuantum.3.010313

[IBM23] IBM: *IBM Quantum Platform: Compute resources.* https://quantum-computing.ibm.com/services/resources. Version: 2023

[JEM+19a] Jones, Tyson ; Endo, Suguru ; McArdle, Sam ; Yuan, Xiao ; Benjamin, Simon C.: Variational quantum algorithms for discovering Hamiltonian spectra. In: *Physical Review A* 99 (2019), Nr. 6, S. 062304

[JEM+19b] Jones, Tyson ; Endo, Suguru ; McArdle, Sam ; Yuan, Xiao ; Benjamin, Simon C.: Variational quantum algorithms for discovering Hamiltonian spectra. In: *Phys. Rev. A* 99 (2019), Jun, 062304. http://dx.doi.org/10.1103/PhysRevA.99.062304. – DOI 10.1103/PhysRevA.99.062304

[KCM16] Kim, Joongheon ; Caire, Giuseppe ; Molisch, Andreas F.: Quality-Aware Streaming and Scheduling for Device-to-Device Video Delivery. In: *IEEE/ACM Transactions on Networking* 24 (2016), Nr. 4, S. 2319–2331. http://dx.doi.org/10.1109/TNET.2015.2452272. – DOI 10.1109/TNET.2015.2452272

[Ket17] *Kapitel* 8. In: Ketkar, Nikhil: *Stochastic Gradient Descent.* Berkeley, CA : Apress, 2017. – ISBN 978–1–4842–2766–4, 113–132

[Kit95] Kitaev, A Y.: Quantum measurements and the Abelian stabilizer problem. In: *arXiv preprint quant-ph/9511026* (1995)

[KWPO+11] Kassal, Ivan ; Whitfield, James D. ; Perdomo-Ortiz, Alejandro ; Yung, Man-Hong ; Aspuru-Guzik, Alá n: Simulating Chemistry Using Quantum Computers. In: *Annual Review of Physical Chemistry* 62 (2011), may, Nr. 1, 185–207. http://dx.doi.org/10.1146/annurev-physchem-032210-103512. – DOI 10.1146/annurev–physchem–032210–103512

[Lan09] *Kapitel* 20. In: Landsman, Nicolaas P.: *Born Rule and its Interpretation.* Berlin, Heidelberg : Springer Berlin Heidelberg, 2009. – ISBN 978–3–540–70626–7, 64–70

[LAS+22] Liu, Xiaoyuan ; Angone, Anthony ; Shaydulin, Ruslan ; Safro, Ilya ; Alexeev, Yuri ; Cincio, Lukasz: Layer VQE: A variational approach for combinatorial optimization on noisy quantum computers. In: *IEEE Transactions on Quantum Engineering* 3 (2022), S. 1–20

[LDG+21] Lin, Sheng-Hsuan ; Dilip, Rohit ; Green, Andrew G. ; Smith, Adam ; Pollmann, Frank: Real- and Imaginary-Time Evolution with Compressed Quantum Circuits. In: *PRX Quantum* 2 (2021), Mar, 010342. `http://dx.doi.org/10.1103/PRXQuantum.2.010342`. – DOI 10.1103/PRXQuantum.2.010342

[Led01] *Kapitel* 1. In: Ledoux, Michel: *The concentration of measure phenomenon*. American Mathematical Soc., 2001 (Mathematical Surveys and Monographs 89)

[LEF+22] Lindauer, Marius ; Eggensperger, Katharina ; Feurer, Matthias ; Biedenkapp, André ; Deng, Difan ; Benjamins, Carolin ; Ruhkopf, Tim ; Sass, René ; Hutter, Frank: SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization. In: *J. Mach. Learn. Res.* 23 (2022), jan, Nr. 1. – ISSN 1532–4435

[LHDC09] Lagatie, Ruben ; Haspeslagh, Stefaan ; De Causmaecker, Patrick: Negotiation protocols for distributed nurse rostering. In: *Proceedings of the 21st Benelux Conference on Artificial Intelligence*, 2009, S. 145–152

