The QuaST Best Practice Guide - A Comprehensive Summary of Research Results

Provided By the QuaST Consortium: quast-quantencomputing.de

February 24, 2025

Contents

1 Purpose and Introduction

Despite numerous advances in bringing quantum computing (QC) closer to the application, exploring the technology from an application perspective remains difficult [1]. From 2022 to 2024, the research consortium "Quantum-enabling services and tools for industrial applications" (QuaST) gathered seven partners from science and industry to tackle this problem within the problem domain of combinatorial optimization. Throughout the three project years, several cornerstone application cases have been investigated with numerous methods from the literature and within the consortium. This document gathers the enormous expertise gained within QuaST and with the joint effort of over 30 researchers and experts from industry and contains key results of over 20 publications.

Whether any quantum advantage can be found for practically relevant optimization problems remains an open question, even more so with the current noisy quantum hardware. However, new algorithms emerge constantly, existing ones are improved, and hardware noise decreases, enabling more and more experiments to aid the quest for quantum advantage and quantum utility. Still, the authors of this guide stress the following point clearly: There is no simple recipe for obtaining a quantum advantage for an application case from this guide, the QuaST project, or (to our knowledge) in general. Similarly, there are too many open research questions to definitively answer which algorithmic settings will yield the best solution to a practical problem. Instead, this guide presents a comprehensive review of the lessons the QuaST consortium learned from its studies of several industrially relevant use cases. In parallel, the project developed the QuaST decision tree, a prototypical framework aiming to enable end users to apply quantum-assisted algorithms to their optimization problems. Together, users can learn how to apply quantum algorithms to their problems and what topics one needs to investigate to improve and tune the algorithms and the solutions they provide.

Put concisely, an industrial optimization problem needs to run through several layers so a quantum-enhanced algorithm can attempt a solution [2]. First, the high-level business description needs to be cast into a mathematically precise problem *formulation* such as the instance of a common optimization problem class like the Traveling Salesperson Problem (TSP). Depending on the size of the optimization problem and its characteristics, a *decomposition* step follows to break it down into smaller parts, some of which (but usually not all) should be handled by a quantum computer. Then, the optimization problem needs to be transformed into a form compatible with a quantum algorithm, often an Ising problem. This step is called *encoding*. Afterward, the hybrid *algorithm* needs to be selected, e.g., a variational quantum algorithm like Quantum Approximate Optimization (QAOA) [3] along with its hyperparameters such as the exact quantum circuit to execute. Analogous choices need to be made for the *classical algorithms* that are involved, such as an optimizer for the parameter updates of a variational algorithm. Finally, *compilation and backend options* need to be weighed against each other to identify the best available quantum hardware backend and enable the execution of quantum circuits. All of these steps require carefully choosing between multiple available options that influence each other, even between the layers, severely complicating the decision-making process. The underlying decision tree structure gives rise to the QuaST decision tree, an important

project to automate and facilitate these decisions for end users.

This compendium complements the QuaST decision tree in gathering and discussing the application-centric knowledge gained inside the QuaST project. First, the executive summary in section 2 lists and explains the key findings of the QuaST project. The specific applications investigated within the QuaST project are presented in section 3. Even though they remain concise, the remaining sections dive into more technical detail and can serve as an entry point into the scientific publications the QuaST researchers produced. The application-centric insights along the layers and topics investigated within the QuaST project are divided into four chapters: section 4 for the formulation and encoding, section 5 for the algorithm selection, section 6 for aspects of decomposition and problem transformations and section 7 for hardware aspects. Our goal is to provide insights into each of these fields that can serve as a starting point for exploring quantum-assisted solutions for other use cases enabling further research and development in industrially applied quantum computing. Ultimately, quantum computing can only be successful if it comes with a real-life benefit obtained through focused research on the applications.

2 Executive Summary

A quantum advantage for general NP-hard combinatorial optimization problems is unlikely. Therefore, QuaST brings together partners from science and industry to investigate *specialized* quantum-enhanced solutions tailored to *specific* industrially relevant use cases. From investigating several applications from logistics to energy networks and software verification, the project extracted six key findings. These provide a general direction in which research around quantum-assisted algorithms for combinatorial optimization should be pursued.

- As of today, there is no quantum advantage (exponential, polynomial, or pre-factor) for general industrially relevant optimization problems. Promising avenues include adapting algorithms to specific problems. This requires testing and benchmarking against specific classical algorithms. Software tools that integrate the different parts of the solution process are necessary to aid the search for some quantum usefulness. They enable intermediate-scale testing and benchmarking and finding solution paths that can inspire fully productive solvers.

- Out-of-the-box solutions with variational quantum algorithms (VQA) need problem-specific adaptions to produce useful results.

- Modeling industrial problem settings requires much effort and introduces manual overhead. Within QuaST, we have expanded the range of use cases to translate into QC-enabling models. The QuaST decision tree is a tool specifically designed to reduce the expertise and manual work in handling industrial optimization problems.

- For quantum computing in the noisy intermediate-scale quantum era, handling complex constraints is challenging. Effective and efficient encodings for quantum-

enhanced algorithms with a good ratio of feasible solutions within the search space are a key ingredient to successful quantum-assisted algorithms.

- Unspecialized or metaheuristic classical optimization methods have troubles finding good parameters in QC loss landscapes. We must develop strategies for finding good parameters for parameterized quantum circuits (fixed angles, good initialization, tweaked optimizers). Some of these strategies have been advanced within the QuaST project.

- VQAs produce bitstrings that exhibit high volatility. The correlations extracted from quantum algorithms are more stable than the bitstrings obtained directly from sampled VQA outputs, showing promising results (e.g., in algorithms like Relax-and-Round and QUBO shrinking) and are a possible path forward in the search for beneficial use of quantum information.

It is part of a realistic view on combinatorial optimization to emphasize how difficult it is to find good solutions for optimization problems with quantum-enhanced methods. Every step of the solution process needs to be highly optimized in dependence of each other, including classical parts and the quantum-classical interfaces and transfer points. Choosing an algorithm and improving its performance with problem-specific adaptions are crucial steps in successfully applying quantum computing to industrial use cases. With the vast array of algorithms available, this can be a daunting task requiring extensive expert knowledge in different fields such as physics, mathematics and computer science. Even with the right algorithm, its performance is significantly impacted by the choice of hyperparameters, optimization methods, and penalty factors. Thus, a holistic view on the entire solution pipeline is encouraged.

To navigate these complexities and allow for a systematic exploration and algorithm setup, the QuaST project partners have developed the QuaST decision tree. This software framework guides users through the process of selecting and improving an algorithm. and provides a structured approach to answering key questions about the problem, algorithm, and optimization strategy, ultimately leading to suitable choices for specific use cases.

2.1 The QuaST Decision Tree

The QuaST decision tree [4] gathers, simplifies, organizes and automates the many decisions on the path from the application to a quantum-enhanced solution. The tree-like modular structure can be use interactively with the user providing necessary input, making the choices freely or accepting the recommendations fed to the decision tree by configurable subroutines (e.g., penalty selection). The flexible and modular framework can take over any computational task necessary for the solution of optimization problems. Different modules can be incorporated as "recommendation engines" and may include even commercial software. Thus, the QuaST decision tree can provide interfaces to any application-facing optimization routine.

From the beginnings of the QuaST project, the consortium has worked towards establishing the needs of end users and researchers. The first concept of the QuaST decision

tree framework [2] indeed dates from the first half of the project. With further research, the initial layout has been tested and revised for stability, flexibility and robustness. The ultimate project result is a unified, light-weight software framework that will undergo further development beyond the QuaST project. It can cover all necessary functions from the application to the invocation of the hardware, with its basic nodes including:

- Problem Loading: Load a problem from a simple JSON file or generate a random instance of predefined problem classes.
- Encoding: Encode the problem directly or via an intermediate encoding-independent step into a QUBO formulation. For constrained problems, this includes penalty factor selection.
- Algorithm Selection: Select a variational quantum algorithm, or generate a comparison benchmark with basic classical methods such as Tabu sampling or even a brute-force approach.
- Hyperparameter Selection: Tune the algorithm by choosing the necessary hyperparameters such as the depth and type of the ansatz, mixer Hamiltonian etc.
- Backend Selection: Choose a simulator backend or, depending on your access, a real quantum computer to send the problem to.

The decision tree provides recommendations in the form of default options for its choices and provides the necessary framework for automating the algorithm building process at various levels. It is set up specifically to allow for any community member to add their own recommendation routines, methods etc. With its modular approach, all information pertaining to the problem instance can be easily modified, pre- and postprocessed. The software is in an alpha stage with an ever-growing set of features like batch handling, decomposition steps and the inclusion of specialized methods from within the QuaST project. Further developments include the integration with various software stacks such as the Munich Quantum Portal allowing for a combined access of quantum computing and conventional high-performance computing resources.

3 Applications

Quantum computing has the potential to greatly impact various application areas in the realm of combinatorial optimization problems. These problems are characterized by a large search space, making it computationally challenging for classical computers to find optimal, and in some cases even good, solutions. Quantum computers can utilize phenomena such as superposition and entanglement, which opens up new possibilities for solving these complex problems more efficiently.

The potential business impact of quantum computing in these application areas is substantial. Efficient solutions to combinatorial optimization problems can translate into significant cost savings, increased productivity, and improved operational efficiency. For important problem classes such as route planning, even small relative increases of 1-2% would have an enormous economic impact justifying the high development costs of quantum computing.

In combinatorial optimization, the expected advantage from quantum computing can manifest from two directions:

- The more efficient search through large solution spaces with the aid of algorithms inspired by Grover search [5].
- The analogy between the tasks of finding the ground state of a physical system (which minimizes energy) and the minimization of the cost function of an optimization problem.

This user guide provides a comprehensive resource for identifying potential application areas and formulating problems with a focus on quantum algorithms like QAOA that aim at an improvement by encoding the problem solution into the ground state of a physdical system. With a focus on combinatorial problems and application areas such as network optimization, route planning, scheduling, and predicting market potential, this guide aims to help identify viable use cases.

In the following, we briefly introduce six use cases explored in QuaST.

3.1 Network Optimization

Many important real-world challenges can be formulated as network problems, where the goal is to optimize the flow, allocation, or configuration of resources within a complex system. Two notable examples of such problems are the unit commitment (UC) problem in power systems and the factory layout (FL) problem in manufacturing.

In the context of power systems, the increasing integration of renewable energy sources, driven by decarbonization efforts, has led to significant challenges in the UC problem. The UC problem involves determining the optimal scheduling of power plants to meet the anticipated demand while minimizing costs and ensuring system reliability. However, the inherent variability and uncertainty associated with renewable power generation, such as wind and solar, make it more difficult to predict the load to be served by conventional

power plants. In practice, mixed-integer programming (MIP) methods are used to solve the UC problem.

Similarly, the FL problem, which aims to optimize the arrangement of resources within a manufacturing facility, can also be viewed as a network optimization problem. Inadequately planned factory layouts can result in higher operational costs, reduced efficiency, and longer production lead times. The planning process for factory layouts is highly complex, as it involves considering multiple parameters simultaneously, such as machine placement, material flow, and space utilization. This complexity leads to large solution spaces, making manual planning a time-consuming and challenging task.

3.2 Route Planning

The logistics and transportation industries are currently undergoing a technological revolution due to the adoption of the Internet of Things (IoT), digitalization, and a strong focus on reducing carbon emissions. Traditionally, complex optimization problems, such as coordinating production across multiple facilities, have been solved using classical algorithms and heuristics. However, there is still room for improvement in speed and performance. For instance, smart sensors installed on containers in waste management can optimize collection routes, ensuring that trucks are dispatched efficiently to full containers. This optimization challenge is known as the Capacitated Vehicle Routing Problem (CVRP), which due to its complexity, currently relies on heuristic methods for finding solutions, especially when dealing with large-scale industry challenges involving hundreds or thousands of nodes. In contrast with the travelling salesperson problem (TSP), the existence of multiple vehicles in the TSP makes the problem much harder.

Quantum computing can potentially eliminate logistical bottlenecks, leading to more cost-effective and environmentally friendly route planning. There are several quantum solution approaches to solving routing problems. A comprehensive study on how to solve the CVRP using QAOA and VQE is available in reference [6].

3.3 Scheduling

The production process is an essential part of every supply chain. Various facilities in different places must work together as a unified global virtual factory to maximize production and minimize costs. However, finding the optimal schedule for each facility becomes increasingly complex as the production process requires more and more steps. For instance, the front-end processes of semiconductor production involve 500 to 1000 operations, which have to be allocated to several machines daily.

The primary objective of production scheduling is to maximize the use of available facilities to complete all production steps in the shortest possible time while maintaining the order of operations. This optimization challenge is known as the Job-shop Scheduling Problem (JSSP). A related problem, the Flexible Job-shop Scheduling Problem, considers operations that can be executed on several machines in the production steps.

A JSSP search space grows exponentially as we increase the number of operations, machines, and time steps of the production process. In the semiconductor production

process example, a JSSP model can easily have thousands of operations and hundreds of machines. This makes the JSSP particularly hard to solve with current classical computers. One natural question is whether quantum computers have the potential to solve this problem faster or better.

With the right problem encoding, quantum computing has the potential to solve this problem more efficiently. A recent paper[7] studied and compared quantum optimization with quantum annealers (a variation of quantum computers specialized on optimization problems) versus some classical approaches, to solve the Flexible JSSP. They found that the solution qualities from quantum annealers were roughly similar or equal to the classical methods, but the quantum annealers reliably delivered low annealing times.

3.4 Social Networks

Modern digitalized economics require spreading products efficiently through large social networks of customers and retailers. This has a big influence on the success of marketing strategies. For example, customers can be organized in communities to share their experience and their way of problem handling of everyday tasks with a specific product. This could be in internal networks hosted and guided by the company or in common social networks that are freely accessible.

The members of these networks are linked or connected, and they can react to other members' contributions or inputs with likes, stars, or sharing. Members who are connected strongly with others because they have a lot of connections and/or many social activities regarding company topics can be seen as key customers or multiplicators. For companies, the identification of multiplicators is extremely beneficial as addressing them directly can support and simplify the communication strategy for commercial purposes or service issues. In a complex network, this can boost the speed and quality of product information and increase the brands' reputation significantly.

Such networks can be mathematically modeled as graphs: The users are the graph's vertices (or nodes), and the edges are the direct communications among the participants. Multiplicators can be seen as a certain subset of the vertices. In that way, separating this subset from the rest would need to cut a maximum number of edges between the other vertices. Indeed, identifying the right subset of vertices that maximizes the number of cuts when separating is a well-known graph-theoretical problem: *The maximum cut probem*. The Max-cut problem is known to be NP-hard and an interesting use case to be solved on a quantum computer, in particular due to its close relationship with Ising models.

3.5 Predictions of Market Potentials

Allocating resources and budgets efficiently is an essential task in all companies. This is also true when it comes to a company's product portfolio, especially when a lot of articles are available from different product categories. The optimum allocation of resources and budget, like for communication, sales, and product development, requires a data-driven decision base. The aim of the Market Potential project is, therefore, to output an

objectively calculated set of products that are worth investing time and money.

One possibility is to translate the use case into the Modern Portfolio Theory, which goes back to Harry Markowitz. [8] Initially intended for the composition of stock portfolios considering investors' risk appetite (risk-return estimation), it is suitable for any system in which an objective function should be optimized.

Portfolio optimization is an NP-hard optimization problem. It takes into account correlated uncertainties in the individual decision variables. The covariance matrix of the individual decisions encodes the correlation of uncertainties and has a decisive influence on the final result. It can then be used to predict the market potential of the product portfolio under a strategy defined by the decision variables (analogous to investment strategy in the stock market).

Quantum computing has the potential to solve the resulting optimization task more efficiently and effectively [9]. The main points for improvement are: finding patterns in small amounts of data only, modeling under uncertainties, and a speed-up advantage for NP-hard problems.

3.6 Software Verification

Software testing is crucial in the software development lifecycle, ensuring reliability, security, and functionality. Early verification and validation can lead to significant cost savings by identifying flaws before they escalate, avoiding costly post-deployment fixes, and protecting user safety and reputation.

In high-stakes systems, such as medical devices, rigorous testing is vital. The Therac-25 incident [10] exemplifies the dire consequences of inadequate testing, highlighting the need for thorough methods. In theory, the ideal method is formal verification, which uses mathematical proofs to ensure the correct output of the software.

Formal verification involves creating mathematical models to verify software against a formal specification, including expected inputs, outputs, and behavioral rules. For instance, a banking system must ensure account balances never go negative. Various techniques exist for building abstract models from source code and specifications, including symbolic execution and reachability analysis. Ultimately, the problem of deciding whether a software is correct translates to deciding the satisfiability of a Boolean formula. Despite its advantages, formal verification faces challenges when handling modern and complex pieces of software. Rice's theorem [11] limits the properties that can be verified, and the halting problem is undecidable.

Quantum computing might give rise to algorithms that allow these processes to be performed more efficiently or even at all. This might lead to higher adoption of systematic verification and more secure and reliable software.

