Optimization Applications as Quantum Performance Benchmarks

THOMAS LUBINSKI*, Quantum Circuits Inc., New Haven, United States
CARLETON COFFRIN, Advanced Network Science Initiative, Los Alamos National Laboratory, Los Alamos, United States
CATHERINE MCGEOCH, D-Wave Systems Inc, Burnaby, Canada
PRATIK SATHE, Department of Physics and Astronomy, University of California Los Angeles, Los Angeles, United States, Theoretical Division (T-4), Los Alamos National Laboratory, Los Alamos, United States, and Research Institute of Advanced Computer Science, Universities Space Research Association, Mountain View, USA
JOSHUA APANAVICIUS, Applied Physics Laboratory, Johns Hopkins University, Baltimore, United States

*QED-C Technical Advisory Committee on Standards and Performance Benchmarks.

†This work was sponsored by the Quantum Economic Development Consortium (QED-C) and was performed under the auspices of the QED-C Technical Advisory Committee on Standards and Performance Benchmarks. The authors acknowledge many committee members for their input to and feedback on the project and this manuscript.
The authors acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBM Quantum team. IBM Quantum, https://quantum-computing.ibm.com/. We acknowledge IonQ for the contribution of access to hardware. The views expressed are those of the authors and do not reflect the official policy or position of IonQ. We acknowledge D-Wave Systems for contributing access to both hardware and software tools. The views expressed are those of the authors and do not reflect the official policy or position of D-Wave Systems. Contributions to this work from Los Alamos National Laboratory were conducted under the auspices of the National Nuclear Security Administration of the U.S. Department of Energy under Contract No. 89233218CNA000001. This research used resources provided by the Los Alamos National Laboratory Institutional Computing Program and was supported by the Laboratory Directed Research and Development program under Project No. 20210114ER. D.B. acknowledges NASA Academic Mission Services (Contract No. NNA16BD14C, funded under Grant No. SAA2-403506). P.S. acknowledges support from the NASA/USRA Feynman Quantum Academy Internship program. Both D.B. and P.S. are supported by NSF Expeditions in Computing program CCF No. 1918549. This work used computational and storage services associated with the Hoffman2 Shared Cluster provided by the UCLA Institute of Digital Research and Education's Research Technology Group.
Authors' Contact Information: Thomas Lubinski, Quantum Circuits Inc., NewHaven, Connecticut, United States; e-mail: tlubinski@quantumcircuits.com; Carleton Coffrin, Advanced Network Science Initiative, Los Alamos National Laboratory, Los Alamos, New Mexico, United States; e-mail: cjc@lanl.gov; Catherine McGeoch, D-Wave Systems Inc, Burnaby, Canada; e-mail: cmcgeoch@dwavesys.com; Pratik Sathe, Department of Physics and Astronomy, University of California Los Angeles, Los Angeles, California, United States, Theoretical Division (T-4), Los Alamos National Laboratory, Los Alamos, New Mexico, United States, and Research Institute of Advanced Computer Science, Universities Space Research Association, Mountain View, California, USA; e-mail: satheprat ik@gmail.com; Joshua Apanavicius, Applied Physics Laboratory, Johns Hopkins University, Baltimore, Maryland, United States; e-mail: apanavicius.josh146@gmail.com; David Bernal Neira, Research Institute of Advanced Computer Science, Universities Space Research Association, Mountain View, California, United States, Quantum Artificial Intelligence Laboratory, NASA Ames Research Center, Mountain View, California, United States, and Purdue University System, West Lafayette, Indiana, United States; e-mail: dbernalneira@usra.edu.

ACM 2643-6817/2024/08-ART18
https://doi.org/10.1145/3678184

DAVID BERNAL NEIRA, Research Institute of Advanced Computer Science, Universities Space Research Association, Mountain View, United States, Quantum Artificial Intelligence Laboratory, NASA Ames Research Center, Mountain View, United States, and Purdue University System, West Lafayette, United States
QUANTUM ECONOMIC DEVELOPMENT CONSORTIUM (QED-C) COLLABORATION†

Combinatorial optimization is anticipated to be one of the primary use cases for quantum computation in the coming years. The Quantum Approximate Optimization Algorithm and Quantum Annealing can potentially demonstrate significant run-time performance benefits over current state-of-the-art solutions. Inspired by existing methods to characterize classical optimization algorithms, we analyze the solution quality obtained by solving Max-cut problems using gate-model quantum devices and a quantum annealing device. This is used to guide the development of an advanced benchmarking framework for quantum computers designed to evaluate the trade-off between run-time execution performance and the solution quality for iterative hybrid quantum-classical applications. The framework generates performance profiles through compelling visualizations that show performance progression as a function of time for various problem sizes and illustrates algorithm limitations uncovered by the benchmarking approach. As an illustration, we explore the factors that influence quantum computing system throughput, using results obtained through execution on various quantum simulators and quantum hardware systems.

CCS Concepts: • **Computing methodologies** → **Modeling and simulation**; *Simulation types and techniques*; • **Information systems**; • **Applied computing** → **Physical sciences and engineering**; **Operations research**;

Additional Key Words and Phrases: Quantum Computing, Benchmarks, Benchmarking, Algorithms, Application Benchmarks, QAOA, Quantum Approximate Optimization Algorithm, Max-cut

ACM Reference Format:
Thomas Lubinski, Carleton Coffrin, Catherine McGeoch, Pratik Sathe, Joshua Apanavicius, David Bernal Neira, and Quantum Economic Development Consortium (QED-C) collaboration†. 2024. Optimization Applications as Quantum Performance Benchmarks. *ACM Trans. Quantum Comput.* 5, 3, Article 18 (August 2024), 44 pages. https://doi.org/10.1145/3678184

1 Introduction

In many application domains, it is of utmost importance to efficiently find near-optimal solutions to problems that involve many variables that affect the cost of some operation or function. For example, in a large power grid, rapidly determining the best allocation of power distribution could prevent a major blackout. These are known as combinatorial optimization problems and are often cited as a potential use case for quantum computing [1–3].

Classical computer algorithms for addressing such problems are substantially advanced and are implemented across industry, government, and academia. They perform critical functions in optimizing resource utilization and minimizing cost. Combinatorial optimization applications are often executed under tight resource constraints (e.g., time, memory, energy, or money), and there is particular emphasis on quantifying the quality of results that could be obtained within a limited budget. Standard techniques for measuring and comparing the performance of alternative solution methods have matured and are in widespread use [1]. An illustrative example of a performance profile is shown in Figure 1.

Quantum computing introduces new techniques for finding solutions to such combinatorial challenges, such as **Quantum Annealing (QA)** [4, 5] and the **Quantum Approximate Optimization Algorithm (QAOA)** [6] that may demonstrate some benefit over classical approaches. Theory and classical simulations indicate that, for some problems, QAOA has the potential to

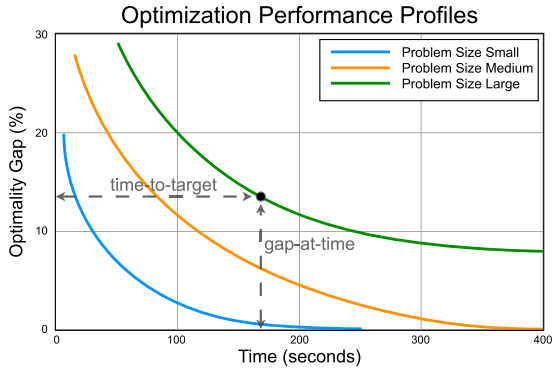


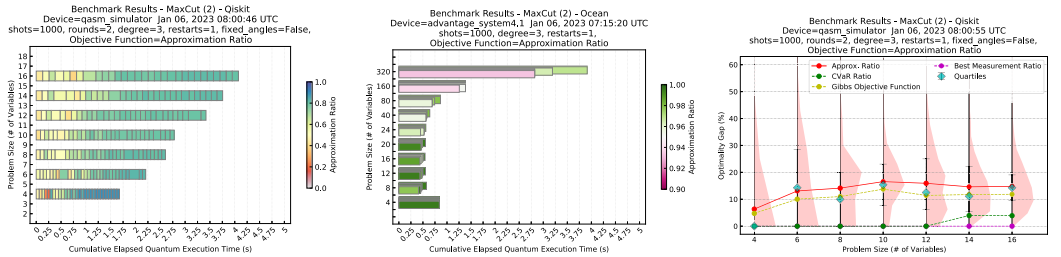
Fig. 1. Illustration of a performance profile for benchmarking optimization methods. *Performance Profile* plots, like the one shown here, are widely used by the Operations Research community to understand, communicate, and compare the performance of optimization methods. The quality of the solution, as the relative difference from optimal (optimality gap), can evolve over time during the execution of an optimization algorithm. This permits the user to gauge the time required to obtain a solution of a desired quality (time-to-target) or the solution quality achieved after a specified amount of time (gap-at-time). The gap-at-time metric is the *de facto* standard used in Operations Research, reflecting use cases for most industrial optimization applications. Performance profiles tend to change with problem size. It is common for problems with a small number of decision variables to converge to an optimal solution reasonably quickly. In contrast, with larger problems, achieving solutions above a quality threshold can be difficult, which is expected due to the NP-HARD nature of challenging optimization tasks.

outperform classical algorithms [7, 8], and some empirical tests of QA systems have demonstrated superior performance over classical alternatives in limited scenarios [3, 9–11].

Numerous efforts have emerged to characterize the performance of quantum computers for applications in optimization (see Section 2). However, we find that for such benchmarks to be accessible to users outside the quantum research community, they must both incorporate emerging methods for quantum computing benchmarking and present results meaningfully to experts in domains such as classical optimization and **Operations Research (OR)**.

In this article, we demonstrate how a properly constructed benchmark program that monitors and characterizes the execution of a combinatorial optimization application on a quantum computing system can provide valuable and critical insights into options for improving its performance and overall throughput. Additionally, analysis and presentation methodology can be structured in ways familiar to quantum computing specialists but are informed by how Operations Research views the quality of results from a solver in addressing optimization problems. These enhanced analysis and visualization techniques can provide useful information about the throughput a quantum computing solution can offer and the factors that can be adjusted to improve performance on these systems. While component and simple application-level benchmarks provide useful information about general performance characteristics, the optimization application supplements this with a detailed understanding of a quantum optimization application's total cost of ownership. While these techniques have long been used in Operations Research, their effective application to quantum computing is still in the early stages.

Concretely, we introduce a methodology and versatile framework for characterizing the performance of combinatorial optimization solvers executed on quantum computing systems based on different underlying technologies. We demonstrate this framework's features and highlight its benefits using the QAOA algorithm for execution on gate model systems. We also demonstrate its adaptability to other types of solvers by using QA on annealing hardware. In future work, we plan



(a) (b) (c)

Fig. 2. Characterizing performance of quantum computing solutions. These new performance profiles depict the trade-off between result quality and execution time for two quantum computing solutions to the unweighted Max-cut problem: (a) the Quantum Approximate Optimization Algorithm and (b) Quantum Annealing. Each row shows a different problem size. The X-axis displays the cumulative execution time, and the rectangle color measures the solution quality defined by the approximation ratio ($= 1 - opt\_gap/100$). For QAOA, successive rectangles depict its iterative execution, tracking the search for appropriate parameters to converge to an optimal solution. For QA, each stacked rectangle represents a distinct execution at increasing anneal times. Shown in panel (c) are several measures of algorithm success, variants of the approximation ratio, plotted over the distribution of final measurements at each problem size.

to extend this to include other technologies, such as cold atoms. We demonstrate the capabilities of our framework using the widely studied the Max-cut [12, 13] problem, in which the goal is to find the maximum cut size of an undirected graph. The Max-cut problem offers a simple early stage target for evaluating the effectiveness of quantum computing solutions. These solutions could also scale to larger applications and incorporate constraints and other problem features that escalate the challenge.

The new optimization application benchmark is provided as an enhancement to the existing open-source QED-C Application-oriented Benchmark suite [14, 15]. This is a diverse collection of algorithmic benchmarks for evaluating the performance of (gate-model) quantum computers on problems not currently related to optimization, with support for execution on multiple systems and for collecting, analyzing, and uniformly presenting performance metrics. Basing our work on this existing framework enabled us to readily extend it with new functionality and make it easy to use and accessible to a broad audience.

The new benchmark exercises multiple components of the integrated hybrid quantum-classical computer systems on which quantum optimization applications run and mimics their real-world use. Combining existing metrics visualization tools with new techniques specific to optimization problems enables the deep exploration of algorithm performance across target systems. To illustrate these analytics features, we present in Figure 2 several visualizations generated by this new benchmark. While many prior benchmarking studies use Max-cut as an example of an optimization application, our approach provides a unique level of flexibility, generality, and customization.

We note that the work described in this article does not include a full-scale comparison between quantum computing systems of different types. Nor does it address benchmarking of classical solutions to optimization problems, as numerous in-depth studies exist in this area (see Section 2.4). The performance results in this article are intended *primarily* to illustrate features and benefits of our benchmarking framework and not to meet methodological expectations for heuristic performance studies from Operations Research.

The inclusion of benchmarking in the QA algorithm has an essential purpose. QA has been extensively studied over decades, and its performance characteristics on annealing hardware are well understood. Including QA illustrates how our benchmark framework readily adapts to

quantum computing technologies other than the gate model. We use QA as a proxy for other solvers that may use quantum technologies in which a large part of the algorithm is executed within the hardware of the remote computing service. We demonstrate how the framework manages the execution of a series of benchmark problems and collects and analyzes metrics consistently across different technologies.

Our work has identified many variables that impact how well a quantum computer will solve an optimization problem. However, we did not perform an exhaustive study of these, nor could we tune vendor-specific hardware settings in all cases to achieve optimal results. As a result, the performance outcomes presented should not be considered generalizable to other test scenarios. A full-scale study of all the factors contributing to quantum performance to tease out the separate contributions of algorithms and hardware is beyond the scope of this article.

Our contributions to quantum benchmarking are threefold:

— Developing a methodology for evaluating the performance of quantum computers running on heterogeneous quantum platforms inspired by standard procedures for assessing classical optimization heuristics.
— Implementing and demonstrating an open-source benchmarking procedure for optimization applications that integrates smoothly with the evolving QED-C benchmarking framework and allows users to implement their performance studies easily.
— Illustrating the capabilities of this framework and the types of performance analysis that it can support using a familiar NP-HARD problem of interest to applications. As an example, we focus on throughput analysis of the application executed on several quantum hardware backend systems as a factor contributing to the total cost of using quantum solutions.

We hope this work sheds light on the practical considerations associated with implementing combinatorial optimization solvers on quantum computing systems and will encourage and enable others to measure and record progress in developing quantum algorithms and computing systems. We propose that the framework could be used to explore many of the recent innovations in quantum algorithms for optimization problems [16–22, 22–24].

The remainder of this article is structured as follows. Background on fundamentals of benchmarking the performance of quantum computers and their applications is provided in Section 2. Enhancements to the QED-C Application-oriented Benchmarks suite are described in Section 3, where we describe the benchmark algorithms. This is followed by a discussion on how we analyze and present the metrics collected by the benchmarks in Section 4. Results from execution on classically implemented quantum simulators validate that results match expectations and highlight the insights that can be gleaned from these benchmarks.

In Section 5, we analyze results obtained from executing these benchmarks on two gate-model quantum hardware systems and a quantum annealing processor using the methods described in the previous section. Several appendices are provided at the end of the manuscript to provide detailed information about quantum solutions to combinatorial optimization and to highlight factors that impact the quality of the result, trade-offs in parameter selection, and challenges in scalability inherent in quantum algorithms.

