QUARK: A Framework for Quantum Computing Application Benchmarking

Jernej Rudi Finžgar∗
BMW Group
Technical University Munich
Munich, Germany
Jernej-Rudi.Finzgar@tum.de

Philipp Ross∗
BMW Group
Munich, Germany
Philipp.Ross@bmwgroup.com

Johannes Klepsch
BMW Group
Munich, Germany
Johannes.Klepsch@bmw.de

Andre Luckow
BMW Group
Munich, Germany
Andre.Luckow@bmwgroup.com

Abstract
Quantum computing (QC) is anticipated to provide a speedup over classical HPC approaches for specific problems in optimization, simulation, and machine learning. With the advances in quantum computing toward practical applications, the need to analyze and compare different quantum solutions increases. While different low-level benchmarks for QC exist, these benchmarks do not provide sufficient insights into real-world application-level performance. We propose an application-centric benchmark method and the QUantum computing Application benchmaRK (QUARK) framework to foster the investigation and creation of application benchmarks for QC. This paper establishes three significant contributions: (1) it makes a case for application-level benchmarks and provides an in-depth "pen and paper" benchmark formulation of two reference problems: robot path and vehicle option optimization from the industrial domain; (2) it proposes the open-source QUARK framework for designing, implementing, executing, and analyzing benchmarks; (3) it provides multiple reference implementations for these two reference problems based on different known, and where needed, extended, classical and quantum algorithmic approaches and analyzes their performance on different types of infrastructures.

Keywords: quantum computing, benchmark, optimization, applications

1 Introduction

Motivation. Quantum computing (QC) is transitioning from research to industrialization. It promises to provide significant improvements to optimization, machine learning, and simulation problems, overcoming the limitations of existing high-performance computing systems [10]. Applications for these problem domains can be found in academia and industry [7]. For example, numerous complex design, manufacturing, logistics, and financial challenges in the automotive industry are promising candidates for quantum-based optimization and machine learning approaches. Quantum chemistry simulations promise to enhance the material research process, e.g., for battery cell chemistry.

Impressive progress has been made, as visible, e.g., in a quantum advantage demonstration [4] by Google. However, it is currently unclear what hardware technology and algorithms will deliver a practical quantum advantage, i.e., a quantum system that provides better quality, runtime, or cost than a classical computing system. The evaluation and benchmarking of quantum systems are becoming increasingly important. Benchmarks are critical to characterize quantum solutions and guide application, algorithm, and hardware developments, and establish communities [66].

State-of-the-art and limitations. Current quantum computing benchmarks often focus on low-level hardware performance [15, 58]. Higher-level algorithm benchmarks, e.g., Lubinski [42] and Martiel [45], consider a set of algorithms and circuits. While these approaches provide important insights, they do not investigate end-to-end application performance and thus, foster holistic advances on all levels that are required for real-world applications.

Key insights, contributions, and artifacts.
In this work, we make three significant contributions: (1) we propose an application-centric approach for developing benchmarks. By using a "pencil and paper" approach (as popularized by the NAS parallel benchmark (NPB) [5]), we allow for multiple problem formulations, e.g., for quantum annealing, noisy intermediate-scale quantum (NISQ) devices, and classical systems. Considering the maturity of quantum hardware and programming systems, we think this approach is best-suited, facilitating innovations and optimizations on all levels, e.g., hardware, control system, operating system and middleware, algorithm, and application level. Specifically, we provide a formulation of two reference problems from the industrial domain, robot path
and vehicle option optimization (see Section 3); (2) we introduce the open-source Quantum computing Application benchmark (QUARK) framework for designing, implementing, executing, and analyzing benchmarks (see Section 4).\(^1\) QUARK addresses critical requirements of application benchmarks, such as the need to abstract realistic workloads and datasets into benchmark kernels, support multiple implementations of these, and to reproducibly capture all results; (3) we demonstrate QUARK’s capabilities by implementing benchmarks for the two reference problems (see Section 5). For this purpose, we develop and characterize several classical and quantum algorithms and implementations (e.g., a novel QUBO formulation of the MAX-SAT problem) utilizing different infrastructures (e.g., D-Wave and simulation).

Limitations. It is challenging to develop representative application benchmarks for quantum computing, as it is unclear which algorithm, qubit modality, and hardware will deliver a quantum advantage. As current quantum systems provide no real practical advantage, the utility of application-level benchmarks is still limited. Further, transferring benchmark results to other applications is often challenging. Finally, QUARK does not cover all types of QC applications, but only optimization cases, in its present form.

2 Background and Related Work

2.1 Quantum Computing Infrastructure

Various realizations of quantum computers have been proposed and are in active development. Distinct implementations of qubits typically possess different characteristics, concerning e.g., gate fidelities, coherence times, and clock speeds. Currently, superconducting and ion-trapped qubits are the most widely used modalities and are available from different vendors on different clouds, e.g., superconducting systems from IBM [33], Google [29], and Rigetti [62] and ion-trap based systems from IonQ [36] and Honeywell [31]. Non-gate-based systems for quantum annealing from D-Wave are also broadly available on the D-Wave [16] and AWS clouds. Finally, approaches such as neutral atom [24], and topological quantum computation [40] could become prominent in the future.

Additionally, classical simulation of quantum systems is crucial for the development of quantum technologies. It aids the design of quantum algorithms and enables the verification of results obtained on quantum devices. Hence, it is necessary to understand the trade-offs and scales of different simulation approaches (see [59] for an overview).

2.2 Benchmarks

Benchmarks are standardized workloads, i.e., sets of inputs (program and data), that are used to compare computer systems [26, 37] and have been instrumental in many areas of computer science and engineering. Benchmarks arise on different levels: Lower-level benchmarks focus on the system level (e.g., gate fidelity), and thus, are difficult to map to application performance. Benchmarking on the algorithmic level focuses on evaluating specific algorithms representing the significant subroutines, and thus, the runtime of different applications. Application-level benchmarks are more holistic and consider the entire stack comprising hardware, operating system, middleware, and classical resources. However, while application-level benchmarks capture critical application characteristics, transferring these insights to other applications and systems is often difficult.

In general, two types of benchmarks exist: (i) specification-based benchmarks that provide a “pen and paper” description of a problem; and (ii) reference implementations. Both approaches have their advantages: specification-based benchmarks are flexible, allowing for innovation, whilst hampering comparisons. Reference implementations typically limit the design space and allow for more controlled yet expensive experiments.