4 From the Application to the Cost Function

From the mathematical side, an optimization problem is defined by a cost function that needs to be minimized as well as a search space of candidate solutions, often restricted through constraints. For the same industry use case and the same optimization goal, there are often multiple ways of formulating the cost function and constraints. In this section, some topics are discussed that should be considered when weighing the options against each other, as well as some tools that can help:

- Section 4.1 brings attention to a very important topic, namely the densitiy or sparsity of encodings. Since complicated constraints are a significant obstacle for the success of quantum algorithms, the ratio of feasible solutions (i. e. those, that satisfy the given constraints) is an important ingredient into choosing good encodings.
- Section 4.2 considers the more general task of representing data on a quantum computer, with different types of data admitting for different encodings. These pure data-based encoding techniques can help and inspire the data representation tasks in combinatorial optimization, e.g., when it comes to input data or mapping of bit strings to solutions.
- Section 4.3 tackles the important question how industry problems where solutions typically need to satisfy constraints can be solved with formulations such as quadratic unconstrained binary optimization (QUBO).
- Section 4.4 complements the previous section in reporting condensed results on the choice of a penalty factor for specific constraints to incorporate them into a mathematically unconstrained cost function.
- Section 4.5 describes an intermediate formulation step called the encoding-independent formulation and discusses advantages and disadvantages of this additional step.
- Section 4.6 highlights an innovative method to reduce the memory overhead for quantum circuits provided certain conditions are met that allow storing multiple binary variables into the same qubit.

4.1 Dense and Sparse Encodings

When solving a problem on a quantum computer, it is necessary to encode it in a way where it can be represented with qubits. For quantum optimization algorithms like VQE and QAOA, measured states are interpreted as bitstrings that can be mapped to solutions for the the optimization problem. Generally, there are many options for choosing an encoding for a specific problem. When choosing such an encoding, there is a trade-off between the density of the encoding and the order of the Hamiltonian (i. e., whether it is quadratic or has terms containing more than two variables). While dense encodings use fewer qubits and produce fewer infeasible bitstrings that do not correspond to a valid solution, the Hamiltonian for such an encoding tends to have higher order terms than.

These higher order terms can be potentially problematic when implemented on a quantum computer.

For example for the Travelling Salesperson Problem (TSP), a common encoding results in a Quadratic Unconstrained Binary Optimization (QUBO) form. For n nodes in the graph, each timestep $t \in \{1, ..., n\}$ uses a one-hot encoding to denote the index of the node visited. Therefore, the binary variable x_{it} is the indicator for the statement "Node i is visited at time step t". If the statement is true, it is 1, otherwise 0. Given an encoding of a valid route, the length of the route to be minimized can then be evaluated via the quadratic cost cuntion

$$C_d = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \sum_{t=1}^{n} x_{it} x_{jt+1} \tag{1}$$

with the distance between nodes i and j given as w_{ij}. However, many variable assignments will lead to infeasible states. This is the case if either multiple cities are visited at the same time step, i.e. the one-hot encoding contains multiple ones, or if the same city is visited multiple times in one route. To combat these infeasible solutions, the QUBO encoding introduces penalty terms

$$C_p^1 = P \sum_{i=1}^{n} \left(1 - \sum_{t=1}^{n} x_{it} \right)^2, \tag{2a}$$

$$C_p^2 = P \sum_{t=1}^{n} \left(1 - \sum_{i=1}^{n} x_{it} \right)^2 \tag{2b}$$

with a large enough penalty factor P. The resulting QUBO formula from combining these terms can then be transformed into a Hamiltonian in order to evaluate energies during optimization. Effectively, the quantum algorithm then solves an unconstrained problems.

Through Higher Order Binary Optimization (HOBO) encodings [12], the portion of feasible solutions out of all candidates can be increased. Instead of denoting the (integer) label of a node at a timestep t with one-hot encoding, this encoding represents the label by a binary number b_t. For this encoding, the first part of the Hamiltonian H_{valid} ensures that the binary index for a timestep does not exceed the number of nodes in the graph. Let $\tilde{b}_{K-1} \dots \tilde{b}_0$ be a binary representation of $n-1$ and define $k^0 \in K_0$ to be indices such that $\tilde{b}_{k^0} = 0$. Then the function H_{valid} takes the form

$$H_{\text{valid}}(b_t) \coloneqq \sum_{k^0 \in K_0} b_{t,k^0} \prod_{k=k^0+1}^{K-1} \left(1 - (b_{t,k} - \tilde{b}_k)^2 \right). \tag{3}$$

Additionally, the Hamiltonian term H_δ is constructed to compare two binary numbers, evaluating to one if they are equal and zero otherwise. This function determines the timesteps at which nodes are visited. All in all, H_δ is defined as

$$H_\delta(b, b') \coloneqq \prod_{k=1}^{K} \left(1 - (b_k - b'_k)^2 \right). \tag{4}$$

The full Hamiltonian for the HOBO encoding is given by

$$C = P\sum_{t=1}^{n} H_{\text{valid}}(b_t) + P\sum_{t=1}^{n}\sum_{t'=t+1}^{n} H_\delta(b_t, b_{t'}) + \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\sum_{t=1}^{n} H_\delta(b_t, i)H_\delta(b_{t+1}, j). \tag{5}$$

Similarly to the QUBO encoding, this formula can be used to create a Hamiltonian which maps quantum states to energies according to the problem structure. However, the interpretation of the single variables is much less evident.

Third, an ideally dense encoding is the permutation encoding [13]. It algorithmically maps every measured state to a valid TSP solution. While this encoding produces only feasible solutions and uses fewer qubits, there is no known efficient construction of a Hamiltonian. Hence, this method is limited to methods like VQE where the Hamiltonian does not need to be implemented on q autnum computer. For those methods, a classical function can be applied to the measured bitstrings to evaluate the energies for the optimization. Depending on the problem, this classical evaluation can be performed efficiently. For small TSP instances, it has been found that the density of the encodings is strongly correlated with the performance of the optimization with denser encodings generally producing better results [13]. Specifically, the permutation encoding performs significantly better than the HOBO and QUBO encoding. Yet, the optimization for the permutation encoding seems more unstable. This might point toward the algorithm struggling to identify the properties of the problem and instead finding a solution by sampling random guesses.

To summarize, for novel applications, different encodings should be considered to transform it into a mathematical problem. Typically, there is a trade-off between simplicity of the encoding and density of the feasible solutions. Ultimately, the encodings need to be tested in how quantum algorithms can travel the loss landscapes efficiently.

4.2 Considerations for Representing Data on a QC @Thomas

Necessarily, each algorithm and optimization method needs application-specific data. Therefore, one needs to represent the data of different types on a quantum device. Whereas an encoding for quantum annealers most often requires the definition of a problem-specific Hamiltonian which is directly mapped to the annealing hardware topology, there is a considerably larger variation on gate-based quantum devices. By chosing an appropriate representation, different kinds of advantages become available to applicationers. For example, if the application requires the manipulation of images, a dense encoding may be used to represent and process all pixels simultaneously due to the superposition principle, while a later retrieval by measurement is typically very cumbersome. On the other hand, less dense representations often require more computational resources, e.g., the number of required qubits, and a detailed bookkeeping at the benefit of enabling an easier information retrieval after the algorithm. The specific application determines this trade-off and guides towards a selection of well-suited data representations. Since a detailed overview over

many different data types is out of scope for this report, we exemplify our approach with images, since they are one of the most important data types in data processing. Many other types of data such as matrices, time series or graphs can be trivially represented by images.

The currently most widely used format is the *Novel Enhanced Quantum Representation* (NEQR) [14] of digital images due to its ability to retrieve pixels accurately by projective measurement [1]. Put briefly, NEQR represents a $2^n \times 2^n$ b-bit image via a quantum state incorporating b qubits for the pixel intensity and $2n$ qubits (in the two-dimensional case) for the pixel indices determining its location on the image. A superposition on the position qubits is generated by applying Hadamard gates, thus the collection of these qubits contains data for every single pixel index. Consequently, the quantum state of NEQR represents *all* pixels and all possible color values at once, contrary to a classical dense data structures. A subsequent unitary transformation is required to set the pixel value for each pixel, i.e., in NEQR there is a sequence of 2^{2n} unitary gates, each considering some pixel position and setting its value. In short, a $2^n \times 2^n$ b-bit image f can be represented in NEQR via the quantum state

$$|I\rangle = 2^{-n} \sum_{x,y=0}^{2^n-1} |f(x,y)\rangle\, |y\rangle\, |x\rangle\,.$$

Image processing algorithms can now exploit the compact encoding by modifying all pixels simultaneously. For example, consider the inversion of the pixel grayscale value by means of inverting the bit string representing the value, e.g., in an 8-bit image this maps the color $85 = 0b01010101 \mapsto 0b10101010 = 170$. Both in classical and quantum implementations, such a bitflip operation is executed by applying a NOT gate on each bit or qubit, respectively. When applied to NEQR, one applies the Pauli X gate on all intensity qubits and thus flips the intensity values of all pixels *simultaneously*. After processing, the image may be obtained by a measurement which yields (disregarding noise) a bitstring representing the position and color of a single pixel. In order to retrieve the overall image, $\mathcal{O}(M2^M)$, where $M = 2n + b$, measurements are required [16].

Clearly, there is a tradeoff between the compactness of the image representation and the effort needed to retrieve its information. Particularly dense encodings therefore show their benefit when only some general properties of the processed image are needed that can be obtained with fewer measurements. In general, the readout complexity can be reduced by employing quantum algorithms that only need specific bits of information on the quantum state.

Naturally, there exist representations dedicated to specific data structures. For optimization problems, important data often comes in the form of a graph. For example, an

[1] Other representations, e.g., the *Flexible Representation of Quantum Images* (FRQI) [15], compress information like color in a single qubit as its probability amplitude. The latter implies that any measurement retrieves the color information only by statistical considerations, rendering accurate measurements impossible with a finite amount of measurements [14]. Even more so, when considering real hardware, the physical noise may also corrupt the measurement results.

undirected loop-free [2] graph $G = (V, E)$ consisting of vertices V and edges E may be represented as a *graph state* $|G\rangle = \prod_{\{a,b\}\in E} U_{ab} \ |+\rangle^{\otimes|V|}$, where each vertex is encoded in a single qubit. The edges are represented by considering that an edge resembles an interaction between two particles, thus any edge $\{a, b\}$ corresponds to a unitary gate U_{ab} acting on the qubits a and b. The complexity of such an encoding clearly depends on the underlying graph, but is very economic in the amount of gates that are needed to produce the superposition and under the consideration that many real-world graphs are rather sparse.

A broader overview over many more data types and their quantum representations can be found in [17].

4.3 Handling Constraints

A QUBO is, by definition, an unconstrained problem in the form of the following equation:

$$f(x) = \sum_{i=1}^{n-1} \sum_{j>1}^{n} q_{ij} x_i x_j + \sum_{i=1}^{n} q_{ii} x_i, \tag{6}$$

where x_i are the problem's binary variables, n the number of variables and q_{ij} the coefficients of the QUBO matrix.

However, most problems of interest contain constraints, e.g. in the Traveling Salesperson Problem (TSP) each city may be visited only once. Thus, it is essential that constrained problems are first transformed into unconstrained problems to make use of QUIBO formulations. This transformation, in practice, means that constraints must be encoded as penalty terms, such that invalid solutions are penalized and valid solutions not. Then, the algorithm will automatically treat the infeasible solutions as inferior ones to feasible ones. Mathematically, one distinguishes between two types of constraints: *Equality Constraints* and *Inequality Constraints*.

Equality Constraints are generally easier to incorporate into the problem formulation. Let's consider a general equality constraint of the form:

$$\sum_{i=1}^{n} c_i x_i = M, c_i \in \mathbb{R} \tag{7}$$

One can encode this constraint into the QUBO formulation by including the following penalty term:

$$P \left(\sum_{i=1}^{n} c_i x_i - M \right)^2, \tag{8}$$

where P is a n appropriately chosen penalty factor such that invalid solutions are penalized, with the best valid solutions becoming the lowest energy solutions of the QUBO. For a

[2] A graph is loop-free if there are no direct edges from a node to itself, whereas cycles involving several nodes are valid, as opposed to an acyclical graph.

more detailed explanation on how to choose P and how it affects the solution quality, see section 4.4. Note that if a solution is valid, $\sum_{i=1}^{n} c_i x_i = M$ and thus the penalty is 0. However, if a solution is invalid, whether because $\sum_{i=1}^{n} c_i x_i < M$ or $\sum_{i=1}^{n} c_i x_i > M$, the solution is penalized.

Inequality Constraints, however, are a more complicated case. Let's also consider a general inequality constraint of the form:

$$\sum_{i=1}^{n} a_i x_i \leq L, a_i \in \mathbb{Z} \tag{9}$$

The typical method for encoding a constraint of this form into a QUBO formulation is by using *Slack Variables*, that is, extra variables. The general idea is to first rewrite the inequality constraint using these slack variables, turning it into an equality constraint:

$$\sum_{i=1}^{n} a_i x_i = DS, \tag{10}$$

where S are the slack variables and D the coefficients associated with them. The number of slack variables m needed to encode a particular constraint is given by the number of variables needed for the binary expansion of L, which has complexity $O(log(L+1))$. Finally, once this transformation has been made, the resulting equality constraint can be encoded into the QUBO using the same approach as in eq. (8). One should note that slack variables lead to an increase not only in the number of qubits needed but also in the depth of the circuit (since more linear and quadratic quantum gates are needed to represent the interactions added by the slack variables). More specifically, the number of linear terms grows with $O(m)$ and the number of quadratic terms with $O(nm + m^2)$.

A method for limiting the number of slack variables needed and consequently both the number of qubits of the resulting circuit and the number of gates is developed in [18]. The basic idea is to divide both sides of eq. (9) by a value ρ. For instance, if $\rho = L$, then the right side of the constraint becomes 1 and a single slack variable is needed to encode the constraint. However, one should be aware that such a method is an approximation.

More recently, a method has been developed to encode inequality constraints without slack variables [19]. It works as follows. One starts by approximating the inequality constraint of eq. (9) given by:

$$L - \sum_{i=1}^{n} a_i x_i \geq 0 \tag{11}$$

using the second-order expansion of the exponential decay function (since a QUBO has at most quadratic terms) and then introduces this function as a penalty term into the QUBO formulation. The method was empirically shown to achieve a higher probability of finding the optimal solution to some combinatorial optimization problems while requiring no slack variables and hence no additional qubits or quantum gates [19].

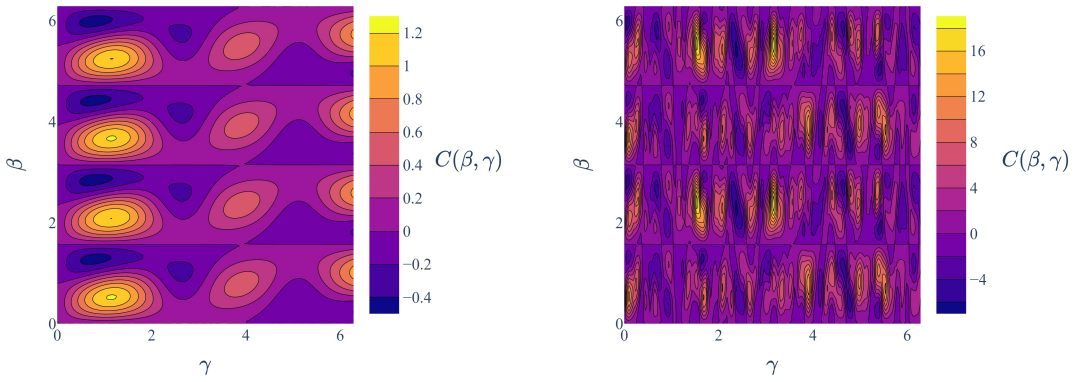
Beyond transforming optimization problems into unconstrained versions, some algorithms can be tuned in order to only explore parts of the solution space [20]. However,

their practical realization is often only efficient enough for current devices for very simple constraints such as Hamming weight preservation (i. e., the number of 1's in the output bitstring is fixed). Therefore, penalty-based formulations currently provide the easier path to formulate combinatorial optimization problems with constraints.

4.4 Penalty Factor Selection

The penalty factor P in eq. (8) plays a crucial role in shaping the cost landscape and influencing the optimization process. It ensures that the optimization algorithm favors valid solutions over invalid ones. From a mathematical standpoint, there is a minimal penalty factor needed to ensure that the optimal feasible solution has a lower effective cost than the lowest infeasible solution. In practice, penalty factors need to be estimated and are often chosen in a way such that *all* feasible solutions have a lower cost than the best infeasible solution.

As the penalty factor increases, more local minima are introduced into the cost landscape, resulting in a rougher surface. Figures 1 illustrate this effect, where a higher penalty factor (P) leads to a more rugged cost landscape.



(a) Optimized penality factor

(b) Large penalty factor

Figure 1: Cost landscape of a one layer QAOA with four qubits with different penalty factors.