2 Background

The benchmarking framework measures performance characteristics of the two leading quantum heuristics for solving combinatorial optimization problems: the Quantum Approximate Optimization Algorithm, which uses a gate-model quantum computer, and Quantum Annealing, which uses an analog quantum computer.

This article presents a benchmark of these algorithms in the context of their application to solving the Max-cut problem. This section provides an overview of the problem's characteristics

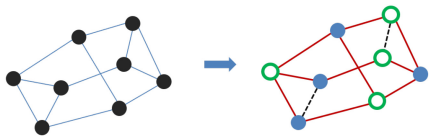


Fig. 3. Max-cut problem. For an undirected graph consisting of nodes, or vertices (V) and edges (E), partition the vertices into complementary sets such that the number of edges between the sets is the greatest. This graph shows one solution to one instance of the Max-cut problem for a graph with eight nodes, using colored nodes and edges. Nodes with different colors belong to the two sets of the solution cut. The number of solid red edges that connect nodes from different sets is the Max-cut of that graph.

and then reviews the quantum solutions we propose to benchmark. We also offer a quick review of existing methodologies for benchmarking and discuss the similarities and differences of our approach with them.

2.1 Max-cut Optimization Problem

The Max-cut problem has emerged as a popular benchmark for quantum optimization [12, 25–27] for two reasons: (1) it is among the most challenging combinatorial optimization tasks, even to obtain an approximate solution, i.e., APX-Hard [13, 28], (2) as an unconstrained discrete optimization task, it has a natural encoding as a **Quadratic Unconstrained Binary Optimization (QUBO)** [29, 30] or an Ising model [5, 31], ideally fitting current quantum optimization algorithms (QAOA, QA). These problems often arise when mapping practical applications [32, 33] to computing hardware and can appear as subroutines in composite algorithms.

The input for a Max-cut problem is an undirected graph consisting of nodes or vertices (V) and edges (E). (Each edge of the graph can be accompanied by a "weight," but we only consider unweighted 3-regular graphs in this article.) A cut is a partition of the graph nodes into two sets. The size of a cut is defined as the number of graph edges that connect nodes belonging to different sets. The Max-cut problem is identifying a cut with the largest size of all possible cuts. (Here, we consider the unweighted version of the Max-cut problem, so that each edge of the graph has the same weight.)

Figure 3 presents a representative eight-node graph in which each node is connected to others by precisely three edges. This type of graph is known as 3-regular or of degree 3. Graphs of degree 3 are considered the most difficult to solve [34]. This figure shows one solution to the Max-cut problem, using colored nodes and edges. The number of red edges that connect the nodes is the maximum cut of the graph.

Due to the challenges associated with finding even approximate solutions to the Max-cut problem at a larger scale (for both classical and quantum algorithms), a quantity called the approximation ratio is often computed to characterize the quality of the solution obtained. For example, in the problem depicted in Figure 3, the Max-cut size is 10 (of 12 total edges). A naive optimization algorithm that randomly tested various cuts but ran out of time to test them all might conclude that the largest cut size was 9. In this case, the approximation ratio would be a number smaller than 9/10 or 0.90, as it is a statistical function of the distribution of all solutions found.

It is common to report the quality of a result in terms of its distance from an optimal solution when benchmarking classical algorithms for optimization problems. The optimality gap is related to the approximation ratio by Equation (1):

$$optimality\_gap = (1.0 - approximation\_ratio) \times 100. \tag{1}$$

Both of these measures of quantum system performance are relevant in distinct contexts. In our work, we incorporate both metrics to facilitate discussions on the outcomes attained with our benchmarking implementation.

2.2 Quantum Algorithms for Optimization Problems

Quantum annealing and circuit-based quantum computers solve a combinatorial optimization problem using fundamentally different strategies. To provide context for our work, we briefly outline how these quantum algorithms function to find solutions to these problems. Additional detail about these algorithms is provided in Appendix B.

Optimization problems, such as Max-cut, can be described by a Hamiltonian H_P that is unique to the problem and represents its variables and constraints. The optimal problem solution then corresponds to this Hamiltonian's ground state(s). The QAOA and QA algorithms use quantum state evolution to compute the energy expectation value for H_P and identify values for variables $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ that yield the lowest energy eigenstate(s) for

$$F_{\boldsymbol{\beta},\boldsymbol{\gamma}} \coloneqq \langle \boldsymbol{\beta}, \boldsymbol{\gamma}| \, H_P \, |\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle \, . \tag{2}$$

The quality of solution, or approximation ratio (*AR*), can then be defined as the ratio of the computed energy state $F_{\boldsymbol{\beta},\boldsymbol{\gamma}}$ and the true ground state energy $E_{\min}$ (assuming it is known). In our case, the Hamiltonian energies are always negative or zero:

$$AR = F_{\boldsymbol{\beta},\boldsymbol{\gamma}} / E_{\min}. \tag{3}$$

Quantum Approximate Optimization Algorithm. QAOA is arguably the leading candidate for solving combinatorial optimization problems using gate-model quantum processors. QAOA belongs to the class of **Variational Quantum Algorithms (VQA)** [35] and is usually implemented iteratively wherein a classical optimizer "trains" a parameterized quantum circuit. QAOA is a heuristic that attempts to solve combinatorial optimization problems, such as QUBO problems. Specifically, the problem is encoded in the form of a specified quadratic function of binary variables, and the objective is to find an assignment for those variables that minimizes the function.

At the core of QAOA is an "ansatz circuit," a parameterized quantum circuit. Measurements in the computational basis at the end of the circuit correspond to sampling from a probability distribution over possible answers to the problem. A classical optimizer is used to obtain parameter values likely to produce optimal or near-optimal solutions by repeatedly taking circuit measurements while varying parameter values.

Quantum Annealing. Quantum annealing effectively addresses optimization problems by using a versatile approach to identifying the global minimum of a function using a systematic process. The algorithmic approach of QA is inspired by the adiabatic theorem from quantum mechanics to transform an easy-to-prepare ground state of an initial Hamiltonian into the ground state of the "target" Hamiltonian that encodes the combinatorial optimization problem.

At a high level, the protocol strives to identify the low-energy states of a user-specified H_{Target} model by conducting an analog interpolation process of the following Hamiltonian, arriving at minimum energy states at the end of the evolution:

$$H(s) = (1 - s)H_{\text{Init}} + (s)H_{\text{Target}}. \tag{4}$$

With quantum annealing, convergence to a solution is performed entirely within the quantum system from the user's view. The problem is mapped to an initial state (equal superposition with respect to the problem basis) on the hardware, and the system is set to anneal toward a solution. Longer annealing times are associated with higher solution quality.

2.3 Benchmarking Quantum Computers

This section reviews concepts and definitions from prior benchmarking work that we reference throughout this manuscript. We focus primarily on the application-oriented level of performance evaluation in our benchmarking of combinatorial optimization applications.

System-level Benchmarks. A large body of reference material exists for gate model computing systems on component and system-level benchmarks [36–42]. We use two well-known system-level performance benchmarks, **Quantum Volume (QV)** [43, 44] and **Volumetric Benchmarking (VB)** [45, 46] as a backdrop in several of our benchmark plots. While these two methods characterize quantum circuit execution quality and scale, neither provides information about the time it takes a program to run, which is a critical factor in evaluating the total cost of any computing solution.

Quantum annealing systems have been available for empirical study since 2011 [47–49]. Examples of component- and system-level approaches to evaluating quantum annealing processors may be found in early papers [47, 50, 51] and in recent proposals for benchmarking of large-scale quantum annealing hardware [52–54].

Application-level Benchmarks. Component and system-level metrics offer valuable insights into overall system capability, but predicting the effectiveness of a machine with a certain level of general performance for a specific application class can be challenging [46, 55]. To address this, application-focused benchmarks run well-defined programs tailored to provide application-specific performance metrics.

Due to its relative maturity, benchmarking of QA tends to involve application-level tests using models with more than 100 qubits, which presents significant challenges for validation by comparison to the classical simulation of ideal quantum systems [3, 56–58]. Such benchmarking work often compares the runtime performance of quantum hardware to that of classical methods [3, 59–62]. Using synthetic optimization problems, a problem instance with a known optimal solution is planted [63–65].

In contrast, early stage gate model quantum computers require benchmarks that involve smaller problems and numbers of qubits. Application-oriented benchmark frameworks typically create circuits that use well-known quantum gate combinations or algorithms, provide inputs and expected outputs, and execute them on quantum simulators or physical hardware [14, 15, 66–70]. Result quality metrics are computed using statistical differences between expected and actual measurements or proximity to an application-specific metric derived from the measurements.

Of particular relevance is the first QED-C application-oriented benchmark suite [14, 15], upon which our work is based. The QED-C suite offers a practical methodology to evaluate the performance of various quantum programs across a range of quantum hardware and simulator systems. Its benchmark programs sweep over a range of problem sizes and input characteristics while systematically capturing key performance metrics, such as quality of result, execution run-time, and quantum gate resources consumed, as shown in Figure 4. Supporting infrastructure and abstractions make these benchmarks accessible to a broad audience. The framework also provides the structure to enable benchmarking of iterative algorithms, such as QAOA, or to execute an algorithm in a single operation, as in QA.

The QED-C benchmarks compute several important figures of merit, which we use throughout this manuscript. The quality of result for individual circuits is given by the "Normalized Hellinger Fidelity," a modification of the standard "Hellinger Fidelity" that scales the metric to account for the uniform distribution produced by a completely noisy device. Resource consumption is quantified as the total number of gate layers, or "Circuit Depth," which can be "Algorithmic" or "Normalized"

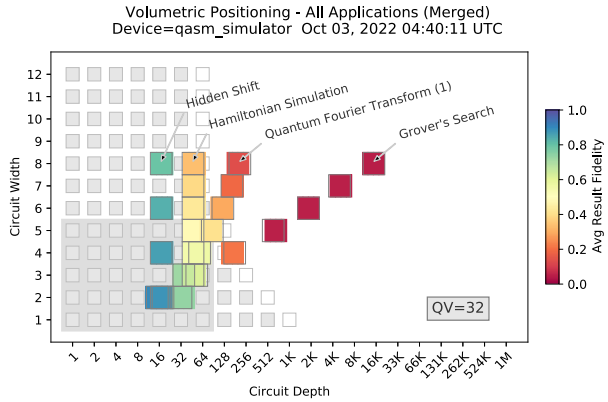


Fig. 4. Application-oriented benchmarks. Here, we present one way to illustrate the result fidelity obtained by executing several application-oriented benchmarks up to 8 qubits on a noisy quantum computer simulator. The plot shows the average result fidelity as a function of circuit width and the circuit depth plotted on a volumetric background to visualize the "profile" and result fidelity of the benchmark circuits. *(Plot produced by QED-C benchmark suite.)*

(transpiled to a normal basis). Execution time is captured as "Elapsed Quantum Execution Time" (wall clock) or "Quantum Execution Time" (reported by quantum provider service) with more granularity possible in some systems. These metrics are defined in detail in our prior work [15].

The framework to capture run-time metrics is fundamental to our new work in benchmarking algorithms for combinatorial optimization. Although execution time was included in the first QED-C benchmarks, benchmarking iterative and hybrid algorithms can provide a more complete picture of a quantum computer's run-time performance. The time required to execute circuits within a quantum application is an essential metric of performance [71–74], often cited when comparing quantum and classical computations. For example, quantum and classical computing times were central to recent demonstrations of quantum advantage [75, 76].

2.4 Quantum Benchmarks for Optimization Tasks

Much background material is available on benchmarking classical solutions to optimization problems [3, 62, 77] and comparing quantum methods to classical solutions [78–80]. This work is often oriented toward improving the performance of quantum algorithms, separate from the analysis of specific hardware platforms. With QA and QAOA, this approach generally involves comparing different implementations or tuning strategies of the quantum algorithm rather than performing runtime comparisons against classical algorithms.

Quantum computation based on quantum annealing techniques has been used to address combinatorial optimization problems for more than a decade [3, 9, 11, 62, 80–83]. In these studies, the primary metrics of interest are expected solution quality and *operation counts* (i.e., circuit depth), which are used as approximate runtime measurements. Within this context, two main threads of execution time analysis have been used in comparing the performance of the quantum annealing algorithm to classical heuristics. The ***time-to-solution* (TTS)** metric determines the expected wall-clock time required to solve a problem to optimality [48, 82–85]. In contrast, the ***time-to-target* (TTT)** metric determines the expected wall-clock time required to solve a problem to a specified *target* solution quality [3, 9, 56, 80]. The typical approach is to measure how TTS or TTT changes as a function of problem size for both the classical and quantum methods (i.e., a scaling advantage) so that one can forecast at what system sizes the quantum solution approach is

likely to be faster than the classical one. In this work, we use these concepts to inform our analysis of the trade-off between the quality of the solution and the time it takes to execute the quantum algorithm.

Significant challenges exist for benchmarking quantum solutions on real-world hardware, as quantum computer noise characteristics and runtime overhead introduce additional requirements. Due to the implementation complexity of configuring optimization tasks for benchmarking quantum hardware, several software frameworks have emerged to support the evaluation of the same (or similar) optimization methods implemented on different platforms. Benchmark frameworks such as SupermarQ [67] and QPack Scores [68] include one or more QAOA applications in their sample benchmarks, while QUARK [69] considers specific optimization problems arising in industry. The Q-score metric [70] is claimed to apply to quantum processors in several categories, measuring the size of the largest graph for which the solver outperforms random guessing within a fixed time limit. All references present results that measure solution validity, feasibility, and runtime on several backend quantum computers, some on both gate model and quantum annealing devices. We used much of this work to guide our development of new benchmarks in the QED-C suite based on combinatorial optimization problems.

3 QED-C Benchmark Framework Enhancements

In this section, we describe the enhancements to the QED-C benchmark suite that collect and analyze application-specific quality and temporal metrics associated with hybrid quantum applications, e.g., QAOA and QA, where the trade-off between the quality of solution and utilization of resources (here execution time) is essential. Our effort has two primary goals: (1) to integrate and enhance critical concepts from other optimization-centric benchmark efforts into the QED-C benchmark suite as a standard feature and (2) to present the results and analysis in ways recognizable by practitioners in the operations research field who are already familiar with benchmarking classical solutions to optimization problems.

This section describes specific features of the framework we developed for cross-paradigm quantum optimization heuristics benchmarking. The enhancements to our original benchmark framework are driven by specific features and challenges of optimization problems and heuristic performance evaluation, distinct from the simple test scenarios used in our initial benchmark suite. The OR community has developed methodologies and tools for evaluating computational performance in this context, some of which we have adapted to the quantum scenario. See Appendix A for a discussion of the theoretical foundations.

Several features distinguish our benchmarking framework from others. Users can evaluate both execution time and solution quality in detail and explore the trade-offs (as opposed to fixing a specific TTS or TTT metric). The platform supports benchmarking of quantum computing hardware that can run quantum annealing or gate model algorithms. It also provides the ability to select problems and inputs of interest beyond our simple illustrations using Max-cut inputs. Presentation of benchmark results is aligned with standard methodologies of Operations Research and the QED-C framework. As quantum computers grow in size, the benchmark framework will be able to support testing on a wide variety of optimization problems.

