18^{th} International Conference on Space Operations, Montreal, Canada, 26 - 30 May 2025.

SpaceOps-2025, ID #267

QUARGS – Quantum Reinforced Ground Station Scheduling

Daniel Alexander Leidreiter[a], Andreas Petrak[b], Michael Alexander Anderle[c], Sven Prüfer[d*]

[a]*German Space Operations Center, daniel.leidreiter@dlr.de.*
[b]*German Space Operations Center, andreas.petrak@dlr.de.*
[c]*German Space Operations Center, michael.anderle@dlr.de.*
[d]*German Space Operations Center, sven.pruefer@dlr.de.*
[*]Corresponding Author

Abstract

Consider a constellation of n satellites in medium-earth orbit as well as k ground stations distributed over the Earth. Such satellites require semiregular contacts with mission control in order to upload telecommands or general purpose maintenances, which is done via ground stations. In order to plan ground station contacts for such a constellation it is necessary to calculate possible visibilities between satellite–ground station pairs, and then build a valid schedule from that data which satisfies all mission-specific constraints. This is a typical challenge for the mission planning system of a satellite constellation. In addition, there is often some kind of relevant optimization goal, for example fairness conditions or minimization of total number of contacts to have more time available for the actual missions objectives.

Depending on the constraints for such ground station contact plans as well as the size of n and k, this problem may be hard to solve optimally. In particular, if k is much smaller than n it may be non-trivial to find any solution at all. For such problems one often employs global optimizers such as Gurobi, CPLEx or SCIP, or problem-specific heuristic search algorithms that try to find a solution that is good enough for practical purposes.

While such approaches can be made to work for most real scenarios, classical solvers tend to struggle to find solutions in a reasonable time for very large satellite constellations. Due to this reason, we investigate potential applications of quantum computers to larger instances of this problem in this paper.

To this end, we developed a library called *QUARGS* (Quantum Reinforced Ground Station Scheduling) which allows solving examples for such ground station scheduling problems using different quantum algorithms. In this paper we compare and evaluate these different solutions against each other as well as against a classical solution using the global optimizer SCIP.

After an overview of similar problems in the literature we define the concrete problem including all constraints as well as possible optimization functions. Then we give a brief overview of quantum algorithms that may be used to solve such scheduling problems. This includes in particular quantum annealing (QA), quantum approximate optimization algorithm (QAOA), variational quantum eigensolvers (VQE), as well as evolutionary variational quantum eigensolvers (E-VQE). The latter in particular was implemented and published in an open-source Python library *QUEASARS* (Quantum Evolving Ansatz Variational Solver).

Besides evaluating these algorithms on sample cases regarding their solution quality and quantum circuit depths, we discuss difficulties for the implementation of various constraints in the algorithms and their potential solutions.

Keywords: Quantum Computing; Optimization; Scheduling; Planning

Acronyms/Abbreviations

E-VQE Evolutionary Variational Quantum Eigensolver
DLR Deutsches Zentrum für Luft- und Raumfahrt e.V.
DSN Deep Space Network
EA Evolutionary Algorithm
F-VQE Filtered Variational Quantum Eigensolver
GSOC German Space Operations Center
MEO Medium Earth Orbit
NISQ Noisy Intermediate Scale Quantum
PUBO Polynomial Unconstrained Binary Optimization
QAOA Quantum Approximate Optimization Algorithm
QCI Quantum Computing Initiative

QEOPS Quantum Earth Observation Planning System
QFT Quantum Fourier Transformation
QPU Quantum Processing Unit
QUARGS Quantum Reinforced Ground Station Scheduling
QUBO Quadratic Unconstrained Binary Optimization
QUEASARS Quantum Evolving Ansatz Variational Solver
SCIP Solving Constraint Integer Programs
SCOTA Spacecraft Orbit and Ground Track Analysis Tool
SPSA Simultaneous Perturbation Stochastic Approximation
SQOS Spacecraft Quantum On-Call Scheduling
VQA Variational Quantum Algorithm
VQE Variational Quantum Eigensolver

1. Introduction

1.1. Ground Station Scheduling

Ever-growing satellite constellations such as Starlink [1], OneWeb [2], GPS [3], or Galileo [4] require an increasing number of ground stations to ensure reliable communication with each satellite. Ground station scheduling is a critical task that aims to optimize the allocation of resources, such as time slots and frequency bands, to meet various objectives like minimizing latency, maximizing throughput, or reducing energy consumption.

The concrete ground station scheduling problem depends very much on the precise mission requirements, so a fully general treatment is implausible. In this paper we consider the special case of a large MEO constellation with only few ground stations and which involves some manual operations, hence requiring some consideration with respect to accumulations of large sets of ground station passes at the same time among which an operator has to split their attention. Satellites in Medium Earth Orbit are typically part of a GNSS constellations, and have rather large visibility windows for a fixed ground station. In our model we consider these visibilities as precomputed by another software, such as e. g. SCOTA (see [5]), and use them as input. The scheduling problem is hence concerned with assigning a set of passes during these visibilities to ground stations and satellites abiding by all relevant constraints.

There are many approaches known in the literature which can be used to solve this problem, including classical optimization techniques (e.g., genetic algorithms, simulated annealing), heuristic approaches, or quantum computing. While many of them come with certain advantages, one typically encounters limits in scalability as most algorithms either take an enormously long time to finish or do not find a satisfactory solution within a fixed time frame. Quantum annealing, on the other hand, has been shown to be a promising approach due to its ability to explore vast solution spaces efficiently. In this paper we focus on certain quantum optimization algorithms (both annealing and gate-based) and compare them with a classical optimizer (SCIP) for solving our discrete optimization problem.

Most notably, there is the work of [6], which uses quantum annealing to solve a scheduling problem similar to ours. However, their approach is tailored towards the Deep Space Network (DSN) and does not consider the specific constraints of our example, in particular the maximum concurrent contacts or the maximum contact separation constraints. In [7] the authors present a multi-objective optimization approach to ground station scheduling, which is more general than our problem, however, their constraints differ from ours. Besides the multi-objective goal, the authors try to apply (classical) evolutionary algorithms. Furthermore, there is [8] which establishes a mixed-integer linear model for the Dove constellation by Planet, taking into account various non-overlap constraints for activities including a model for the battery state of the satellite. However, their model is very much adapted to their specific use case as the optimization via simulated annealing is done in steps to keep the complexity manageable. Still, the paper presents a very interesting and realistic approach to satellite constellation scheduling.

Regarding our problem version, the variables, constraints, and relevant reformulations for the investigated quantum algorithms are described in detail in Sections 2 and 3. Afterwards, we describe these algorithms and their caveats in Section 4. Notice that this is a huge field, and we are not able to go into all details, so we focus on a broad overview. Results of the evaluation of these algorithms on real-world data as well as some lessons learned are presented in Section 5 and the raw data is included in Section 5.3.

1.2. Acknowledgments

This project was made possible by the DLR Quantum Computing Initiative and the Federal Ministry for Economic Affairs and Climate Action; `https://qci.dlr.de/en/qmpc/`.