When selecting the value of the penalty factor, there is a trade-off to consider:

- If the penalty factor is too low, invalid solutions may become optimal in terms of the cost function, leading to suboptimal or incorrect results.
- If the penalty factor is too high, the cost landscape becomes increasingly difficult to optimize, with numerous local minima that can trap optimization algorithms, especially local optimizers.

Empirically, a penalty factor that is too high can also result in a loss of accuracy: The resulting algorithm may not distinguish anymore between the quality of two feasible solutions and thus effectively produces feasible solutions (but not optimal ones) at random. Depending on the use case, producing feasible solutions can be hard (e.g., creating a valid shift plan for a company) or easy (e.g., a TSP route containing each city exactly once).

Choosing an appropriate penalty factor value is crucial for achieving good optimization performance. However, significant performance differences empirically appear with drastic changes in the penalty (compared to the scale of the cost function, e.g. typical length differences of TSP routes). A balance must be struck between penalizing invalid solutions sufficiently while maintaining a cost landscape that is amenable to optimization algorithms. In some cases, it may be necessary to employ techniques such as adaptive penalty factor adjustment or incorporate problem-specific knowledge to select an effective penalty factor value. Since the feasibility of solutions can typically be verified easily in the case of NP-hard optimization problems, it is recommended to always check for feasibility in practice to ensure that the penalty factor was not too small.

4.5 Encoding-Independent Formulation

Solutions of an optimization problems need to be represented as bitstrings comprised of binary variables. The cost function maps the individual strings on their respective function value. However, many optimization problems have no native representation in binary variables. Instead, it can be useful to employ an intermediate representation with discrete integer variables. From the QuaST project, this has been proposed in [21]. A central advantage is the increased modularity of such an approach: A new optimization use case only needs to be translated into one encoding-independent representation after which no additional work is needed to take the step to binary variables.

Following [21], the conversion of the integer cost function to a binary one is achieved with *value indicator functions* relating the integer variables to bitstrings. This allows an elegant, but theoretical representation of the cost function and problem-specific constraints. Additionally, so-called *core constraints* are introduced that ensure that the bitstring leads to a valid integer variable. For example, bitstrings of the one-hot encoding must contain a single 1 and be 0 otherwise. The full problem formulation is then given by the binary cost function, the problem-specific constraints as well as the core constraints. Optionally, some (or all) constrains can be included in the cost Hamiltonian through penalization. Ultimately, the handling of the constraints is left to the algorithm.

Ref. [21] showcases the encoding-independent formulation for a range of optimization problems and encodings. Further attempts at using the theoretical foundation for a practical implementation of the QuaST decision tree proved to lead to a significant computational overhead, however. The symbolic computations required to resolve the indicator functions in the cost function lead to a slow, cumbersome process. In conclusion, the encoding-independent formulation is still useful at enabling multiple different encodings easily, but solutions optimizes for production should implement direct encoding methods from the application.

4.6 Reducing Memory Footprint by Random Access Coding

In the case that the memory need is about twice as high as the hardware provides, there is an encoding technique, that might allow the problem to still be evaluated. This is possible with Random Access Codes, that allow up to three variables to be encoded in a single qubit. We will use the code on the QUBO formulation and briefly explain what drawbacks it carries.

To pack three variables into one qubit, they are encoded in different bases. This is only possible if the variables commute, a specific property that makes them compatible. A sufficient test is for the corresponding vertices in the weighted graph, given by the QUBO, to have the same color in any valid coloring.

Encoding three variables in one qubit does come with problems. A measurement can only target a single variable and only extracts the correct value with high probability. It is therefore necessary otfperform the procedure multiple times, while measuring in different bases to allow all values to be extracted.

At the same time, the new QUBO is a relaxation of the original one, with some interesting properties. While there might be better solutions added, than there were before, they turn out to be linear combinations of the original optimal solutions. Measuring the new optimum, rounds them to the original optimum. This adds a bit of tolerance to the optimization procedure. Here, a new problem arises. There is no guarantuee to measure the same optimum in every repetition. A rounding schema is needed to interpret the results.

In conclusion, the procedure can allow oversized problems to still be executed, at the cost of a higher number of repetitions and the need for schemus to interpret and incorporate the measurement results.

5 Choosing and Improving the Algorithm

After an application has been encoded into a concrete optimization problem, an algorithm needs to be selected for its solution. From the quantum and classical algorithms at hand, the QuaST project has been focused on variational algorithms, namely the variational quantum eigensolver (VQE) and the quantum approximate optimization algorithm (QAOA). These algorithms have been evaluated. It turns out that the classical optimization is hugely important for the success of the algorithm. The best algorithm always depends on the exact problem formulation, structure of the solution space as well as the available quantum and classical computing resources.

For completeness, this chapter starts with a short description of VQE and QAOA. Afterwards, several aspects and improvements from within the QuaST projects are discussed:

- Section 5.2 discusses evaluation methods for quantum-assisted algorithms, in particular which metrics can be used to assess their performance and solution quality.
- Section 5.3 attempts to eliminate or reduce the load caused by the variational parameter updates in QAOA.
- Section 5.4 discusses general aspects about the classical optimization routine, namely considerations to use global or local optimizers for variational algorithms.
- Section 5.5 showcases how post-processing VQA results can help in obtaining better solutions.

5.1 Basic Algorithms

5.1.1 VQE

The Variational Quantum Eigensolver (VQE), introduced in [22], has garnered considerable attention from the research community in recent years. By leveraging the variational principle, VQE is capable of calculating the ground state energy of a Hamiltonian, a crucial task in quantum chemistry and materials science. Additionally, VQE has been used to solve optimization problems. The limitations of conventional computing methods, which struggle to accurately model the exponentially growing electronic wave function of many-electron systems, make VQE an attractive solution. By enabling the modeling of these complex wave functions in polynomial time, VQE is one of the most promising near-term applications of quantum computing. For a detailed description as well as additional application areas we refer the reader to [23].

5.1.2 QAOA

The Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum-classical algorithm designed to solve combinatorial optimization problems. It is particularly well-suited for NISQ (Noisy Intermediate-Scale Quantum) devices due to its relatively shallow quantum circuits and has been widely studied in recent literature.

QAOA uses a parameterized quantum circuit called an ansatz, that applies alternating layers of unitary operations based on two Hamiltonians: The cost Hamiltonian Unitary $U_C(\gamma) = e^{-i\gamma H_C}$ (sometimes also referred to as the problem Hamiltonian), which is based on the Ising formulation of the QUBO problem and has been established in section 4, and a mixing Hamiltonian Unitary $U_M(\gamma) = e^{-i\beta H_M}$.

The full QAOA circuit alternates between these two unitaries for p layers, where p can be seen as a hyperparameter that controls the algorithm's accuracy. The final state prepared by the quantum circuit is:

$$|\phi(\beta,\gamma)\rangle = U_M(\beta_p)U_C(\gamma_p)...U_M(\beta_1)U_C(\gamma_1)|+\rangle^{\otimes n} \tag{12}$$

After this quantum state has been prepared, the expectation value of the cost Hamiltonian $C(\beta,\gamma) = \langle\phi(\beta,\gamma)|H_C|\phi(\beta,\gamma)\rangle$ is estimated. This expectation value serves as the objective function for a classical optimizer, which adjusts the parameters β and γ to minimize the cost. The classical optimizer iteratively refines the parameters β and γ until convergence, ideally finding parameters that prepare a quantum state that closely approximates the optimal solution to the problem (see section 5.4). However, another strategy for the selection of these parameters can be based on fixed-parameter strategies (see section 5.3), bypassing a possible time- and resource-consumning optimization process.

Once a set of (sub-) optimal parameters are found, the quantum circuit is run using these parameters, and the resulting state is measured in the computational basis. The measurement results correspond to candidate solutions to the optimization problem, and the best solution is chosen based on the cost function value.

The optimization of the parameters β and γ can be a challenging task and the selection of the classical optimizers and the right initialization of the parameters is crucial for the performance of the algorithm. Additionally, the number of layers as well as the previously defined cost Hamiltonian greatly influence the solution quality.

5.2 Performance Metrics

The performance of quantum computing solutions is currently measured with a varied set of performance metrics. Ultimately, application-level metrics [24] are the closest to the final goal of improving industrial optimization problems. However, in practice, derived metrics are used frequently to evaluate specific aspects of a solution. In principle, algorithm performance can be evaluated in different aspects: *performance in speed*, *performance in accuracy*, *performance in probability* and *performance in scale*. Typically, these performance metrics are not indepentend. For example, a better accuracy needs also a better performance in probability and often leads into a better scalability as well.

5.2.1 Performance in Speed

Performance in speed is what everybody expects when talking about quantum computing. It forms the most basic and perhaps most pure form of quantum advantage: A quantum computer carries out a specific computation faster than a classical computer. This should result in finding a problem's solution faster with appropriate quantum algorithm than it

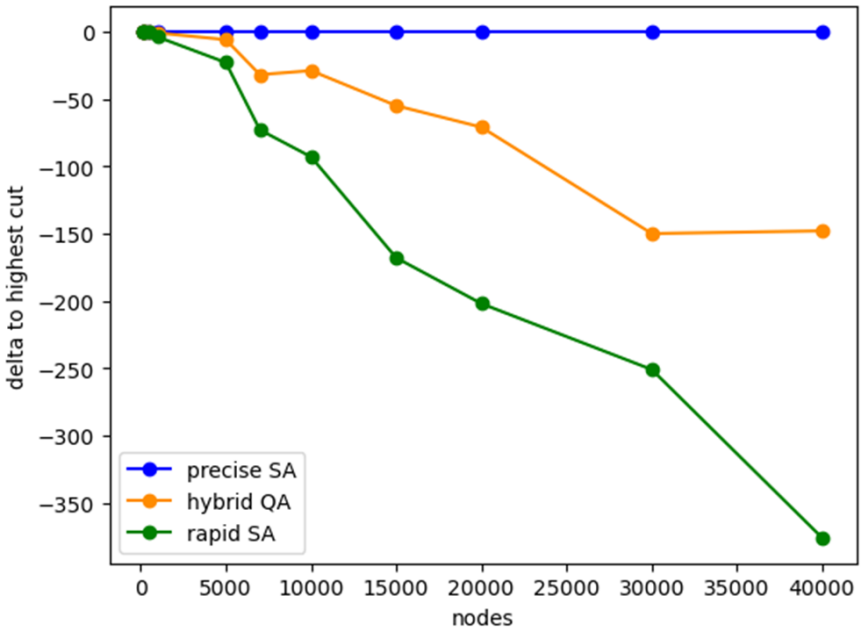


Figure 2: Comparison of Max-Cut between Simulated and Quantum Annealing (Hybrid Solver)

would be possible with any classical approach. However, no one could show a practical quantum advantage so far. Present-day NISQ quantum computers are not yet big and powerful enough to be able to achieve this. Hence, performance in speed is not yet the main focus when doing research on solving of industrial optimization problems.

5.2.2 Performance in Accuracy

Performance in accuracy helps monitoring if better results are achievable with the use of quantum algorithms than with classical approaches in a similar time frame. This is particularly relevant for optimization problems where the quality of efficient approximations is bounded, or where typical algorithms cannot be easily tuned to invest more runtime into a better approximation. With quantum annealing, especially when using a hybrid solver, a better accuracy can be observed already on problem sizes that are relevant for real-world use cases - under certain circumstances. In an exemplary manner, this can be seen in fig. 2 that shows the results when looking for MaxCut values on sparse graphs representing communication structures on a social network (also refer to section 3.4). The investigated graphs are randomly generated with up to 40,000 nodes.

To build reference results (on bigger graphs, the real MaxCut value can no longer be reliably determined), a time-intensive classical computation with simulated annealing (SA) is performed (blue line in Fig. 2). The time invest for the reference results is up to 200 minutes for the 40,000 nodes graph that corresponds to 3 hours and 20 minutes.

The quantum annealing (QA) experiments need only up to 4 minutes for the 40,000

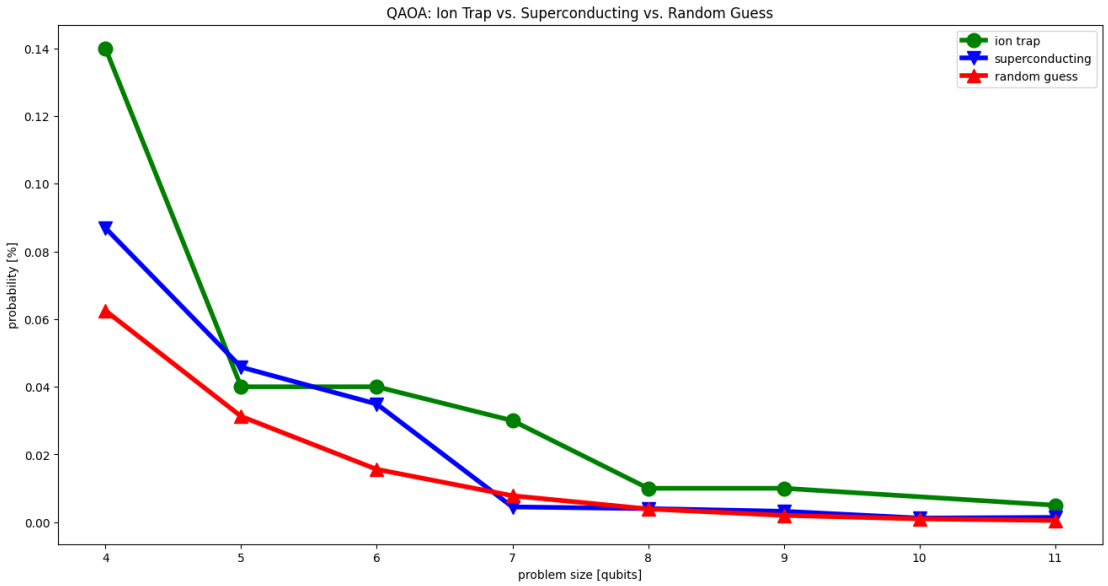


Figure 3: QAOA: Ion Trap vs. Superconducting vs. Random Guess

nodes graph to achieve heuristic MaxCut results which are indeed lower (worse) than the reference values (yellow line in Fig. 2). However, when comparing these results with a SA configuration that runs in a similar time window as the QA algorithm of about 5 minutes (green curve in Fig. 2), QA delivers the better results, especially when the problem size increases. With the same invest of time, QA (hybrid solver) has a better accuracy than SA when solving MaxCut problems on sparse graphs.

5.2.3 Performance in Probability

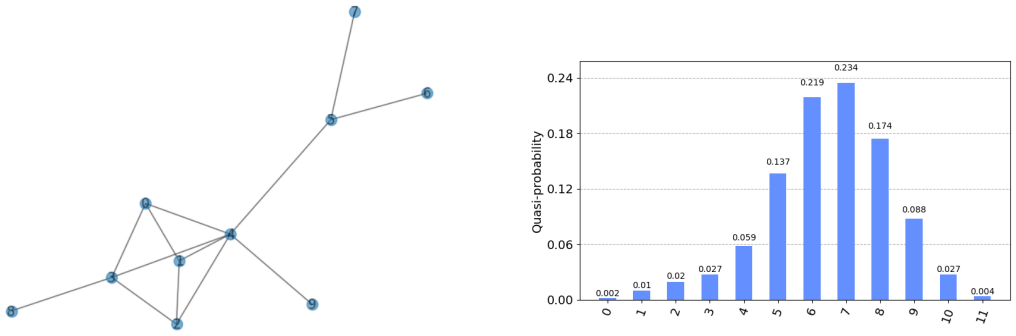
Because of the probabilistic nature of a quantum computer, an interesting performance metric is the probability of finding the optimum solution, or a solution of a specific quality. A quantum algorithm is typically designed to deliver the sought result with a maximum probability compared to all other results. Unfortunately, this is not easy to achieve in many practical cases. The probability of the optimum result can be veiled depending on the problem size, the noise behavior of the specific quantum device or the chosen hyperparameters of the particular algorithm. In the worst case, the QC acts like a slot machine and just guesses randomly solutions.

Each problem has a certain probability for predicting the optimum solution accidentally depending on the problem type and size. This probability can be seen as the statistical threshold. The performance in probability metric measures how much the quantum algorithm outperforms this threshold.

For example, fig. 3 illustrates this for the portfolio optimization problem, based on Markowitz Portfolio Theory. For details of the corresponding use case refer to section 3.5. The diagram shows the comparison of an ion trap (green curve) and a superconducting quantum computer (blue curve) with the statistical threshold for just guessing the

optimum solution (red curve). The x-axis shows the problem size, meaning here the number of considered products for which the optimum resource allocation is sought. The y-axis displays the probability. The result shows that already at the problem size of 7 products the superconducting QC is guessing the solution while the ion trap QC delivers the optimum solution still with a certain higher probability. That suggests that an ion trap quantum computer works better for this particular use case in the sense that it delivers the desired optimum solution with a higher probability. That typically also leads to a better accuracy than using a superconducting device.