We focus on its application to a combinatorial optimization problem to demonstrate the key enhancements, using Max-cut as a specific example. Unlike the simple algorithms used in the initial benchmark suite, where circuit execution fidelity is the key metric, the enhanced benchmark must derive a solution quality metric that is application-specific and accounts for the fact that solutions to optimization problems are often approximate. Results from its execution on a classically implemented circuit-based quantum simulator illustrate how key metrics are collected and presented.

ALGORITHM 1: Benchmark Algorithm for QAOA

```
target ← backend_id
initialize_metrics()
for size ← min_size, max_size do
    circuit_def ← define_problem(problem, size, args, rounds)
    for restart_id ← 1, max_restarts do
        cost_function ← define_cost_function(problem)
        circuit, num_params ← create_circuit(circuit_def)
        cached_circuit ← compile_circuit(circuit)
        params[β, γ] ← random(num_params)
        while minimizer() not done do                          ▷ minimizing
            circuit ← apply_params(cached_circuit, params)
            counts ← execute(target, circuit, num_shots)
            energy, quality ← cost_function(counts)
            store_iteration_metrics(quality, timing)
            params[β, γ] ← optimize(params[β, γ, energy])
            done ← True if lowest(energy) found
            done ← True if iteration_limit_reached()
        end while
        compute_and_store_restart_metrics()
    end for
    compute_and_store_group_metrics()
end for
```

The QED-C benchmark framework includes shared functions to manage the execution of benchmark algorithms over a range of problem definitions, collect metrics during execution, and present results consistently across backend targets. Both the Max-cut QAOA and QA benchmark algorithms operate on a target system *backend_id*, sweeping over a range of problem sizes [*min_size*, *max_size*], to solve a *problem* defined by input *args*. An inner loop, controlled by the *max_restarts* argument, provides the ability to execute the benchmark algorithm multiple times at each problem size.

For each problem size tested, we consider a set of random 3-regular graphs using the *networkx* package [86] and determine the maximum cut size for each using the *gurobi* package [87]. These are used to generate the quantum circuits for testing and to determine solution quality after execution, respectively.

A key practical difference between the QAOA and QA algorithms lies within the restart loop, where the specific solvers are applied to the input, and the quality of the solution is evaluated over increasing execution times. A gate model device iterates through a series of quantum circuit executions, testing parameter values, to find a set that yields a low-energy state. In contrast, a quantum annealing system gradually attempts to reach its lowest energy state as a transverse Ising model undergoing quantum mechanical evolution. Comparing the evolution of the state over time in these systems requires different data collection and presentation. Below, we detail the related algorithms used to benchmark these solutions and highlight differences between them and how they impact the results, omitting some details for brevity.

3.1 Benchmark Algorithm for QAOA

The QAOA benchmarking method is defined in Algorithm 1. Nested within the first and second for loops is the QAOA algorithm, which defines a cost_function based on the problem specifics and a gate model quantum circuit that implements the Hamiltonian associated with the problem and

ALGORITHM 2: Benchmark Algorithm for QA

```
target ← backend_id
initialize_metrics()
for size ← min_size, max_size do
    for restart_id ← 1, max_restarts do
        compute_quality ← define_compute_quality(problem)
        for a_time ← min_anneal_time, max_anneal_time do
            embedding ← define_problem(problem, size, args)
            sampler ← create_sampler(target, embedding)
            samples ← sample_ising(sampler, a_time, reads)
            quality ← compute_quality(samples)
            store_iteration_metrics(quality, timing)
        end for
        compute_and_store_restart_metrics()
    end for
    compute_and_store_group_metrics()
end for
```

is parameterized by variables $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. The quantum circuit used with QAOA can be replicated by some number of *rounds* (often referred to as p in code).

Starting with a random or fixed set of parameters *params* (corresponding to $|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle$ from Equation (11). the quantum circuit is executed *shots* times to obtain the measurement *counts* and compute a value for the cost function. Classical optimizer code explores the parameter landscape by varying the set of parameters to obtain measurement counts representing the Hamiltonian's lowest energy state, iterating until either the lowest energy is determined or an iteration limit is reached. A relevant *quality* metric is calculated and stored along with metrics that track the quantum and classical *timing* information. Although we use random starting parameters for the benchmark, users may have some information about a reasonable starting point in practice, which could result in a better or faster solution (see Appendix E.2).

The results of this algorithm's execution can be affected by a number of factors unique to QAOA, such as the number of shots and rounds, the type of classical optimizer employed, and, most importantly, the noise level in the target system. The quality of the results can also be constrained by limiting the number of iterations the classical optimizer performs. In the QAOA benchmark, this is a configurable option, but we set this limit to 30 by default to avoid runaway execution on costly hardware.

The Max-cut benchmark can be executed using two different methods, enabling the study of these factors independent of the complete QAOA algorithm. Method (1) executes one instance of the ansatz circuit for a specific problem using configurable shots and rounds, permitting detailed analysis of these factors. Method (2) executes the complete QAOA algorithm and provides the option to specify the classical optimizer, with COBYLA as the default. Additionally, one of several variants of the cost function may be selected. These variants are described below in Section 4.1.

3.2 Benchmark Algorithm for QA

The QA benchmarking method is described in Algorithm 2. The core of the QA benchmark algorithm is within the first and second `for` loops. It uses a special metrics collection loop unique to the QA benchmark.

From the user's perspective, the convergence to a solution using a quantum annealer is performed in a single step, entirely within the quantum system. A Hamiltonian describing the

problem is embedded into the quantum components of the device and is evolved in time to settle to the lowest energy state, representing an optimal solution.

In our QA benchmark, the QA algorithm is executed multiple times *from the start* within this particular collection loop. The *anneal_time* is initialized to 1μs and doubled after each execution until it reaches 256 μs (the range of annealing times is a configurable option in the benchmark). Each time, the problem is mapped to an initial state (equal superposition with respect to the problem basis) on the hardware, and the system is set to anneal toward a solution. Longer annealing times are associated with higher solution quality.

Using this approach, we are able to provide a measure of the time versus quality trade-off comparable to the QAOA benchmark by capturing the quality of the solution obtained at each value of the annealing time. It is impossible to monitor the solution's quality as it evolves within a single execution of the QA algorithm.

For each of the executions performed in the benchmark, the algorithm queries (*reads*) *samples* the same number of times as we do shots in QAOA. As with benchmarking QAOA, a relevant *quality* metric is calculated after each execution and stored along with metrics that track the quantum and classical *timing* information.

The results of this algorithm's execution are affected primarily by the number of shots or samples, the annealing time used, and the noise level in the target execution system. The Max-cut benchmark can be executed using two different methods, enabling the study of these factors independently of the entire QA algorithm. Method (1) executes at a single, configurable annealing time for each problem size using configurable shots, permitting detailed analysis of these factors. Method (2) executes the complete QA described in this section algorithm. Note that in this case, Method (1) is the same as Method (2), with the range collapsed to a single value.

4 Analysis of Max-cut Benchmark Metrics

In this section, we discuss our analysis of the data collected as the benchmarks execute. First, we discuss the application-specific quality metrics explicitly computed for the Max-cut problem, produced in common for both QAOA and QA algorithms. The benchmarking framework generates optimality gap plots and cut-size distribution plots as practical tools to visualize these additional metrics succinctly. Furthermore, metrics other than the approximation ratio can also be used to assess result quality, which we discuss in this section.

We follow this with our visualization of the trade-off between execution time and the quality of the result that can be obtained. While displaying these data in simple line charts is typical, we generate informative plots that use color to represent the quality of execution, horizontal length to represent the execution time of individual iterations, and vertical/horizontal position to indicate problem size and the cumulative execution time. The section ends with our analysis of the effect of shot count and the number of rounds on the quality of result for the QAOA implementation of the benchmark.

4.1 Application-specific Result Quality Metrics

The QAOA and QA algorithms attempt to converge on a solution to an optimization problem by finding the lowest energy state of a Hamiltonian after a sequence of parameter tests (QAOA) or annealing operations (QA) on a target system. In both cases, the result is an energy value computed as a function of the distribution of the energy samples obtained at the end of the execution.

This energy value can be compared against the precomputed optimal solution (i.e., the lowest energy of the Hamiltonian). The ratio between the actual and expected energy values, or the approximation ratio [6, 30, 88], is the metric commonly used to quantify the quality of the results for the Max-cut problem, as discussed in Section 2.1. It is formally defined as follows.

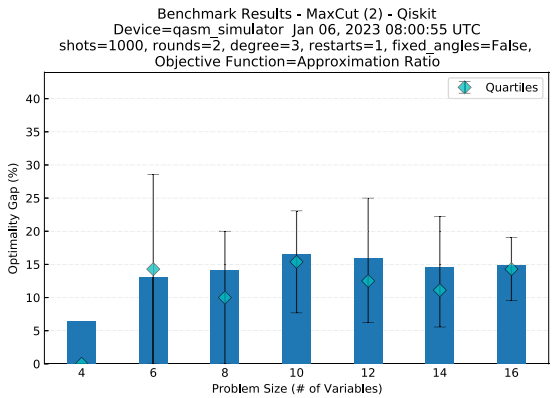


Fig. 5. Showing final optimality gap. As the QAOA program executes, the minimizer finds an optimal solution to the Max-cut problem, represented by the approximation ratio computed after the final iteration. The optimality gap for each problem size is computed from these values and shown in this bar chart, along with quartile marks showing the distribution of the final measurement results. In some cases, the quality of the results could improve with additional execution time, but we limit the benchmark to 30 iterations to conserve computing resources.

Let M denote the number of shots so that for given values of (vectors) $\boldsymbol{\beta}, \boldsymbol{\gamma}$, we obtain M cut-sizes, one corresponding to each of the bit-strings obtained from computational basis measurements. Let these energies be denoted by $E_1, \ldots, E_M$, arranged in non-decreasing order. Since these energies are ≤ 0, $|E_1|, \ldots, |E_M|$ are non-negative integers arranged in non-increasing order. Then, the energy expectation value is approximated by

$$F \approx \frac{\sum_{i=1}^{M} E_i}{M}. \tag{5}$$

Normalizing the result of this computation to the range $[0, 1]$ is convenient. Hence, we define the approximation ratio in

$$\text{Approximation Ratio} = \frac{F_{\boldsymbol{\beta},\boldsymbol{\gamma}}}{|E_{\min}|}, \tag{6}$$

where $E_{\min} < 0$ is the actual ground state energy of the problem Hamiltonian.

Optimization performance studies typically use the complement of this, the optimality gap. In Figure 5, we show the final optimality gap computed at the end of the execution of the benchmark algorithm for each of the problem sizes tested. In this plot, we include quartile bars, which provide information on the width of the distribution in addition to the mean. Although these results were produced using the QAOA algorithm on a classically implemented quantum simulator, the results of our QA algorithm are also plotted in this fashion.

The approximation ratio is a valuable measure of a quantum computing system's ability to solve the Max-cut problem. The higher the mean of the cut sizes found in the distribution, the more likely the algorithm will produce a cut size that is the maximum. However, we note that the final output of these quantum algorithms is not the approximation ratio but the best-measured cut (i.e., the cut corresponding to the largest cut size) obtained across all iterations of the algorithm.

The best-measured cut is often a poor measure of the quality of the result, because it is numerically unstable, particularly with smaller numbers of samples. However, it has inspired a few other objective functions, such as the Conditional Value at Risk or CVaR [89], and the Gibbs

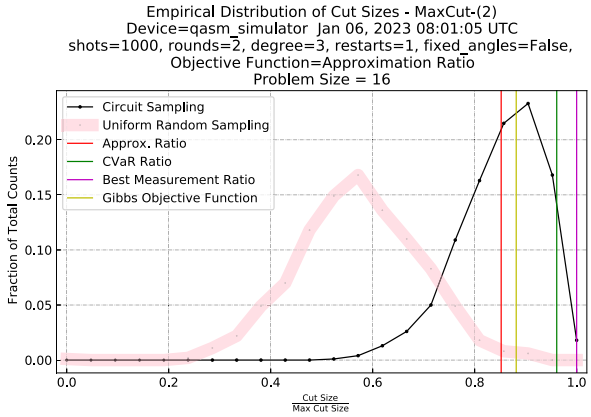


Fig. 6. Cut-size distribution: The quality of the final output of QAOA can be understood by inspecting the distribution of the cut-size values obtained at the final optimizer iteration. A distribution peaked closer to the right indicates higher result quality. Also plotted here are the various metrics (vertical lines) and the distribution corresponding to a uniform random sampling of bit-strings (pink line).

objective function [90]. Both metrics focus on the tail end of the distribution rather than treating all measurements equally.

To illustrate how these different quality metrics relate to each other, in Figure 6, we illustrate the distribution of cut sizes produced from the execution of our benchmark on a noiseless quantum simulator at a problem size of 16 with 1,000 samples taken (shots). The distribution obtained from our benchmark is shown with a black line. A wide pink line shows a simulated distribution that would be obtained by executing the algorithm on a computing system that returns uniformly random results. A distribution that peaks closer to the right indicates a higher result quality. In this case, the best-measured result is shown at 1.0, indicating that the algorithm returned the expected optimal Max-cut. The CVar and Gibbs ratios fall between the approximation and best-measured ratios.

Formally, CVaR [89], for a chosen value of parameter $\alpha \in (0, 1]$, is defined as

$$\mathrm{CVaR}_\alpha(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \frac{1}{\lceil \alpha N \rceil} \sum_{i=1}^{\lceil \alpha N \rceil} E_i, \tag{7}$$

where $\lceil . \rceil$ denotes the ceiling function. CVaR_α denotes the mean value of (the negative of) cut-sizes over the lower α-tail of the measured energy distribution. The limit $\alpha \to 0$ corresponds to the ground state energy value (i.e., E_1), while $\alpha = 1$ corresponds to the energy expectation value $F_{\boldsymbol{\beta},\boldsymbol{\gamma}}$. The value of the metric depends on the choice of α. While the value of this parameter can be configured, we default to $\alpha = 0.1$ in all plots and analyses.

Another choice is the Gibbs objective function [90], which is defined as

$$f_\eta(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \ln \langle \boldsymbol{\beta}, \boldsymbol{\gamma} | \, e^{-\eta H} \, | \boldsymbol{\beta}, \boldsymbol{\gamma} \rangle \,, \tag{8}$$

with $\eta > 0$, and where H denotes a Hamiltonian whose ground state is sought. The parameter η determines the relative weights of the low energy states of H in the expression. The parameter η tunes f_η between two extreme values (similar to α in CVaR): $f_{\eta=0}(\boldsymbol{\beta}, \boldsymbol{\gamma}) = F_{\boldsymbol{\beta},\boldsymbol{\gamma}}$, while $f_{\eta\to\infty} = E_{\min}$, i.e., the lowest measurable energy of H in the state $|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle$.