Table 1 summarizes benchmarks for HPC and QC for different levels, which we describe in the following sections.

| Level | Classical | Quantum |
|-------|-----------|---------|
| Application | ImageNet [64], Glue [73], QScore [45], QED-C [42], Fermi-Hubbard Model [18, 27] | |
| Algorithm | Linpack [23], NPB SPEC [68], TSPLib95 SAT competition [35] | |
| System | SPEC HPC, ACCEL SPEC [52], OMP [68], QV [15], Volumetric benchmarking [11], randomized gate benchmarking [15], Arm compiler benchmark [3], CLOPS [72] | |

Table 1. Important benchmarks for the different layers: From system-level to application-level benchmarks.

\(^1\)QUARK will be available on GitHub soon.
Later, the MLCommons benchmark suite [50] (also known as MLPerf [46]) emerged, proposing runtime performance, solution quality, and costs as metrics. Together, these metrics enable benchmark users to understand the relationship between time-to-solution and solution quality systematically.

Several benchmarks for common optimization tasks have been proposed. The SAT competition [35] comprises different kernels, e.g., for model verification, database repair, and MAX-SAT problems. Various benchmark datasets for scheduling problems exist, e.g., employee scheduling [65] or compute job scheduling [76]. Chook [56] is a tool to generate higher-order binary optimization problems of desired complexity and a portfolio of classical solver techniques. For the traveling salesperson problem, TSPLib [60, 61] provides more than 100 sample datasets for different problem variations.

### 2.2.2 Quantum System Benchmarks

Benchmarking quantum systems is a complex endeavor, mainly due to the high dynamism of the field, which is rapidly evolving on different layers of the stack. Quantum system benchmarks focus on low-level aspects. For example, the quantum volume (QV) benchmark captures important aspects, such as the number of qubits, circuit size, and gate fidelity [15]. The largest circuit with equal width (number of qubits) and depth (number of circuit layers) that a system can successfully execute defines the QV.

Blume-Kohout et al. [11] extend the QV metric to more diverse circuit types, addressing some of its limitations. Particularly, it removes the constraint of square circuits, allowing for rectangular circuits with different numbers of qubits and layers. Further, the authors propose, in addition to randomized circuits used by QV, the use of other circuit types, e.g., Grover iterations and Hamiltonian simulations, and additional quality metrics (complementing the heavy output probability metric used by QV). While the QV metric emphasizes the quality of qubits, the circuit layer operations per second (CLOPS) metric focuses more holistically on the speed of execution [72]. The benchmark is based on parametrized circuits, i.e., a circuit, which is static but will be configured with a set of parameters at runtime. Parameterized circuits are used in quantum machine learning, quantum optimization, and quantum chemistry, particularly in NISQ-era algorithms. The metric considers the circuit execution time including overheads like preparation and queuing time.

### 2.2.3 Quantum Application Benchmarks

**Characterizations:** D-Wave annealing systems have been thoroughly investigated regarding their performance, tunability, and limitations for different applications ranging from science to finance and industry. Grant et al. [30] utilize a portfolio optimization use case to analyze the effects of different control parameters of quantum annealers. In particular, they monitored how the solution quality changes with different embeddings, annealing times, and spin reversal routines. Their work does not consider other metrics, e.g., solution quality and time-to-solution. Pang, et al. [54] investigate the performance of quantum annealing for finding the ground state of a spin glass with ferromagnetic coupling in three different parameter regimes and compare it to several classical methods. Perdomo et al. [55] investigate an industrial optimization problem, specifically the combinational circuit fault diagnosis (CCFD) problem, focusing on the scalability of annealing approaches. Yarkoni et al. [78] investigate annealing for paint shop optimization. They found that on small scale, the quantum annealing and the Hybrid Solver Service (HSS) could perform better than random.

Various characterizations of gate-based systems and applications exist. Willisch et al. [77] investigate the performance of quantum approximate optimization algorithm (QAOA) and annealing and their ability to discover the optimal solution for artificial Max-Cut and 2-SAT problems. Performance aspects, e.g., the time-to-solution, are not investigated.

For other problem domains such as quantum chemistry, benchmarks have been proposed. McCaskey et al. [47] propose a benchmark for the variational quantum eigensolver (VQE) algorithm using three different alkali metal hydrides materials. The authors utilize the discovered ground state to compare a Rigetti and IBM quantum system. Similarly, Dallaire-Demers et al. [18] use VQE for the Fermi-Hubbard model to benchmark Google’s Sycamore processor.

**Benchmarks:** While previous examples focus on specific application scenarios, Mills et al. [48] emphasize the need for more holistic benchmarks. For this purpose, the authors propose three circuit designs: shallow, square and deep circuit. The proposed approach is similar to the volumetric benchmark approach proposed in [11]. While these circuit types can be mapped to more concrete applications on a high level, it is difficult to predict performance on concrete applications (e.g., for specific problem types and sizes).

Martiel et al. [45] propose an application-centric optimization benchmark called Q-Score. The Q-Score is based on performing the Max-Cut algorithm using QAOA on different sizes of standardized Erdős-Rényi graphs. Q-Score is limited because it relies on a single algorithm.

Lubinski et al. and the QED-C [42] propose application-oriented benchmarks to assess quantum systems using a volumetric framework. Currently, the framework comprises 11 different algorithms. The proposed algorithms are exclusively gate-based. While most of these algorithms provide important building blocks for quantum applications, the analysis is not conducted in the context of industry applications. Important application domains, such as optimization and machine learning, are not addressed. The framework relies on a normalized fidelity metric, which compares the output distributions between the optimal result and experiment.
Discussion: Most approaches focus on specific applications and systems, investigating different configurations to improve understanding. Further, they often rely on application-agnostic quality metrics which are difficult to map to real-world application performance. Finally, they often lack an end-to-end perspective and ignore hidden costs, e.g., the time required to move data between classical and quantum interfaces. The current state reflects the maturity of the quantum ecosystem, which is yet to deliver a practical advantage. The standardization of metrics, datasets, benchmarking methods, and reproducibility will become increasingly important considering the rapid progress toward real-world applications.

3 Applications and Workloads

In this section, we describe two representative application examples from the optimization domain: robot path planning and vehicle option planning and provide a rigorous formulation of the problems.

3.1 Robot Path Planning

Application. Robots are a crucial enabler for automation in industrial manufacturing, driving quality, efficiency, and scale improvements. However, the deployment of robot systems comprising software and hardware is challenging. One particular example is planning paths for complex multi-robot systems [51]. Robots have to follow a pre-defined path to execute multiple tasks in such systems.