Another example how performance in probability can be depicted, can also be explained based on the social network MaxCut problem. It is illustrated on a small 10 nodes toy example for a social network. The use of a gate-based ion trap quantum computer only allows currently experiments with small problem sizes with the access to up to 12 qubits. The example graph and its individual statistical distribution of all possible cut values are depicted in fig. 4

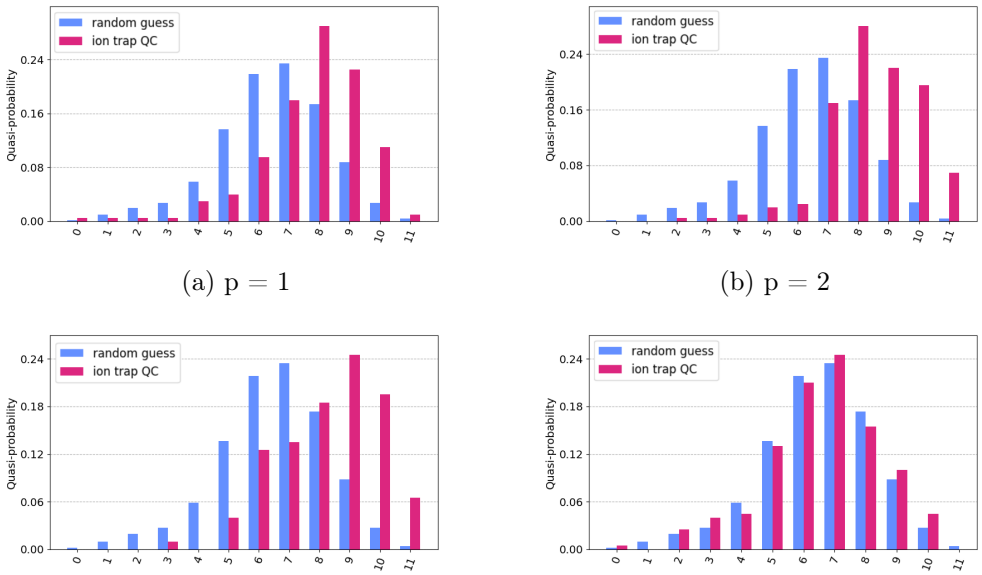


(a) Social Network Graph (b) Statistical Cut Distribution

Figure 4: Social Network MaxCut Problem (10 Nodes)

The natural cut distribution of the graph (shown in fig. 4b) can be compared to the result histograms of the used quantum algorithm. In fig. 5 the comparison with QAOA result histograms is illustrated for different p-factors, with 11 being the MaxCut value. The four diagrams show that even with p-factor of one, there is a shift to the right of the QAOA histogram compared to the pure statistical cut distribution, meaning a shift into the direction of the MaxCut value that the algorithm is looking for. With increasing p-factor the shift in MaxCut direction gets stronger up to the p-factor of 4. Then, the QAOA result distribution approximates the statistical distribution. Finally with $p = 7$, the QAOA histogram only coincides with the natural cut distribution of the graph (fig. 5d). At this point, the quantum computer is just guessing the result, and the MaxCut values cannot be found anymore (no red bar on the MaxCut value of 11 in fig. 5d).

This performance metric representation shows that for this specific optimization problem the optimal value for the hyperparameter p of the QAOA algorithm is four.



(a) p = 1 (b) p = 2

(c) p = 4 (d) p = 7

Figure 5: QAOA MaxCut Results

5.2.4 Performance in Scale

Performance in scale is a metric to extrapolate towards industrial problem sizes to estimate what to expect when solving specific real-world problems. There are two aspects that needs to be considered: First, what is the maximum problem size of a particular problem that can be solved on a real quantum computer? Second, where is the limit of the algorithm itself? That means what is the maximum problem size to be solved on an *ideal* quantum computer running without any errors during the computation? An ideal quantum computer can be simulated by an *quantum simulator* without using any noise model.

An example for performance in scale with quantum annealers for solving certain MaxCut problems in context of social network analysis is shown in Fig. 2. The graphic shows clearly the increasing gap between the cut results of the quantum annealer (yellow curve) and the simulated annealing algorithm (green curve) when computations are done in similar runtimes (Fig. 6).

5.3 Fixed-angle QAOA

As a variational algorithm, QAOA relies on alternating classical and quantum computational steps. The quantum computer calculates the cost function for a specific parameter set, then the classical optimizer performs an optimization of the parameters. However, this iterative procedure leads to a large communication overhead and, in general, inefficient use of the computational resources. Furthermore, the optimization landscapes of such algorithms are known to be swamped with bad local minima [25] and, depending on the

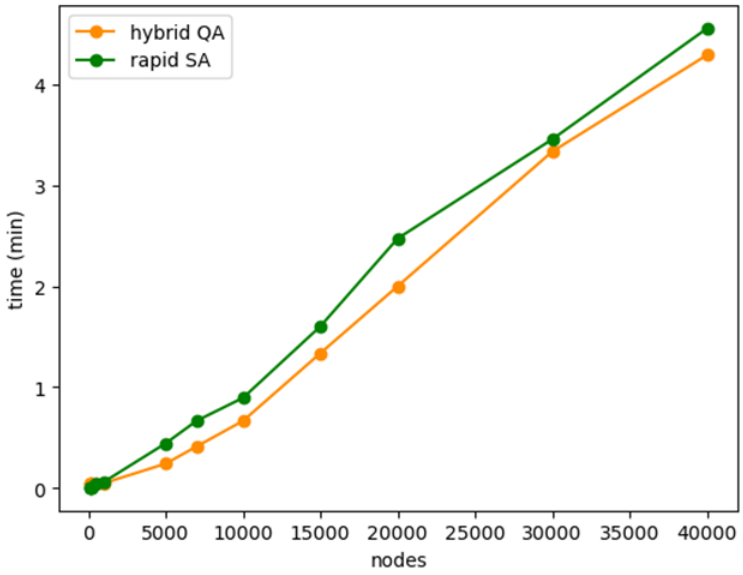


Figure 6: Runtimes of QA and SA in Dependence of Graph Size

ansatz and observables, they can exhibit Barren Plateaus (BPs). There, the variance of the gradient decays exponentially with the number of qubits or, in other words, that the loss function is concentrated. In practice, if a circuit exhibits BPs, they become untrainable as the number of qubits increases for both gradient-based and gradient-free methods [26]. Last, gradient-based optimization is quite expensive on a real quantum computer since the computation of a single gradient, e. g. by the parameter-shift rule [27] requires many evaluations of the cost function. While on simulators efficient techniques such as Backpropagation and Adjoint Differentiation can be used, on real hardware they require 2 circuit executions for each parameter, in total $2 \times p$ executions for a p-layer QAOA.

For these reasons, fixed-angle QAOA-based methods are of interest, where the angles are initialized using some heuristic and no optimization is necessary.

5.3.1 Tree angles

If a given problem graph $G(V, E)$ is sufficiently sparse, then the distance-p subgraphs of the individual nodes or edges of the graph are not expected to contain many loops. This would mean that these subgraphs are with high probability trees, i.e. graphs without any loops. If the problem graph has fixed regularity d, meaning that every node has exactly d neighbors, these subgraphs would be d-regular trees with high probability. It is a well known fact that a sequence of tensors associated to a tree structure can be contracted with a cost that scales exponentially with the depth p. The cost scaling of computing local expectation values in such cases is thus crucially not exponential, but at most polynomial, in the number of nodes $|V|$ of the problem graph.

The evaluation of the cost function of a QAOA circuit for a d-regular graph problem

can be expressed as such tensor-network contractions over trees. The contraction sequence from outer leaf to root of the tree can be written down in anylytical form as recursive formulae. These formulae can be minimized to give (near-)optimal QAOA angles in order to solve unweighted QUBO problems on regular graphs [28].

The tree angles can however also be reused in different scenarios. Either as fixed, and hopefully near-optimal angles, or alternatively as initializations for the optimizer. For example, if the problem graph has average regularity $\overline{d} = \frac{1}{|V|}\sum_{i\in V} d_i \ll N$ with small standard error, meaning that the graph is sparse but not regular, the tree angles should still give good results. An example of such cases are unweigthed QUBO problems on Erdös-Rényi graphs with fixed average regularity $\overline{d}$. For weighted problems defined on sparse graphs, the approach can be twofold: either the formulas in Ref. [28] can be modified to explicitly incorporate the weights, either a first initial guess can be simply provided by the unweigthed case, where the unweigthed angles should be renormalized by the average weigth.

5.3.2 Linear Ramp Initialization

In [29], the authors initialize the parameters using a linear-ramp schedule, which is inspired by Quantum Annealing Theory. Given the parameters $\Delta\beta$ and $\Delta\gamma$ and a number of layers p, the parameters β and γ are initialized as:

$$\beta_i = (1 - \frac{1}{p})\triangle_\beta \quad and \quad \gamma_i = \frac{i+1}{p}\triangle_\gamma \tag{13}$$

Thus, one can execute the resulting quantum circuit with no need for further optimization. Furthermore, the authors of [29] propose the following conjecture:

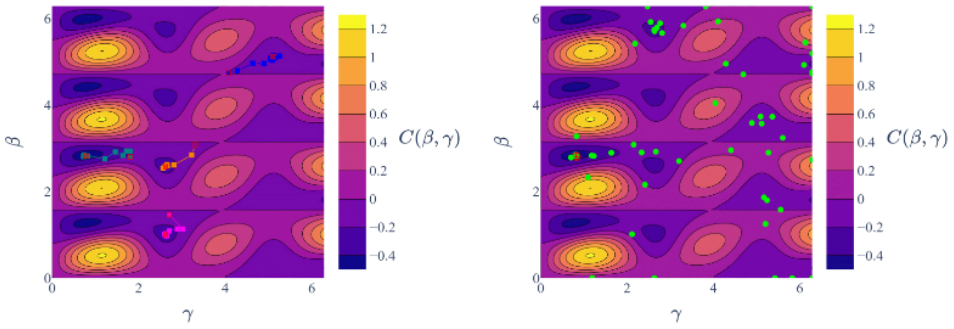
$$p(x^*) = 1/2^{\nu N_q/p}, \tag{14}$$

where $p(x^*)$ is the probability of sampling the optimal solution x^* to the original problem, ν a factor and N_q the number of qubits and p the number of layers. In particular, one can see that, if $N_q = p$, the probability of finding the optimal solution remains constant. In other words, if this conjecture holds true (and [29] provide empirical evidence), the probability of the optimal solution would remain constant independent of the problem size if the number of layers equals the number of qubits. However, this comes at the cost of an increasing circuit depth.

5.4 Global and Local Optimization

5.4.1 Cost Landscapes

The shape of the cost landscape for a given problem is influenced by various parameters that depend on the specific problem at hand and the characteristics of the algorithm being used. These parameters can significantly impact the performance of classical optimization methods, making it crucial to thoroughly examine their impact on the underlying cost landscape.



(a) Local optimization (b) Global optimization

Figure 7: Comparison of local and global optimizer on a random one layer QAOA instance.

The cost landscape represents the objective function or the cost function that needs to be optimized. The shape of this landscape can vary from smooth and convex (easy to optimize) to highly non-convex with multiple local minima or maxima (difficult to optimize). The presence of local minima or maxima can cause optimization algorithms, especially local optimizers, to get trapped and fail to find the global optimum.

In the context of quantum algorithms like QAOA (Quantum Approximate Optimization Algorithm) and VQE (Variational Quantum Eigensolver), the cost landscape can be influenced by factors such as the problem size, the choice of ansatz (the parameterized quantum circuit), and the specific problem instance being solved. Even for well-designed QAOA cost landscapes, optimization can remain challenging, particularly for local optimizers, as discussed in Section 5.

The parameters of most of the variational quantum algorithms covered in this guide can be optimized with a classical optimizer. The large class of optimizers can be seperated in local and global optimizers.

Global optimization algorithms aim to find the best solution by exploring the entire feasible search space, considering all possible combinations of variables and constraints. These algorithms are designed to locate the global minimum or maximum of a given objective function. However, global optimization techniques can be computationally expensive, especially for problems with large search spaces or complex constraints.

Local optimization algorithms on the other hand, focus on finding the best solution within a specific region of the search space, typically starting from an initial guess or point. These algorithms analyze the local behavior of the objective function near the starting point and iteratively improve the solution by moving towards a nearby minimum or maximum. Local optimization methods are generally faster and more efficient than global optimizers, as they do not explore the entire search space. However, they can get trapped in local minima or maxima, and their effectiveness heavily depends on the quality of the initial starting point.

A illustration of the two optimization regimes is depicted in figure 7. In both surface plots, the same loss landscape of a representative instance of a one layer QAOA is shown.

In the left plot, three randomly initiated local optimizers converge to three different minima, of which only one is the global minima. In the right plot, an instance of a global optimizer is depicted which converges to the global minima.

When choosing between global and local optimization algorithms, it is essential to consider the trade-off between the computational cost and the desired solution quality. Global optimization algorithms are more likely to find the global optimum, but can be computationally expensive, especially for large-scale problems. Local optimization algorithms are often more efficient but may converge to suboptimal solutions if the initial starting point is not well-chosen or if the objective function has multiple local minima or maxima.

5.5 Quantum Relax-and-Round (QRR)

QRR is a classical post-processing method, inspired by the classical relax-and-round algorithm, that might increase overall performance for certain problems, and can be applied on the bitstrings sampled from a quantum algorithm. It was first proposed by Maxime Dupont and Bhuvanesh Sundar [30], and was tailored especially for solving the MaxCut problem of certain graph types with QAOA.

The QRR algorithm can be summarized as follows:

1. Optimize a regular QAOA circuit at depth p (low, e.g., $p = 1$), with respect to the cost function of the problem (this step basically performs the regular QAOA algorithm).

2. Find the 2-point correlation matrix, $\mathbf{Z}$, by sampling the solution from the optimized QAOA circuit. The matrix entry at index (i, j), $\mathbf{Z}_{ij}$, can be calculated by $\mathbf{Z}_{ij} = (\delta_{ij} - 1)\langle \hat{Z}_i \hat{Z}_j \rangle$, where δ_{ij} is the Kronecker delta and $\langle \hat{Z}_i \hat{Z}_j \rangle$ is the measurement expectation of Pauli-Z on the i-th and j-th qubit. If the QAOA circuit has N number of qubits, the size of the $\mathbf{Z}$ matrix will be $N \times N$.

3. "Relax" step: Find the set of eigenvectors $\{\vec{z}\}$ of the correlation matrix $\mathbf{Z}$. From the $\mathbf{Z}$ matrix with size $N \times N$, we will obtain N eigenvectors.

4. "Round" step: Apply the sign function to the eigenvectors entry by entry, so round non-negative entries to +1 and others to -1. Repeat the process once more, with their sign flipped (round non-negative entries to -1, and others to +1) to cover for cases where the problem has a non-degenerate solution. This doubles the number of vectors from the "relax" step, from N to $2N$ eigenvectors.

5. From all eigenvectors obtained, check which eigenvector gives the best value when evaluated with the cost function of the problem. This best eigenvector is the solution bitstring of the QRR algorithm (the output of this algorithm is a single bitstring).

In QRR [30], the authors focus mainly on the approximation ratio as the comparison metric. The approximation ratio is calculated as a ratio of the cost function expectation value to the theoretical optimal value. They compare the expectation value from just the

QAOA (averaged from all sampled bitstrings of QAOA) versus the solution bitstring of QRR.

From an application-centric view, this comparison is not entirely fair. Notably, it hides the fact that one might find the optimal bitstring out of all sampled bitstrings from QAOA. QRR improves QAOA expectation value by a good margin. However, during the QAOA procedure, one can find some bitstrings sampled from QAOA to be a better solution than the QRR solution. This leads to the conclusion that QRR might bring a clear advantage is when QAOA failed to sample the optimal bitstring since the eigendecomposition and rounding process in QRR can retrieve the optimal bitstring nonetheless.

The idea of QRR also opens up a new possibility of using different post-processing techniques to increase the quality of the bitstrings sampled from QAOA. Since QRR is basically an eigendecomposition, it has a computational cost of roughly $\mathcal{O}(N^3)$. An interesting research outlook is to test other bitstring post-processing techniques that have similar (or even lower) computational costs and compare their performance to QRR. For example, randomly flipping 3 bits for each sampled bitstring from QAOA is a post-processing that also roughly has $\mathcal{O}(N^3)$ computational cost.

In theory, QRR can technically be applied to other quantum algorithms as well, e.g., VQE. However, its performance has to be evaluated on an algorithm-by-algorithm basis. Since it depends on the initial performance of the algorithm, it will also vary between different applications.

6 Decomposing and Transforming the Problem

Decomposition steps are crucial for the success of quantum computing in optimization for two reasons: First, quantum computers should only be used for tasks where they outperform classical ones. Typically, this applies only to specific subproblems. Second, as current quantum computers are still limited on the size of treatable problems, it is often not possible to encode optimization problems directly on quantum hardware. To this end, in this chapter two approaches are presented to tackle this problem. Both methods show potential of increasing the performance of optimization algorithms on noisy quantum computers.