2. Discrete Problem Specification

As the visibilities of satellites and ground stations in MEO are in the order of hours, the scheduling problem is naturally of a mixed-integer type, where start and end times of contacts are continuous variables. In order to apply quantum optimization algorithms, the continuous variables need to be discretized, hence we define a discrete ground station scheduling problem based on binary variables in the following. To do so, we subdivide the available passes for each ground station and satellite into multiple fixed-length contact slots. Note, that ground station and satellite outages as well as visibility constraints are easily incorporated, by simply not including contact slots during times restricted by such constraints.

The discrete ground station scheduling problem then aims to find a contact plan P for a set of ground stations G and a set of satellites S, during a time horizon T, so that all constraints are satisfied, and some optimization goal is minimized. Note that the contact plan P is a subset of all available contact slots C.

2.1. Variables

The discrete ground station scheduling problem is expressed in binary decision variables x_i, one for each contact slot $c_i \in C$. They individually represent, whether a contact slot has been selected to be scheduled. As a result the combined state X of all decision variables x_i expresses an attempted, but not necessarily valid, contact plan P. Going forward we will use x_i to refer to both to the binary decision variable and the contact slot c_i it represents. We will further assume, that the variables and contact slots are ordered and indexed in such a way, that $i < j$ implies $x_i\,.\,\mathrm{start} < x_j\,.\,\mathrm{start}$.

Table 1. Definitions of used variables in this paper

Term	**Meaning**
C	Set of all contact slots
$C(s)$	Set of all contact slots for the satellite s
c_i	Contact slot that may or may not be planned
x_i	Binary decision variable AND shorthand for c_i
X	Vector of all x_i decision variables
T	Planning time horizon
$T\,.\,\mathrm{start}$, $T\,.\,\mathrm{end}$	start / end time of the planning time horizon
$x_i\,.\,\mathrm{start}$, $x_i\,.\,\mathrm{end}$	start / end time of the contact slot c_i

2.2. Constraints

For an attempted contact plan to be valid, the combined state X needs to satisfy the following list of constraints.

2.2.1. Forbidden Combination Constraints

There are multiple constraints which impose that certain passes cannot be scheduled simultaneously. This includes an *interference constraint* which forbids contacts from being scheduled, if their communication in certain frequency bands would interfere with each other. It also includes the *minimum separation constraint*, which forbids contacts from being scheduled for a satellite, which are closer together than some fixed time difference, similarly for ground stations. In particular, every satellite and ground station can only have a single contact scheduled at every point in time. For such constraints let us define $\mathrm{forbidden}(x_i, x_j)$ to be true if x_i and x_j may not be scheduled together and false otherwise. Then the following needs to hold:

$$\forall x_i, x_j \in C \wedge i < j : (x_i \wedge x_j) \implies \neg\,\mathrm{forbidden}(x_i, x_j)$$

2.2.2. Maximum Concurrent Contacts Constraint

Another constraint is that at any point in time during the contact schedule no more than m contacts may be scheduled at once. Let's define $t \in T$ to be any point in time during the problem's time horizon. Let's further define $\mathrm{during}(t, x_i)$ to be true, if $x_i\,.\,\mathrm{start} < t < x_i\,.\,\mathrm{end}$. Then it needs to hold that

$$\forall t \in T : \left(\sum_{x_i \in C} x_i \wedge \mathrm{during}(t, x_i) \right) \leq m,$$

see Figure 1.

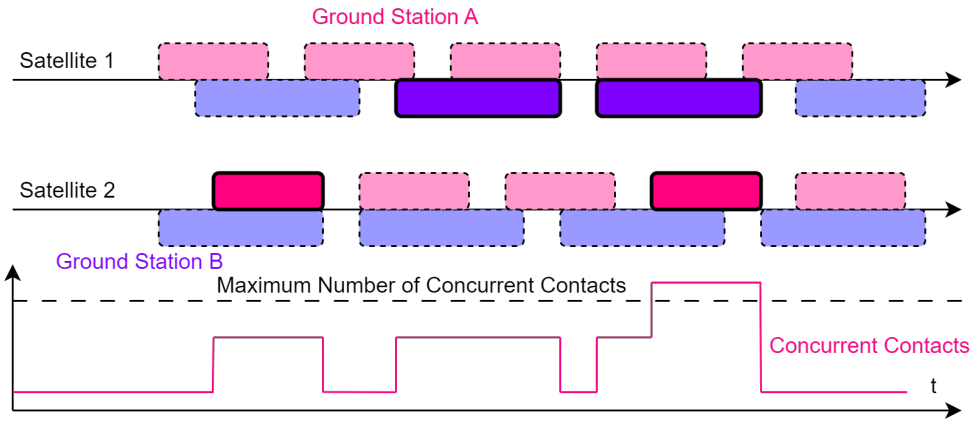


Fig. 1. In case of an upper bound of $m = 1$ for concurrent contacts, then the shown contact schedule is invalid.

Note that all subsets of contact slots that are mutually overlapping in time can be precomputed, denote the set of all such subsets as $\mathrm{overlapping(C)}$. As it is clearly enough to check this constraint for such subsets which contain more than m contact slots, we can implement the constraint by finitely many conditions

$$\forall O \in \mathrm{overlapping}(C) \wedge |O| > m : \left(\sum_{x_i \in O} x_i \right) \leq m.$$

2.2.3. Maximum Contact Separation Constraint

One more important constraint poses, that any two planned contacts of a satellite may not be further apart than some maximum separation $\mathrm{max_{sep}}$, so that telemetry and telecommands may be exchanged frequently. This is equivalent to requiring that for any sub interval of length $\mathrm{max_{sep}}$ within the time horizon T there must be at least one contact scheduled for that satellite. To this end we define a *sliding window* $\mathrm{following}(x_i)$ as the subset of contact slots for the same satellite as x_i which start within $(x_i.\,\mathrm{end}, x_i.\,\mathrm{end} + \mathrm{max_{sep}}]$. Note that there are finitely many such sliding windows, for each one we require

$$\forall s \in S\ \forall x_i \in C(s) : \left(\sum_{x_j \in \mathrm{following}(x_i)} x_j \right) \geq 1.$$

We add two such conditions at the beginning and the end of the planning horizon, respectively, and consider the above constraint only for those intervals which are fully contained within the planning horizon, see Figure 2.

2.3. Optimization Goal

For the discrete ground station scheduling problem, various optimization goals can be defined. A very simple optimization goal is to maximize the amount of scheduled contacts, which is equivalent to minimizing the cost function

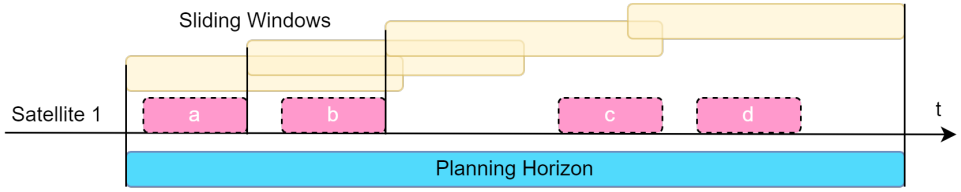


Fig. 2. For every satellite we define sliding windows beginning at the end of possible passes as well as starting at the beginning and the end of the planning horizon. The constraint requires that at least one contact must be scheduled within each sliding window.

$$f(X) = -\sum_{x_i \in C} x_i$$

A more realistic optimization goal is to distribute the contact while trying to adhere as closely as possible to a satellite specific target separation length t_s between consecutive contacts. Let $\mathrm{distances(X, s)}$ now yield the set of all time distances between consecutively planned contacts. Then this yields the more realistic optimization function:

$$f(X) = \sum_{s \in S} \left(\sum_{d \in \mathrm{distances}(X,s)} (t_s - d)^2 \right) \tag{1}$$

Notice that this more realistic goal is also expressible via quadratic constraints or cost functions, but this might require very large sums. In this paper we mostly focus on the maximization of scheduled contacts.