An example is the polyvinyl chloride (PVC) sealing process, in which spaces on the vehicle body, e.g., between joint sheets, are sealed using PVC, a thermoplastic material, to increase waterproofness and prevent corrosion.

The real-world system is highly complex – for example, each robot has multiple tools and configuration settings, like the number and type of nozzles used by each robot. Typically, multiple robots (up to four) work parallel during this process.

Thus, spatial constraints to avoid collisions must be enforced. The objective is to find the shortest valid path that fulfills the following requirements: (1) all seams need to be sealed; (2) the robot always must start and end at a particular home position; and (3) no collision between the different robots must occur.

Due to current quantum computing hardware limitations, we make several simplifications. First, we only consider single robot systems, meaning that no collisions have to be avoided. Second, we simplify the dataset and aggregate data across some dimensions, e.g., different available tool and configuration parameters. Third, we only consider two different tools and configuration settings. Finally, we decrease the problem size to allow execution on current hardware. For this purpose, we remove seams from the real-world problem graph deterministically to ensure reproducibility. Listing 1 illustrates the data.

Listing 1. Simplified data model for the PVC use case. S is the index of the seam, N the number of the node (2 per seam), C the configuration of the robot, T the chosen tool, and COST is the combined cost for interseam and intraseam movement. [-1 -1 -1 -1] is the special home position where every valid solution has to start.

| S | N | C | T | COST |
|---|---|---|---|------|
| [1 1 2 1] | 5.65 |
| [1 1 1 1] | 4.42 |

Problem Class. The problem is related to the NP-hard traveling salesperson problem (TSP). TSP can be formulated as a weighted graph, which encodes the distances between all possible pairs of nodes. The goal is to find a combination of nodes representing the shortest path, and thus, the shortest time necessary.

While robot path planning is related to the TSP, there are some key differences: (1) There are two nodes per seam, but only one of these nodes needs to be visited to seal that seam; (2) there are numerous tools and configuration settings in which a node can be visited; (3) the costs from one node to the other with a specific tool/configuration setting are not symmetric; and (4) the graph is not fully connected as not all moves are possible.

Mathematical Model. We define $x^{i}$ as a binary variable, which we set to 1 if the robot is at the node $(s, n, c, t)$ at time-step $i$, where $s$ denotes the seam number, $n$ the node number, $c$ the configuration and $t$ the tool setting. Overall there are $N_{seams} + 1$ time-steps as we need to visit all seams plus the special home position for a path to be valid. The cost function comprises the following components:

\[
\begin{align*}
\text{f}_{\text{distance}}(x) &= \sum_{i=1}^{N_{seams}+1} \sum_{(s,n,c,t)} \sum_{(s',n',c',t')} d_{snt}^{s'n'c't'} x_{snt}^{i} x_{s'n'c't'}^{i}, \\
\text{f}_{\text{time}}(x) &= \sum_{i=1}^{N_{seams}+1} \left[ \sum_{(s,n,c,t)} x_{snt}^{i} - 1 \right]^{2}, \\
\text{f}_{\text{complete}}(x) &= \sum_{s=1}^{N_{seams}} \sum_{i=1}^{N_{seams}+1} \left[ \sum_{(s,n,c,t)} x_{snt}^{i} - 1 \right]^{2},
\end{align*}
\]

where we have collected all $x_{snt}^{i}$ into a vector $x$. Note that the home position is included in $(s, n, c, t)$ for simplicity. The total distance covered by the robot is $\text{f}_{\text{distance}}$, with $d_{snt}^{s'n'c't'}$ representing the distance between $x_{snt}$ and $x_{s'n'c't'}$. Additionally, we defined two constraint terms: $\text{f}_{\text{time}}$ and $\text{f}_{\text{complete}}$. The constraint term $\text{f}_{\text{time}}$ ensures that only a single node is visited per time-step, while $\text{f}_{\text{complete}}$ ensures that every task is performed exactly once, i.e., every seam is sealed and the home position is visited.
which are notoriously hard to solve. SAT problems pose the number of qubits
\[ N \]
we can compute the number of qubits
\[ f = \text{distance}(x) + \lambda \left[ f_{\text{complete}}(x) + f_{\text{time}}(x) \right], \tag{2} \]
where \( \lambda \) is the Lagrange parameter determining the magnitude of the constraint terms. The resulting QUBO instance can be optimized using quantum approaches such as quantum annealing, QAOA, or classical algorithms. In a post-processing step, we reorder the solution to ensure that the home is the start position. Using
\[ N_{\text{qubits}} = (2 \text{ (nodes per seam)} \ast N_{\text{seams}} + 1 \text{ (home position)}) \ast N_{\text{configs}} \ast N_{\text{tools}} \ast N_{\text{time-steps}} \tag{3a} \]
\[ N_{\text{time-steps}} = N_{\text{seams}} + 1 \text{ (home position)} \tag{3b} \]
we can compute the number of qubits \( N_{\text{qubits}} \) needed to encode the optimization objective Eq. (2) on a quantum device (see Table 2).

### 3.2 Vehicle Options

**Application.** Before a new vehicle model can be deployed for production, several tests have to be carried out on pre-series vehicles to ensure the feasibility and gauge the functionality of specific configurations of components. Naturally, the manufacturer wants to save resources and produce as few pre-series vehicles as possible while still performing all desired tests. Further, not all feature configurations can realistically be implemented in all vehicles, leading to constraints that the produced vehicles must satisfy.

**Problem Class.** The vehicle options optimization problem belongs to the family of satisfiability (SAT) problems, which are notoriously hard to solve. SAT problems pose the question of determining whether a configuration of Boolean variables exists, such that the given Boolean formula evaluates to 1.\(^2\) SAT problems are NP-complete and not only lie at the center of contemporary theoretical computer science research but also appear in a wide range of fields such as artificial intelligence [53], circuit design [32] and computational biology [44], to name a few. Additionally, one can consider an optimization extension of SAT problems termed maximum satisfiability (MAX-SAT). In MAX-SAT, one searches for the configuration that maximizes the number of satisfied clauses in a SAT problem. Due to its theoretical importance and applicability, the study of (MAX-)SAT is an active area of research [35] – for a review, see [41].