We normalize each of these objective functions so that they lie in the range $[0, 1]$ and thus define the following quantities:

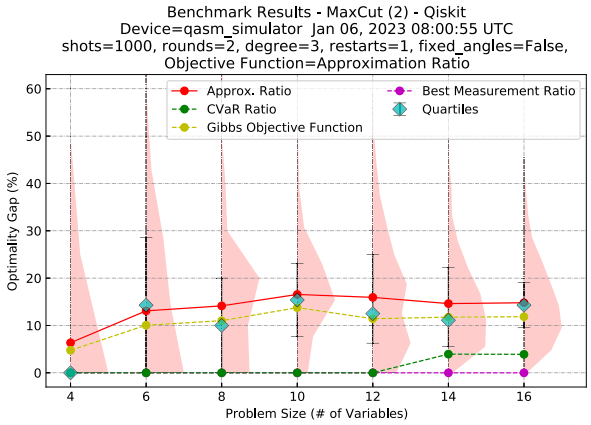


Fig. 7. Detailed optimality gaps plots. A variety of metrics can be used to assess the quality of the final distribution of outputs of QAOA. For each implemented problem size or circuit width (along the X-axis), we plot the optimality gap (along the Y-axis). The obtained distributions of optimality gaps are shown as pink half-violin plots. The optimality gap values regarding the CVaR ratio, approximation ratio, Gibbs Ratio, and Best Cut ratio are shown as line plots.

$$\begin{aligned} \text{CVaR}_\alpha \text{Ratio} &= \frac{\text{CVaR}_\alpha}{|E_{\min}|}, \\ \text{Gibbs Ratio} &= \frac{f_\eta}{\eta |E_{\min}|}, \\ \text{Best Measurement Ratio} &= \frac{E_1}{|E_{\min}|}. \end{aligned} \tag{9}$$

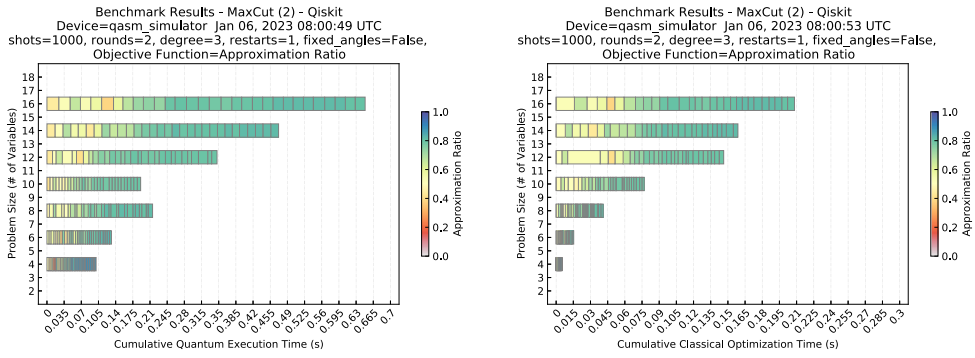
In our benchmarking framework, the objective function may be set to any of these. The approximation ratio is commonly used in studies of quantum computing solutions to optimization problems, and the other ratios appear less often in the literature. These are measures of the quality of the solution where a value of 1.0 is optimal.

Our framework also generates "detailed optimality gap plots" (e.g., Figure 7). For each problem size, the empirically obtained distribution of cut sizes is shown using a half-violin plot. (The plotted distribution is that of the quantity $1-\frac{\text{Cut Size}}{\text{Optimal Cut Size}}$, so it is normalized to be between [0,1]). The four metrics in Equations (6)–(9) are shown in terms of their optimality gap, i.e., $(1-metric\_value)*100$. This provides a detailed snapshot of the quality of the result as a function of problem size in terms of various quality metrics.

4.2 Result Quality and Time of Execution

In this section, we introduce a new method for visualizing the relationship between solution quality and execution run-time in the results from our Max-cut benchmark. The methodology is inspired by the typical visualizations used in Operations Research (e.g., the performance profile in Figure 1). Still, it is enhanced in ways that yield valuable insights about execution time unique to hybrid quantum computing algorithms. Some aspects of this approach are especially relevant in the early stages of quantum computing maturity. They provide critical information about bottlenecks and other drag factors that impact system throughput more than realized.

Our approach can be applied to both QAOA and QA, although the visuals vary slightly in ways that mirror algorithm differences. In Figure 8, we illustrate the time versus quality trade-off for the QAOA algorithm using a novel performance profile referred to as an "area plot." Similar to



(a) (b)

Fig. 8. Iterative execution of QAOA max-cut algorithm. In QAOA, a classical minimizer function iteratively executes an ansatz, varies its parameter values, and computes a cost function to converge to an optimal solution. In panel (a), each horizontal row represents successive iterations at each problem size (number of qubits), where the position on the X-axis represents the cumulative quantum execution time, and the color tracks the approximation ratio of each iteration of the optimizer. In panel (b), the X-axis represents the cumulative classical execution time (optimizer). Both of these times contribute to the total elapsed time that a user experiences.

the volumetric plot of Figure 4, it shows the circuit width (problem size) on the Y-axis and uses the color of rectangles to illustrate a metric score. In the area plot, the horizontal width of an individual rectangle represents the execution time for a single ansatz, and its location along the X-axis indicates the cumulative execution time, including prior iterations.

A key difference between the QAOA and QA benchmarks is in evaluating the execution times to be plotted on the X-axis. With QAOA, the algorithm inherently executes in a series of iterations, and the execution time accumulates with each, represented by stacked rectangles. With QA, however, the complete algorithm is executed in a single step. To evaluate how well the algorithm performs at different times, the algorithm is executed from the start each time with different annealing times. Figure 2(b) shows how we visualize the re-initialization with rectangles that overlap instead of being stacked. More detail about QA execution can be found in Section 5.3.

In the area plots shown here, we use the approximation ratio as the default figure of merit to gauge the quality of the result. The approximation ratio ranges from 0 to 1.0, but for QAOA, it usually starts above 0.5 and oscillates as the optimizer converges to a solution. Due to the annealing computer's nature, the QA's starting point is often above 0.9. We use a different color scale for QA to emphasize the fundamental difference between the algorithms. However, when running the benchmarks, any objective functions in the previous section can be selected as the figure of merit shown in the plots.

The benchmarking framework collects multiple measures of execution time. The first plot of Figure 8 shows the cumulative quantum execution time or the time spent executing the quantum processor. The second plot shows the cumulative classical execution time, primarily consisting of the time taken by the optimizer in QAOA or the setup time in QA. In Figure 2(a), we show the cumulative elapsed quantum execution time, which is the total wall clock time that includes both of these plus other setup times such as compilation or time to load the program into the quantum processor. We include a detailed analysis of these and other essential times related to QAOA and QA algorithms in Appendix D

The quality versus time visualization we use here, the area plot, significantly enhances the information presented to users about executing a hybrid quantum algorithm such as QAOA or QA. For example, this QAOA and QA evolution analysis can provide information about the *incremental time units consumed by execution.* With some of the newer hybrid systems and the use of error mitigation, it is extremely valuable to inform the user of the bottlenecks or anomalies in the execution.

For example, in the plots of Figure 8, the width of several of the rectangles representing the time of each iteration is not uniform. With quantum computing in its early stages of maturity, the execution pipeline often contains many steps that involve non-deterministic classical computation, some of which are unique to quantum computing, such as error mitigation. These plots effectively convey a measure of the level of unpredictability in the execution times that may contribute to throughput degradation.

Other types of information unique to quantum are also transmitted in these plots. For example, QAOA can require classical pre-processing, specifically compilation, and transpilation to a target topology and gate set from an intermediate representation. In contrast, QA requires embedding the problem graph onto the device topology before execution. With gate model computers, intermittent calibration processes can sometimes interrupt program execution and appear as rectangles with larger widths. In addition, there is often a start-up cost associated with executing any circuit component of these algorithms.

While we only show a few examples here, the area plots allow users to view all of these things at a glance and can assist them in quickly interpreting how execution time impacts the quality of results and overall throughput of the quantum algorithm.

4.3 Factors Affecting QAOA Ansatz Fidelity

Several factors can impact the results obtained from executing the QAOA algorithm. Here, we use Method (1) of the QAOA benchmark to analyze how the number of shots and rounds can affect a quantum computing system's ability to execute the ansatz circuit used in the algorithm.

Each iteration of the QAOA algorithm involves repeatedly measuring a parameterized circuit, executed with parameter values $|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle$ determined by a classical optimizer routine. The algorithm's success relies on the ability of the quantum subroutine to compute an accurate value of the objective function. If the measurement probabilities obtained by the quantum subroutine do not match sufficiently well with the probabilities from the ideal distribution $P_{\text{ideal}}(s) = |\langle s \mid \boldsymbol{\beta}, \boldsymbol{\gamma}\rangle|^2$, then the effectiveness of the classical optimizer can be negatively affected.

Even in a noiseless simulator, perfect fidelity can be achieved only within the limit of an infinite number of shots. On quantum hardware, noise and decoherence can exacerbate the drop in fidelity, as can limited connectivity between qubits. To quantify circuit fidelity, we use both the Hellinger fidelity and the normalized Hellinger fidelity as defined in our initial work on application-oriented benchmarks [15]. The normalized fidelity is most useful in our context, recalling that a circuit fidelity of 0 corresponds to a uniformly random probability distribution. In contrast, a fidelity of 1 corresponds to the ideal distribution.

In Figure 9, the number of measurements per iteration (which we call the number of "shots") is shown to affect the circuit fidelity significantly. For example, on the noiseless simulator used here, the normalized circuit fidelity falls below 0.6 at 8 qubits with 1,000 shots, while it does so at twelve qubits with 5,000 shots. As the circuit's width increases, the number of shots required to distinguish between the ideal and random distributions increases. With the variant of QAOA used here, larger problems require a larger number of shots to effectively capture the cut sizes in the resulting larger distributions.

Figure 10 illustrates how circuit fidelity is impacted as one of the arguments for the QAOA ansatz definition, the number of *rounds* (referred to as p in code), is increased from 1 to 8.

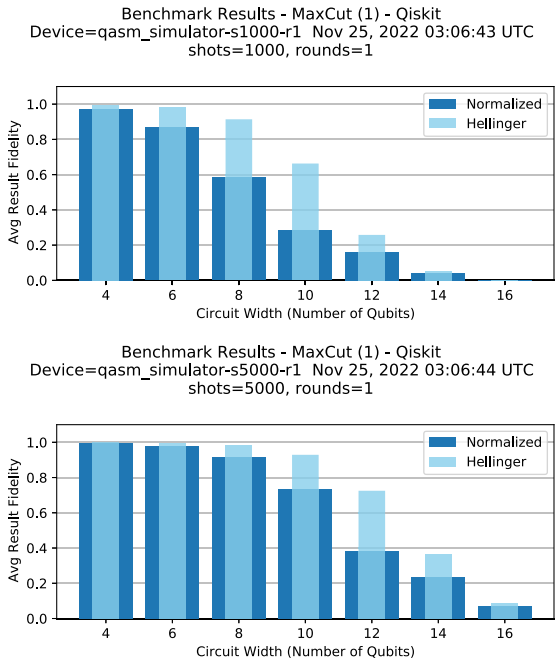


Fig. 9. Impact of shots on fidelity of ansatz execution. Here, we illustrate the difference in fidelity when executing the same Max-cut ansatz circuit at circuit widths ranging from 4 to 16 qubits, with 1,000 shots and again with 5,000 shots, on an ideal quantum simulator. For each problem size, we use a single graph, which defines an *instance* of the Max-cut problem. The resulting fidelity is greater when using a larger number of shots.

Execution fidelity is expected to degrade not only as circuits become wider (i.e., comprise more qubits) but also deeper (i.e., have a larger number of gate layers). More rounds result in deeper circuits. The volumetric plot uses a color scale to represent the fidelity at each circuit width and depth tested. In this case, the circuit was executed with 1,000 shots on a quantum simulator with noise characteristics that mimic a typical quantum computer (one- and two-qubit gate error rates of 0.003 and 0.03, respectively) and with a quantum volume equal to 32 (the region shown in the dark rectangle). As the "rounds" parameter grows, the circuit becomes correspondingly deeper, and the result fidelity degrades as a function of depth. A consequence is that the theoretical benefit of using a larger number of rounds is countered by the lower fidelity that results from executing a deeper circuit with a larger gate count.

5 Execution on Quantum Hardware

This section presents results from executing the Max-cut benchmark on several representative quantum hardware systems based on underlying quantum technologies. Our objective is to demonstrate the robust capability of the benchmark framework to accurately capture key performance metrics that highlight fundamental distinctions between technologies.

This presentation can serve as a valuable resource for providers of these systems, providing insight into incremental performance improvements across successive generations of hardware. It can also equip users with tools to form a comprehensive understanding of the trade-offs inherent in utilizing these emerging technologies. Particular attention is placed on the analysis of solution quality versus execution run-time.

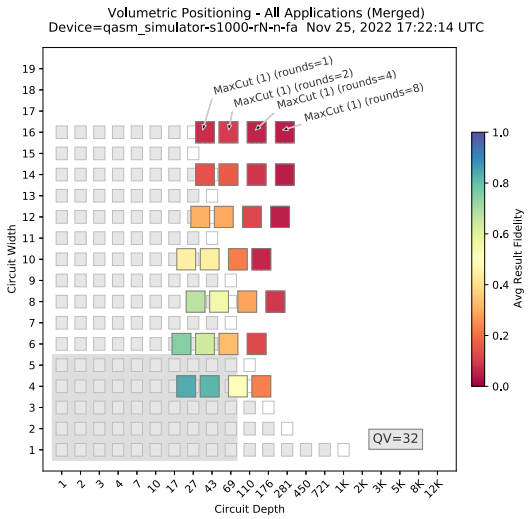


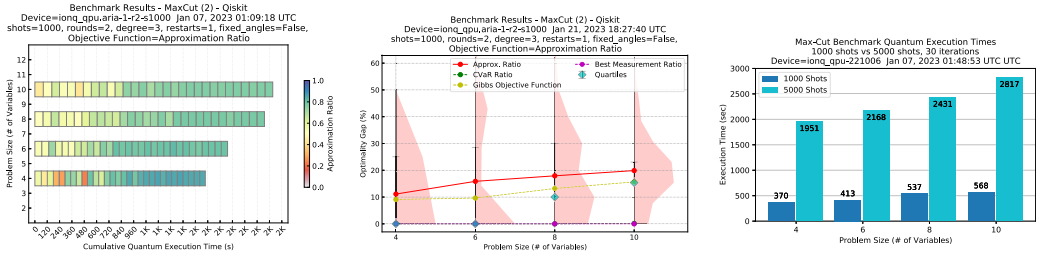
Fig. 10. Volumetric presentation of fidelity and impact of rounds. The fidelity metrics obtained for the execution of any quantum circuit are influenced by both the width of the circuit and its depth or its total number of quantum gate layers. Here, the Max-cut ansatz circuit with varying rounds is executed on a noisy quantum simulator with a quantum volume of 32 (one- and two-qubit gate error rates 0.003 and 0.03, respectively). The (normalized) result fidelity at a specific width and depth is represented by the color shown in the rectangle at that location and degrades with increasing rounds (depth) or problem size (width).

However, we emphasize that the results in this section should not be taken as representative of the comparative performance of these quantum platforms in general. They are designed for illustrative purposes and are not intended to be a formal comparison between quantum systems. Furthermore, we use the respective manufacturers' default software and parameter settings. Together with the quantum systems, these software tool sets are developing rapidly; therefore, our conclusions about quantum system performance represent a snapshot of progress over time. We hope that users will use the framework to create more thorough benchmark tests and utilize them to draw conclusions.

Several critical factors affect benchmark algorithm outcomes on current quantum computing hardware. Errors from gate infidelity and decoherence can lead to significant differences between the obtained measurement distribution and an ideal system, especially with larger circuits. These errors accumulate in iterative algorithms like QAOA, reducing the quality of the results. Noise in QA, such as thermal energy and control line fluctuations, can disrupt qubit states. Inefficient mappings or embeddings onto specific hardware topologies worsen these effects.

Apart from purely quantum computation, the quality of solutions returned by QAOA depends significantly on the quality of classical computations, such as compilation and optimization of beta and gamma for each round. Similarly, the quality of solutions returned by QA depends on the classical operations of minor embedding and post-processing and the user parameters that control the quantum computation. In this sense, our benchmark framework should be viewed as a tool to evaluate the performance of the quantum *system* performance in combination with algorithmic choices and parameter settings rather than the performance of a standalone circuit.