3. Unconstrained Cost Function

While classical global optimizers can typically deal with these constraints directly, most quantum algorithms require an *unconstrained* optimization problem. Hence, we add penalty terms to the cost function penalizing constraint violations.

3.1. Encoding the Problem as QUBO

For the Quantum Annealing algorithm, the optimization problem needs to be in the Quadratic Unconstrained Binary Optimization (QUBO) form:

$$f(X) = \sum_i q_i x_i + \sum_{i<j} q_{i,j} x_i x_j,$$

where q_i are linear coefficients and $q_{i,j}$ are quadratic coefficients, X is a vector of binary variables. Of course, the linear term can also be included into the quadratic one since we want to find the minimum of this function only. Now, we need to transform our constraints into polynomials of maximum degree 2 and sum them up together with an objective function (as described in Section 2.3) in QUBO formulation to get the final QUBO. In order to simplify the resulting terms, we will ignore constants, since they do not influence the argmin we are searching for, and we will make use of the fact that $x_i^2 = x_i$ for $x_i \in \{0, 1\}$.

3.1.1. Forbidden Combination Constraint

As mentioned in Section 2.2.1 there are multiple reasons restricting the scheduling of specific precomputed combinations of contact slots. We construct a QUBO by penalizing the concurrent scheduling of such combinations

$$\sum_{i<j} a_{i,j} x_i x_j \qquad \text{with } a_{i,j} = \begin{cases} 1, \text{ if } \mathrm{forbidden}(x_i, x_j) \\ 0, \text{ otherwise} \end{cases}.$$

3.1.2. Maximum Concurrent Contacts Constraint

Suppose we have precomputed subsets of mutually overlapping contact slots $O \in \text{overlapping}(C)$ as discussed in Section 2.2.2. For each of these subsets O the number of scheduled contacts $n = \sum_{x_i \in O} x_i$ may not exceed the maximum allowed number of concurrent contacts m, i. e. $n \leq m$. However, since QUBOs cannot directly represent inequality constraints, we have to model this using a slack variable $s \in \mathbb{N}$,

$$(n - m + s)^2 = 0,$$

where $s \geq 0$ is a slack variable. Using a binary encoding for $s = \sum_i 2^i s_i$ with $m \leq \sum_i 2^i < 2m$, we can represent the slack variable s using the $s_i \in \{0, 1\}$ as qubits. Since $s \geq 0$, the function on the left side can take its minimum 0 if and only if $n - m \leq 0$.

In order to compute $(n - m + s)^2$, we first perform some pre-calculations. As mentioned above, we use $x_i^2 = x_i$ and $s_i^2 = s_i$ to obtain

$$n^2 = \sum_i x_i + 2\sum_{i<j} x_i x_j \qquad s^2 = \sum_i 2^{2i} s_i + \sum_{i<j} 2^{i+j+1} s_i s_j$$
$$2ns = \sum_{i,j} 2^{j+1} x_i s_j \qquad 2ms = m\sum_i 2^{i+1} s_i$$

Now, we can compute

$$\begin{aligned}(n - m + s)^2 =& n^2 + m^2 + s^2 - 2nm + 2ns - 2ms \\ =& \sum_i x_i + 2\sum_{i<j} x_i x_j + m^2 + \sum_i 2^{2i} s_i + \sum_{i<j} 2^{i+j+1} s_i s_j \\ & - 2m\sum_i x_i + \sum_{i,j} 2^{j+1} x_i s_j - m\sum_i 2^{i+1} s_i.\end{aligned}$$

Again, we ignore the constant m^2 and obtain the quadratic function to be minimized

$$(1 - 2m)\sum_i x_i + \sum_i \left(2^{2i} - 2^{i+1}m\right) s_i + 2\sum_{i<j} x_i x_j + \sum_{i<j} 2^{i+j+1} s_i s_j + \sum_{i,j} 2^{j+1} x_i s_j.$$

3.1.3. Maximum Contact Separation Constraint

As seen in Section 2.2.3, the maximum contact separation constraint consists of checks that over sets of contact slots in certain time intervals at least one has to be scheduled. Let I be a time interval and $n = \sum_{x_i \in I} x_i$ be the number of contact slots (for a given satellite) that are scheduled in I. For such an interval I it has to hold that $n \geq 1$. Rephrasing this using a slack variable s yields

$$(n - 1 - s)^2 = 0.$$

Using the calculations from above, we get the function to be minimized

$$-\sum_i x_i + \sum_i \left(2^{2i} + 2^{i+1}\right) s_i + 2\sum_{i<j} x_i x_j + \sum_{i<j} 2^{i+j+1} s_i s_j - \sum_{i,j} 2^{j+1} x_i s_j$$

Notice that there are various cases in which we can simplify this considerably, e. g. when there are only one or two possible contact slots within the interval.

3.2. Encoding the Problem as PUBO

Both QAOA and E-VQE require the optimization problem in the form of a Hamiltonian, whose minimal eigenvalue is being approximated by these algorithms. Well known mappings from QUBOs to Hamiltonians also apply to Polynomial Unconstrained Binary Optimization problems (PUBOs), allowing for higher degree monomials in the

objective function. Expressing the cost function as a PUBO and subsequently translating this to a Hamiltonian allows to reduce the amount of slack variables and hence needed qubits, which are very limited on current gate-based quantum computers.

A PUBO problem has the form

$$f(X) = \sum_i q_i x_i + \sum_{i<j} q_{ij} x_i x_j + \sum_{i<j<k} q_{ijk} x_i x_j x_k + \dots$$

where $q_i, q_{ij}, q_{ijk}, \dots$ are the coefficients and X is a vector of binary variables. As any valid QUBO is also a valid PUBO, we do not need to reformulate the *Forbidden Combination Constraint*, as it does not need any slack variables.

3.2.1. Maximum Concurrent Contacts Constraint

Translating inequality constraints into a PUBO while avoiding slack variables, we can directly penalize any combination of contact slots where $n > m$. Let $\mathrm{overlapping}(C)$ again be the set of all mutually overlapping contact slot combinations, then we can penalize any illegal set using

$$\sum_{\substack{O \in \mathrm{overlapping}(C) \\ |O| > m}} \prod_{x_i \in O} x_i.$$

3.2.2. Maximum Contact Separation Constraint

Similar to the previous constraint we again can directly penalize any invalid combination of scheduled contact slots in the PUBO. To that purpose we find for any satellite any combination where two contact slots x_i and x_j are scheduled with no other slot scheduled between them and their separation exceeds $\max_{\mathrm{sep}}$. Similarly, at the boundary of the planning horizon within $\max_{\mathrm{sep}}$ of T's edges we need to penalize the case when no contact slots are scheduled for a satellite.