**Mathematical Model.** Consider the set of \( N_v \) test vehicles \( \{v^{(1)}, ..., v^{(N_v)}\} \), where each vehicle is exactly defined by its configuration of \( N_f \) features. That is for each \( i \), \( v^{(i)} \in \{0, 1\}^{N_f} \) is a binary vector of dimension \( N_f \), where its \( j \)th component \( v^{(i)}_j \) encodes the information whether feature \( j \) is absent \( (v^{(i)}_j = 0) \) or present \( (v^{(i)}_j = 1) \) in this particular vehicle.

In a realistic setting, not all of the \( 2^{N_f} \) possible configurations are feasible (e.g., a vehicle cannot simultaneously have a V4 and V8 engine) leading to the introduction of \( N_k \) constraints \( \phi_k \). Each constraint can be specified as a Boolean expression involving some subset of features. For example, the condition that vehicle \( i \) must contain at least one of the features 1 or 2, and not include feature 3 can be formulated as follows:
\[ \phi_{\text{example}}(v^{(i)}) = \left( v^{(i)}_1 \lor v^{(i)}_2 \right) \land \neg v^{(i)}_3. \]

Since all of the \( n \) vehicles have to satisfy each of the \( p \) constraints, this means that we demand that
\[ \bigwedge_{j=1}^{N_v} \bigwedge_{i=1}^{N_k} \phi_j(v^{(i)}) = 1 \tag{4} \]
holds.

Additionally, we want to perform \( N_e \) different tests on the vehicles. We model this by introducing a collection of \( N_e \) test requirements \( \theta_k \) – we demand each of the \( \theta_k \) to be satisfied by at least one of the \( N_v \) vehicles:
\[ \bigwedge_{k=1}^{N_e} \bigwedge_{i=1}^{N_v} \theta_k(v^{(i)}) = 1. \tag{5} \]

Combining the buildability constraints and the test requirements, we can state the full mathematical formulation of the vehicle options problem as:
\[ \bigwedge_{j=1}^{N_v} \bigwedge_{i=1}^{N_k} \phi_j(v^{(i)}) \land \bigwedge_{k=1}^{N_e} \bigwedge_{i=1}^{N_v} \theta_k(v^{(i)}) = 1. \tag{6} \]

In practice, a related question is asked: given that a certain quantity of vehicles can be produced, what is the configuration of features of the produced vehicles that maximizes the number of tests that can be performed on them. Due to the limited capabilities of current quantum devices, we limit ourselves to finding the optimal configuration of features for a single vehicle. This approach can be interpreted as a single step of the optimization procedure for multiple vehicles. After one finds the vehicle that satisfies the most tests, the

| \( \text{N}_{\text{seams}} \) | \( \text{N}_{\text{tools}} \) | \( \text{N}_{\text{configs}} \) | \( \text{N}_{\text{time-steps}} \) | \( \text{N}_{\text{qubits}} \) |
|---|---|---|---|---|
| 1 | 2 | 2 | 2 | 24 |
| 2 | 2 | 2 | 3 | 60 |
| 3 | 2 | 2 | 4 | 112 |
| ... | ... | ... | ... | ... |
| 60 | 4 | 4 | 61 | 118096 |
| 70 | 4 | 4 | 71 | 160176 |

\( 0 \equiv \) False and \( 1 \equiv \) True throughout this manuscript.

\(^2\)We use False \( \equiv 0 \) and True \( \equiv 1 \) throughout this manuscript.
tests that have been satisfied can be removed from consideration. The next chosen vehicle is chosen by maximizing the number of the remaining tests.

Thus, the optimal configuration is defined as:

$$\mathbf{v}^* = \arg \max_{\mathbf{v} \in \Phi} \left( \sum_{k=1}^{N_h} \phi_k(\mathbf{v}) \right),$$

where $\Phi = \{ \mathbf{v} \mid \bigwedge_{k=1}^{N_h} \phi_k(\mathbf{v}) = 1 \}$ is the set of the configurations that satisfy all buildability constraints. This formulation of the problem is an instance of MAX-SAT, with the buildability constraints and test requirements corresponding to hard and soft constraints from the MAX-SAT literature, respectively.

For simplicity, we limit ourselves to MAX-3SAT (i.e., MAX-SAT where all clauses are length 3) instances in conjunctive normal form (CNF), since any MAX-SAT instance can efficiently be brought into this form.

To utilize quantum devices, we have to transform our problem into a suitable form. In our case, this amounts to rewriting the given MAX-3SAT instance as a QUBO problem. We extend the QUBO formulation by Dinneen [22] to be able to prioritize satisfying hard over soft constraints.

Consider a clause

$$\xi_i = (x_{i1} \lor x_{i2} \lor x_{i3}), \quad x_{ij} \in \{0,1\}.$$

Using the fact that we can represent negation of binary variables as $\bar{v} \iff (1-v)$, we can equivalently state the clause $\xi_i$ as a cubic polynomial in the binary variables $x_{ij}$:

$$\xi_i = x_{i1} + x_{i2} + x_{i3} - x_{i1} x_{i2} - x_{i1} x_{i3} - x_{i2} x_{i3} + x_{i1} x_{i2} x_{i3}. \quad (8)$$

By introducing an ancillary binary variable $z_i$, we can reduce the degree of the polynomial on the r.h.s. of Eq. (8) as

$$x_{i1} x_{i2} x_{i3} = \max_{z_i \in \{0,1\}} z_i (x_{i1} + x_{i2} + x_{i3} - 2). \quad (9)$$

If we make use of the fact that for binary variables $x = x^2$, we can thus write each clause as a purely quadratic polynomial by combining Eqs. (8) and (9).

Let us denote with $\bar{\phi}_j(v, z_h)$ and $\bar{\phi}_k(v, z_s)$ quadratic polynomials corresponding to the hard and soft constraints transformed in this manner. Here $z_h$ and $z_s$ are binary vectors of dimensions $N_h$ and $N_s$, with their components being the ancillary variables introduced to reduce the degree of the hard and soft constraints, respectively. The vehicle options MAX-3SAT problem can then be formulated as finding the maximum of the following QUBO problem:

$$C_{\text{MAX-3SAT}}(v, z_h, z_s) = \lambda \sum_{j=1}^{N_h} \bar{\phi}_j(v, z_h) + \sum_{k=1}^{N_s} \bar{\phi}_k(v, z_s), \quad (10)$$

where $\lambda$ is a hyperparameter. If we set $\lambda$ to be the number of soft constraints $q$, it is never favorable to violate a hard constraint in order to satisfy a soft constraint. In that case

$$v_{\text{opt}} := \arg \max_v \left( \max_{z_h, z_s} C_{\text{MAX-3SAT}}(v, z_h, z_s) \right) \quad (11)$$

is guaranteed to be the optimal configuration for the given instance. Conversely, we can minimize $-C_{\text{MAX-3SAT}}$ using (quantum) annealing approaches to obtain $v_{\text{opt}}$. Note that this approach uses $N_q + N_h + N_s$ binary variables, and therefore qubits, to encode the vehicle options problem.