[Llo96] Lloyd, Seth: Universal Quantum Simulators. In: *Science* 273 (1996), Nr. 5278, 1073-1078. `http://dx.doi.org/10.1126/science.273.5278.1073`. – DOI 10.1126/science.273.5278.1073

[LMB+20] Luengo, David ; Martino, Luca ; Bugallo, Mónica ; Elvira, Víctor ; Särkkä, Simo: A survey of Monte Carlo methods for parameter estimation. In: *EURASIP Journal on Advances in Signal Processing* 2020 (2020), Nr. 1, S. 1–62

[LWM+12] Lenzen, Christoph ; Wörle, Maria T. ; Mrowka, Falk ; Spörl, Andreas ; Klaehn, Rüdiger: The Algorithm Assembly Set of Plato. In: *12th International Conference on Space Operations (SpaceOps 2012)*, 2012. – ePoster

[LY16a] *Kapitel* 1. Introduction. In: Luenberger, David G. ; Ye, Yinyu: *Linear and Nonlinear Programming*. Cham : Springer International Publishing, 2016. – ISBN 978–3–319–18842–3, 321–355

[LY16b] *Kapitel* 11. Constrained Minimization Conditions. In: Luenberger, David G. ; Ye, Yinyu: *Linear and Nonlinear Programming*. Cham : Springer International Publishing, 2016. – ISBN 978–3–319–18842–3, 321–355

[Mai20] Mainzer, Klaus: *Quantencomputer*. Springer Berlin, Heidelberg, 2020. – 1–5 S. `https://link.springer.com/book/10.1007/978-3-662-61998-8`. – ISBN 978–3–662–61997–1

[Man60] Manne, Alan S.: On the job-shop scheduling problem. In: *Operations research* 8 (1960), Nr. 2, S. 219–223

[MF19] Mitarai, Kosuke ; Fujii, Keisuke: Methodology for replacing indirect measurements with direct measurements. In: *Phys. Rev. Res.* 1 (2019), Aug, 013006. http://dx.doi.org/10.1103/PhysRevResearch.1.013006. – DOI 10.1103/PhysRevResearch.1.013006

[MRBAG16] McClean, Jarrod R. ; Romero, Jonathan ; Babbush, Ryan ; Aspuru-Guzik, Alán: The theory of variational hybrid quantum-classical algorithms. In: *New Journal of Physics* 18 (2016), feb, Nr. 2, 023023. http://dx.doi.org/10.1088/1367-2630/18/2/023023. – DOI 10.1088/1367–2630/18/2/023023

[NBH20] Nakerst, Goran ; Brennan, John ; Haque, Masudul: *Gradient descent with momentum — to accelerate or to super-accelerate?* 2020

[PKAY22] Patti, Taylor L. ; Kossaifi, Jean ; Anandkumar, Anima ; Yelin, Susanne F.: Variational quantum optimization with multibasis encodings. In: *Phys. Rev. Res.* 4 (2022), Aug, 033142. http://dx.doi.org/10.1103/PhysRevResearch.4.033142. – DOI 10.1103/PhysRevResearch.4.033142

[PMS+14] Peruzzo, Alberto ; McClean, Jarrod ; Shadbolt, Peter ; Yung, Man-Hong ; Zhou, Xiao-Qi ; Love, Peter J. ; Aspuru-Guzik, Alán ; O'Brien, Jeremy L.: A variational eigenvalue solver on a photonic quantum processor. In: *Nature Communications* 5 (2014), jul, Nr. 1. http://dx.doi.org/10.1038/ncomms5213. – DOI 10.1038/ncomms5213

[Pow64] Powell, M. J. D.: An efficient method for finding the minimum of a function of several variables without calculating derivatives. In: *The Computer Journal* 7 (1964), 01, Nr. 2, 155-162. http://dx.doi.org/10.1093/comjnl/7.2.155. – DOI 10.1093/comjnl/7.2.155. – ISSN 0010–4620