Let's define $\mathrm{between}(x_i, x_j, s)$ to yield all $x_k \in C(s)$ where $x_i\,.\mathrm{start} < x_k\,.\mathrm{start} < x_j\,.\mathrm{start}$. The first penalty can then be achieved with

$$\sum_{s \in S} \left(\sum_{\substack{x_i, x_j \in C(s) \\ i<j \\ x_j\,.\mathrm{start} - x_i\,.\mathrm{end} > \max_{\mathrm{sep}}}} x_i x_j \cdot \prod_{x_k \in \mathrm{between}(x_i, x_j)} (1 - x_k) \right),$$

and similarly for the boundary cases.

4. Overview of Quantum Algorithms for Scheduling Problems

4.1. Quantum Annealing (QA)

Given the objective function in QUBO form, our discrete ground station scheduling problem can in principle be solved using a quantum annealer, similarly to [6]. It remains to choose a good relative weighting of the constraints in the QUBO objective function. The difficulty is that no constraint should be overrepresented or underrepresented compared to the other constraints, requiring some experience or trial and error. We use three different techniques to achieve a proper balancing of the constraints:

Every constraint is multiplied with a weight that can be set by the user to balance the constraints, e. g. after some hyperparameter optimization.

Looking at the constraints Section 3.1 more closely, one notices that some constraints can be modeled with a single QUBO (e. g. the satellite minimum contact separation constraint), while other constraints require the sum of many QUBOs (e. g. the satellite maximum contact separation constraint). As a consequence, the penalty terms of

the multi-QUBO constraints may be higher and hence these constraints can be overrepresented. We counter this effect by multiplying each QUBO with a weight to balance the constraints, in this case we divide each QUBO of a multi-QUBO constraint by the number of QUBOs representing that constraint.

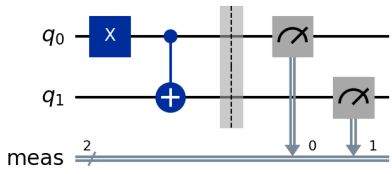
Similarly, the size of the individual QUBOs for the satellite maximum contact separation constraint depends on the number of possible contacts contained in an interval, hence overrepresenting those with many of them. To counteract this, we introduce an additional factor with which each QUBO of the satellite maximum contact separation constraint is multiplied, namely

$$\left(3\frac{n_{max}-n_x}{n_{max}-n_{min}}+0.3\right)^2,$$

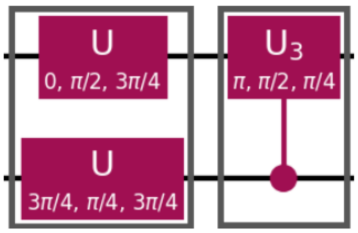
where n_x is the number of possible contacts in the interval $(x_i\,.\,\mathrm{end}, x_i\,.\,\mathrm{end}+\max_{\mathrm{sep}}]$, n_{max} is the maximum number of possible contacts in all maximum separation intervals, and n_{min} is the minimum number of possible contacts in all maximum separation intervals. This factor is determined heuristically and may be improved in the future.

4.2. Variational Quantum Algorithms

As an alternative to Quantum Annealing, an option is to use gate-based quantum computing hardware to solve optimization problems. On such hardware, quantum gates are used to manipulate the state of a register of quantum bits (qubits). The result can then be retrieved by measuring the state of the quantum register, which collapses the quantum state and yields a classical bit string. This state preparation and measurement is typically done repeatedly to retrieve a probability distribution of bit strings. The quantum register, quantum gates and measurement instructions form a quantum circuit, see Figure 3a.



(a) Example for a quantum circuit. The quantum gates affect the state of the qubits q_0 and q_1, which is then measured.



(b) A single E-VQE individual with two circuit layers on two qubits and concrete parameters.

Fig. 3. Quantum circuit examples

Gate-based quantum optimization algorithms with a theoretically proven speedup, like Grover's algorithm [9], prepare the quantum state based on the optimization problem in such a way, that measuring that state yields the intended solution with a high likelihood. For realistic optimization problems this necessitates the usage of extremely high amounts of quantum gates [10]. As, on current hardware, quantum gates are noisy in nature, noise quickly adds up invalidating any results [11].

The alternative to such algorithms are so-called Variational Quantum Algorithms (VQAs). These algorithms are heuristics that offload a part of the computation to a classical computer, so that the quantum state preparation can be achieved using significantly fewer quantum gates. This makes VQAs a prime candidate for current noisy quantum computing hardware [12]. To achieve this, VQAs employ a so-called parameterized quantum circuit (PQC). Such a circuit produces a quantum state $|\psi(\theta)\rangle$ which is dependent on some classical parameter values θ. This PQC, which is usually also referred to as a VQA's ansatz, enables the iterative VQA optimization loop. First a quantum state is created based on some parameter values. It is measured and rated based on some cost function $C(|\psi(\theta)\rangle)$ that encodes the optimization problem. Then, using the determined cost, an optimization algorithm on a classical computer is used to improve the parameter values. This optimization loop is repeated until termination. Measuring the obtained quantum state should then yield a good solution to the optimization problem with reasonable likelihood.

4.2.1. Quantum Approximate Optimization Algorithm (QAOA)

The quantum approximate optimization algorithm (QAOA) was originally proposed in [13]. It is a VQA which uses an ansatz designed to approximate the quantum annealing process. To achieve this its ansatz consists of p repeating layers each of which first apply a so-called mixer Hamiltonian H_m and then the problem Hamiltonian H_p. For each layer, a classical parameter β_i and γ_i determine to which degree the respective Hamiltonian is applied. In QAOA this ansatz provides a discretization, which for $p \to \infty$ yields the quantum annealing process exactly [14].

4.2.2. Evolutionary Variational Quantum Eigensolver (E-VQE)

The ansatz choice for a VQA has a strong influence on how well it performs. It has for instance been shown, that various aspects of an ansatz may cause barren plateaus, which can make the parameter optimization of a VQA very difficult [15, 16]. This difficulty in ansatz design leads to the idea to let the ansatz structure be optimized during the optimization routine as well. In recent years many such approaches have been investigated [17–20].

E-VQE's (see [20]) approach to this is particularly interesting. It uses an evolutionary algorithm (EA), which is an optimization heuristic that mimics biological evolution, to optimize both the ansatz structure and parameter values in its VQA routine. The following explanation of E-VQE's evolutionary optimization process is intentionally short. For more detail, see either the original authors' work [20] or our previous investigation of E-VQE [21].

EA's work on a population of individuals, each of which represent a possible solution to the optimization problem. In E-VQE, such an individual represents a combination of a certain ansatz structure, specified as a sequence of quantum circuit layers, paired with the accompanying parameter values. These quantum circuit layers then in turn can consist of arbitrary placements of either U3, controlled U3 or if nothing else is possible, identity gates. How good of a solution such an individual provides is rated by its fitness. In E-VQE this fitness is determined based on the VQA's cost function penalized by the depth and complexity of the used ansatz. This is meant to help find a good solution to the optimization problem, while avoiding the pitfalls of barren plateaus due to overtly deep or complex ansatz circuits.