While the procedure presented above works for the MAX-3SAT problem, we also include a direct QUBO formulation for MAX-SAT instances with arbitrary (even varying) clause lengths in QUARK. The formulation relies on mapping the SAT problem to the maximum independent set problem, and is an extension of the encoding introduced by Choi [14].

### 4 QUARK Benchmarking Framework

The QUARK framework aims to facilitate the development of application-level benchmarks. The framework simplifies the end-to-end process of designing, implementing, conducting, and communicating application benchmarks. As applications are highly diverse, it is essential to provide a flexible framework that focuses on investigating system performance in terms of application-level quality metrics (e.g., the path length for TSP applications), bridging the gap between existing system benchmarks and applications. The framework addresses essential benchmarking requirements, allowing for rapid development and refinement of application benchmarks. It provides reproducibility, verifiability, high usability, and customizability. It ensures that benchmark results can be easily collected and distributed. Furthermore, it is vendor-agnostic, ensuring the neutrality of the system.
4.1 Architecture

The framework is written in Python and designed to be modular and extensible, facilitating new application and problem types, algorithms, and devices. Figure 1 shows the architecture of the QUARK framework. The framework comprises five components: The Benchmark Manager is responsible for orchestrating the overall execution of the benchmark. The Application, Mapping, Solver and Device components encapsulate different aspects of a benchmark. Each component provides an abstract base class that can be extended for the concrete realizations of a functionality. The modular approach accommodates changes and extensions to benchmark implementations with minimal effort.

Application. The application component defines the workload, comprising a dataset of increasing complexity, a validation, and an evaluation function. We provide examples for utilizing real-world, synthetic data, and existing benchmark datasets (e.g., TSPLib95 [61]). The application module can be configured using a shared, framework-wide configuration management system. For example, different problem sizes can be generated depending on the configuration, accommodating the limitations of current quantum hardware and simulation devices. The validation function checks whether the provided solution is valid. For example, for the robot path problem, the function determines whether a valid path comprising a visit of all seams was generated. The validation function assumes that the result can be validated using a classical system, which is the case for most problems. The task of the evaluation function is to compute and return a metric that aids the quantitative comparison of the discovered solution. The benchmark developer can utilize particular quality scores for this purpose.

Mapping. The task of the mapping module is to translate the application’s data and problem specification into a mathematical formulation suitable for a solver. For example, quantum-based solvers for combinatorial optimization problems usually require the problem to be specified in a QUBO or Ising formulation [28]. The mapping is highly application-specific, requiring domain-specific knowledge. To implement the mapping, developers can utilize higher-level abstractions, e.g., PyQubo [79], or re-use available formulations in libraries, such as Ocean [70] and Qiskit Optimization [21].

Solver. The solver is responsible for finding feasible and high-quality solutions of the formulated problem, i.e., of the defined objective function. Various algorithms for solving QUBO problems exist, e.g., quantum annealing as provided by D-Wave machines, QAOA [25] and VQE [57] for NISQ devices, and Grover Adaptive Search for fault-tolerant hardware [12]. Quantum SDKs like Qiskit [1], Pennylane [8] and Braket [2] provide circuit templates or even higher abstraction levels for solving QUBO and Ising problem formulations. Specifically, for the TSP, we provide a Braket, Pennylane, and Qiskit implementation of QAOA.

Device. Several quantum devices (e.g., IonQ, Rigetti, IBM, Google), simulators (e.g., Braket’s SV1, Qiskit’s QASM simulator, Atos’ QML, Qulacs), and services (e.g., Amazon Braket and Azure Quantum) exist. Each environment has its characteristics and API. Adapting applications and benchmarks to this heterogeneous landscape is challenging, requiring the manual customization of API (e.g., for job submission) and translation between data formats (e.g., different QUBO/Ising matrix representations).

The device class abstracts away details of the physical device, such as submitting a task to the quantum system. QUARK currently supports different simulators, e.g., Braket, QULACs, and Qiskit, and quantum hardware, i.e., annealing, gate-based superconducting and ion-trap based quantum computers via Amazon’s Braket service. It can easily be extended to additional simulators and quantum hardware systems.

Benchmark Manager. The benchmark manager is the main component of QUARK orchestrating the overall benchmarking process. The benchmarking process is highly customizable, i.e., every module is configurable using a central configuration file. Custom parameter settings can be added for all components, allowing a straightforward evaluation of different parameters. This configuration system ensures that benchmarks and parameters can easily be standardized. Based on the configuration, the benchmark manager will create an experimental plan considering all combinations of configurations, e.g., different problem sizes, solver, and hardware combinations. It will then instantiate the respective framework components representing the application, the mapping to the algorithmic formulation, solver, and device. After executing the benchmarks, it collects the generated data and executes the validation and evaluation functions. Data is processed according to the tidy specification [74], allowing for straightforward analysis. Data is stored with critical metadata, such as the used configuration. Further, various analysis plots are automatically generated.

Figure 2 illustrates an example of concrete instances of the abstract components. For example, the robot path planning application generates a synthetic application graph mimicking real-world data and stores it as a NetworkX graph object. The current implementation provides different mapping options, e.g., a custom, or a predefined (from e.g. Qiskit) QUBO mapping. The QUBO formulation is then used to solve the problem using quantum annealing, QAOA or classical methods like simulated annealing. The device abstraction provides the means to execute application tasks.

\[ \lambda \geq \sum_{k=1}^{N_t} w_k \]

\footnote{If one considers the weighted extension, then we have to set \( \lambda \geq \sum_{k=1}^{N_t} w_k \).}
We illustrate the capabilities of QUARK by applying it to the given problem but to showcase the flexibility and power of QUARK, and the value of providing real-world applications. The intention of the vehicle options “pen and paper” benchmarks. We present some initial results for these applications. The performance of quantum devices. While we have conducted some micro-experiments to identify suitable configurations, hyperparameters, and factors, we focused on understanding out-of-the-box performance rather than deeply profiling a single configuration.