[Pow08] Powell, James: The Quantum Limit to Moore's Law. In: *Proceedings of the IEEE* 96 (2008), 09, S. 1247 – 1248. http://dx.doi.org/10.1109/JPROC.2008.925411. – DOI 10.1109/JPROC.2008.925411

[PSPS23] Prüfer, Sven ; Sajko, Wanja ; Pomplun, Nikolas ; Spörl, Andreas: Evolving Spacecraft Quantum On-Call Scheduling. In: *17th International Conference on Space Operations (SpaceOps 2023)*, 2023

[PSR21] Plewa, Julia ; Sieńko, Joanna ; Rycerz, Katarzyna: Variational Algorithms for Workflow Scheduling Problem in Gate-Based Quantum Devices. In: *COMPUTING AND INFORMATICS* 40 (2021), Dec., Nr. 4, 897–929. http://dx.doi.org/10.31577/cai_2021_4_897. – DOI 10.31577/cai_2021_4_897

[PSS+21] Prüfer, Sven ; Scherer, Antonius ; Spörl, Andreas ; Guggemos, Tobias ; Pomplun, Nikolas ; Lenzen, Christoph: Quantum Shift Scheduling - A Comparison to Classical Approaches. In: Chien, Steve (Hrsg.): *12th International Workshop on Planning and Scheduling for Space (IWPSS 2021)*, 2021

[RSR+17] Ristè, Diego ; Silva, Marcus P. ; Ryan, Colm A. ; Cross, Andrew W. ; Córcoles, Antonio D. ; Smolin, John A. ; Gambetta, Jay M. ; Chow, Jerry M. ; Johnson, Blake R.: Demonstration of quantum advantage in machine learning. In: *npj Quantum Information* 3 (2017), Apr, Nr. 1, 16. http://dx.doi.org/10.1038/s41534-017-0017-3. – DOI 10.1038/s41534–017–0017–3. – ISSN 2056–6387

[Rud17] Ruder, Sebastian: *An overview of gradient descent optimization algorithms.* 2017

[SA19] Shaydulin, Ruslan ; Alexeev, Yuri: Evaluating Quantum Approximate Optimization Algorithm: A Case Study. In: *2019 Tenth International Green and Sustainable Computing Conference (IGSC)*, IEEE, oct 2019, 1.6

[SGGC+21] Scherer, Antonius ; Guggemos, Tobias ; Grundner-Culemann, Sophia ; Pomplun, Nikolas ; Prüfer, Sven ; Spörl, Andreas: OnCall Operator Scheduling for Satellites with Grover's Algorithm. In: *International Conference on Computational Science* Springer, 2021, S. 17–29

[Sho94] Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, S. 124–134

[SIKC20] Stokes, James ; Izaac, Josh ; Killoran, Nathan ; Carleo, Giuseppe: Quantum Natural Gradient. In: *Quantum* 4 (2020), Mai, 269. http://dx.doi.org/10.22331/q-2020-05-25-269. – DOI 10.22331/q–2020–05–25–269. – ISSN 2521–327X

[SIR+21] Saha, Debadrita ; Iyengar, Srinivasan S. ; Richerme, Philip ; Smith, Jeremy M. ; Sabry, Amr: Mapping quantum chemical dynamics problems to spin-lattice simulators. In: *Journal of Chemical Theory and Computation* 17 (2021), Nr. 11, S. 6713–6732

[SJAG19] Sim, Sukin ; Johnson, Peter D. ; Aspuru-Guzik, Alá n: Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms. In: *Advanced Quantum Technologies* 2 (2019), oct, Nr. 12. http://dx.doi.org/10.1002/qute.201900070. – DOI 10.1002/qute.201900070

[SYRY21] Suzuki, Yudai ; Yano, Hiroshi ; Raymond, Rudy ; Yamamoto, Naoki: Normalized Gradient Descent for Variational Quantum Algorithms. In: *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2021, S. 1–9