- As a prerequisite, section 6.1 summarizes the connection between QUBO problems and the MaxCut optimization problem of dividing a given graph's vertices into two sets with a maximal number of connections between the sets.
- Section 6.2 presents a graph shrinking algorithm that applies to MaxCut problems and leverages correlations from either quantum or classical resources to recursively simplify the problem. Using classical subroutines, allows to shrink the problem until it can be encoded a quantum computer, while using quantum resources, like introduced in [31], [32], increases the performance significantly even though the problem could be encoded on the quantum computer immediately without any reduction in the decision variables.
- Then, section 6.3 introduces a circuit cutting algorithm in combination with the graph shrinking. The algorithm allows us to cut the quantum circuit in the middle, thus making it possible to run an optimization problem on a circuit that employs half of the qubits of the original algorithm. This circuit cutting comes however at the cost of running the algorithm more often compared to the uncut circuit.

6.1 Transformation from QUBO to weighted MAXCUT problem

Both the graph shrinking and the circuit cutting algorithms work with weighted MaxCut instances. Despite the simple structure of the MaxCut problem, it is a popular example of a combinatorial optimization problem since any quadratic unconstrained binary optimization (QUBO) problem can be transformed into a MaxCut problem.

MaxCut Formulation A MaxCut problem instance is defined by a weighted undirected graph $G = (V, E)$ with vertices $V = \{i\}$, edges $E = \{e\}$ and edge weights w_e. We denote the number of vertices by $n = |V|$. In MaxCut , the task is to find a subset of nodes that maximizes the weight of edges connecting the chosen node subset and its complement. Formally, we want to find a node partition $W \subseteq V$ such that the edge set $\delta(W) \coloneqq \{ij \in E \mid i \in W, j \in V \setminus W\}$ maximizes its weight defined as $\sum_{e \in \delta(W)} w_e$.

The weighted MaxCut problem can also be formulated as maximizing an integer

quadratic unconstrained cost function $C(\mathbf{z})$ in the form of

$$C(\mathbf{z}) = \frac{1}{2} \sum_{ij \in E} w_{ij}(1 - z_i z_j). \tag{15}$$

Here, $\mathbf{z} \in \{-1, 1\}^n$ and z_i indicates whether vertex i is in the subset W or not. Furthermore, $w_{ij} \in \mathbb{R}$ represent the edge weights.

Transformation QUBO to MaxCut For understanding the transformation from QUBO to MaxCut , we also introduce its binary formulation with $\mathbf{x} \in \{0, 1\}^n$:

$$C'(\mathbf{x}) = \sum_{ij \in E} w_{ij}\big(x_i(1 - x_j) + x_j(1 - x_i)\big) = \mathbf{x}^T \mathbf{W} \overline{\mathbf{x}}. \tag{16}$$

Here, $\mathbf{W} = \{\omega_{ij}\}$ is the weight matrix and $\overline{\mathbf{x}} = e - \mathbf{x}$, where e is the all-one vector in $\mathbb{R}^n$. It is possible to transform any QUBO problem with n decision variables to a MaxCut problem with $n + 1$ nodes. The description follows [33]. For this, we define the QUBO problem by an $n \times n$ QUBO matrix $\mathbf{Q}$ and the incidence vector $\mathbf{x} \in \{0, 1\}^n$ that contains the decision variables:

$$\mathbf{x}^T \mathbf{Q} \mathbf{x} = \sum_{ij} x_i Q_{ij} x_j = -\sum_{ij} x_i Q_{ij}(1 - x_j) + \sum_{ij} Q_{ij} x_j = \mathbf{r}^T \mathbf{x} - \mathbf{x}^T \mathbf{Q} \overline{\mathbf{x}} \tag{17}$$

with $r_j = \sum_i q_{ij}$. Notice that the right part on the left-hand side is already a MaxCut problem as defined by eq. (16), while we need to introduce an additional node in the MaxCut problem graph to include the left part. From this, we can deduce an equivalent MaxCut problem graph $G' = (V', E')$ with nodes $V' = 0, 1, ..., n$ and edges $E' = E \cup E^0$. These edges are defined as:

$$E = \{(i, j) : Q_{ij} \neq 0; \;\; i, j = 1, 2, ..., n; \;\; i \neq j\} \tag{18a}$$

$$E^0 = \{(0, i) : r_i \neq 0; \;\; i = 1, 2, ..., n\}, \tag{18b}$$

and we introduce the edge weights:

$$w_{ij} = \begin{cases} -Q_{ij}, & \text{if } (i, j) \;\in E \\ -r_j, & \text{if } (i = 0, j) \;\in E^0, \end{cases} \tag{19}$$

This graph is now equivalent to the QUBO problem, up to a constant factor $\alpha = \sum_i r_i$.

6.2 Graph Shrinking

Introduced in [31], [32], recursive QAOA (RQAOA) is an algorithm that calculates correlations from standard QAOA routine and uses these correlations to successively shrink the optimization problem. In addition, [34] and [35] introduce new means of computing correlations, namely from linear programming (LP) and semi-definite programming (SDP),

to use in shrinking algorithms for optimization problems. This can be used for shrinking a problem until it can be solved optimally with future quantum computers, that can only solve problems with limited size. But it is also significant to mention that it also introduces new classical approximation algorithms that make use of this shrinking routine to outperform its classical counterparts.

6.2.1 Means of computing correlations

The core of the algorithm is formed by the correlations between the decision variables of the MaxCut problem. Each edge $ij \in E$ in the problem graph is assigned a correlation $b_{ij} \in [-1, +1]$. Ideally, we want this correlation to be indicative of the correlation between variables in high-quality solutions of the MaxCut problem. In this case, a large negative (positive) correlation between two variables indicates that in good candidate solutions, these two variables take mostly opposite (equal) values on average. In the language of MaxCut , this translates to an edge predominantly being cut for negative correlations and not cut for positive correlations.

The shrinking algorithm works identically irrespective of the correlations that are used. However, for different situations, different means of computing correlations might be better suited to achieve the best possible results. [35] introduces three means of computing correlations here, namely, using LP and SDP relaxations, as well as QAOA. For the sake of concreteness, we will introduce the correlations calculated from QAOA in detail, while just referring to [35] for the other approaches.

For the QAOA correlation, we first follow the standard QAOA approach. We use a classical subroutine that optimizes the parameters $(\boldsymbol{\beta}, \boldsymbol{\gamma})$ such that the expectation value

$$F(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \langle \Psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) | \, H_C \, | \Psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) \rangle \tag{20}$$

is maximized. Thus, we hope to prepare a superposition of high quality candidate solutions.

As follows from eq. (15), for MaxCut the cost Hamiltonian H_C can be written as

$$H_C = \frac{1}{2} \sum_{ij \in E} w_{ij} (I - Z_i Z_j), \tag{21}$$

where I is the identity operator and Z_i is the Pauli-Z operator acting on qubit i. After maximizing the expectation value $F(\boldsymbol{\beta}, \boldsymbol{\gamma})$, correlations b_{ij} between nodes i and j connected by an edge can be defined as the expectation value with respect to the QAOA state:

$$b_{ij}^{\text{QAOA}} = \langle Z_i Z_j \rangle \equiv \langle \Psi(\boldsymbol{\beta}_{\text{opt}}, \boldsymbol{\gamma}_{\text{opt}}) | \, Z_i Z_j \, | \Psi(\boldsymbol{\beta}_{\text{opt}}, \boldsymbol{\gamma}_{\text{opt}}) \rangle \, , \tag{22}$$

where $\boldsymbol{\beta}_{\text{opt}}$ and $\boldsymbol{\gamma}_{\text{opt}}$ stand for the optimized parameters after maximization.

As we can see, these correlations measure the relationship between the spins Z_i and Z_j. If we measure $+1$ we conclude that the corresponding decision variables are in the same partition and in the case -1 in opposite partitions. Thus, by measuring the expectation value of low-energy states, we can make educated guesses about the relationship between two variables. Naturally, we measure the quality of the correlation by the absolute value of the correlation.

6.2.2 Shrinking procedure

MaxCut is particularly suitable for shrinking algorithms, as there is a natural way of reducing the problem such that the shrunk problem is still a valid MaxCut problem. However, extensions of the shrinking presented here can be applied to other problems as well [36].

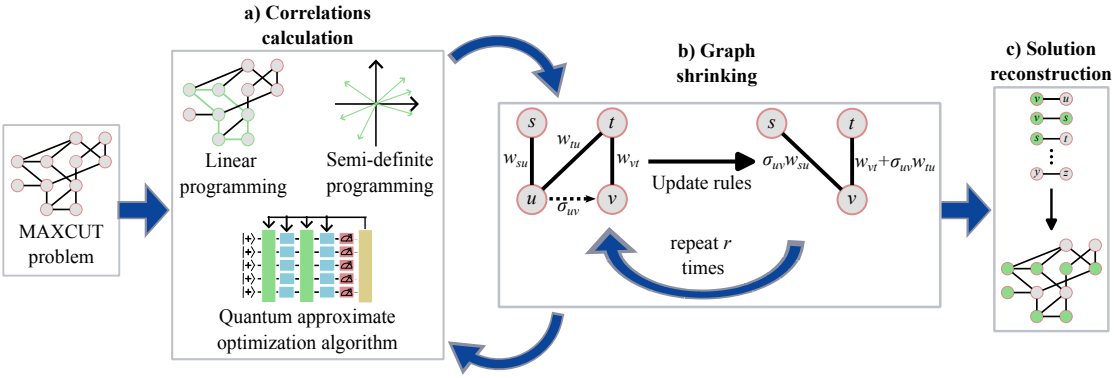


Figure 8: Depiction of the shrinking algorithm used in this work: First, the MaxCut problem instance is modeled according to the input requirement of the chosen routine for computing the correlations. In step a) we use the chosen routine to compute the correlations between variables. Then in b) the problem graph is shrunk to a smaller MaxCut problem by fixing a variable based on the calculated correlations and given update rules. The shrinking is repeated r times before a new set of correlations is calculated for the simplified graph. The steps a) and b) are applied in an alternating manner until the problem is fully simplified. In the final step c) the solution to the original problem is reconstructed based on the fixed variables. Figure taken from Ref. [35].

The shrinking procedure used in this work is similar to RQAOA [32], [37] as well as the algorithm introduced in [34]. For a better overview, a schematic of the algorithm is shown in fig. 8. In general, the problem can be divided into three major steps a) to c).

After modeling the MaxCut problem, we first start with calculating correlations between the decision variables of the problem in step a). As explained in detail in [35] and section 6.2.1, there are several ways of obtaining these correlations in polynomial time. However, as they rely on approximation algorithms, the solutions are typically not optimal and the performance of the approaches varies depending on the problem graph. Nevertheless, when using perfect correlations for every shrinking step, i.e., correlations obtained from an optimal cut of the problem, the shrinking algorithm is guaranteed to return the optimal cut.

Then in step b), we use these correlations to successively reduce the number of nodes in the problem graph. We do this by combining two nodes by fixing whether they are in the same or opposite partition, generating a new MaxCut instance that has one node fewer. We start the correlation with the largest absolute value and ties are broken randomly. This way, the algorithm first selects the edges that have the strongest tendency to be

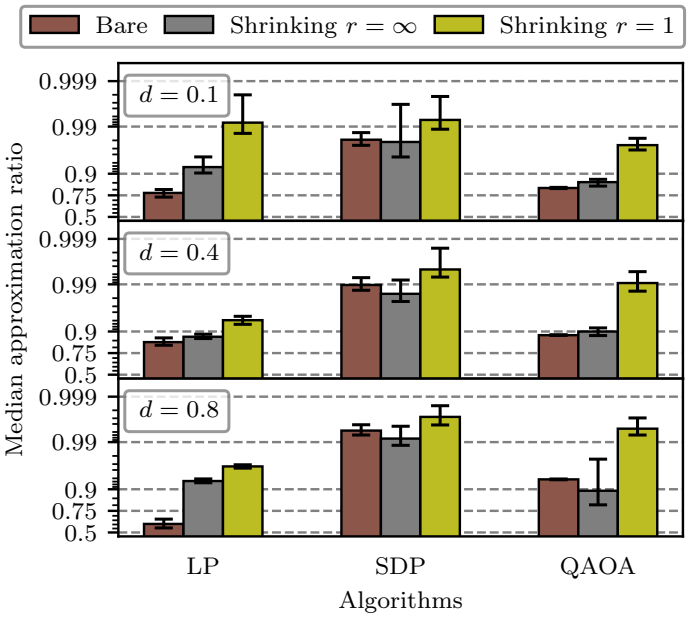


Figure 9: Median approximation ratio R_A of the bare LP, GW and QAOA ($p = 1$) algorithms as well as their shrinking counterparts with recalculation intervals $r = 1$ and $r = \infty$. The algorithms are applied to 80 different randomly generated Erdős-Rényi graphs of size 100 for each of the densities 0.1, 0.4 and 0.8. The lower and the upper error bars represent the first and third quartiles, respectively. Figure taken from Ref. [35].

cut or not cut. The reduced problem is equivalent to the original problem under the additional constraint implied by the correlation. Importantly, since the shrunk problem is again a valid MaxCut problem, we can calculate new correlations for the shrunk problem using the same method as for the original problem. This can be done either after every shrinking step or after a given number r of shrinking steps.

Lastly, in step c), we recreate a solution to the original instance from a solution to the shrunk instance by undoing the shrinking steps. Starting from a solution of the shrunk graph, one simply has to backtrack through the performed shrinking steps. At each shrinking step, we either add the shrunk node to the vertex partition or exclude it, depending on the sign of the shrinking.

6.2.3 Comparison to the underlying algorithms

Interestingly, the shrinking algorithm not only works as a way to shrink problems until they can be solved by quantum computers, but it also outperforms its classical, well established MaxCut approximation algorithms, from which the correlations are inspired.

For the LP bare algorithm we calculate the same correlations as for the shrinking but for rounding we use the spanning tree heuristic, which was also previously used in Refs. [38]–

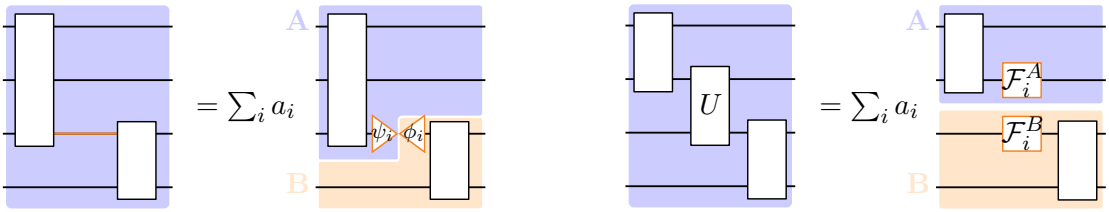
[40]. For SDP, we use the standard Goemans-Williamson algorithm [41] as the bare algorithm. For QAOA, the graphs are too large to perform a classical simulation of the standard quantum algorithm. Efficient sampling from the output state of a QAOA $p = 1$ circuit with one hundred qubits already exceeds the capabilities of classical hardware [42]. To this end, we calculate the expectation value of the energy returned by $p = 1$ QAOA using analytical formulae from [43].

In fig. 9 we see the comparison of shrinking with recalculating only once at the beginning ($r = \infty$) of the algorithm and recalculating every shrinking step ($r = 1$) to the bare algorithms. We applied the respective algorithms to randomly generated Erdős-Rényi graphs with 100 nodes with densities densities 0.1, 0.4 and 0.8.

Starting with LP, we observe that the shrinking with no recalculations outperforms the spanning tree heuristic for all of the considered graph densities, even though the correlations are identical in both cases. In addition, there is a clear improvement in performance when increasing the amount of recalculations performed (and, thus, decreasing r). This also holds true for SDP and QAOA. However, in contrast to SDP and QAOA, which show a roughly constant approximation ratio for all densities, LP performs significantly better at lower densities than on high densities. As shown in [35], when going to even lower densities, which are not shown in fig. 9, the LP correlations will outperform all other correlations, making them the best correlations for low-density MaxCut instances.

The bare GW algorithm and the SDP shrinking algorithm perform well across all densities. The shrinking algorithm with recalculation interval $r = 1$ performs best with median approximation ratios above 99% for all densities. Beyond the field of quantum optimization, this makes the shrinking algorithm with SDP correlations a new GW-inspired approximation algorithm, that manages to achieve better results than its original algorithm for most of the here considered problem instances. However, the bare GW algorithm slightly outperforms the shrinking algorithm with no recalculations.

Finally, for QAOA the shrinking algorithm with recalculation interval $r = 1$ (equivalent to RQAOA) sees a significant improvement over the bare algorithm: For the different densities, the median approximation ratio increases roughly from around 90% for the bare QAOA to approximately 99% for the $r = 1$ shrinking algorithm. This behavior is in line with previous results from the literature [31], [32]. Given that these results for a low-depth ($p = 1$) QAOA are similar to the GW algorithm, this suggest that the correlations from quantum resources manage to capture the relations between the decision variables and by going to higher depths we can expect to further improve the results. The improvement for going to higher depths has also been shown in [35] from smaller regular-3 graphs of 50 nodes, which makes this a promising approach to solve optimization problems with quantum resources. Interestingly, QAOA shows the largest relative improvement compared to the bare algorithm of the three different means of computing correlations. The bare algorithm and the shrinking with no recalculations perform quite similarly.



a) Wire cutting

b) Gate cutting

Figure 10: Schematic visualization of circuit cutting. a) Wire cutting: We replace the identity channel by a linear combination of measure-and-prepare channels. b) Gate cutting: We replace the corresponding unitary channel by a linear combination of local channels including measurements.