We investigate different classical solvers and D-Wave quantum annealers for all applications. For TSP, we also use a simulation and QAOA/VQE. The annealing problems are run on the two D-Wave machines available on AWS Braket: (D-Wave Advantage 4.1 with 5760 Qubits and 2000Q 6 with 2048 Qubits). It is insightful to compare quantum annealing to its classical counterpart, simulated annealing. We use an implementation of simulated annealing given in the Neal library [17], using the default parameters. For all annealing methods, we used 500 reads. Although a QUBO formulation is typically not the most efficient mathematical representation for simulated annealing, this approach aids a direct comparison between quantum and simulated annealing.

We investigate the solution validity \( V \), quality \( Q \), and time-to-solution \( TTS \). In all experiments, \( TTS \) is mainly determined by \( T_{\text{solver}} \). The other components of \( TTS \) do not significantly change for different problem sizes. For example, the annealing of the robot path problem \( T_{\text{solver}} \) accounts for more than 99% of the overall \( TTS \). In the case of quantum annealing, \( T_{\text{solver}} \) also includes the embedding time. The error bars (where visible) display the minima and maxima across different runs of a solver.

### 5.2 Robot Path Planning (PVC Sealing)

Fig. 3 summarizes the results for \( TTS \), the path length \( Q \), and for the ratio of valid solutions \( V \). A path is valid if it starts from the home position and visits all seams. In addition to simulated annealing, we implemented three other classical algorithms as a baseline: greedy, reversed greedy, and random. The greedy and reversed greedy algorithms make the best and worst possible local move at each step, respectively. The random solver makes a random choice at every time-step to decide which node to visit next.

While quantum annealing outperforms the reverse greedy and random algorithms, it performs worse than simulated annealing – particularly striking is the difference between the ratios of valid solutions. Since both simulated and quantum annealing use the same problem formulation, this suggests that the capabilities of presently available quantum devices, rather than the encoding, are the decisive factor for the worse performance of quantum annealing.
Another limitation of current D-Wave devices is that embedding larger problem sizes is impossible after a few seams. On D-Wave 2000Q we can only solve two seams, while on the larger Advantage 4.1 problems up to three seams can be embedded. It is possible, however, that a QUBO formulation tailored to the particular architecture of D-Wave devices, would perform significantly better, both in terms of the solution quality and in time-to-solution.

Traveling Salesperson. Since the TSP can be regarded as a simplification of the PVC sealing problem, we use it to provide a well-recognized and established standard benchmark. By integrating the TSPLib95 [61] dataset into QUARK, we can easily benchmark quantum TSP solutions against state-of-the-art solutions.

Fig. 4 illustrates the performance obtained using the dsj1000 TSPLib95 dataset, which we reproducibly simplified by removing nodes until reaching the desired problem size. The QUBO formulation for this problem is constructed from the graph using the Ocean library [70], and requires $N^2_{\text{nodes}}$ qubits. We compare quantum and simulated annealing to classical algorithms: the greedy algorithm from the NetworkX library [19] and the previously described reversed greedy and random algorithms.

While on average, the greedy solver returns shorter paths, we find, for up to 8 nodes, at least one annealing run with a better solution (e.g. all three annealing options for 6). This highlights the probabilistic nature of annealing methods, and demonstrates that only considering average-case performance might be deceiving when analyzing them.

Corroborating our findings for PVC sealing, simulated annealing exhibits better performance than quantum annealing. It is interesting to note, however, that while for PVC sealing, simulated annealing outperforms the greedy algorithm, the converse is true for the TSP. This is because the greedy algorithm never changes its tool and config setting during a tour, as it is never locally optimal to do so.

As for PVC sealing, above a certain problem size, finding an embedding for the quantum annealers is impossible. For the D-Wave 2000Q we only can solve problems involving 8 nodes, while on the larger D-Wave Advantage 4.1 instances with up to 14 nodes are feasible. However, starting at 8 nodes, we observe a drop in the rate of valid solutions for both quantum annealers. Moreover, for more than 10 nodes, no valid solutions could be found with D-Wave Advantage 4.1.

Variational Algorithms. Variational quantum algorithms, such as QAOA [25] and VQE [57], promise to provide viable solutions to combinatorial optimization problems. We surveyed different hyperparameter configurations of the QUBO
formulations of the TSP from the Ocean [69] and Qiskit libraries [20]. Further, we evaluated three QAOA implementations (AWS Braket [2], PennyLane [8] and Qiskit [1]) and a VQE implementation (Qiskit). For all the different implementations, we explored different hyperparameters, particularly the depth $p$, the number of iterations, stepsize, and various configurations of the classical optimizer. For all configurations, we used 500 shots.

We were unable to consistently obtain valid solutions for any of the QAOA implementations and hyperparameter configurations considered. Only the Qiskit VQE implementation converged for three nodes but failed for four or more nodes – see Fig. 5 for a summary of the results obtained using the QASM simulator.

The poor performance of variational algorithms could be due to the inefficient QUBO formulation of the TSP, which in turn leads to wide quantum circuits. It appears that the large width and depth of the resulting parametrized circuits makes finding their optimal parameters a difficult endeavor, with the optimizers unable to escape local minima corresponding to invalid solutions [75]. Thus, it appears that finding a more efficient QUBO formulation of the TSP is of paramount importance if variational algorithms are to become a viable method for this class of problems. Alternatively, one could utilize the QUARK framework to test other hyperparameter settings or classical optimizers, yielding a setup that would reliably avoid local minima [63, 75].

5.3 Vehicle Options

Finally, we display the results for the vehicle options inspired instances of MAX-3SAT. We generate random MAX-3SAT instances for a range of total feature (variable) numbers $N_f$ up to 110, which is the largest problem instance we can encode on a quantum annealer. For each $N_f$ we generate 10 different MAX-3SAT instances with $N_h = 2N_f$ hard constraints and $N_s = \left[4.2N_f\right]$ soft constraints. Additionally, we ensure that no variable appears more than once within each clause.

We utilize the QUBO formulation presented in Section 3.2 (using $\lambda = N_f$) to solve these problems using two different quantum annealing devices and a classical simulated annealing algorithm. With the given problem specifics, the number of qubits needed to encode the generated instances scales as linearly as $[7.2N_f]$. We benchmarked the annealing-based approaches against the designated classical MAX-SAT solver RC2 [34]. For each problem instance, we perform three solver runs, resulting in 30 runs per solver for each $N_f$.