On their population EA's apply evolutionary operators, such as evaluation, selection, variation and replacement to enforce evolutionary pressure, which over multiple generations improves the solution quality in the population. In E-VQE, during evaluation, the parameters of the last ansatz layer of an individual are first optimized to determine the effect of previously newly added circuit layers. This last layer optimization step entails an entire VQA subroutine. Then, with the slightly improved parameters, the cost of the quantum state produced by the individual's ansatz is evaluated, from which the individual's fitness can be determined. With the fitness for all individuals having been determined during the evaluation step, individuals are selected for the next generation of the population probabilistically, so that individuals with a better fitness have a higher chance of being selected, and the population remains of a fixed size. After the selection the individuals are varied to explore slightly different solutions. The variation operators include parameter search, which uses a VQA subroutine to optimize all parameters of an individual; topological search, which adds a random circuit layer to the ansatz of an individual; and layer removal which may remove one or multiple layers from the ansatz of an individual. Each variation operator is applied with a certain likelihood to an individual. These varied individuals then replace the individuals of the previous population. This procedure is repeated until a termination criterion is hit, see Figure 4b for a graphical representation of this process.

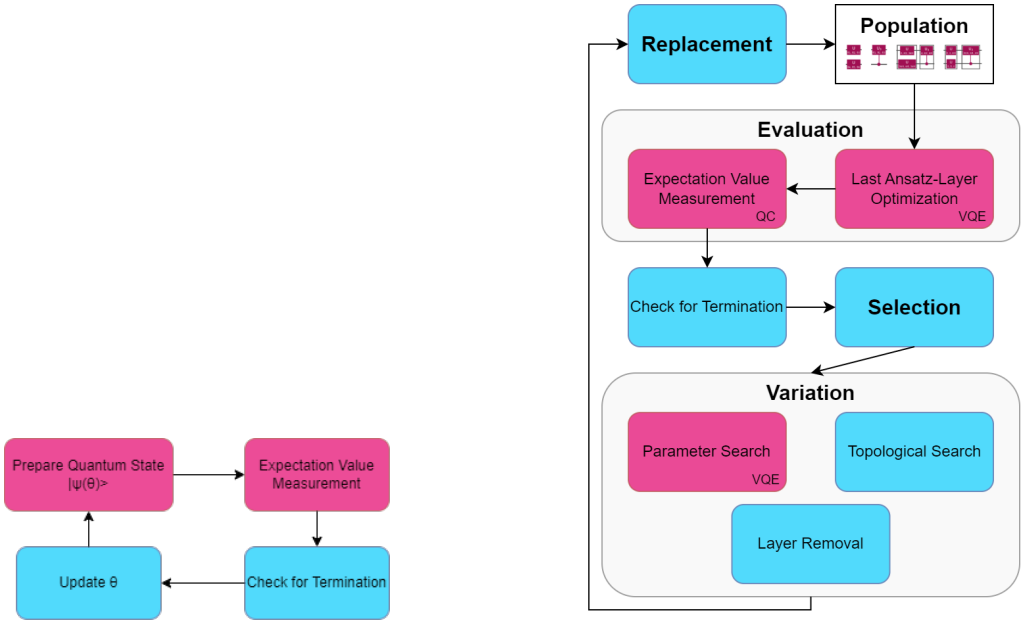
5. Results and Discussion

5.1. Evaluation and Methodology

To evaluate the presented quantum algorithms, we benchmarked the algorithms on problem instances which were generated from real, anonymized data on satellite–ground station visibilities. Quantum algorithms were simulated on classical machines without noise. In the following we will first outline how such problem instances are generated. Following that, we will explain the methodology of our benchmarks. Finally, we provide the results.

5.1.1. Generating Problem Instances

The base of our generated problem instances is real-world data on satellite–ground station visibilities. Since we need discrete contact slots within a limited time horizon for our quantum algorithms, we broke up the continuous data into small discrete problem instances, given some parameters. In particular, we fixed how long the time horizon



(a) The optimization loop of a VQA algorithm, the red steps involve quantum computing.

(b) The E-VQE evolutionary optimization routine. The red steps involve quantum computing.

Fig. 4. Workflows of VQA algorithms

should be, how long the contact slots should be, how many satellites or ground stations should be involved and roughly what percentage of the original visibility should be covered by the contact slots. We then randomly select a fitting time horizon, satellites and ground stations, so that the satellites share at least some visibilities with the same ground station, which makes the scheduling problem interesting. Then the visibilities are cut up into slots of the selected length, so that the slots are equally distributed and cover the specified portion of the visibility.

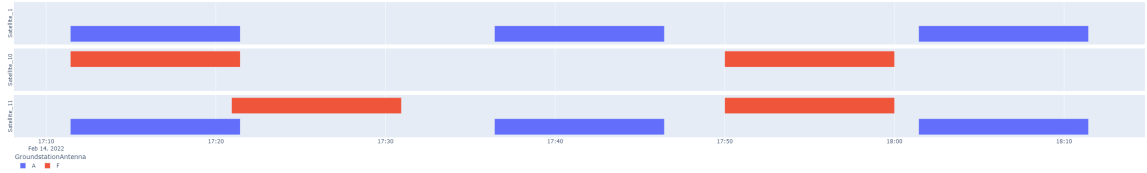


Fig. 5. Example problem instance with 10 contact slots for 3 satellites and 2 ground stations. The colored bars represent the contact slots, with the color representing the involved ground station. The separate light blue bar which they belong to indicate the satellite.

We configured this process and selected instances which match our requirements for the amount of contact slots, as these are a major factor in the amount of needed qubits.

5.1.2. Comparison Benchmarks

Since simulating gate-based quantum computers is exponentially expensive in the amount of qubits, we could only benchmark QAOA and E-VQE on problem instances with very limited amounts of contact slots. As a result, we generated 16 problem instances with 5, 10, 15 and 20 contact slots, 4 instances for each size. Since for such low amounts of contact slots, for each satellite only 2 to 3 contacts may be scheduled, trying to target a specific average contact separation is not particularly useful. Therefore, for all of these problem instances we set the goal to maximize the amount of scheduled contacts. To these problem instances we apply the minimum contact separation, maximum contact separation and maximum concurrent contact constraints. We ran our benchmark once on each

problem instance for each algorithm. As a point of comparison, we also ran the classical solver SCIP to solve each of these problem instances optimally and used to result for determining whether a quantum result is optimal. All benchmarks were executed on a 24-core i9-13590HX CPU with 128 GB memory.