6.3 Circuit Cutting

As stated in previous sections, the application of quantum-computing algorithms to tasks such as combinatorial optimization is currently restricted to toy-sized problem instances. As a promising tool for bridging the gap to practical relevant quantum computing, graph shrinking was introduced in section 6.2. In this section, we describe another technique aimed at optimizing the use of available quantum computing resources, which is known as circuit cutting or circuit knitting [44]–[46]. We furthermore showcase the combination of graph shrinking and circuit cutting in the social network MaxCut use case introduced in section 3.4.

Circuit cutting is a technique to reduce the size of a quantum circuit by "cutting" gates or qubit wires that link different partitions. Once all connections have been cut, the partitioned circuits can be run independently on distinct, smaller hardware or even sequentially on the same quantum computer. Conceptually, circuit cutting can be viewed as a form of modular quantum computing where the entanglement shared among modules is substituted by classical communication links. Work done in the QuaST project has significantly advanced circuit-cutting methods [47], [48] and showcased several possible applications [49]–[51].

Mathematically, the quantum channel $\mathcal{F}$ representing a quantum gate or a qubit wire is written as a linear combination $\mathcal{F} = \sum_i a_i \mathcal{F}_i$, where each channel $\mathcal{F}_i$ acts locally on the partitions, as illustrated in fig. 10. Circuit cutting has shown promising results in applications such as chemical simulation [52], error mitigation [53], and combinatorial optimization [49], [54]. Despite these successes, application of circuit cutting still is restricted to toy instances. The reason is that circuit cutting incurs a sampling overhead, defined as the factor of more samples required to estimate an expectation value of an observable to the same accuracy as without circuit cutting. This sampling overhead per cut is in the order of κ^2 where the quantity $\kappa = \sum_i |a_i|$ is determined by the specific cutting method. The total sampling overhead is κ_{tot}^2, where κ_{tot} is the product of the individual κ of each gate or wire cut. Thus, the total sampling overhead grows exponentially in the number of cuts, which quickly becomes prohibitive if circuit cutting is performed naively.

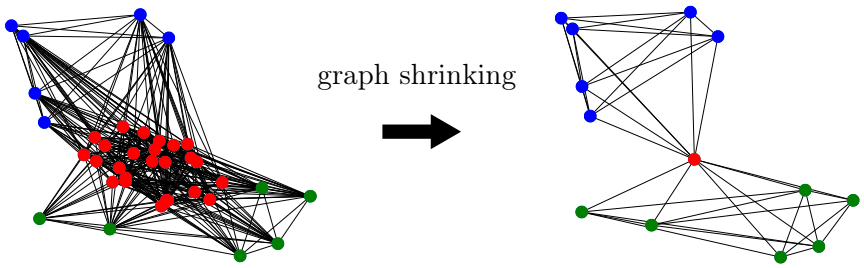


Figure 11: Modified shrinking procedure for generating graph with a small cardinality vertex separator and approximately equally sized partitions. We first calculate a vertex separator in the original graph (red nodes in the left graph) such that the resulting partitions (blue and green nodes) have approximately equal size. Then, we shrink the vertex separator to a specified target size K, illustrated on the right side for the case $K = 1$. Figure taken and modified from [49].

In a recent study [49], we utilized graph shrinking to reduce the number of cuts necessary to partition QAOA quantum circuits. As a result we reduced the sampling overhead of circuit cutting significantly. In the following, we illustrate this method at the social network graphs of section 3.4. Our approach involves a variant of the graph shrinking procedure introduced in section 6.2. This variant produces a graph, which can be partitioned into two vertex sets with approximately equal node counts such that the partitions are connected by a vertex separator of of cardinality K. Figure 11 illustrates this procedure for the case $K = 1$.

Although determining an appropriate shrinking sequence for the two partitions is NP-hard, practical solutions can be found quickly. The cardinality K is an input parameter to the shrinking algorithm. The shrinking process is illustrated in fig. 11 for $K = 1$.

In the second step, we map the shrunk graph to QAOA circuits. As shown in fig. 12, the size of the vertex separator corresponds to the number of wire cuts per layer [54] which are required to achieve independent partitions. Here, we use two different cutting techniques. Recently, the author of [55] have shown that $\kappa_{\text{nc}} = 4$ for an individual wire cut can be reduced to $\kappa_{\text{cc}} = 3$ if we allow classical communication. Here, circuit cutting with classical communication refers to a protocol where the operation performed on one partition depends on the outcome of a measurement performed on the other partition. In our work [49], we proposed to alternate the κ_{nc} and κ_{cc} cutting. This guarantees that classical information only needs to be communicated in one direction (one-way classical communication). Although two-way classical communication could further reduce the sampling overhead, this would necessitate two quantum computers (one for each partition) linked with real-time communication. In contrast, one-way communication allows for a sequential evaluation on the same device. After training the QAOA algorithm, for example by following section 5.3, we evaluate the solution by sampling bitstrings. Figure 13 compares the bitstring distribution for standard and cut QAOA circuits. The histograms of the standard QAOA peak at the right, indicating high-quality solutions. In contrast, the bitstring distributions obtained from the cut QAOA circuits are shifted towards smaller

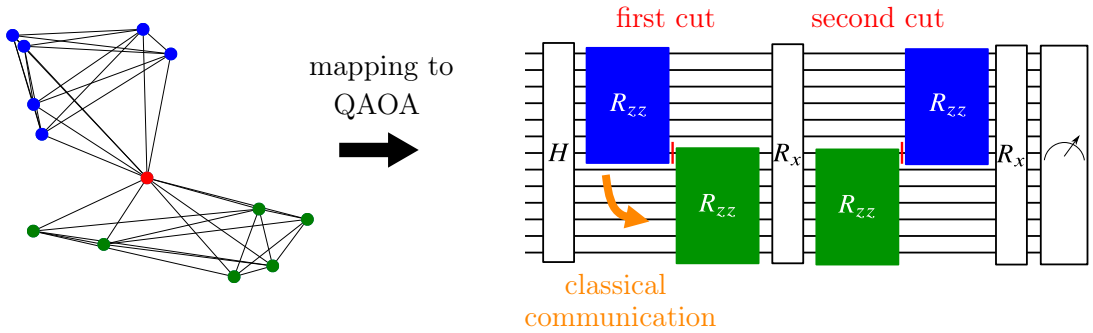


Figure 12: Mapping of a graph with a small cardinality vertex-separator to a QAOA circuit. The cardinality K of the vertex separator translates into the number of cuts needed per QAOA layer. The figure shows an example with $K = 1$. In order to reduce the sample overhead imposed by circuit cutting, we alternatingly employ two different cutting methods. Figure taken and modified from [49].

values. Thus, at first sight, circuit cutting seems not meaningful for sampling tasks. In the QuaST project, however, we derived quality guarantees on the bit string probability distribution [49], [54]. More precisely, in order to match the probability of obtaining a specific bitstring in the un-cut case, we must sample by a factor of κ_{tot} more often in the cut case. That is, if we need to draw N samples from the un-cut distribution to obtain a bitstring s with probability at least $1-\delta$, for some $\delta > 0$, we have to draw $\kappa_{\text{tot}} \cdot N$ samples from the uncut distribution to obtain bitstring s also with probability at least $1 - \delta$. For a two-layer QAOA with alternating cut type, we therefore find $\kappa_{\text{tot}} = \kappa_{\text{nc}}\kappa_{\text{cc}}$=12. We believe that these results form an initial step into a promising research direction that could bring optimization problems amenable to current or near-term quantum hardware. Future work will refine and advance these methods, helping to reduce the gap between today's experiments and practical applications of quantum computing.

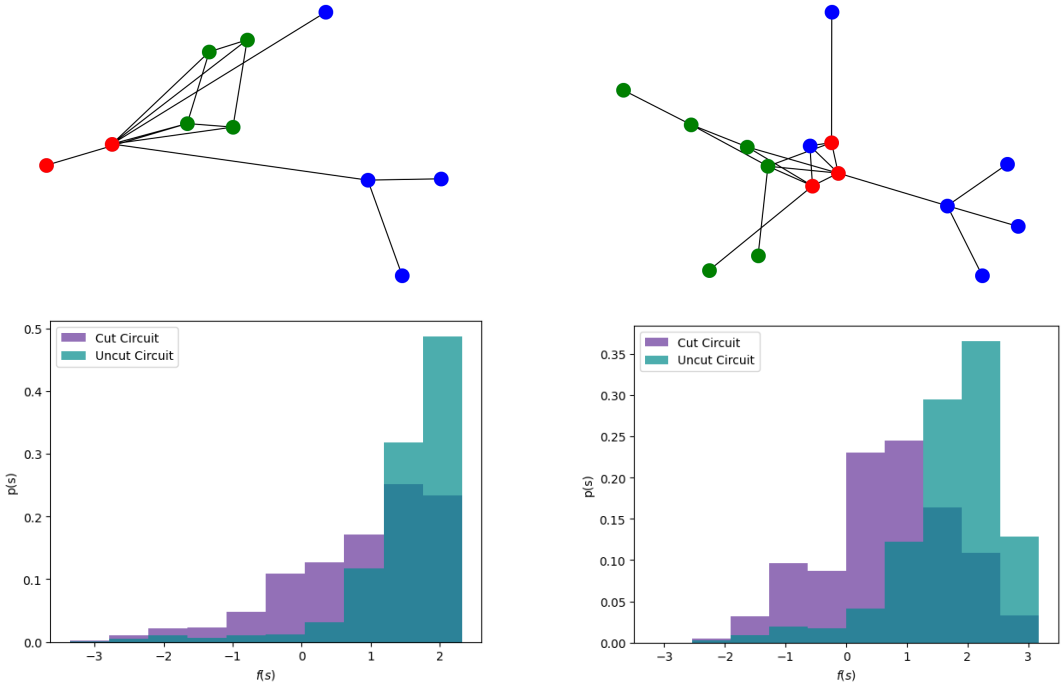


Figure 13: Experimental results for applying shrinking and cutting to two social network MaxCut instances from section 3.4. On the top row, we draw the original graphs having 10 (left) and 15 (right) vertices. In red, we highlight the vertex separator which is shrunk to a single vertex via graph shrinking. In the bottom row, we visualize the histograms resulting from the cut and uncut QAOA circuits. Here $f(s)$ denotes the normalized objective value of bitstring s (larger is better) and $p(s)$ denotes the corresponding probability.

7 Quantum Computing Hardware

Even from and end user side, or when doing research in quantum algorithms, one aspect cannot be left out: quantum hardware. There are different possible technologies to realize qubits, and even more manufacturers offering some form of experimental or even commercial access to their systems. How an optimization problem should be handled always depends on the quantum hardware as well. Some of these aspects are discussed in this section.

- Section 7.1 discusses the difference in research and evaluation results when using simulators or quantum hardware.
- Section 7.2 gathers the QuaST expertise in handling and evaluation quantum computers based on superconducting qubits, ion traps and neutral atoms as well as superconducting annealing systems. All technologies have distinct advantages and disadvantages, with direct comparisons currently made difficult due to the wide

spread in technological maturity especially of the access models.

7.1 Simulators and Quantum Hardware

In quantum computing, researchers and developers often rely on two distinct approaches to test and run quantum algorithms: simulators and real hardware. Simulators are classical computer programs designed to emulate the behavior of quantum systems. They provide a controlled environment where quantum circuits can be executed without external noise or disturbances and serve as an essential tool for designing, testing, and debugging quantum algorithms before running them on actual quantum hardware. However, they require keeping track of the state vector, which increases exponentially with the number of qubits. Thus, simulators are limited in the number of qubits one can simulate, see fig. 14.

Real quantum hardware consists of physical systems such as superconducting qubits or ion traps which are susceptible to noise and environmental disturbances. Noise can arise from various sources, such as imperfect gate operations, decoherence, and readout errors. The presence of noise can significantly impact the accuracy and reliability of quantum computations and can completely rule out any benefits from quantum computing [56].

Another difficulty of real quantum hardware compared to simulators arises when looking at variational quantum algorithms and their trainability. In simulators, techniques like backpropagation and the adjoint method can efficiently compute gradients and optimize variational quantum algorithms [57]. These techniques allow for the direct computation of gradients with respect to all trainable parameters of the quantum circuit. However, these methods are not possible on real quantum hardware. Instead, one relies on the parameter-shift rule [27] for gradient estimation, which requires 2 circuit evaluations for each parameter of the quantum circuit, which can be increasingly costly on real hardware and leads to inefficiencies.

In conclusion, simulators provide a noise-free environment for designing and testing quantum algorithms but are limited in the number of qubits one can simulate. Real hardware, despite being subject to noise and other unfavorable properties, is necessary for running large-scale quantum algorithms.

7.2 Different hardware options

The theoretical algorithms formulated to solve an optimization problem are often independent of the specific quantum hardware. This leads to the question regarding which hardware concept best suits a specific problem. However, the answer is not simply a matter of comparing result quality; it necessitates careful benchmarking considerations. We can categorize the hardware into two main types: one that employs algorithms described with gates and another that utilizes adiabatic processes to derive solutions. For gate-based quantum computers, the primary distinction among different systems lies in their hardware capabilities, qubit connectivity, and gate sets. These three factors will significantly influence the performance of the designed circuit on the respective hardware. For instance, in systems with low connectivity, minimizing long-range entanglement between qubits is advisable. Quantum Annealers, on the other hand, mimic an adiabatic

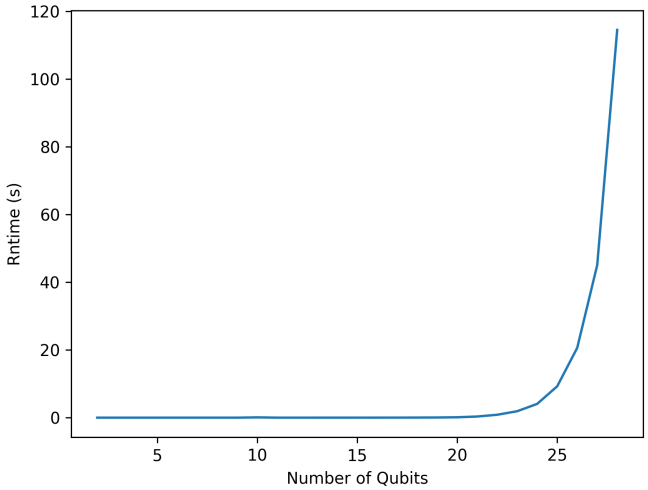


Figure 14: Runtime of a simulator (time needed for circuit execution) in seconds of a simple hardware-efficient quantum circuit with 5 layers as the number of qubits increases.

process evolving a quantum system into its ground state. Thus, they do not inherit a universally applicable gate set but are limited to solving certain types of optimization problems.

In the following sections, we will examine each type of hardware and its various providers, discussing access methods and best practices drawn from firsthand experience with quantum hardware.

7.2.1 Superconducting Qubits

A central advantage of quantum computers based on superconducting qubits is their execution speed. They deliver the fastest results for variational algorithms compared to the other quantum hardware tested (excluding annealing which cannot run variational algorithms in the usual sense). The mein drawback, however, is the limited connectivity of the qubits. When developing an ansatz for this hardware, it is crucial to consider how qubits are entangled. If carelessly designed, the SWAP gate overhead might undermine any chance of interpretable results because of noise.

For experiments on this hardware, a VQE algorithm is used in combination with a hardware efficent ansatz. The circuit comprises one layer of RY rotations and one entangling layer of CZ rotation where we skip the non-local gate between the first and last qubit to decrease the SWAP gate overhead (which was the best-performing architecture on the simulator). We also keep the circuit depth low for hardware noise considerations.