In Fig. 6 we display the $TTS$, and the ratio and average quality of valid solutions returned by each solver. The quality of solutions is taken to be the ratio of satisfied soft constraints and is only displayed for valid solutions, i.e., solutions that satisfy all hard constraints. These results reveal several features of the solvers we analyzed. Firstly, one notices that the annealing-based approaches do not consistently return valid solutions – this would suggest that increasing the $\lambda$ parameter (see Eq. (10)) is required. However, the ratio of satisfied
soft constraints roughly coincides with that expected from random assignments, which is 87.5%. This suggests that the annealing methods completely disregard soft constraints – increasing $\lambda$ would only exacerbate this problem. This issue is often encountered when formulating constrained problems as QUBO instances, which are inherently unconstrained. Hence, one has to carefully balance enforcing constraints and optimizing the objective [43].

Secondly, there is a big difference in solution quality between the RC2 classical solver and the annealing-based approaches. However, the $TTS$ of RC2 increases roughly exponentially, especially for larger problem sizes ($N_s \geq 40$). While more efficient approximate classical algorithms exist [38], this gives hope that quantum annealing could become a viable alternative with improved encoding and devices. Such improvements could come from tuning hyperparameters (e.g., $\lambda$) of the QUBO mappings presented within our framework or from finding more efficient encodings that potentially better suit the topology of current annealing devices [9, 13].

While one (on average) expects a monotonic decrease in performance of annealing algorithms with increasing problem sizes [5, 80], this is not strictly the case in our study (see middle panel of Fig. 6). This behavior can be explained by the fact that we generate a limited number of instances at each $N_s$. These instances can, in principle, be of varying complexity, which in turn leads to varying performance of the solvers – the trend towards worsening efficacy as the problem sizes increase is, however, evident. Varying instance complexity manifests itself in fluctuating solution validity for annealing-based approaches and in variation of $TTS$ for the classical solver (note error bars in the top panel of Fig. 6).

Finally, we can observe that the quantum annealing and simulated annealing approaches yield comparable results. Moreover, it is interesting to note that quantum annealing managed to provide valid solutions for some problem sizes where no valid solution was found with simulated annealing. The fact that, at least for this use case, quantum annealing seems to have started catching up with its classical counterpart portends optimism as quantum annealing devices are improved.

6 Conclusion and Future Work

Benchmarks for applied quantum computing are instrumental for measuring progress, encouraging new and innovative solutions, accelerating adoption, establishing best practices, and predicting the viability of algorithms and hardware solutions.

In this paper, we make a case for application-centric benchmarks to connect progress in the QC hardware realm to real-world application performance. For this purpose, we propose a “pen and paper” benchmark approach to address the uncertainty concerning practical quantum advantages. QUARK automates and standardizes critical parts of a benchmarking system, ensuring reproducibility and verifiability. The modular architecture enables benchmark developers to investigate and automate large-scale benchmark scenarios across a diverse set of infrastructures. The framework provides the ability to develop and characterize quantum solutions, e.g., understand performance bottlenecks, compare different solutions and configurations, understand resource requirements.

We demonstrate the benchmark development lifecycle from specification, implementation to execution using QUARK using two significant and representative industrial applications: robot path and vehicle configuration optimization. Our results provide valuable insights into the current state of quantum computing. Unsurprisingly, classical solvers outperform quantum algorithms, in that they more reliably return valid solutions, which are also of higher quality. Specifically, our data shows the limitations of variational algorithms such as VQE concerning producing valid solutions to real-world, industrial problems. However, the roughly exponential scaling of the $TTS$ for the classical solver in the vehicle options problem (Section 3.2) may serve as a reminder of the potential benefits that will be attainable as quantum computing advances. While our results show limitations of current quantum approaches, we hope that QUARK will be valuable for advancing application benchmarks.

Future Work. We will evolve QUARK by adding new problem classes (e.g., machine learning and chemistry) and frameworks (e.g., AWS Braket Jobs, Qiskit Runtime). Particularly, we will add the functionality of comprehensively analyzing hybrid algorithms, facilitating the in-depth characterization of all classical and quantum components. Further, we will enrich the collected data and metrics, e.g., by providing support for lower-level metrics like gate fidelities to better understand the system’s behavior.

We will evolve the presented reference implementations into standardized benchmarks. Standardizing all aspects of benchmarks is crucial to advance the uptake, utility, and impact. In addition to technical aspects, it is crucial to engage interested parties in a community-driven process of the technology industry, application users, and academia.

Acknowledgments

We thank Stefan Benesch, Yannick van Dijk, Marvin Erdmann, Christian Mendl, Lukas Müller and Carlos Riofrío for valuable feedback. Additionally, we thank AWS, specifically Kyle Brubaker, Helmut Katzgraber, Henry Montagu, Mauricio Resende, and Martin Schuetz, for providing a QUBO formulation of the TSP.

---

6To see that, notice that for each clause, there is a single invalid configuration. Hence, a random assignment has a probability of $1 - 1/2^3 = 87.5\%$ to satisfy each clause (of length 3).
References

[1] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, Francisco Jose Cabrera-Hernandez, Jorge Carballo-Franquis, Adrian Chen, and Chun-Fu Chen. 2019. Qiskit: An Open-source Framework for Quantum Computing. https://doi.org/10.5281/zenodo.2562110

[2] Amazon 2022. Amazon Braket: Accelerate quantum computing research. Retrieved January 25, 2022 from https://aws.amazon.com/braket/

[3] Arline Benchmark 2021. Standard Performance Evaluation Corporation (SPEC) Benchmark. Retrieved January 25, 2022 from https://github.com/ArlineQ/arline_benchmarks

[4] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Bupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Kaysh, Alexander Korotkov, Fedor Kostyuk, Mike Landhuis, Kyle Lindmark, Erik Lucero, Dmitry Lyakh, Salvador Mandra, Jarrod R. McClean, Matthew McEwen, Anthony Megnart, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neely, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sankey, Kevin J. Saitzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. 2019. Quantum supremacy using a programmable superconducting processor. Nature 574, 7779 (Oct. 2019), 505–510. https://doi.org/10.1038/s41586-019-1666-5

[5] Marlon Azinović, Daniel Herr, Bettina Heim, Ethan Brown, and Matthias Troyer. 2017. Assessment of Quantum Annealing for the Construction of Satisfiability Filters. SciPost Physics 2, 2 (April 2017). https://doi.org/10.21468/scipostphys.2.2.013

[6] D.H. Bailey, E. Barszcz, J.T. Barton, D.S. Browning, R.L. Carter, L. Dagum, R.A. Fatoohi, P.O. Frederickson, T.A. Lasinski, R.S. Schreiber, H.D. Simon, V. Venkatakrishnan, and S.K. Weeratunga. 1991. The Nas Parallel Benchmarks. The International Journal of Supercomputing Applications 5, 3 (1991), 63–73. https://doi.org/10.1177/109434229100500306