When quantum optimization applications are run on hardware, the quality of the result will degrade compared to a simulator, as the programs will be negatively impacted by noise. To illustrate the practical limits to execution on hardware, the benchmarks are executed on three



(a) (b) (c)

Fig. 11. Execution on Ion trap system. These plots present results from executing the Max-cut benchmark on the IonQ Aria QPU at different problem sizes (30 iterations, each with 1,000 shots). The area plot (a) shows the approximation ratio improving for each problem size as the cumulative quantum execution time increases (to around 2,400 s at ten qubits). The violin plot (b) shows the final optimality gap for each computed ratio and illustrates how the approximation ratio declines with larger problem sizes (to around 20% at ten qubits). Plot (c) presents data from a different system, IonQ Harmony, comparing the total quantum execution times, using 1,000 shots (568 s at ten qubits) and again using 5,000 shots (2,817 s at ten qubits). This indicates that the cost of executing the Max-cut algorithm on these systems is nearly proportional to the number of shots used. (*Data collected via cloud service*).

different classes of quantum computers: ion trap, superconducting transmon, and quantum annealing system.

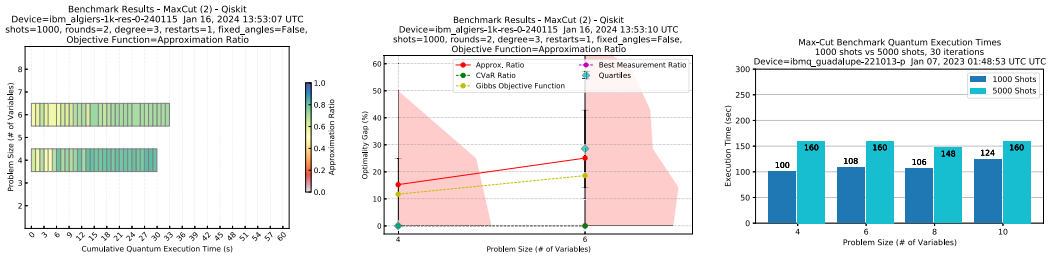
5.1 Execution on Ion Trap Systems

We first show results from executing the Max-cut benchmark on two remotely accessed gate model quantum computing systems that use ion trap technology. Quantum computers based on ion traps offer all-to-all connectivity and high fidelity, but this advantage is offset by longer execution times than with other technologies.

Figure 11 presents results obtained using the IonQ Aria QPU, a second-generation ion trap computer, and its first-generation predecessor, IonQ Harmony. At each problem size, from 4 qubits to 10 qubits, we executed the benchmark on both systems with a maximum of 30 iterations using 1000 shots each.

The first plot (a) uses the area plot of Section 4.2 to illustrate, for each problem size, how the approximation ratio improves as the cumulative quantum execution time increases with each iteration (to ~ 2,400 s at ten qubits on Aria). At larger problem sizes, the ansatz circuit becomes deeper, which increases the total execution time and lowers the quality. The degradation in the final result quality is visible in the violin plot (b), which shows the final optimality gap for each computed ratio increasing with problem size (to ~ 20% at ten qubits.) However, note that the best measurement ratio gap is 0%.

Plot (c) of this figure presents data generated using IonQ Harmony, on which we executed the Max-cut benchmark twice, once using 1,000 shots and again using 5,000 shots. For all problem sizes, the increase in the cumulative quantum execution time is roughly proportional to the rise in the number of shots. For example, at ten qubits, the time increases from 568 to 2,817 s, a factor of 4.959. The height difference between the bars at a specific problem size represents the difference in time spent executing an additional 4,000 shots over 30 iterations, since both runs share the same initialization time. From this, we can compute the execution time per shot. For example, at six qubits, this is $(2,168 - 413)/(4,000 * 30)$ or 14.6 ms/shot for IonQ Harmony. These results indicate that the total cost to a user to execute the Max-cut or similar algorithms on this class of quantum computers includes not only the financial cost, which depends on the



(a) (b) (c)

Fig. 12. Execution on superconducting transmon system. These plots present results from executing the Max-cut benchmark on the IBM Quantum *ibm_algiers* system at different problem sizes (30 iterations, each with 1,000 shots). Execution was performed using the Sampler primitive without error mitigation (resilience level 0). The area plot (a) shows the approximation ratio improving for each problem size as the cumulative quantum execution time increases (to ~ 33 s at six qubits). The violin plot (b) shows the final optimality gap for each computed ratio and illustrates how the approximation ratio declines with larger problem size (to ~ 25% at six qubits.) Plot (c) presents data from a different system, IBM Quantum *ibm_guadalupe*, comparing the total quantum execution times, using 1,000 shots (124 s at ten qubits) and again using 5,000 shots (160 s at ten qubits). This indicates that the cost of executing the Max-cut algorithm on this system is only marginally increased using more shots. (*Data collected via cloud service.*)

shot count but also a reduction in total throughput that is a consequence of the longer execution times.

We observe that execution times on IonQ Aria are longer than on IonQ Harmony. For example, at ten qubits, we see approximately 2,400 and 568 s approximately, respectively. The Aria device applies an error mitigation scheme to measurement data to improve results. Still, we did not investigate the reasons behind the increased execution time and improvement in quality on Aria.

5.2 Execution on Superconducting Transmon Systems

Here, we present results from executing the Max-cut benchmark on two gate-model quantum computing systems that use superconducting transmon technology and can be remotely accessed. These results highlight and quantify the characteristics of quantum program execution inherent in hardware implementations built on this technology. Quantum computers based on superconducting transmons execute more quickly than other technologies. However, this advantage is offset by reductions in fidelity that can result from the introduction of swap operations to compensate for limited connectivity between qubits.

Execution on both of the systems was performed using the Qiskit Sampler primitive run through the IBM Cloud Qiskit Runtime service [91]. Error mitigation was turned off by setting the "resilience_level" execution argument to 0. While the Sampler offers automatic error mitigation, we elected not to enable it in our hardware demonstrations. Users are encouraged to gauge for themselves the impact of selecting this option on both the quality of the result and the total execution time.

The plots in Figure 12 present results from executing the Max-cut benchmark on the IBM Quantum *ibm_algiers* system at different problem sizes (30 iterations, each with 1,000 shots). The area plot (a) shows the approximation ratio improving for each problem size as the cumulative quantum execution time increases with each iteration (to ~33 s at six qubits). With deeper ansatz circuits at larger problem sizes, the total execution time grows longer, and the result quality declines. However, note this system's overall run-time performance. Each iteration of 1,000 shots requires ~1 s

at four qubits and ~1.1 s at six qubits. This results in completing the QAOA execution, limited to 30 iterations, in 30 and 33 s, respectively.

The violin plot (b) shows the final optimality gap for each computed ratio increasing with larger problem size (to ~ 25% at six qubits.) as the approximation ratio declines. However, the best measurement ratio gap is 0%, indicating that the algorithm could identify the maximum cut at both problem sizes.

Plot (c) presents data from the execution of the benchmark on a different system, IBM Quantum *ibm_guadalupe*. This is an earlier generation system with execution times that are longer than on *ibm_algiers*. However, we use these results to illustrate an important aspect of quantum program execution on superconducting transmon computers. Here, we compare the total quantum execution times obtained when running the benchmark using 1,000 shots (124 s at ten qubits) and again using 5,000 shots (160 s at ten qubits). Since both runs perform similar initialization processing, the height difference between the bars at each problem size represents the difference in time required to execute an additional 4,000 shots over 30 iterations. This lets us determine the time to execute a shot on this quantum computing device. For example, at six qubits, the execution time per shot can be computed as $(160 - 108)/(4,000 \times 30)$ or 0.43 ms/shot.

These results suggest that using more shots to execute quantum programs only marginally increases the cost of executing the Max-cut optimization or similar algorithms on this system. Users may take advantage of significantly higher throughput on this class of devices to execute the circuit repeatedly to search for an optimal result while still achieving a shorter total execution than with alternative technologies. Additionally, the ability to execute more shots could be explored to achieve higher-quality results.

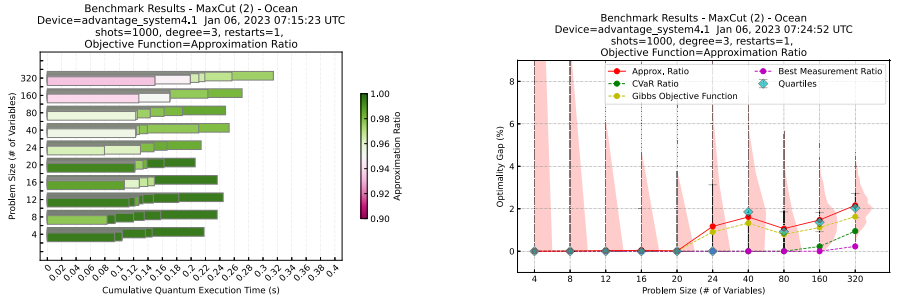
5.3 Execution on a Quantum Annealing System

An important focus of our work was to structure the QED-C benchmark suite to enable the execution of some benchmarks on back-end systems implemented using quantum technologies other than gate model quantum circuits. In this section, we describe the execution of the Max-cut benchmark on a D-Wave Advantage system, accessed through LEAP Solver "advantage_system4.1," as a way to illustrate the framework's support for test orchestration with varying parameters, capture of relevant performance metrics, and presentation of results consistently across quantum technologies.

The plots in Figure 13 present results from executing the Max-cut benchmark on the D-Wave *advantage_system4.1* quantum annealing system at problem sizes ranging from 4 to 320 variables (each executed with 1,000 shots). As in the corresponding gate-model displays, the area plot (a) shows the approximation ratio improving for each problem size as the cumulative quantum execution time increases (to ~ 0.31 s at 320 variables). Similarly, the violin plot (b) shows the final optimality gap for each computed ratio, and we see the approximation ratio declining with larger problem sizes (to ~2% at 320 variables.)

However, the QA benchmark implementation (described in Section 3.2) differs from the QAOA benchmark, affecting how the results are presented. At each problem size, the QA algorithm is re-executed from the beginning, doubling the annealing time in steps from 1 to 128 μs. In the annealing version of the plot (a), for each problem size, the rectangles are drawn overlaid to highlight this difference from the QAOA benchmark algorithm. Each rectangle represents a complete execution but with a larger anneal time.

In this way, we illustrate the quality versus time trade-off for quantum annealing in the same way we do for the QAOA algorithm but unambiguously convey the difference between the algorithms. The intent is to permit a user to quickly see the level of quality that can be expected for a



(a) (b)

Fig. 13. Execution on quantum annealing system. These plots present results from executing the Max-cut benchmark on the D-Wave *advantage_system4.1* quantum annealing system at different problem sizes (each with 1,000 shots). The problem is embedded once and executed over a range of annealing times from 1 ms to 256 ms to evaluate the impact of quantum annealing time on the resulting quality. The area plot (a) shows the approximation ratio improving for each problem size as the cumulative quantum execution time increases (to ~ 0.31 s at 320 variables). The violin plot (b) shows the final optimality gap for each computed ratio and illustrates how the approximation ratio declines with larger problem sizes (to ~2% at 320 variables.) (*Data collected via cloud service*).

specific annealing time, which is essential to evaluate the overall cost of the quantum annealing solution.

The total quantum execution time for the annealing algorithm increases only slightly as the problem grows, from ~0.22 s at 4 variables. to ~0.32 s at 320 variables. This is because the quantum execution reported in the plot includes the non-quantum operations of chip programming and readout in addition to the actual annealing time, which is small relative to these. This suggests that while increasing the annealing time of the computation may increase the financial cost, it does not impact the throughput that can be achieved when using quantum annealing.

At the problem sizes tested in this benchmark, the approximation ratios obtained with the QA benchmark are above 0.90 in all cases. Therefore, the QA area plots use a different color scale to represent the approximation ratio, making the evolution of the quality visible in this different range of values. Note also that the QA algorithm benchmark identifies the best cut size for problems with up to 160 variables.

5.4 Discussion of Hardware Results

The Max-cut extension to the QED-C benchmark suite enables user control over problem definition and size, shots, rounds, restarts, initial angles, and the choice of optimizer and its parameters. The number of possible combinations of these settings is large, and we explored a limited subset. In the hardware tests above, we execute benchmark problems of different sizes on three distinct classes of quantum computers and explore how the quality of the solution varies as execution time increases. In two of the tests, we also varied the number of shots (1,000 and 5,000) to gauge the impact of this parameter on system throughput.

In other tests, using simulations implemented with noise characteristics of these target systems, we found that setting the rounds parameter to 2 provides a good balance between the QAOA algorithm's effectiveness and the degradation from noise in longer circuits. Other tests indicated that at the small scale to the tests run using QAOA on gate model hardware, a setting of 1,000 shots and two rounds offers the best default configuration for executing on quantum hardware to minimize utilization of hardware resources during benchmarking.

For interested readers, we reference Appendix E, in which we discuss more extensive demonstrations executed to determine how the many different parameters affect result quality. The results suggest that multiple restarts could improve benchmark results. Importantly, using fixed initial angles combined with multiple restarts could be an effective way to provide a standard optimization benchmark that does not require complete QAOA execution to evaluate the effectiveness of a target system, reducing the resources necessary for benchmarking. Also presented in that section is a parameter-tuning strategy developed with the help of these benchmarks to identify the best combination of several of the parameters for QAOA execution.

In the results presented above, we depict cumulative quantum execution time using area plots to illustrate how the quality of the result improves as execution progresses. This time, reported by hardware providers, reflects the quantum processor (or simulator) usage. It holds significant importance as it directly impacts the financial cost of utilizing quantum computers, which varies substantially across systems. Our analysis revealed significant variations in execution throughput across different classes of quantum computing technology. When evaluating the utility of a quantum computer, it is essential to consider both the financial cost and the time required to complete tasks.

Another critical factor to consider in evaluating the total cost of a quantum computing solution is other time costs beyond execution time in the quantum processing. Preparing the quantum program for execution involves resources overhead. This can include compilation time, transpilation to the target topology and gate set, loading the compiled program, and data transfer into and out of the quantum system. These overhead components directly impact the optimization application's total throughput, as every execution will include some or all of them.

These throughput factors can vary widely between vendors, particularly within the execution environment, and uniquely between users. For example, pre-compilation of the program or co-location of the classical and quantum processors can dramatically impact total throughput. There are also numerous vendor-specific hardware settings that we did not test that could potentially improve results.

Furthermore, the QAOA and QA algorithms, tailored to match their corresponding gate model and annealing-based hardware paradigms, possess different properties and structures, significantly impacting computation time and solution quality. This complexity can obscure the effects of hardware and system performance. For QAOA, the choice of optimizer and its parameters can dramatically impact benchmark results.

As stated earlier, the performance results in our study are intended for illustration purposes only and should not be taken as representative of relative performance in general. Too many variables contribute to both the quality of the solution and the total run time for these results to be viewed as a definitive characterization of these systems' performance. For these reasons, we do not provide an exhaustive analysis of these factors. A full-scale study of all the factors contributing to quantum performance to tease out the hardware contribution is beyond the scope of this article. Instead, we propose that users include the total cost of execution, including these additional overheads, in any of their studies using the benchmark suite presented here.