For QAOA and E-VQE, we used the PUBO formulation of our problem, so that the amount of needed qubits matches the amount of contact slots, while for Quantum Annealing we used the QUBO formulation of our problem, hence using additional slack variables. We used Qiskit's implementation of QAOA and our own open source implementation of E-VQE, QUEASARS [22]. Both QAOA and E-VQE's subroutines were configured to use the SPSA classical optimization algorithm. As QAOA and E-VQE, as well as the SPSA algorithm offer many hyperparameters, we configured them using mostly the same hyperparameters as in our previous work where we used automatic algorithm configuration to improve these hyperparameters for solving the job shop scheduling problem [21]. We only deviated by adjusting the quantum circuit measurements to use 1024 measurement shots to introduce some randomness, and by using a less strict termination criterion for QAOA, as in our previous work, QAOA never terminated. In particular, we defined that QAOA should terminate once the measured cost does not change by more than 5 percent for 5 optimization iterations. We also configured QAOA to terminate after 5000 evaluations of the quantum circuit (VQA cost function) and E-VQE after 15000 evaluations. We are more permissive for E-VQE due to the overhead incurred by evaluating 10 individuals in parallel. For quantum annealing we used the simulated annealing sampler from D-Wave's Ocean SDK and configured it to use 1000 measurement shots.

5.1.3. Quantum Annealing Specific Benchmarks

For Quantum Annealing more qubits can be simulated, allowing us to benchmark simulated Quantum Annealing on problem instances with more contact slots. We first tested several sizes of the QUARGS problem with constraints only and no optimization goal. Since these problems are sufficiently big, we then also tested a QUARGS problem with the goal to evenly distribute the planned contacts for varying contact separation targets.

5.2. Results

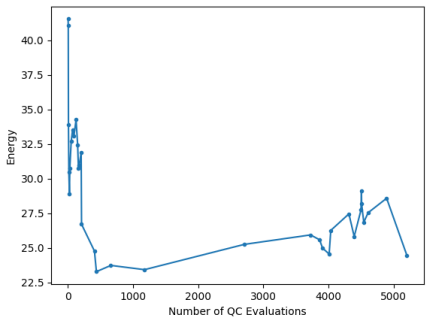
5.2.1. General Results

As mentioned in Section 4.2, variational quantum algorithms are an iterative optimization loop. The cost of the VQA's cost function (also called Energy when using problems in Hamiltonian form), can be tracked during this optimization loop. This allows us to investigate the convergence speed, stability and termination during the VQA optimization loop. In Figure 6, as an example, we show the energy history of one optimization run for QAOA and E-VQE for the problem sizes of 10 and 20 contact slots.

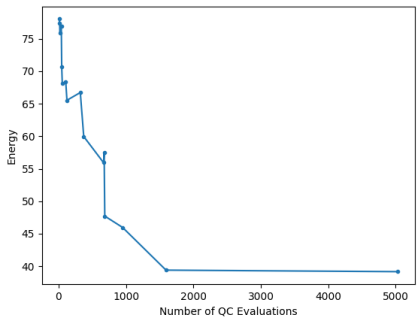
Comparing Figures 6a and 6c, we can see that E-VQE quickly reaches states with an energy much lower than those found by QAOA. E-VQE's energy history is also much more stable, allowing it to terminate, once it cannot find a better energy level. QAOA's history in comparison is much more unstable. Note also that the points in the QAOA graph lie far apart. For QAOA this is caused by SPSA's blocking option preventing any optimization step that would worsen the energy too much. This indicates that SPSA very often finds no optimization step that would improve the energy. The instability in optimization also leads to the fact that QAOA does not terminate naturally and runs into the limit of 5000 circuit evaluations. This shows that our termination criterion for QAOA still needs some work. The same can mostly be said for the 20 contact slot problem instances (see Figures 6b and 6d). Although it can be seen that E-VQE also has a harder time in finding good energies for the more difficult problem, forcing it to terminate when running into the limit of 15000 circuit evaluations.

Taking a look at the results for all the benchmarks (see Section 5.3), we can observe that QAOA finds optimal results for the small problem instances with some significant probability. Yet as the problem sizes increase, QAOA does not find the optimal solution anymore. In those cases QAOA mostly finds sub-optimal solutions that do not violate any constraints, but with very small measurement likelihoods. This does seem to imply that the basic version of QAOA we benchmarked does not scale well with the problem size. Similarly to the individual cases, we see that QAOA often does not terminate before its limit suggesting that we need to improve on this criterion.

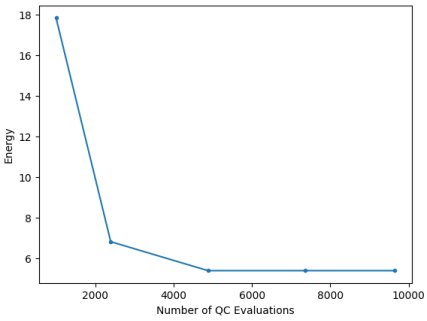
For E-VQE it can be seen that it finds the optimal solution very often, with a significant measurement likelihood. This only changes for the largest 20 contact slot problem instances, where E-VQE only finds the optimal solution for 2 out of 4 problem instances. Generally, it can be seen that while E-VQE copes better with the larger problem



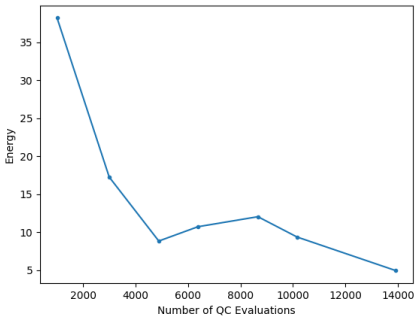
(a) QAOA - 10 contact slots



(b) QAOA - 20 contact slots



(c) E-VQE- 10 contact slots



(d) E-VQE- 20 contact slots

Fig. 6. Plots of the energy history for the QAOA and E-VQE optimization runs on the same 10 contact slot and 20 contact slot problem instances. The y-axis shows the energy while the x-axis shows how often the quantum circuit had to be evaluated during the optimization routine to reach this point. For QAOA this shows the energy of all SPSA optimization steps which were not blocked. For E-VQE this shows the energy of the best individual after each generation.

instances than QAOA this comes at the cost of a stark increase in the amount of circuit evaluations needed, so that for the largest problem instances E-VQE also runs into the configured circuit evaluation limit.

The most constant algorithm in performance over the problem sizes is QA. It finds optimal solutions for about half of the problem instances, with this behavior seemingly not changing as the problem sizes increase. Where QA does not find optimal solutions, it mostly finds suboptimal but valid solutions that do not violate any constraints. Yet, as the problem sizes increase, a decrease for the measurement likelihoods for these solutions can be observed. Note that the low measurement likelihoods for the best observed state are not necessarily indicative of bad performance for QA, as the measurements are distributed among many low energy but not optimal solutions. See Figure 7 for an example distribution of measurements.

Note also that the amount of measurements for QA was kept constant for all problem instances, which shows how well QA scales, as both QAOA and E-VQE required many more circuit evaluations as the problem size increased. This difference in scaling can also be seen in Figure 8, showing that while E-VQE is best at finding the optimal solution to the optimization problem, QA scales better with the problem size, as long as a valid but suboptimal solution is sufficient.