[7] Andreas Bayerstader, Guillaume Becciuin, Julia Binder, Thierry Botter, Hans Ehm, Thomas Ehmmer, Marvin Erdmann, Norbert Gaus, Philipp Harbach, Maximilian Hess, Johannes Klepsch, Martin Leib, Sebastian Luber, Andre Luckow, Maximilian Mansky, Wolfgang Mauerer, Florin Neukart, Christoph Niedermeier, Lilly Palackal, Ruben Pfeiffer, Carsten Polenz, Johanna Sepulveda, Tammo Sievers, Brian Standen, Michael Strief, Thomas Strohm, Clemens Utschig-Utschig, Daniel Volz, Horst Weiss, Fabian Winter, and Quantum Technology and Computing Applications Consortium – QUTAC. 2021. Industry quantum computing applications. EPJ Quantum Technology 8, 1 (Nov 2021). 25. https://doi.org/10.1007/s40850-021-00114-x

[8] Ville Bergholm, Josh Isaac, Maria Schuld, Christian Gogolin, M. Sohaib Alam, Shahawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, Keri McKiernan, Johannes Jakob Meyer, Zeyue Niu, Antal Száva, and Nathan Killoran. 2020. PennyLane: Automatic differentiation of hybrid quantum-classical computations. arXiv:1811.04968 [quant-ph]

[9] Zhengbing Bian, Fabian Chudak, William Macready, Aidan Roy, Roberto Sebastiani, and Stefano Varotti. 2020. Solving SAT (and MaxSAT) with a quantum annealer: Foundations, encodings, and preliminary results. Information and Computation 275 (Dec 2020), 104609. https://doi.org/10.1016/j.ic.2020.104609

[10] Matteo Biondi, Anna Heid, Nikolaus Henke, Niko Mohr, Lorenzo Pautasso, Ivan Ostojic, Linde Wester, and Rodney Zemmel. 2021. Quantum computing: An emerging ecosystem and industry use cases. McKinsey & Company, https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/quantum-computing-use-cases-are-getting-real-what-you-need-to-know.

[11] Robin Blume-Kohout and Kevin C. Young. 2020. A volumetric framework for quantum computer benchmarks. Quantum 4 (Nov. 2020), 362. https://doi.org/10.22331/q-2020-11-15-362

[12] D. Bulger, W. P. Baritompa, and G. R. Wood. 2003. Implementing Pure Adaptive Search with Grover’s Quantum Algorithm. Journal of Optimization Theory and Applications 116, 3 (March 2003), 517–529. https://doi.org/10.1023/A:1023612186844

[13] N. Chancellor, S. Zohren, P. A. Warburton, S. C. Benjamin, and S. Roberts. 2016. A Direct Mapping of Max-XSAT and High Order Parity Checks to a Chimera Graph. Scientific Reports 6, 1 (Nov. 2016). https://doi.org/10.1038/srep37107

[14] Vicky Choi. 2011. Different Adiabatic Quantum Optimization Algorithms for the NP-Complete Exact Cover and 3SAT Problems. Proceedings of the National Academy of Sciences 108, 7 (Feb. 2011), E19–E20. https://doi.org/10.1073/pnas.1018310108 arXiv:1010.1221

[15] Andrew W. Cross, Lev S. Bishop, Sarah Sheldon, Paul D. Nation, and Jay M. Gambetta. 2019. Validating quantum computers using randomized model circuits. Phys. Rev. A 100 (Sep 2019), 032328. Issue 3. https://doi.org/10.1103/PhysRevA.100.032328

[16] D-Wave 2022. D-Wave Leap. Retrieved January 25, 2022 from https://cloud.dwave.com/yes/

[17] D-Wave Systems. 2021. dwave-Neal. D-Wave Systems. https://docs. ocean.dwavesys.com/projects/neal/en/latest/reference/sample.html

[18] Pierre-Luc Dallaire-Demers, Michal Stechly, Jerome F. Gonthier, Ntwali Toussaint Bashige, Jonathan Romero, and Yudong Cao. 2020. An application benchmark for fermionic quantum simulations. arXiv:2003.01862 [quant-ph]

[19] NetworkX Developers. last accessed: Nov. 2021. https://networkx.org/ documentation/stable/reference/algorithms/generated/networkx.algorithms.approximation.traveling_salesman.greedy_tsp.html

[20] Qiskit Optimization Developers. 2022. Max-Cut and Traveling Salesman Problem. Retrieved January 25, 2022 from https://qiskit.org/documentation/optimization/tutorials/06_examples_max_cut_and_tsp.html

[21] Qiskit Optimization Developers. 2022. Qiskit Optimization. Retrieved January 25, 2022 from https://github.com/Qiskit/qiskit-optimization

[22] Michael J. Dinneen. 2016. Maximum 3-SAT as QUBO. Retrieved January 25, 2022 from https://canvas.auckland.ac.nz/courses/14782/files/574983/download?verifier=1xqRikUTEBw8mPtOB6YVmKdeEhZ9U8axW8HwpW&wrapp=1

[23] Jack J. Dongarra, Piotr Luszczek, and Antoine Petitet. 2002. The LINPACK benchmark: Past, present, and future.

[24] Sepehr Ebad, Tout T. Wang, Harry Levine, Alexander Keesling, Giulia Semeghini, Ahmed Omran, Dolev Bluvstein, Rhine Samajdar, Hannes Pichler, Wen Wei Ho, Soorwon Chai, Subir Sachdev, Markus Greiner, Vladan Vuletic, and Mikhail D. Lukin. 2021. Quantum phases of matter on a 256-atom programmable quantum simulator. Nature 595, 7866 (July 2021), 232–232. https://doi.org/10.1038/s41586-021-03528-4

[25] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028 [quant-ph]

[26] Domenico Ferrari. 1978. Computer Systems Performance Evaluation. Prentice-Hall. https://books.google.com/books?id=geBQAAAAMAAJ

[27] Bryan T. Gard and Adam M. Meier. 2021. A Classically Efficient Quantum Scalable Fermi-Hubbard Benchmark. arXiv:2111.00044 [quant-ph]

[28] Fred W. Glover and Gary A. Kochenberger. 2018. A Tutorial on Formulating QUBO Models. CoRR abs/1811.11538 (2018). arXiv:1811.11538 http://arxiv.org/abs/1811.11538