The two main players in superconducting hardware, IBM and IQM, provide cloud access to their systems. We access the most up to date hardware that would be able to solve our problem. Figure 15, compares both provider on multiple VQE runs over the

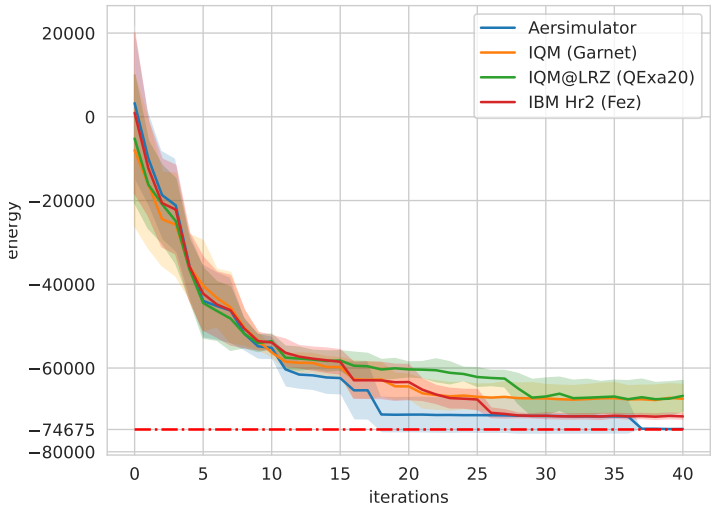


Figure 15: Comparison of IQM and IBM optimization process. Here, we differentiate between IQM access through the LRZ and their cloud access. The setup here is the same, i.e, same circuit (one layer of RY and one of CZ), same optimizer seed to make results as reproducible as possible.

same graph. The circuit is simply transpiled into the different basis gates defined by sets dependent on the hardware. Indeed, IQM has X and Y rotation as well as CZ for the entangling gate, whereas IBM Heron uses X and Z rotation gates. That makes the transpiled circuit slightly different for the two backends regarding gate count, but the depth stays the same in both cases. Two types of access were available for IQM, cloud and via the LRZ; both methods of access are presented here for comparison.

Following application-specific performance measure, Figure 16, illustrates the comparison in feasibility ratio, which represents the proportion of feasible TSP paths corresponding to the best result achieved during the optimization process, across the two hardwares. The AerSimulator feasibility ratio is simply here to illustrate the simulator baseline, which is comprising of only the optimal solution. This experiment is conducted without any error mitigation techniques and includes shot noise. However, the number of feasible states appears to correlate with the lower energy values attained by the IBM hardware.

Naturally, we also examined the runtime differences across all available hardware access options. The job timing was measured consistently by summing the runtime from when the job is submitted until the results are received. This includes any potential network or internal overhead from the provider. We chose this approach because each provider has a different method for measuring job execution times. From Figure 17, two interesting observations can be made. First, there is a noticeable difference between the jobs run on the cloud IQM access (Garnet) and the QExa20 system hosted at LRZ. This can be explained by the fact that for the cloud access, the QPU was reserved exclusively for our jobs, whereas on the LRZ machine, we were sharing the QPU with other users. Second,

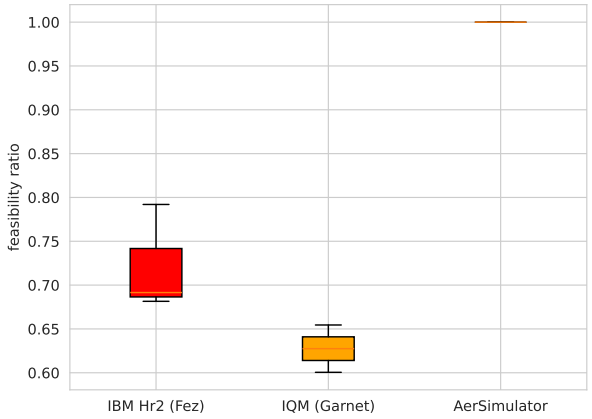


Figure 16: Comparison of IQM and IBM feasibility ratios at their respective final parameter state. AerSimulator shows a feasibility of 1.0 because the final state is only comprised of the optimal solution.

we observe a clear disadvantage for the IBM Heron 2, despite it being run in Session mode, which effectively locks the machine to only our jobs. A key factor here is that the QPU is hosted in North America, which increases network latency and contributes to longer runtimes. If these providers had more transparent timing data for their executions, we could better understand the reasons behind such significant differences between them.

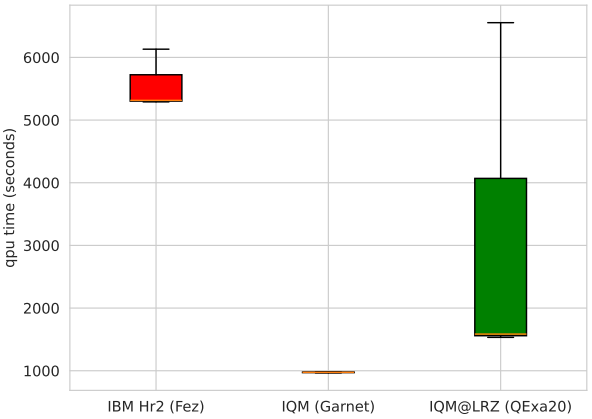


Figure 17: Runtime comparison between all three superconducting hardware

7.2.2 Ion traps

In comparison with superconducting qubits, ion-trap quantum computers generally have better noise behavior but are worse in speed. Additionally, compared to today's superconducting IBM systems, the number of qubits available is still more limited. While IBM offers systems with about 130 and 156 physical qubits, ion trap QCs typically have only 35 (IonQ, 2024) or 12 (AQT, 2024) available.

On the other hand, ion trap devices offer an ideal connectivity. AQT, for example, offers an all-to-all connection on their 12-qubit system. Hence, there is no restriction in generating entanglement between the qubits when running a quantum algorithm. In particular, this might be a significant advantage for solving problems corresponding to graphs with a high density.

This quality, in conjunction with a relatively low noise ratio, can lead to computation results that come close to noiseless simulation. Figure 18 shows a comparison between the AQT ion trap quantum computer IBEX™ and the Qiskit Aer simulator.

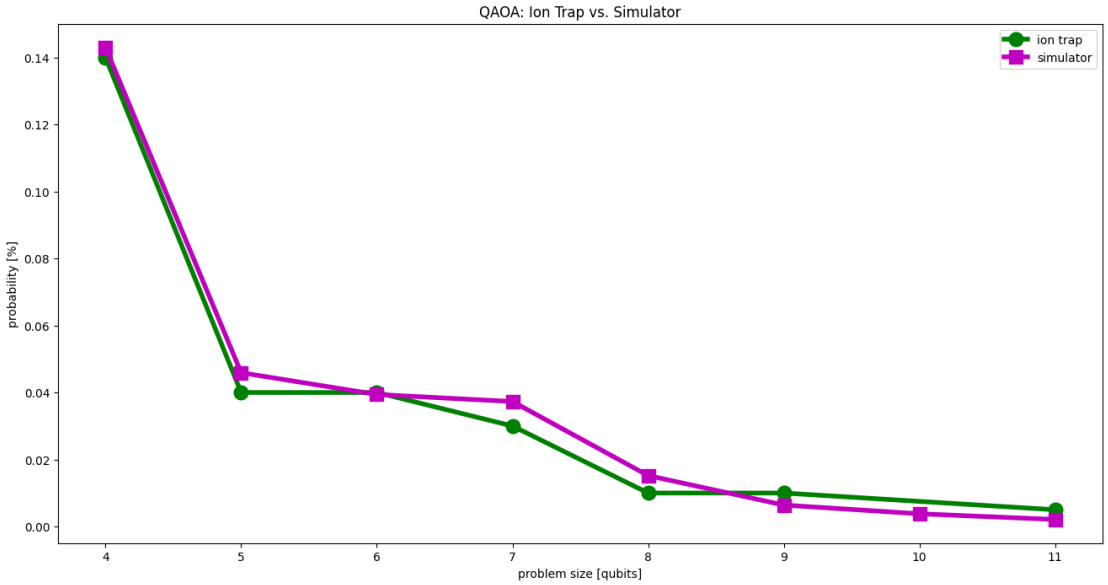


Figure 18: Probability of the optimal solution of a Markowitz optimization problem in dependence of problem size in qubits, performed by QAOA

7.2.3 Neutral atoms

Neutral atom devices are unique in how the algorithm can be implemented and encoded on the hardware. They operate in two computational modes. The first is a gate-based mode, where sequences of laser pulses are directed at specific qubits to create the gates. The second mode is analog, in which computation occurs by addressing the entire qubit array and gradually evolving the system toward the ground state of the target Hamiltonian through the design of suitable pulse shapes.

Another specificity of neutral atoms is encoding the qubits on the atom lattice. This is a critical part of the algorithm since the placement determines the connectivity. There is no universal encoding for general problems; each graph instance requires a different qubit arrangement. The idea is to minimize the interaction term from the underlying general Hamiltonian by efficiently placing the atoms on the lattice. Figure 19 gives an example placement for a problem requiring 4 qubits.

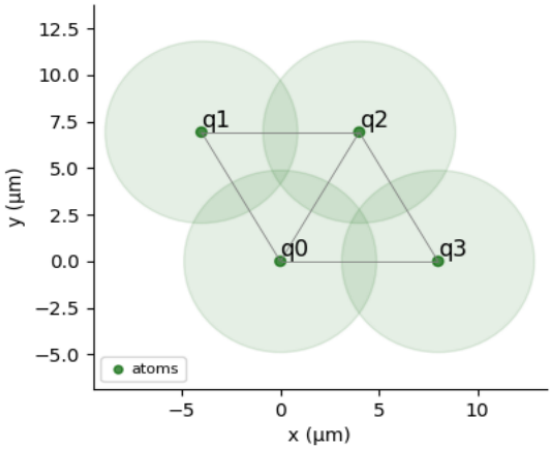


Figure 19: Encoding of 4 qubits on a triangular atom lattice in preparation for evaluating a PASQAL quantum computer

Pasqal's devices can only be operated in analog mode. Figure 20 gives an example of a pulse sequence. Here, two parameters, Ω, the amplitude, and δ, the detuning, are varied across 4 microseconds. Since we have no prior knowledge of what the pulse shape should be, we optimize it in a similar way as the variational circuits. We take 10 points (5 for each Ω and δ), and we interpolate between these points while a Bayesian optimization procedure finds the best point configuration (see Figure 21).

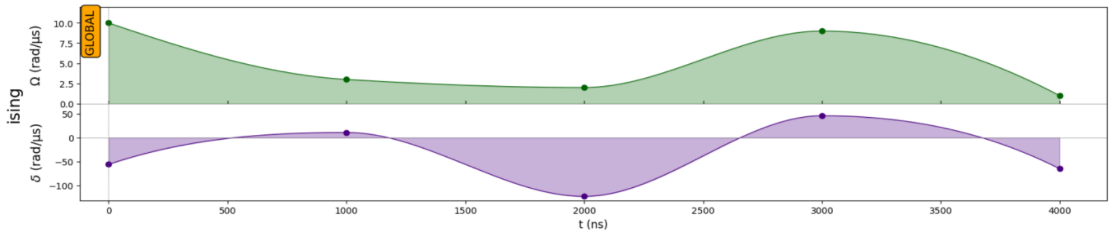


Figure 20: Best performing pulse sequence found by smoothly interpolating between points that act as the parameters we optimize for.

In Figure 22, we show results for a 3-node graph. However, the performance of this algorithm drops quite rapidly as we increase the number of nodes in our graph. There might be multiple reasons, beyond the hardware noise levels, for the subpar results compared to other hardware types. The first one might be the lack of digital mode, i.e., we cannot address individual qubits, which heavily restricts our algorithm choice. Secondly,

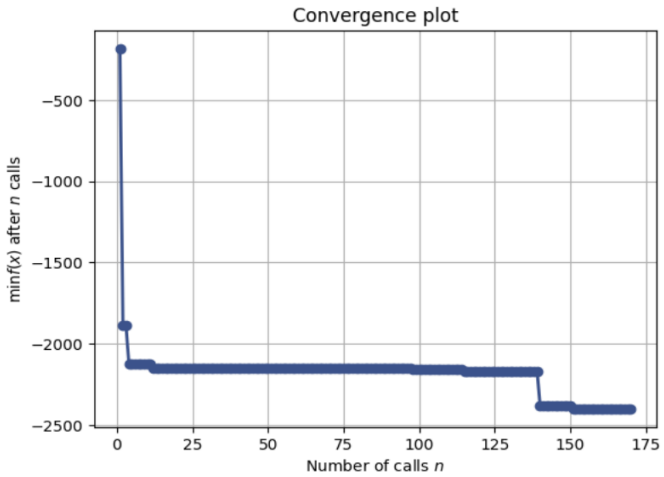


Figure 21: Bayesian optimization procedure solving a 3-node graph on Pasqal neutral atom hardware

the Pasqal QPU is locked, at the moment, to a triangular lattice shape, which means that we cannot freely optimize the placement of our atoms. Third, a 4000 nanoseconds cap for the time evolution on the hardware makes the results not as good as a simulation where this number can be increased. The slower the evolution, the better the results. Moreover, it also restricts us from how much we can vary the pulse shape because of the unphysical derivative values of the curve that might arise.

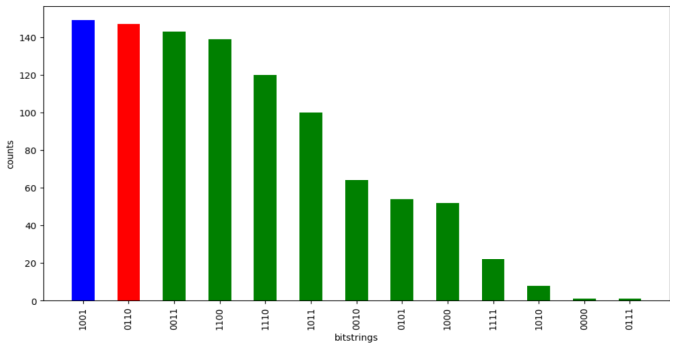


Figure 22: Sampled state after optimization of the pulse shape. Red and blue states are the optimal path and equivalent

7.2.4 Superconducting annealing

Quantum annealing is a computational method that uses quantum fluctuations to solve optimization problems. As an example, in fig. 23, the TSP is solved using D-Wave's proprietary methods. Due to the maturity of this hardware compared to the previous ones, we can solve larger problems. Figure 23 presents the difference between the best classically computed solution and the best solution found by the quantum processing

unit (QPU). Since this method is still based on sampling the quantum state to get the solution, running it multiple times increases the likelihood of hitting a solution closer to the optimal one. From graph sizes 5 to 9, the optimal solution is always reached. However, as we increase the number of nodes in the graph, it becomes increasingly harder to find the optimal solution (see 23).

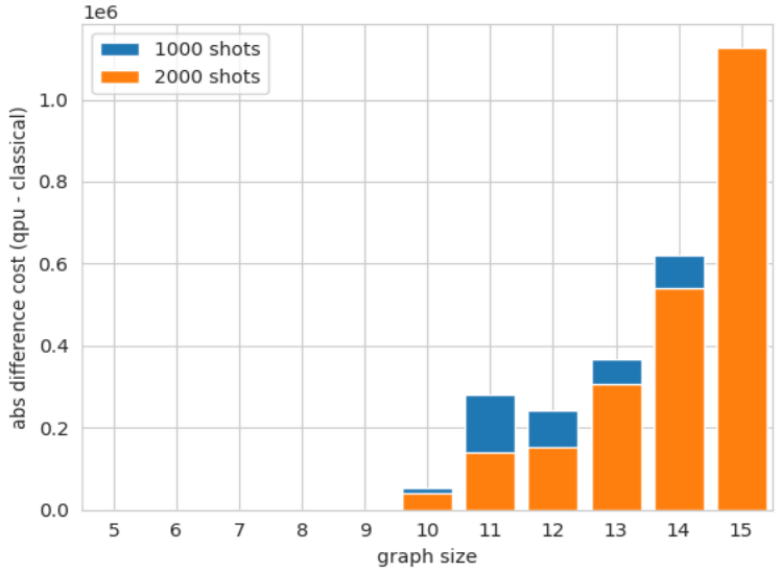


Figure 23: Difference between the best classical and quantum annealing solution. From 5 to 9 nodes, when sampling, the QPU finds the best solution for a TSP.

Another advantage of D-Wave's system, compared to other hardware, is the time to solution (see Figure 24), which seems to scale linearly with the graph size.

Moreover, another parameter we need to consider is the encoding of the problem to the physical qubits. Indeed, the QPUs of D-Wave's Advantage2™ do not possess all-to-all connectivity, which makes the embedding crucial for good performance. Our formulation of the TSP scales quadratically with the number of nodes in the graph. On top of that, the minor embedding, i.e., the encoding from variable to physical qubit, adds a power of two to the scaling, resulting in $(n-1)^4$ in the worst case, where n is the graph size. Figure 25 illustrates this point by looking at graphs of size up to 15.