It is worth noting that inherent variations can significantly influence the quality of results in the quantum algorithms employed in these benchmarks. While our study did not delve into such variations, our benchmark framework offers a platform to explore recent advancements in optimization algorithms. For instance, the framework could be configured to incorporate innovations such as pre-computing diagonal matrices [16], investigating the impact of multiple rounds [17, 18], adopting a multi-angle ansatz [19–22], utilizing an expressive ansatz [23], or employing a large-scale solver with few qubits [24]. As quantum computing hardware progresses to incorporate fault tolerance and error correction features, it becomes imperative to advance

algorithms in tandem. Utilizing more focused versions of the tests illustrated here will be crucial in assessing the concurrent improvement in performance.

6 Summary and Conclusions

While the current generation of quantum computers may be limited in computational power, these systems are expected to rapidly evolve and become capable of performing increasingly complicated tasks. It is thus critical to this advancement effort to establish accurate and validated methods for measuring progress that are readily available to the developers of these systems and the users who utilize the resource in solving real-world problems.

To this end, we built on the existing open-source QED-C Application-oriented Benchmark suite, enhancing it to support benchmarking of hybrid quantum-classical solutions to combinatorial optimization problems, often cited as a use case for quantum computing. Multiple factors affect the ability of a quantum computer to produce solutions to combinatorial optimization problems effectively. The algorithms used to find these solutions provide only an approximate answer, and the quality of the results is typically a function of the time available for processing, often under tight constraints. Our benchmarks are designed to provide a mechanism to evaluate these and to provide valuable and critical insights into options for improving its performance and overall throughput.

We demonstrate the features of this framework and highlight its benefits using the Quantum Approximate Optimization Algorithm algorithm for execution on gate model systems and demonstrate its adaptability to other types of solvers by using a Quantum Annealing algorithm executed on an analog quantum computer system. We demonstrate the capabilities of our framework using the widely studied Max-cut problem, which offers a simple early stage target for evaluating quantum computing solutions but can scale to larger application challenges in the future. In future work, we plan to extend this demonstration to include other technologies, such as cold atoms.

A primary goal of this work was to structure our analysis and presentation methodology to use methods familiar to quantum computing specialists but inspired by how Operations Research views the quality of results from a solver in addressing optimization problems. The methods are enhanced to account for specific characteristics of quantum solutions, such as statistical sampling or the iterative nature of the QAQA algorithm. They can inform the user of bottlenecks or anomalies in execution, which is extremely valuable.

These enhanced analysis and visualization techniques can provide useful information about a quantum computing solution's throughput and a detailed understanding of its total cost of ownership. Results from these benchmarks can inform the user about the factors that can be adjusted to improve performance on these systems. While these techniques have been used in Operations Research for a long time, applying them effectively to quantum computing is still in the early stages.

This article does not include a full-scale comparison between quantum computing systems of different types, nor does it address benchmarking of classical solutions to optimization problems, as numerous in-depth studies exist in this area. The performance results in this article are intended to illustrate the features and benefits of our benchmarking framework. We include benchmarking the QA algorithm on annealing hardware as a proxy for other solvers using different quantum technologies. Our work has identified many parameters that impact how a quantum computer solves an optimization problem well. However, we did not perform an exhaustive study of these or all options that might produce optimal results. We do not attempt to provide coverage or analysis over all possible settings of algorithm parameters or vendor-specific execution options.

While we generally show that one class of device may provide higher-quality results while another may execute more quickly, the emphasis in this article is on how the benchmark framework is structured and how it can be used to explore quantum algorithm execution and performance. Users can execute these benchmark programs on devices to which they have access and evaluate

for themselves the total cost of ownership of this technology. This is crucial to understanding how and when quantum computers may be able to offer measurable value.

We consider this work to be forward-looking with respect to the available technology. The complexity of discrete optimization problems motivates the development of methods, such as QAOA, QA, and others, that may be challenging in the worst-case complexity analysis yet provide value in practice. Performance-based metrics and benchmarking tools can quantify the progress of these proposed methods and provide a way of comparing alternative solutions whose capabilities are beyond those currently available. As quantum computers evolve, the benchmark methods we have defined here will be critical to gauge performance improvement.

Code Availability

The code for the benchmark suite introduced in this work is available at https://github.com/SRI-International/QC-App-Oriented-Benchmarks. Detailed instructions are provided in the repository.

Appendices

A Methods for Combinatorial Optimization

We present a general introduction to the theoretical foundations of combinatorial optimization problems and their implications for developing the hardware demonstrations to study solver performance (a solver is an algorithm or heuristic implemented in software or hardware). The issues discussed here informed our decisions about the choice of inputs and performance metrics in designing the QED-C benchmarking framework.

A.1 Combinatorial Optimization Theory

For concreteness, we consider the class of combinatorial optimization problems defined on n integer-valued variables $x = \{x_1, \ldots x_n\}$, containing m constraint functions $c(x)$, and one objective function f that is a polynomial in x, as follows:

$$\begin{aligned} &\min : f(x) \\ &\text{s.t.:} c_i(x) \leq 0 \quad \forall i \in \{1, \ldots, m\} \\ &x_i \in \mathbb{Z} \quad \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{10}$$

Given a problem thus described, the algorithmic goal is to find an assignment of integer values to x that obeys all the constraints and minimizes the value of $f(x)$. For example, this simple problem,

$$\begin{aligned} f(x) &= x_1 + 2x_2 \\ c_1(x) &: -x_1 + 1 \leq 0 \\ c_2(x) &: -x_2 + 1 \leq 0, \end{aligned}$$

asks to find two positive integers that minimize $f(x)$; an optimal solution $x_1 = 1, x_2 = 1$ has the objective value $f(x) = 3$.

This notation is general enough to cover an enormous variety of optimization problems of interest to all industry sectors. To name just a few:

- The **Job Shop Scheduling** problem and its variations are ubiquitous in industry scheduling problems associated with the efficient assignment of multiple resources to multiple tasks.
- The **Portfolio Optimization** problem is of interest to finance. For example, given a list of items to purchase, select a subset of items to maximize profit and minimize risk.
- The **Airport Gate Scheduling** problem in the transportation industry is as follows: Given a list of airport arrival times and passenger connections, assign gates to airplanes to minimize the total distance passengers must walk to the connecting gates.

— **Machine Learning (ML)** is a core problem of Artificial Intelligence. Most ML techniques require access to good-quality optimization heuristics as part of an inner-loop computation that may be performed hundreds or thousands of times. The heuristic finds input/output pairs that constitute diverse samples of the near-optimal solution space of a given optimization problem.

The complexity class NP-OPT contains optimization problems (including (A1)) that are defined in terms of an objective function with a numerical result, as opposed to decision problems with binary outcomes (e.g., Yes/No or True/False), which inhabit the more famous class NP.

Every problem P in NP can be reformulated (also called translated) as a problem P-OPT in NP-OPT. For example, in the binary **Satisfiability (SAT)** problem, *Given the Boolean expression B, does there exist an assignment of variables such that B evaluates to True?* can be translated to an equivalent problem in NP-OPT: *Given B-OPT, find an assignment of variables that maximizes the number of satisfied clauses.* The transformation guarantees that an optimal solution to SAT-OPT is a yes answer to SAT. In this case, the maximum number of satisfied clauses equals the total number of clauses in $B - OPT$, then the answer to the binary problem is *yes.*

The translated SAT-OPT problem is called NP-HARD, because a polynomial-time algorithm for SAT-OPT could be used to solve SAT, and by extension, every problem in NP could be solved in polynomial time. Problems that are both NP-HARD and in NP (i.e., binary decision problems) are called NP-COMPLETE. The famously open question *Does $P = NP$?* captures the current unhappy state of knowledge about these problems: no polynomial-time algorithm is known to exist, and no one can prove that they cannot exist.

Solving Problems by Translation Among many approaches to solving problems in NP and NP-OPT, solution-by-transformation has been studied for a handful of problems and algorithms.

This approach is attractive to practitioners when a single solver for the target problem T can be applied to a wide variety of problems that arise in practice: that is when the overhead cost of translating individual inputs to match the formulation for T is less than the overhead cost of implementing a problem-specific solver for each new problem that arises.

The most widely studied versions of this approach involve a subset of problems formulated as (A1), known as integer linear problems, which can be solved in polynomial time when the objective function $f(x)$ and constraints $c_i(x)$ are all linear. Another common approach is translating problems to SAT or SAT-OPT, for which efficient heuristics are sometimes known.

The **quadratic unconstrained binary optimization (QUBO)** problem has also been considered as a general-purpose target formulation, especially for problems defined on graphs, before quantum computing came onto the scene [92, 93]. The emergence of quantum annealing processors that solve QUBOs natively in hardware has sparked recent interest in QUBO and its variation, the **Ising Model (IM)** problem, often used in physics applications. The two problems are identical, except for the change in the domain from binary variables $x \in \{0, 1\}$ (QUBO) to spin variables $s \in \{-1, +1\}$ (IM).

The theory of NP-COMPLETEness tells us that, in principle, any input for a problem formulated as (A1) can be transformed in polynomial time into a formulation that can be solved directly using a quantum computer. See References [29, 31] for tutorials on formulating general optimization problems expressed by (A1) as QUBOs and IMs. However, due to their small size, the problems currently being tested on quantum platforms are significantly restricted.

B Quantum Heuristics for Optimization Problems

The benchmarking framework measures performance characteristics of the two leading quantum heuristics for solving combinatorial optimization problems: the QAOA, which uses a gate-model

quantum computer, and QA, which uses an analog quantum computer. This article presents a benchmark of these algorithms in the context of their application to solving the Max-cut problem.

The input for a Max-cut problem is an undirected graph consisting of nodes or vertices (V) and edges (E). In general, each edge of the graph can be accompanied by a "weight," but we only consider unweighted 3-regular graphs in this article. A cut partitions the graph's nodes into two sets. Its size is defined as the number of graph edges connecting nodes belonging to different sets. The Max-cut problem is identifying a cut with the largest size out of all possible cuts.

Max-cut has emerged as a popular benchmark for quantum optimization [12, 25–27] for two reasons: (1) it is among the most challenging combinatorial optimization tasks, even to obtain an approximate solution, i.e., APX-Hard [13, 28], (2) as an unconstrained discrete optimization task, it has a natural encoding as a QUBO [29, 30] or an Ising model [5, 31], ideally fitting current quantum optimization algorithms (QAOA, QA). Although Max-cut provides a reasonable first step for benchmarking current methods, testing more complex optimization tasks, including problems with constraints, will be important in future work to demonstrate that quantum-accelerated optimization can impact a broad range of optimization applications.

B.1 Quantum Approximate Optimization Algorithm

The Quantum Approximate Optimization Algorithm [6] is arguably the leading candidate for solving combinatorial optimization problems using gate model quantum processors. QAOA belongs to the class of VQA [35] and is usually implemented iteratively wherein a classical optimizer "trains" a parameterized quantum circuit. QAOA is a heuristic that attempts to solve combinatorial optimization problems such as QUBO problems. Specifically, the problem is encoded in the form of a specified quadratic function of binary variables, and the objective is to find an assignment for those variables that minimizes the function.

At the core of QAOA is an "ansatz circuit," a parameterized quantum circuit. Measurements in the computational basis at the end of the circuit correspond to sampling from a probability distribution over possible answers to the problem. A classical optimizer obtains parameter values with a significant probability of producing optimal or near-optimal solutions. Finally, repeatedly measuring the circuit with the parameter values the optimizer determines provides approximate solutions to the problem.

The problem is first codified as a Hamiltonian H_P, such that an optimal problem solution corresponds to a ground state(s) or lowest energy eigenstate(s). For a given choice of the number of "rounds" denoted by p, the QAOA ansatz is given by

$$|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle = e^{-i\beta_p H_M} e^{-i\gamma_p H_P} \dots e^{-i\beta_1 H_M} e^{-i\gamma_1 H_P} |+\rangle\,, \tag{11}$$

where $H_M = \sum_i X_i$, is the so-called mixer Hamiltonian, and $|+\rangle = \otimes_i H |0\rangle$ is the equal superposition state. Here, X is the Pauli X matrix, defined by $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. The ansatz state is thus obtained by implementing repeating and alternating rotations about H_P and H_M as shown in Figure 14.

The Max-cut problem can be framed in terms of obtaining the ground state of the Hamiltonian

$$H_P = \frac{-1}{2} \sum_{\langle i,j\rangle \in E} (1 - Z_i Z_j), \tag{12}$$

where E denotes the set of (undirected) edges of the graph, and Z is the Pauli-Z matrix satisfying $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$. Each computational basis vector corresponds to a possible cut, and its energy represents the negative of the cut size. Note that the eigenvalues of H_P are all nonnegative

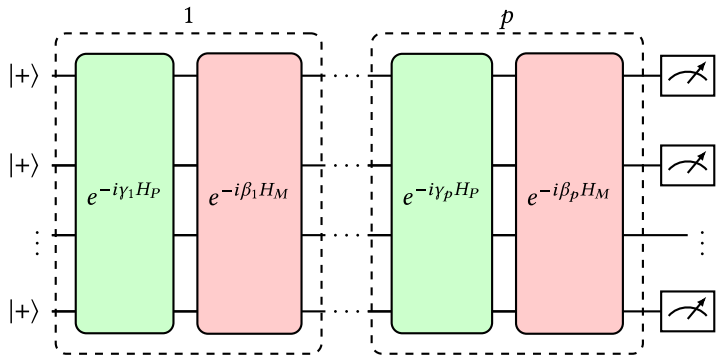


Fig. 14. QAOA circuit consists of p repeating parameterized blocks. First, each qubit is acted upon with the Hadamard gate to obtain the $|+\rangle$ state. Each block further consists of a rotation involving the problem Hamiltonian H_P, followed by a rotation involving the mixer Hamiltonian H_M. Finally, all qubits are measured on the computational basis $\{|0\rangle, |1\rangle\}$.

integers and that Equation (12) corresponds to an Ising model with all the coupling constants set to $1/2$.

A quantity called the approximation ratio is usually computed to characterize the quality of solutions. The approximation ratio r is defined as the ratio of the energy expectation value $F_{\boldsymbol{\beta},\boldsymbol{\gamma}} \coloneqq \langle \boldsymbol{\beta}, \boldsymbol{\gamma}| H_P |\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle$, and the ground state energy value $E_{\min}$:

$$r_{\boldsymbol{\beta},\boldsymbol{\gamma}} = \frac{F_{\boldsymbol{\beta},\boldsymbol{\gamma}}}{E_{\min}} = \frac{\langle \boldsymbol{\beta}, \boldsymbol{\gamma}| H_P |\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle}{E_{\min}}. \tag{13}$$

Note that the numerator is less than or equal to zero, whereas the denominator, which is the negative of the largest cut size, is strictly negative. Consequently, $0 \leq r \leq 1$. The classical optimizer routine aims to obtain optimal values of the angles $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, i.e., values corresponding to the highest approximation ratio value. $F_{\boldsymbol{\beta},\boldsymbol{\gamma}}$, and hence $r_{\boldsymbol{\beta},\boldsymbol{\gamma}}$ cannot be computed exactly and are instead approximated by measuring $|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle$ many (say M) times, or *shots*, in the computational basis at the end of the circuit (see Figure 14). Specifically, $F_{\boldsymbol{\beta},\boldsymbol{\gamma}}$ are approximated by the empirical average of energy.