[TBB+23] Truger, Felix ; Barzen, Johanna ; Bechtold, Marvin ; Beisel, Martin ; Leymann, Frank ; Mandl, Alexander ; Yussupov, Vladimir: *Warm-Starting and Quantum Computing: A Systematic Mapping Study.* 2023

[TCC+22] Tilly, Jules ; Chen, Hongxiang ; Cao, Shuxiang ; Picozzi, Dario ; Setia, Kanav ; Li, Ying ; Grant, Edward ; Wossnig, Leonard ; Rungger, Ivan

; Booth, George H. ; Tennyson, Jonathan: The Variational Quantum Eigensolver: A review of methods and best practices. In: *Physics Reports* 986 (2022), 1-128. http://dx.doi.org/https://doi.org/10.1016/j.physrep.2022.08.003. – DOI https://doi.org/10.1016/j.physrep.2022.08.003. – ISSN 0370–1573. – The Variational Quantum Eigensolver: a review of methods and best practices

[TPMR20] Tomasiewicz, Dawid ; Pawlik, Maciej ; Malawski, Maciej ; Rycerz, Katarzyna: Foundations for Workflow Application Scheduling on D-Wave System. In: Krzhizhanovskaya, Valeria V. (Hrsg.) ; Závodszky, Gábor (Hrsg.) ; Lees, Michael H. (Hrsg.) ; Dongarra, Jack J. (Hrsg.) ; Sloot, Peter M. A. (Hrsg.) ; Brissos, Sérgio (Hrsg.) ; Teixeira, João (Hrsg.): *Computational Science – ICCS 2020*. Cham : Springer International Publishing, 2020. – ISBN 978–3–030–50433–5, S. 516–530

[Tre11] Trevisan, Luca: Combinatorial optimization: exact and approximate algorithms. In: *Standford University* (2011)

[TSB+21] Tang, Ho L. ; Shkolnikov, V.O. ; Barron, George S. ; Grimsley, Harper R. ; Mayhall, Nicholas J. ; Barnes, Edwin ; Economou, Sophia E.: Qubit-ADAPT-VQE: An Adaptive Algorithm for Constructing Hardware-Efficient Ansätze on a Quantum Processor. In: *PRX Quantum* 2 (2021), Apr, 020310. http://dx.doi.org/10.1103/PRXQuantum.2.020310. – DOI 10.1103/PRXQuantum.2.020310

[WHJR18] Wang, Zhihui ; Hadfield, Stuart ; Jiang, Zhang ; Rieffel, Eleanor G.: Quantum approximate optimization algorithm for MaxCut: A fermionic view. In: *Physical Review A* 97 (2018), feb, Nr. 2. http://dx.doi.org/10.1103/physreva.97.022304. – DOI 10.1103/physreva.97.022304

[Woe03] *Kapitel* 17. In: Woeginger, Gerhard J.: *Exact Algorithms for NP-Hard Problems: A Survey*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2003. – ISBN 978–3–540–36478–8, 185–207

[ZDZ+19] Zhang, Jian ; Ding, Guofu ; Zou, Yisheng ; Qin, Shengfeng ; Fu, Jianlin: Review of job shop scheduling research and its new perspectives under Industry 4.0. In: *Journal of Intelligent Manufacturing* 30 (2019), S. 1809–1830

[ZMS+23] Zoufal, Christa ; Mishmash, Ryan V. ; Sharma, Nitin ; Kumar, Niraj ; Sheshadri, Aashish ; Deshmukh, Amol ; Ibrahim, Noelle ; Gacon, Julien ; Woerner, Stefan: Variational quantum algorithm for unconstrained black box binary optimization: Application to feature selection. In: *Quantum* 7 (2023), jan, 909. http://dx.doi.org/10.22331/q-2023-01-26-909. – DOI 10.22331/q–2023–01–26–909