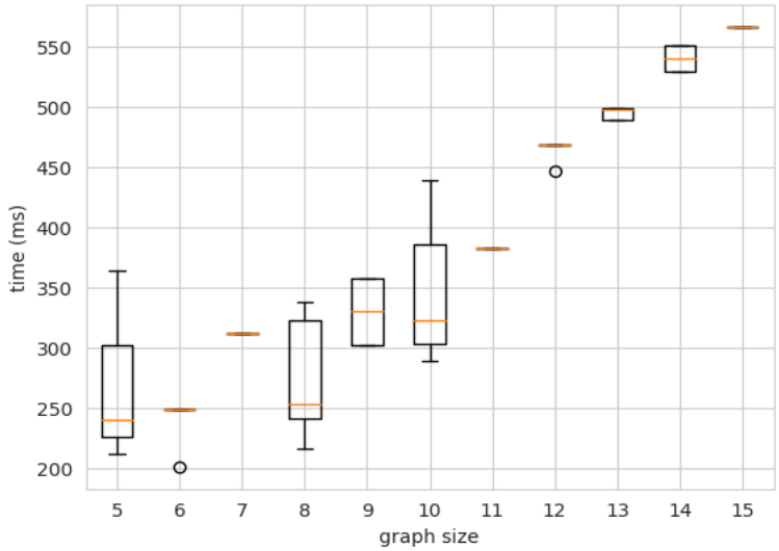


Figure 24: Runtime as we increase the number of nodes in the graph to solve

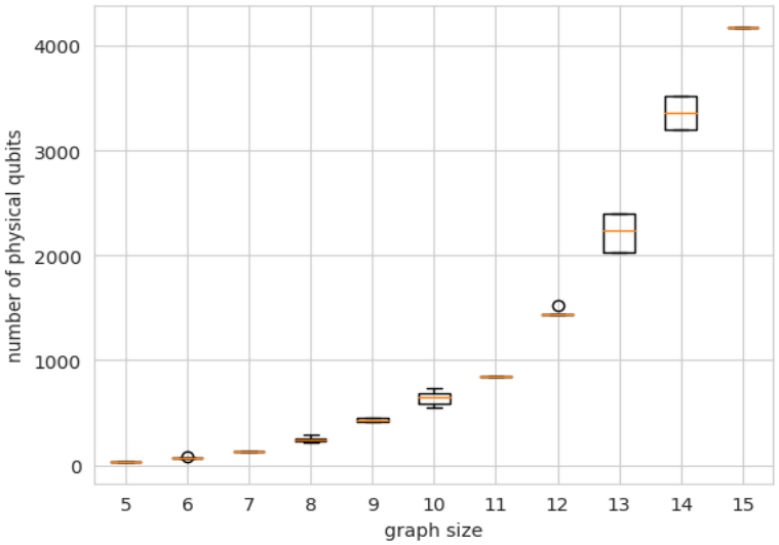


Figure 25: Number of physical qubits after minor embedding of our QUBO on the QPU

References

[1] A. Bayerstadler, G. Becquin, J. Binder, *et al.*, "Industry quantum computing applications," *EPJ Quantum Technology*, vol. 8, no. 1, p. 25, Nov. 2021, ISSN: 2196-0763. DOI: 10.1140/epjqt/s40507-021-00114-x. [Online]. Available: https://doi.org/10.1140/epjqt/s40507-021-00114-x.

[2] B. Poggel, N. Quetschlich, L. Burgholzer, R. Wille, and J. M. Lorenz, "Recommending Solution Paths for Solving Optimization Problems with Quantum Computing," in *2023 IEEE International Conference on Quantum Software (QSW)*, IEEE, Jul. 2023. DOI: 10.1109/qsw59989.2023.00017. [Online]. Available: http://dx.doi.org/10.1109/QSW59989.2023.00017.

[3] E. Farhi, J. Goldstone, and S. Gutmann, "A Quantum Approximate Optimization Algorithm," 2014. DOI: 10.48550/ARXIV.1411.4028. [Online]. Available: https://arxiv.org/abs/1411.4028.

[4] B. Poggel, X. Runge, A. Bärligea, and J. M. Lorenz, *Creating Automated Quantum-Assisted Solutions for Optimization Problems*, _eprint: 2409.20496, 2024. [Online]. Available: https://arxiv.org/abs/2409.20496.

[5] L. K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96, event-place: Philadelphia, Pennsylvania, USA, New York, NY, USA: Association for Computing Machinery, 1996, pp. 212–219, ISBN: 0-89791-785-5. DOI: 10.1145/237814.237866. [Online]. Available: https://doi.org/10.1145/237814.237866.

[6] L. Palackal, B. Poggel, M. Wulff, H. Ehm, J. M. Lorenz, and C. B. Mendl, "Quantum-assisted solution paths for the capacitated vehicle routing problem," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01, 2023, pp. 648–658. DOI: 10.1109/QCE57702.2023.00080.

[7] B. Denkena, F. Schinkel, J. Pirnay, and S. Wilmsmeier, "Quantum algorithms for process parallel flexible job shop scheduling," *CIRP Journal of Manufacturing Science and Technology*, vol. 33, pp. 100–114, 2021, ISSN: 1755-5817. DOI: https://doi.org/10.1016/j.cirpj.2021.03.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1755581721000432.

[8] H. M. Markowitz, "Foundations of portfolio theory," *The journal of finance*, vol. 46, no. 2, pp. 469–477, 1991.

[9] J. S. Baker and S. K. Radha, "Wasserstein solution quality and the quantum approximate optimization algorithm: A portfolio optimization case study," *arXiv preprint arXiv:2202.06782*, 2022.

[10] N. Leveson and C. Turner, "An investigation of the therac-25 accidents," *Computer*, vol. 26, no. 7, pp. 18–41, Jul. 1993, ISSN: 0018-9162. DOI: 10.1109/mc.1993.274940.

[11] H. G. Rice, "Classes of recursively enumerable sets and their decision problems," *Transactions of the American Mathematical Society*, vol. 74, no. 2, pp. 358–366, 1953, ISSN: 1088-6850. DOI: 10.1090/s0002-9947-1953-0053041-6.

[12] A. Glos, A. Krawiec, and Z. Zimborás, "Space-efficient binary optimization for variational quantum computing," *npj Quantum Information*, vol. 8, no. 1, p. 39, 2022, ISSN: 2056-6387. DOI: 10.1038/s41534-022-00546-y. [Online]. Available: https://doi.org/10.1038/s41534-022-00546-y.

[13] M. Schnaus, L. Palackal, B. Poggel, *et al.*, *Efficient encodings of the travelling salesperson problem for variational quantum algorithms*, 2024. arXiv: 2404.05448 [quant-ph]. [Online]. Available: https://arxiv.org/abs/2404.05448.

[14] Y. Zhang, K. Lu, Y. Gao, and M. Wang, "NEQR: A novel enhanced quantum representation of digital images," *Quantum Information Processing*, vol. 12, no. 8, pp. 2833–2860, Aug. 2013, ISSN: 1570-0755, 1573-1332. DOI: 10.1007/s11128-013-0567-z. [Online]. Available: http://link.springer.com/10.1007/s11128-013-0567-z (visited on 05/19/2024).

[15] P. Q. Le, F. Dong, and K. Hirota, "A flexible representation of quantum images for polynomial preparation, image compression, and processing operations," *Quantum Information Processing*, vol. 10, no. 1, pp. 63–84, Feb. 2011, ISSN: 1570-0755, 1573-1332. DOI: 10.1007/s11128-010-0177-y. [Online]. Available: http://link.springer.com/10.1007/s11128-010-0177-y (visited on 04/25/2024).

[16] S. Semmler, K. Dremel, D. Suth, *et al.*, "N-Dimensional Image Encoding on Quantum Computers," *e-Journal of Nondestructive Testing*, vol. 28, no. 3, 2023, ISSN: 14354934. DOI: 10.58286/27740.

[17] T. Lang, A. Heim, K. Dremel, *et al.*, *Representation of Classical Data on Quantum Computers*, 2024. DOI: 10.48550/ARXIV.2410.00742.

[18] A. Verma and M. Lewis, "Variable reduction for quadratic unconstrained binary optimization," *arXiv preprint arXiv:2105.07032*, 2021.

[19] J. Montanez-Barrera, D. Willsch, A. Maldonado-Romo, and K. Michielsen, "Unbalanced penalization: A new approach to encode inequality constraints of combinatorial problems for quantum optimization algorithms," *Quantum Science and Technology*, vol. 9, no. 2, p. 025 022, 2024.

[20] F. G. Fuchs, K. O. Lye, H. M. Nilsen, A. J. Stasik, and G. Sartor, *Constraint preserving mixers for QAOA*, 2022. DOI: 10.48550/ARXIV.2203.06095. [Online]. Available: https://arxiv.org/abs/2203.06095.

[21] F. Dominguez, J. Unger, M. Traube, B. Mant, C. Ertler, and W. Lechner, *Encoding-Independent Optimization Problem Formulation for Quantum Computing*, _eprint: 2302.03711, 2023.

[22] A. Peruzzo, J. McClean, P. Shadbolt, *et al.*, "A variational eigenvalue solver on a photonic quantum processor," *Nature communications*, vol. 5, no. 1, p. 4213, 2014.

[23] J. Tilly, H. Chen, S. Cao, *et al.*, "The variational quantum eigensolver: A review of methods and best practices," *Physics Reports*, vol. 986, pp. 1–128, 2022.

[24] W. van der Schoot, R. Wezeman, P. T. Eendebak, N. M. Neumann, and F. Phillipson, "Evaluating three levels of quantum metrics on quantum-inspire hardware," *Quantum Information Processing*, vol. 22, no. 12, p. 451, 2023.

[25] E. R. Anschuetz and B. T. Kiani, "Quantum variational algorithms are swamped with traps," *Nature Communications*, vol. 13, no. 1, p. 7760, 2022.

[26] M. Larocca, S. Thanasilp, S. Wang, *et al.*, "A review of barren plateaus in variational quantum computing," *arXiv preprint arXiv:2405.00781*, 2024.

[27] D. Wierichs, J. Izaac, C. Wang, and C. Y.-Y. Lin, "General parameter-shift rules for quantum gradients," *Quantum*, vol. 6, p. 677, 2022.

[28] E. Wybo and M. Leib, *Missing puzzle pieces in the performance landscape of the quantum approximate optimization algorithm*, 2024. eprint: `arXiv:2406.14618`.

[29] J. Montanez-Barrera and K. Michielsen, "Towards a universal qaoa protocol: Evidence of quantum advantage in solving combinatorial optimization problems," *arXiv preprint arXiv:2405.09169*, 2024.

[30] M. Dupont and B. Sundar, "Extending relax-and-round combinatorial optimization solvers with quantum correlations," *Phys. Rev. A*, vol. 109, p. 012 429, 1 Jan. 2024. DOI: `10.1103/PhysRevA.109.012429`. [Online]. Available: `https://link.aps.org/doi/10.1103/PhysRevA.109.012429`.

[31] S. Bravyi, A. Kliesch, R. Koenig, and E. Tang, "Hybrid quantum-classical algorithms for approximate graph coloring," en, *Quantum*, vol. 6, p. 678, Mar. 2022, ISSN: 2521-327X. DOI: `10.22331/q-2022-03-30-678`. [Online]. Available: `https://quantum-journal.org/papers/q-2022-03-30-678/` (visited on 08/10/2022).

[32] S. Bravyi, A. Kliesch, R. Koenig, and E. Tang, "Obstacles to Variational Quantum Optimization from Symmetry Protection," en, *Physical Review Letters*, vol. 125, no. 26, p. 260 505, Dec. 2020, ISSN: 0031-9007, 1079-7114. DOI: `10.1103/PhysRevLett.125.260505`. [Online]. Available: `https://link.aps.org/doi/10.1103/PhysRevLett.125.260505` (visited on 04/04/2023).

[33] A. P. Punnen, "The quadratic unconstrained binary optimization problem," *Springer International Publishing*, vol. 10, pp. 978–3, 2022.

[34] F. Wagner, J. Nüßlein, and F. Liers, "Enhancing quantum algorithms for quadratic unconstrained binary optimization via integer programming," *arXiv preprint arXiv:2302.05493*, 2023.

[35] V. Fischer, M. Passek, F. Wagner, J. R. Finžgar, L. Palackal, and C. B. Mendl, "The role of quantum and classical correlations in shrinking algorithms for optimization," *arXiv preprint arXiv:2404.17242*, 2024.

[36] J. R. Finžgar, A. Kerschbaumer, M. J. Schuetz, C. B. Mendl, and H. G. Katzgraber, "Quantum-informed recursive optimization algorithms," *PRX Quantum*, vol. 5, no. 2, p. 020 327, 2024.

[37] S. Bravyi, A. Kliesch, R. Koenig, and E. Tang, "Hybrid quantum-classical algorithms for approximate graph coloring," *Quantum*, vol. 6, p. 678, 2022.

[38] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt, "An application of combinatorial optimization to statistical physics and circuit layout design," *Operations Research*, vol. 36, no. 3, pp. 493–513, 1988.

[39] F. Liers, M. Jünger, G. Reinelt, and G. Rinaldi, "Computing exact ground states of hard ising spin glass problems by branch-and-cut," *New optimization algorithms in physics*, pp. 47–69, 2004.

[40] T. Bonato, M. Jünger, G. Reinelt, and G. Rinaldi, "Lifting and separation procedures for the cut polytope," *Mathematical Programming*, vol. 146, pp. 351–378, 2014.

[41] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM (JACM)*, vol. 42, no. 6, pp. 1115–1145, 1995.

[42] E. Farhi and A. W. Harrow, *Quantum supremacy through the quantum approximate optimization algorithm*, 2016. DOI: 10.48550/ARXIV.1602.07674. [Online]. Available: https://arxiv.org/abs/1602.07674.

[43] A. Ozaeta, W. Van Dam, and P. L. McMahon, "Expectation values from the single-layer quantum approximate optimization algorithm on Ising problems," *Quantum Science and Technology*, vol. 7, no. 4, p. 045 036, Oct. 2022, ISSN: 2058-9565. DOI: 10.1088/2058-9565/ac9013. [Online]. Available: https://iopscience.iop.org/article/10.1088/2058-9565/ac9013 (visited on 02/12/2024).

[44] H. F. Hofmann, "How to simulate a universal quantum computer using negative probabilities," *J. Phys. A: Math. Theor.*, vol. 42, p. 275 304, 2009. DOI: 10.1088/1751-8113/42/27/275304. [Online]. Available: https://dx.doi.org/10.1088/1751-811/42/27/275304.

[45] T. Peng, A. W. Harrow, M. Ozols, and X. Wu, "Simulating large quantum circuits on a small quantum computer," *Phys. Rev. Lett.*, vol. 125, p. 150 504, 2020. DOI: 10.1103/PhysRevLett.125.150504. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.125.150504.

[46] K. Mitarai and K. Fujii, "Constructing a virtual two-qubit gate by sampling single-qubit operations," *New J. Phys.*, vol. 23, p. 023 021, 2021. DOI: 10.1088/1367-2630/abd7bc. [Online]. Available: https://doi.org/10.1088/1367-2630/abd7bc.

[47] C. Ufrecht, M. Periyasamy, S. Rietsch, D. D. Scherer, A. Plinge, and C. Mutschler, "Cutting multi-control quantum gates with ZX calculus," *Quantum*, vol. 7, p. 1147, 2023. DOI: 10.22331/q-2023-10-23-1147. [Online]. Available: https://doi.org/10.22331/q-2023-10-23-1147.

[48] C. Ufrecht, L. S. Herzog, D. D. Scherer, *et al.*, "Optimal joint cutting of two-qubit rotation gates," *Phys. Rev. A*, vol. 109, p. 052 440, 2024. DOI: 10.1103/PhysRevA.109.052440. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.109.052440.

[49] L. S. Herzog, F. Wagner, C. Ufrecht, *et al.*, "Improving quantum and classical decomposition methods for vehicle routing," *arXiv:2404.05551*, 2024.

[50] P. Seitz, M. Geiger, and C. B. Mendl, "Multithreaded parallelism for heterogeneous clusters of qpus," *arXiv:2311.17490*, 2023.

[51] P. Seitz, M. Geiger, C. Ufrecht, *et al.*, "SCIM MILQ: An HPC quantum scheduler," *arXiv:2404.03512*, 2024.

[52] G. Gentinetta, F. Metz, and G. Carleo, "Overhead-constrained circuit knitting for variational quantum dynamics," *Quantum*, vol. 8, p. 1296, 2024. DOI: 10.22331/q-2024-03-21-1296. [Online]. Available: https://doi.org/10.22331/q-2024-03-21-1296.

[53] R. Majumdar and C. J. Wood, "Error mitigated quantum circuit cutting," *arXiv:2211.13431*, 2022.

[54] A. Lowe, M. Medvidović, A. Hayes, *et al.*, "Fast quantum circuit cutting with randomized measurements," *Quantum*, vol. 7, p. 934, 2023. DOI: https://doi.org/10.22331/q-2023-03-02-934. [Online]. Available: https://quantum-journal.org/papers/q-2023-03-02-934/.

[55] H. Harada, K. Wada, and N. Yamamoto, "Https://arxiv.org/abs/2303.07340," *arXiv:2303.07340*, 2023. DOI: https://doi.org/10.48550/arXiv.2303.07340. [Online]. Available: https://arxiv.org/abs/2303.07340.

[56] Y. Pan, Y. Tong, S. Xue, and G. Zhang, "Efficient depth selection for the implementation of noisy quantum approximate optimization algorithm," *Journal of the Franklin Institute*, vol. 359, no. 18, pp. 11 273–11 287, 2022.

[57] V. Bergholm, J. Izaac, M. Schuld, *et al.*, "Pennylane: Automatic differentiation of hybrid quantum-classical computations," *arXiv preprint arXiv:1811.04968*, 2018.