B.2 Quantum Annealing

With quantum annealing, an optimization problem is encoded into the machine, after which the solution is determined through quantum adiabatic evolution to arrive at a near-optimal final state. The algorithmic approach of quantum annealing is to take advantage of the dynamic evolution of a quantum system to transform an *initial* ground state (which is easy to prepare) into the ground state of a *target* Hamiltonian, which is unknown and difficult to compute by other means. At a high level, the protocol strives to identify the low-energy states of a user-specified H_{Target} model by conducting an analog interpolation process of the following Hamiltonian:

$$H(s) = (1 - s)H_{\text{Init}} + (s)H_{\text{Target}}. \tag{14}$$

The interpolation process starts with $s = 0$ and in the ground state of H_{Init}. The annealing process involves a smooth interpolation of s from 0 to 1. For a sufficiently long annealing time, the adiabatic theorem demonstrates that a quantum system remains at the minimal eigenvector of the interpolating Hamiltonian, $H(s)$ [94–96], and therefore arrives at minimum energy states of H_{Target} at the end of the evolution.

Currently, available quantum annealing hardware focuses on a particular case of Equation (14) that is limited to the Transverse Field Ising model,

$$H(s) = A(s)\left(\sum_i X_i\right) + B(s)\left(\sum_i h_i Z_i + \sum_{i,j} J_{ij} Z_i Z_j\right), \tag{15}$$

where X_i denotes the Pauli X operator applied to qubit i, Z_i denotes the Pauli Z operator applied to qubit i, and Z_iZ_j is the tensor product of Z operators on qubits i and j. The two interpolation functions $A(s)$ and $B(s)$ control a transition from a strong H_{Init} and weak H_{target} to a weak H_{Init} and strong H_{target}; that is, $A(0) \gg B(0)$ and $A(1) \ll B(1)$. The hardware implements a default annealing "path" through these functions, which user parameters can modify. This way, this hardware, and the QA algorithm can find ground and low-energy states of a user-specified classical Ising model specified on the Z basis via the parameters h and J, which encode the local fields and coupling strengths, respectively. Note that the Max-cut problem considered in this work is encoded in this model by setting $h = 0$ and $J_{ij} = +1$ for each edge $(i, j) \in E$ that appears in the given Max-cut graph. If the Max-cut graph cannot be encoded naively in the quantum annealing hardware (i.e., the edge set of the Max-cut problem is not a subgraph of the Z_iZ_j terms in the quantum annealing hardware), then a process known as *minor embedding* [97] is used to map the Max-cut problem into a mathematically equivalent hardware-native problem.

It is interesting to note that the QAOA algorithm outlined in Appendix B.1 can be interpreted as a Trotterized version of Equation (15) where the number of rounds p determines the Trotter order. That is, the limit of a QAOA circuit can model the smooth analog QA transition as $p \longrightarrow \infty$. The approximation ratio is computed for QA in the same way as QAOA by transforming the samples obtained after annealing to an equivalent distribution.

C Problem and Implementation Details

To generate the results presented in this article, we execute the Max-cut benchmark on a single problem instance at each problem size (or number of qubits). The instance is a randomly chosen 3-regular graph at each size. A data set defining each instance is contained in the QED-C benchmark repository at https://github.com/SRI-International/QC-App-Oriented-Benchmarks. The benchmark can be modified to use other graphs if desired.

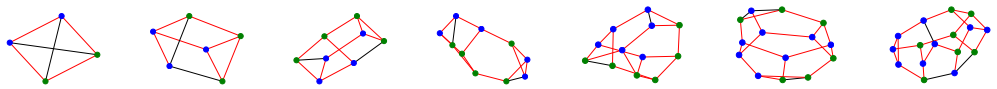


Fig. 15. Graph instances chosen for the benchmark implementations. For each problem size ranging from 4 to 16 in increments of 2, we used both QAOA and QA to solve the Max-cut problem for one 3-regular graph with that number of nodes. (For QA, larger graphs up to 320 nodes were also generated). These graphs show one solution to the Max-cut problem using colored nodes and edges. Nodes with different colors belong to the two sets of the solution cut. The number of red edges connecting nodes from different sets is the Max-cut for that graph.

For reference, in Figure 15, we present some of the graphs used for different size problems. Each graph shows one solution to the Max-cut problem using colored nodes and edges, as described in the caption.

In Figure 16, we show the QAOA ansatz circuit generated for the problem of size 6 (number of variables/qubits) used in this benchmark. The caption describes how the components of the circuit represent the Hamiltonian that defines the problem.

Below, we show the J matrix required to specify the 6-variable Max-cut problem for the quantum annealing hardware according to Equation (15):

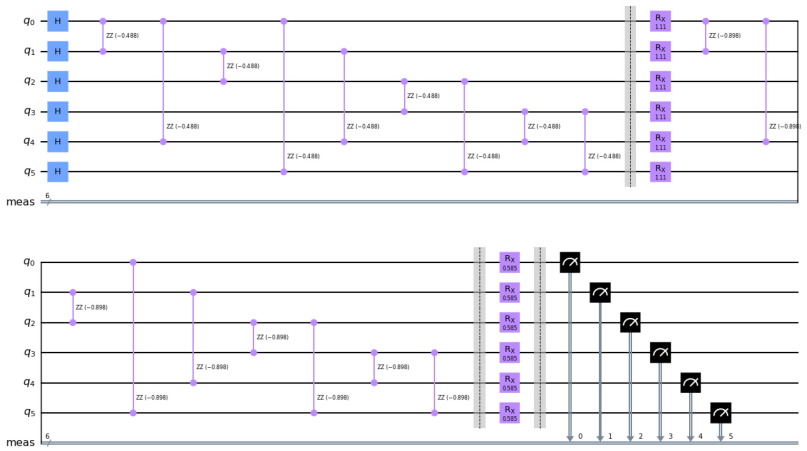


Fig. 16. Sample QAOA ansatz circuit diagram. The quantum circuit shown here is the ansatz created for the Max-cut problem with 6 variables shown above, implemented using two rounds on 6 qubits. Each of the two sets of nine parameterized RZZ gates maps the edges within the graph to the circuit and represents the problem Hamiltonian. The two sets of parameterized RX gates represent the mixer Hamiltonian. This circuit implements what is shown in Figure 14 for the specific graph used in the benchmark.

$$J = \begin{bmatrix} 0 & -1 & 0 & 0 & -1 & -1 \\ -1 & 0 & -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & 0 & -1 & -1 \\ -1 & -1 & 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{16}$$

D Analysis of Execution Time

Execution Time in QAOA. The total time the QAOA algorithm consumes includes both quantum and classical components. There is a time associated with executing the quantum ansatz circuit on the quantum processor. Additionally, time is spent on a classical processor to perform the minimization function that computes new parameters from measurements obtained after each execution of the ansatz.

The time to execute the quantum portion of the algorithm itself is broken down into several components. One is the "Quantum Execution Time," defined as the time to execute N shots of a quantum circuit within a quantum processor. Quantum computer hardware providers typically report this time in a result record and include the time required to initialize the quantum system before execution and the delay between shots [98] as shown in Equation (17):

$$t_{\text{quantum}} = t_{\text{init}} + N_{\text{shots}} \cdot (t_{\text{shot}} + t_{\text{delay}}). \tag{17}$$

Using QAOA, a quantum circuit is executed repeatedly but with varying parameters. The total time to execute the circuit, the "Elapsed Quantum Execution Time," includes the time required to either compile the circuit or to apply parameters before execution, and to validate and load the compiled circuit for execution. Another highly variable component is the time spent in a queue awaiting execution. The elapsed quantum execution time is defined in Equation (18) and must be collected as part of the benchmarking algorithm, as we did not find this metric directly available

in most systems:

$$t_{\text{elapsed\_quantum}} = t_{\text{queue}} + t_{\text{compile}} + t_{\text{load}} + t_{\text{quantum}}. \tag{18}$$

The sum of the quantum and elapsed quantum times for all iterations of the QAOA algorithm, the "Cumulative Quantum Execution Time" and "Cumulative Elapsed Quantum Execution Time," respectively, are defined in Equations (19) and (20). The financial cost of quantum computation is often tied to the cumulative quantum execution time in many hardware systems. The elapsed time for each iteration depends on the execution parameters and may be influenced by the system's ability to support parameterized execution or the inclusion of hidden classical post-processing time, such as error mitigation:

$$t_{\text{cum\_quantum}} = \sum_{iter=1}^{N_{\text{iter}}} t_{\text{quantum(iter)}}, \tag{19}$$

$$t_{\text{cum\_elapsed\_quantum}} = \sum_{iter=1}^{N_{\text{iter}}} t_{\text{elapsed\_quantum(iter)}}. \tag{20}$$

"Classical Execution Time" for QAOA is the sum of the time needed to create the ansatz with specific parameters and the time used by the minimizer to process measurement results and generate new parameters during a particular iteration $iter$, as in Equation (21). The "Cumulative Classical Execution Time" is the sum of the classical execution times for all iterations as in Equation (22). For small problems, this time is typically insignificant but can increase with problem size and rounds. It may also be impacted by the system's ability to support parameterized execution and reduce creation time:

$$t_{\text{classical}} = t_{\text{create}} + t_{\text{optimize}}, \tag{21}$$

$$t_{\text{cum\_classical}} = \sum_{iter=1}^{N_{\text{iter}}} t_{\text{classical(iter)}}. \tag{22}$$

The total execution time for QAOA is the sum of the cumulative elapsed quantum time $t_{\text{cum\_elapsed\_quantum}}$, and classical compute time $t_{\text{cum\_classical}}$. Variability due to different processing options or choice of classical optimizer can result in widely differing result quality and execution times.

Execution time in QA. For QA, the "Quantum Execution Time" is defined as the time the **quantum processing unit (QPU)** takes to execute N reads (samples) using a specified anneal time. This time is reported by the hardware as "qpu_access_time" and includes "qpu_programming_time" of ~ 16ms, and "qpu_sampling_time," which is "anneal_time" plus "readout_time" (~ 0.25ms), multiplied by the number of reads. Quantum execution time is defined in Equation (23):

$$t_{\text{quantum}} = t_{\text{qpu\_access}} = t_{\text{qpu\_programming}} + t_{\text{qpu\_sampling}}. \tag{23}$$

The "Elapsed Quantum Execution Time" includes the time required to issue the sample command to the (remote) backend hardware system, wait for it to complete, and receive a fully resolved sample set. It is defined in Equation (24). It includes the quantum execution time, along with the time for computing a minor embedding of the input to match the specific qubit connection structure inside the QPU, and the time to resolve solutions by mapping them back to the original (unembedded) problem. The embedding cost is measured once for each annealing time we test, which is not always necessary in practice, because embeddings can be reused:

$$t_{\text{elapsed\_quantum}} = t_{\text{queue}} + t_{\text{embed}} + t_{\text{sample}} + t_{\text{quantum}} + t_{\text{resolve}}. \tag{24}$$

For QA, the cumulative times reported in Figure 2 reflect measurements of t_{quantum} for increasing anneal times, as specified on line 6 of the benchmarking code. The $t_{\text{elapsed\_quantum}}$ metric includes all the time needed to perform the annealing operation and obtain a final sample set. This is comparable to the cumulative times in QAOA. To illustrate the difference between the two, we visualize the data in a slightly different style. Each bar represents a different anneal time and has a slight vertical offset from the one before it to convey that it represents the time starting at 0. See Section 5.3 for a presentation of these metrics collected from execution on quantum annealing hardware.

E Result Quality and Hyper-Parameters

The performance of an optimization algorithm is often studied in terms of the trade-off between the quality of the obtained result and the resources required to achieve it. In many real-world applications, a high-quality result is required in a limited time. It is desirable to predict whether obtaining a solution with acceptable quality within the available time budget and to determine the parameter values that result in high-quality outputs is possible.

However, many options (or "hyper-parameters") can be used to control the execution of QAOA, and conclusions must not be drawn from just one set of results obtained with limited exploration. While various hyper-parameters, such as the number of shots, choice of the classical optimizer, number of iterations, and rounds, affect result quality, we focus on the effects of values of initial angles on the performance in this section. We end this section by discussing a parameter tuning strategy to help identify good hyper-parameters for QAOA execution. (Similar strategies can be developed for QA but are not discussed here.)

While this section refers to QAOA in the context of the Max-cut problem, most of the conclusions hold more generally for QAOA. Throughout this section, we use the terms cut size and energy interchangeably, where energy refers to the eigenvalues of the Hamiltonian in Equation (12).

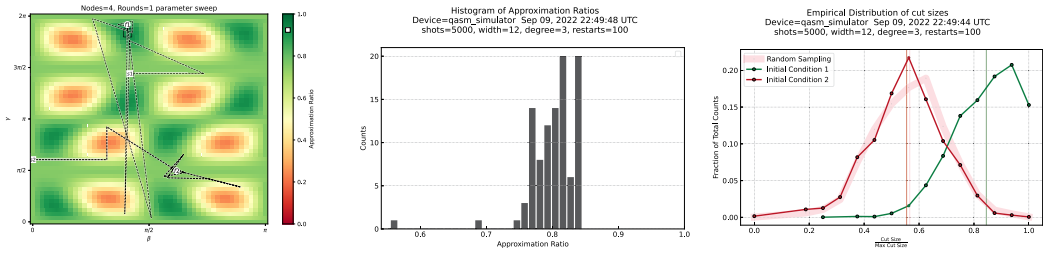
E.1 Initial Angles and Restarts

While several hyper-parameters, such as rounds, shots, number of optimizer iterations, and so on, affect the ability of the iterative QAOA execution to obtain a high-quality output, perhaps the most critical and non-trivial choice is that of the initial values of the angles.

The classical optimization routine faces several challenges. Finding the optimal angles for QAOA has been shown to be an NP-HARD problem [99]. Additionally, the landscape of the objective function suffers from "barren plateaus," a condition where the gradient of the objective function is close to zero, hindering training of the angles [100]. Barren plateaus can also be exacerbated by the choice of objective function [101], noise in quantum hardware [102], or large entanglement in the ansatz [103].

A consequence of these challenges is that the choice of the initial angles (i.e., $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$) can substantially affect the optimizer's ability to reach the optimal value of the objective function. For example, Figure 17(a) shows the trajectories of the angles probed by the optimizer for two randomly chosen initial angles. The distribution of the cut sizes obtained at the end of 30 optimizer iterations is substantially different, as shown in Figure 17(c). While one choice results in an output practically indistinguishable from a random sampling of bitstrings, the other results in a high-quality distribution of cuts. We also plot a histogram of the approximation ratios in Figure 17(b) from 100 random initializations.

These issues have spurred substantial research to address and overcome these challenges. Although many proposals have been put forth, keeping in mind our objective of benchmarking the performance of quantum solutions available to end users, we focus on implementing the most



(a) (b) (c)

Fig. 17. Values of initial angles affect result quality. (a) *Parameter trajectories and approximation ratio landscape:* The COBYLA optimizer navigates the parameter space differently depending on the initial parameter values. The trajectory taken for two randomly chosen initial angle values (labeled $s1$ and $s2$) is shown in the background of the approximation ratio landscape, obtained from a state-vector simulation. The parameters at the end of the 30 iterations are labeled $f1$ and $f2$, respectively. (b) *Histogram of approximation ratios* obtained at the end of 30 COBYLA iterations from QAOA simulations for 100 restarts succinctly shows the variability associated with initial conditions. Although most restarts result in an approximation ratio between 0.75 and 0.85, some result in a substantially lower approximation ratio. (c) The distribution of cut sizes at the end of 30 iterations for two initial conditions is substantially different, with initial condition 2 almost overlapping with a random sampling of bit-strings, while initial condition 1 results in a relatively high approximation ratio of ≈ 0.83.