5.2.2. Quantum Annealing Specific Results

As mentioned in Section 5.1.3, we tested different scopes and larger problem sizes for the simulated Quantum Annealing solver to get a rough idea how it performs when scaling up and to confirm that the individual constraints and goal functions work as expected. When using only the constraint QUBOs, so essentially solving a constraint

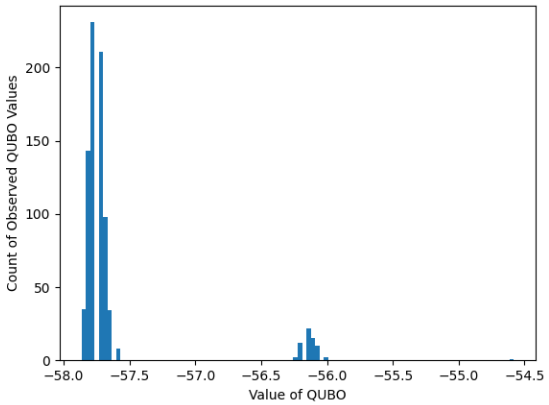


Fig. 7. Measurement distribution of the observed solution for the second 20 contact slot problem instance obtained using the QA algorithm. Note that, while the optimal solution to the very left was seldom observed, many solutions close to the optimum are frequently observed.

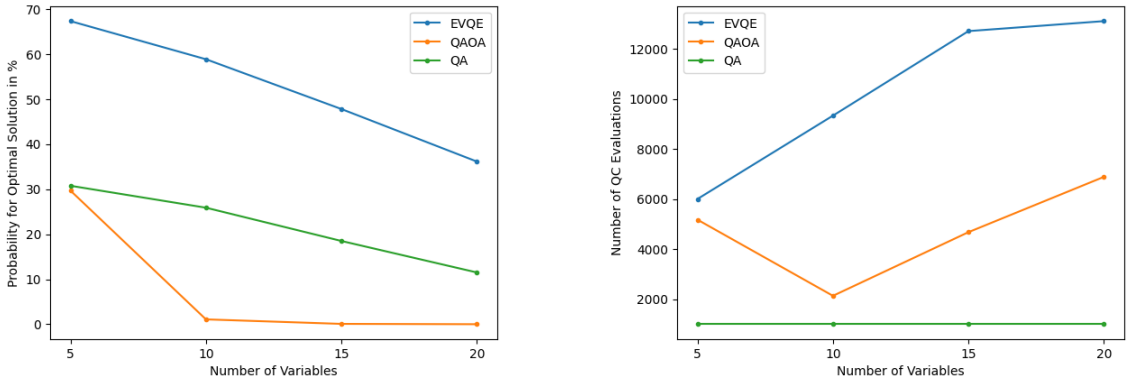
satisfaction problem, we first ran QA with only few possible contacts (e. g. 12 contact candidates) and a single constraint using 1000 shots. In all cases we found around 50–80 valid solutions which occurred several times each, hence the constraints were implemented correctly. One larger test case with 1324 contact candidates and five constraints was run with just 50 shots. In this case a uniform distribution of 50 valid solutions was found by the solver. A single medium test case with 441 contact candidates and six constraints, running for 100 shots, yielded again a uniform distribution of 100 solutions, however only around 90 of them were valid. In particular, the solution with the lowest energy level was a valid solution. When testing only the objective function for uniform contact distributions for 182 contact candidates and 100 shots, the QA algorithm did indeed found evenly distributed contacts for each satellite, although we observed 100 distinct solutions. The contacts within the individual solutions were distributed sufficiently uniformly, as expected.

The combined test with an objective function, again with an even distribution of contacts per satellite, and seven constraints with 814 possible contacts and 500 shots, was tested for four different separation goals (see Section 2.3): 4 hours, 5 hours, 6 hours and 7 hours. Non-surprisingly, the size of the QUBO increased with the desired separation, since the amount of to-be-considered possible contacts corresponds to the length of the considered time interval. As a result we found 500 schedules which occurred once each. As can be seen in Table 2, the number of valid solutions decreased with increasing separation goal. In some cases the algorithm did not work as expected, since even the solution with the lowest energy was an invalid solution.

Separation Goal	**QUBO Monomials**	**Valid Solutions**
4h	442942	~ 499
5h	454543	~ 497
6h	464702	~ 444
7h	473192	~ 237

Table 2. The result of the test combining a separation goal and several constraints

All in all, we can summarize that for small problem sizes we get good results using a QA algorithm. For bigger problem sizes, however, we sometimes still observed invalid results, hence requiring some more tuning. In any case, due to the inherently probabilistic nature of the QA algorithm, a post-processing step is necessary to filter out invalid schedules.



(a) Scaling comparison of the average measurement likelihood for the optimal solution.

(b) Scaling comparison of the average amount of circuit evaluations to termination.

Fig. 8. These figures compare the scaling of QA, QAOA and E-VQE over problem instances of increasing size.

5.3. Discussion & Outlook

In this paper we have shown, how quantum algorithms can be used to solve a discrete version of the ground station scheduling problem. A large part of the difficulty of this derives from formulating this problem as QUBO or PUBO. While forbidding precomputed pairs of contacts translates relatively easily, the inequalities in the maximum concurrent contacts constraint and in the maximum contact separation constraint, are expensive to embed in a QUBO / PUBO, requiring either many slack variables or many polynomials of high degrees. Note that the amount of slack variables / degree of the polynomials increases with the size of the time window / maximum amount of allowed concurrent contacts. This means that for realistic use-cases these constraints are much more expensive to enforce than for the small problem instances we investigated.

The results retrieved by our benchmarks show, that QA and both QAOA and E-VQE are capable of solving the ground station scheduling problem. Of these algorithms, E-VQE seems to be the best at finding the optimal solution which comes at the expense of a very high amount of computations on the quantum computer, which scale with the problem size. QA on the other hand retrieves results at a much lower computational cost and scales much better, but does not always yield optimal results. If valid but suboptimal results are sufficient, this may be a desirable trade-off. Finally, QAOA yielded the worst results, with it not scaling particularly well with the problem size.

Interestingly, in our experience there are different issues for the algorithms. For QA, it seems, that the crucial point is the relative weighting of the constraints and the objective function(s). These weights might be a worthwhile topic for further analyses and optimization. For the VQAs on the other hand the classical optimization algorithm and its convergence stability as well as its termination criterion seem to be critical. Here future work should investigate, how QAOA and its classical optimization algorithm may be improved to ensure a more stable convergence and termination.

Finally, we would also like to run benchmarks on a real quantum annealer / gate-based quantum computer in the future. This will currently only be possible for small problem instances. For QA, this is due to our QUBOs which contain many quadratic monomials which means that both of the involved qubits have to be connected physically, and hence many single qubits have to be connected to many others. This requires an embedding in the QPU where many physical qubits have to be combined to a single logical qubit, reducing the number of available qubits. For the VQAs the limit to small problem instances arises due to the limited amount of qubits on current gate-based quantum computers, which currently offer qubit amounts in the low hundreds.

References

[1] SpaceX, *Starlink*. 2020. URL: `https://www.starlink.com/` (visited on 04/07/2025).

[2] Eutelsat Group, *Oneweb*. 2025. URL: `https://oneweb.net/` (visited on 04/07/2025).

[3] US Department of Defense, *GPS*. 1995. URL: https://www.gps.gov/ (visited on 04/07/2025).

[4] EUSPA, *Galileo*. 2016. URL: https://www.euspa.europa.eu/eu-space-programme/galileo (visited on 04/07/2025).

[5] Gross, E. M.-L., Fruth, T., Dauth, M., Petrak, A., and Mrowka, F., "SCOTA: The Mission Planning Orbit Analysis Tool at GSOC". *IAF Space Operations Symposium 2021 at the 72nd International Astronautical Congress, IAC 2021*. Oktober 2021.

[6] Guillaume, A., Goh, E. Y., Johnston, M. D., Wilson, B. D., Ramanan, A., Tibble, F., and Lackey, B., "Deep space network scheduling using quantum annealing". *IEEE Transactions on Quantum Engineering* 3 (2022), pp. 1–13.

[7] Petelin, G., Antoniou, M., and Papa, G., "Multi-objective approaches to ground station scheduling for optimization of communication with satellites". *Optimization and Engineering* 24.1 (2023), pp. 147–184.

[8] Monmousseau, P., "Scheduling of a constellation of satellites: Creating a mixed-integer linear model". *Journal of Optimization Theory and Applications* 191.2 (2021), pp. 846–873.

[9] Grover, L. K., "A Fast Quantum Mechanical Algorithm for Database Search". *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 212–219. ISBN: 0897917855. DOI: 10.1145/237814.237866.

[10] Chauhan, A. K. and Sanadhya, S. K., "Quantum resource estimates of grover's key search on aria". *Security, Privacy, and Applied Cryptography Engineering: 10th International Conference, SPACE 2020, Kolkata, India, December 17–21, 2020, Proceedings 10*. Springer. 2020, pp. 238–258.

[11] Wang, Y. and Krstic, P. S., "Prospect of using Grover's search in the noisy-intermediate-scale quantum-computer era". *Physical Review A* 102.4 (2020), p. 042609.

[12] Tilly, J., Chen, H., Cao, S., Picozzi, D., Setia, K., Li, Y., Grant, E., Wossnig, L., Rungger, I., Booth, G. H., et al., "The variational quantum eigensolver: a review of methods and best practices". *Physics Reports* 986 (2022), pp. 1–128.

[13] Farhi, E., Goldstone, J., and Gutmann, S., *A Quantum Approximate Optimization Algorithm*. 2014. DOI: 10.48550/ARXIV.1411.4028.

[14] Bharti, K., Cervera-Lierta, A., Kyaw, T. H., Haug, T., Alperin-Lea, S., Anand, A., Degroote, M., Heimonen, H., Kottmann, J. S., Menke, T., et al., "Noisy intermediate-scale quantum algorithms". *Reviews of Modern Physics* 94.1 (2022), p. 015004.

[15] Wang, S., Fontana, E., Cerezo, M., Sharma, K., Sone, A., Cincio, L., and Coles, P. J., "Noise-induced barren plateaus in variational quantum algorithms". *Nature communications* 12.1 (2021), p. 6961.

[16] Holmes, Z., Sharma, K., Cerezo, M., and Coles, P. J., "Connecting ansatz expressibility to gradient magnitudes and barren plateaus". *PRX quantum* 3.1 (2022), p. 010313.

[17] Grimsley, H. R., Economou, S. E., Barnes, E., and Mayhall, N. J., "An adaptive variational algorithm for exact molecular simulations on a quantum computer". *Nature communications* 10.1 (2019), p. 3007.

[18] Zhu, L., Tang, H. L., Barron, G. S., Calderon-Vargas, F., Mayhall, N. J., Barnes, E., and Economou, S. E., "Adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer". *Physical Review Research* 4.3 (2022), p. 033029.

[19] Bilkis, M., Cerezo, M., Verdon, G., Coles, P. J., and Cincio, L., "A semi-agnostic ansatz with variable structure for variational quantum algorithms". *Quantum Machine Intelligence* 5.2 (2023), p. 43.

[20] Rattew, A. G., Hu, S., Pistoia, M., Chen, R., and Wood, S., "A domain-agnostic, noise-resistant, hardware-efficient evolutionary variational quantum eigensolver". *arXiv preprint arXiv:1910.09694* (2019).

[21] Leidreiter, D. A., *Investigating Evolving Ansatz VQE Algorithms for Job Shop Scheduling*. https://elib.dlr.de/206087/. 2023.

[22] DLR, *QUEASARS*. 2024. URL: https://github.com/DLR-RB/QUEASARS (visited on 04/07/2025).

Additional Data

The following tables contain the benchmark result data for the general benchmarks. **# Slots** refers to the amount of contact slots that may be scheduled in a problem instance. **Instance** is the index of the problem instance. Note, that the same index on a different problem size (# Slots) refers to a different problem instance. **Optimal** refers to whether the algorithm found an optimal solution for the problem instance (Y = yes, N = no). **# Violations** refers to the number of constraints that were violated by the best solution. A solution is only valid if that number is zero. **Probability** refers to the probability with which the best found solution was observed from the final quantum state. **# Evaluations** refers to the amount of quantum evaluations were needed. For the VQAs this is the amount of times the VQA cost function was evaluated by measuring the quantum circuit. For QA this is the amount of configured measurement shots.

# Slots	Instance	Optimal	# Violations	Probability	# Evaluations
5	1	N	0	99.9%	1000
	2	Y	0	99.6%	1000
	3	N	0	99.4%	1000
	4	Y	0	23.5%	1000
10	1	N	0	22.3%	1000
	2	N	1	55.8%	1000
	3	N	1	40.5%	1000
	4	Y	0	7.3%	1000
15	1	N	0	4.0%	1000
	2	Y	0	68.8%	1000
	3	Y	0	5.2%	1000
	4	N	0	8.0%	1000
20	1	N	0	0.1%	1000
	2	Y	0	0.2%	1000
	3	Y	0	45.8%	1000
	4	N	0	0.2%	1000

Table 3. Benchmark results for the QA simulations.

# Slots	Instance	Optimal	# Violations	Probability	# Evaluations
5	1	Y	0	21.8%	5116
	2	Y	0	45.9%	5002
	3	Y	0	26.7%	5035
	4	Y	0	24.4%	5545
10	1	Y	0	0.2%	64
	2	Y	0	2.8%	3199
	3	Y	0	0.2%	67
	4	Y	0	1.1%	5206
15	1	Y	0	0.1%	5869
	2	N	0	0.2%	7216
	3	N	0	0.1%	5521
	4	N	0	0.1%	97
20	1	N	0	0.2%	5035
	2	N	1	0.1%	11194
	3	N	1	0.2%	6208
	4	N	0	0.1%	5101

Table 4. Benchmark results for the QAOA simulations.

# Slots	Instance	Optimal	# Violations	Probability	# Evaluations
5	1	Y	0	66.6%	7366
	2	Y	0	56.4%	4089
	3	Y	0	71.8%	4584
	4	Y	0	74.7%	7960
10	1	Y	0	62.3%	8762
	2	Y	0	57.5%	8455
	3	Y	0	60.6%	10445
	4	Y	0	55.2%	9653
15	1	Y	0	64.4%	10356
	2	Y	0	50.6%	14910
	3	Y	0	76.3%	9158
	4	N	0	62.4%	16359
20	1	Y	0	15.6%	13930
	2	N	0	50.0%	15009
	3	N	1	85.8%	8564
	4	Y	0	43.2%	14910

Table 5. Benchmark results for the E-VQE simulations.