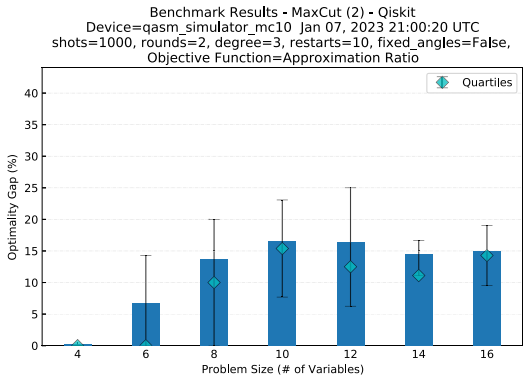


Fig. 18. Optimality gap with multiple restarts. We show the results obtained when executing the Max-cut benchmark 10 times at each problem size. The result shown for each problem size represents the "best" result obtained for that problem size across all 10 restarts, defined as the result showing the highest approximation ratio.

basic approach for mitigating some of these effects. Specifically, we implement multiple "restarts," i.e., we run QAOA multiple times using random angles as initial angles for the optimizer and report the output corresponding to the best restart.

To this end, our benchmarking framework allows users to specify the number of restarts through a parameter called `max_circuits`. This parameter is set to 1, and all the initial β and γ angles are set to 1. Thus, all the results in Section 5 use these starting angles. For restarts > 1, for each problem size, the output corresponding to the best restart is displayed in the plots. Figure 18 shows the output corresponding to 10 restarts for the same parameters as Figure 5. The quality of the results is noticeably better for smaller problem sizes. The user can also specify initial angles manually using the `thetas_array` parameter.

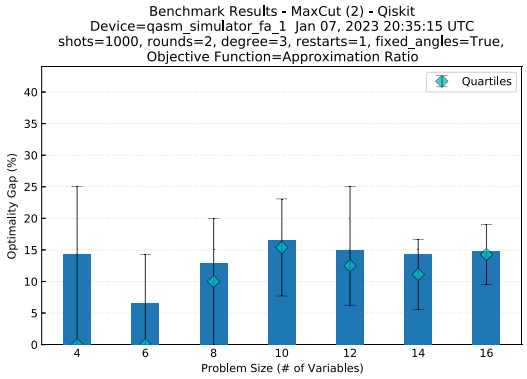


Fig. 19. Optimization-free QAOA implementation using fixed angles. To avoid restarts and the costly optimization loop, one may use the "fixed angles" guaranteed to produce a good quality output [104]. Here is the optimality gap using fixed angles for rounds=2 without implementing the minimizer routine.

E.2 Fixed Angle Conjecture

Although multiple initializations or restarts mitigate some of the difficulties faced by the optimizer, the cost of implementing the optimizer routine multiple times can be substantial, requiring many-fold quantum) processing unit access time.

A recent study [104] proposes an optimization-free QAOA implementation, executing the ansatz for each problem only once using the so-called "fixed angles." The authors show that at these angles, the approximation ratio is higher than the threshold for every problem instance for 3-regular graphs. Although these angles are not the global maxima of the approximation ratio landscape, they guarantee close to optimal performance without performing the costly optimizer loop. For example, Figure 19 shows that the optimality gap for all problem sizes (except 4) with fixed angles is practically the same as in Figure 18, which required 10 restarts with 30 optimizer iterations each. This corresponds to a reduction in QPU access time by a factor of ≈300 while yielding similar quality results.

Hence, the benchmark framework includes a provision for choosing the initial angles to be the fixed angles by setting the `use_fixed_angles` flag to `True`. The optimizer iterations can be set simultaneously to 1 to avoid using the optimizer routine.

In Figure 20, we present results from a test run using this benchmark feature to explore "parameter concentration" [105]. For problem sizes ranging from 4 to 20 qubits on 3-regular graphs, 100 random initial angles were tested using the Max-cut benchmark, with 30 optimizer iterations each. This plot shows the γ values obtained as final values by the optimizer and the corresponding approximation ratios. The angles obtained by the optimizer are shown to cluster around four values, most of which match the values proposed in Reference [104]. The choice of initial angles influences the algorithm's outcome, and a strategy for selecting these angles is critical for optimal performance.

E.3 Parameter Selection Strategy

Previously, we showed how the choice of initial angles and the number of rounds, shots, and restarts could affect the performance of the QAOA implementation. In addition, the performance could vary from one problem instance to another. These considerations raise the following question: For previously unseen problem instances, can we predict parameter values that are likely to result in the best performance? Specifically, given a notion of resource (e.g., QPU access time) and

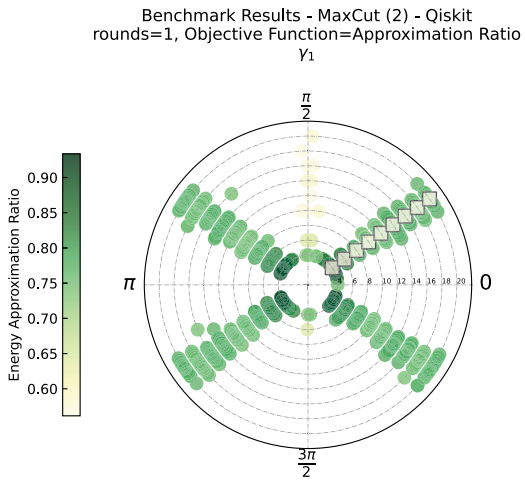


Fig. 20. Angles cluster around certain values. For 3-regular graphs with sizes ranging from 4 qubits to 20 qubits, we choose 100 random initial angles and run 30 (COBYLA) optimizer iterations each. The final angles obtained by the optimizer cluster around four values. The γ values are shown here, along with the corresponding approximation ratios.

a metric for result quality (e.g., approximation ratio), what parameter values should be used to get optimal performance given a resource budget?

With that goal in mind, a benchmarking framework is being developed for parameterized stochastic optimization algorithms such as QAOA and quantum annealing in a parallel effort [106]. Although this framework [107] applies to other algorithms, we apply it to the QAOA simulations using results obtained from the QED-C benchmarking framework. This framework generates parameter recommendations over a grid of resource values and also plots the corresponding performance compared to the best performance seen in the data.

The input to the framework consists of performance data obtained empirically by implementing an algorithm on various problem instances. The data includes the quality of the result, which we call the performance metric, corresponding to many algorithm executions over a range of parameter value settings. A function is provided to compute the resources expended for each execution.

The framework splits the problem instances into testing and training sets. A statistical analysis of the training set data is then used to identify the parameter values likely to lead to high performance when applied to the test set. However, for each instance in the test set, parameter values as a function of the resource corresponding to the highest result quality found so far are identified from the available data for all resource grid values. These are summarized in a curve denoted as "virtual best."

Thus, the parameter values corresponding to the virtual best simulate knowing ahead of time for each instance what the best parameters would be for any resource value. The virtual best provides a bound on the performance that any parameter-setting strategy using the data provided for the analysis can provide. Thus, the recommended and virtual best parameters are plotted together in a "strategy plot." The virtual best performance is plotted in a separate plot along with the performance obtained on the test set using the recommended parameters.

We now present an analysis of QAOA using this framework. Figure 21 shows the obtained performance profile, while Figure 22 shows the strategy plots generated by the framework, using an 80%–20% train/test instances split. The QAOA algorithm uses noiseless simulations with two

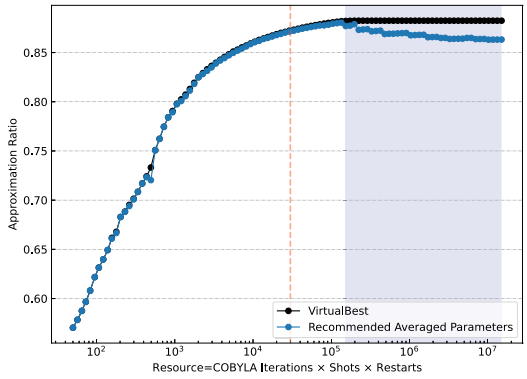


Fig. 21. Solution quality vs. total resource utilization: The virtual best provides a bound on the best performance attainable by any parameter strategy. Here are the performance profiles of the virtual best, along with the performance obtained from the parameters recommended by the stochastic-benchmarking framework [107]. The red dashed vertical line corresponds to the resource value used throughout the hardware section (30 iterations, 1,000 shots, 1 restart). The shaded area highlights the regime, after which the approximation ratio drops with increasing resources.

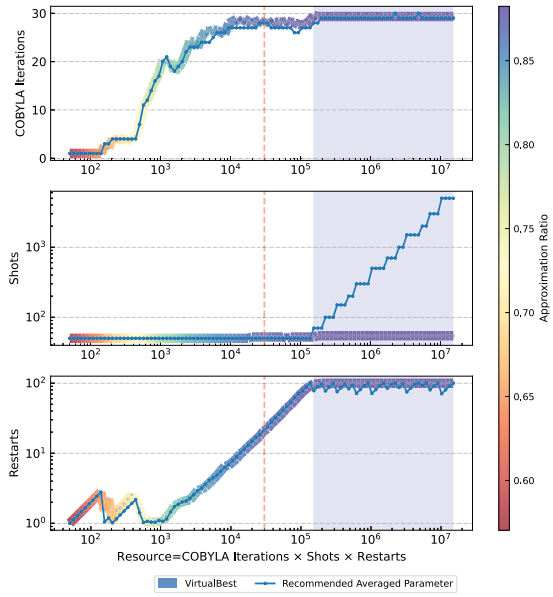


Fig. 22. Strategy plots: For each resource value, the framework recommends values for minimizer iterations, shots, and the number of restarts likely to lead to the best performance. For comparison, the virtual best parameters are also plotted as colored curves, with the colors indicating the corresponding approximation ratio. The red dashed vertical line corresponds to the resource value used throughout the hardware section (30 iterations, 1,000 shots, 1 restart). The shaded area highlights the regime, after which the approximation ratio drops with increasing resources.

rounds for 50 distinct 3-regular graphs of size 12. We implemented runs corresponding to a range of values for restarts [1,…,100], number of classical optimizer (COBYLA) iterations [1,…,30], and shots [50,…,5,000]. We capture the number of times the processing unit was accessed by defining the resource as the product of these parameters.

Figure 21 can be used to determine the relative performance of the recommended parameter values with respect to the optimistic bound given by the virtual best. In this example, we observe that both lines almost overlap, showing that good parameter values are shared across the training (recommended parameter values) and testing (virtual best parameter values) instances. In particular, these results show that, with increasing access to the processing unit, the quality of the response increases, as measured by the approximation ratio up to a certain point. The shaded area in this figure shows a regime of the resource quantity in which the performance metric decreases with increasing resources. This is counterintuitive and reveals that, given the data used to generate this analysis, the best parameter values are given with the COBYLA iterations set to 30 and 100 restarts and only 50 shots. Combinations of parameter values that yield larger resource usage can diminish the approximation ratio. This observation suggests that increasing the number of shots deteriorates the performance if allowed more processing unit access.

Moreover, it highlights that the number of classical minimizer iterations should be increased before the number of restarts when more resources become available, always aiming to keep the number of shots small. The dashed red line shows the equivalent resource usage of the simulations in the remaining of Appendix E. Notice how the recommended parameter values, as seen in Figure 22, i.e., 27 COBYLA iterations, 20 restarts, and 50 shots, are not the same as the ones used in the other hardware demonstrations, i.e., 30 COBYLA iterations, 1 restart, and 1000 shots. Using this analysis and specifying a performance metric and a resource function, empirical data can be used to inform parameter setting values. Moreover, these results can also inform about the instances themselves. In this example, the problem instances are relatively small, i.e., 12 node graphs with a solution space of size $2^{12} = 4,096$. When solving the problem with QAOA, sampling the output distributions extensively with many shots does not improve the approximation ratio as much as reinitializing the problem (restarts) or allowing more classical optimization iterations.

These parameter-setting strategy analyses provide practical recommendations for using algorithms like the one discussed in this manuscript.

Acknowledgments

D-Wave, Ocean, and Advantage are trademarks of D-Wave Systems, Inc. IBM, Qiskit, IBM Q, and IBM Quantum System Two are trademarks of International Business Machines Corporation. IonQ, IonQ Harmony, and IonQ Aria are trademarks of IonQ, Inc. We acknowledge Jerry Gamble of Verizon Corporation for his contribution to code development and editorial efforts on this manuscript. We acknowledge Jason Necaise in the Department of Physics and Astronomy, Dartmouth College (previously with D-Wave Systems), for his contribution to code development. We thank Mark Johnson (D-Wave), Andrew Wack (IBM), David McKay (IBM), Paul Nation (IBM), Luning Zhao (IonQ), Ananth Kaushik (IonQ), Farshud Sorouifar (Ohio State University), Amos Anderson (Quantum Circuits), Steve Reinhardt (Quantum Machines), Davide Venturelli (USRA/NASA), Filip Maciejewski (USRA/NASA), and others for providing comments on this manuscript.

References

[1] Ehsan Zahedinejad and Arman Zaribafiyan. 2017. Combinatorial Optimization on Gate Model Quantum Computers: A Survey. Retrieved from http://dx.doi.org/10.48550/ARXIV.1708.05294

[2] Daniel J. Egger, Claudio Gambella, Jakub Marecek, Scott McFaddin, Martin Mevissen, Rudy Raymond, Andrea Simonetto, Stefan Woerner, and Elena Yndurain. 2020. Quantum computing for finance: State-of-the-art and future prospects. *IEEE Trans. Quantum Eng.* 1 (2020), 1–24. DOI: http://dx.doi.org/10.1109/TQE.2020.3030314

[3] Catherine C. McGeoch and Cong Wang. 2013. Experimental evaluation of an adiabiatic quantum system for combinatorial optimization. In *Proceedings of the ACM International Conference on Computing Frontiers (CF'13).* ACM, New York, NY, Article 23, 11 pages. DOI: http://dx.doi.org/10.1145/2482767.2482797

[4] A. B. Finnila, M. A. Gomez, C. Sebenik, C. Stenson, and J. D. Doll. 1994. Quantum annealing: A new method for minimizing multidimensional functions. *Chem. Phys. Lett.* 219, 5 (1994), 343–348. DOI: http://dx.doi.org/10.1016/0009-2614(94)00117-0

[5] Tadashi Kadowaki and Hidetoshi Nishimori. 1998. Quantum annealing in the transverse Ising model. *Phys. Rev. E* 58 (Nov. 1998), 5355–5363. Issue 5. DOI: http://dx.doi.org/10.1103/PhysRevE.58.5355

[6] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A quantum approximate optimization algorithm. Retrieved from https://arXiv:1411.4028. DOI: http://dx.doi.org/10.48550/arXiv.1411.4028

[7] Gavin E. Crooks. 2018. Performance of the Quantum Approximate Optimization Algorithm on the Maximum Cut Problem. Retrieved from https://arxiv:quant-ph/1811.08419

[8] Jonathan Wurtz and Peter Love. 2021. MaxCut quantum approximate optimization algorithm performance guarantees for p>1. *Phys. Rev. A* 103, 4 (Apr. 2021), 042612. DOI: http://dx.doi.org/10.1103/PhysRevA.103.042612

[9] Immanuel Trummer and Christoph Koch. 2015. Multiple Query Optimization on the D-Wave 2X Adiabatic Quantum Computer. Retrieved from http://dx.doi.org/10.48550/ARXIV.1510.06437

[10] Yuchen Pang, Carleton Coffrin, Andrey Y. Lokhov, and Marc Vuffray. 2021. The potential of quantum annealing for rapid solution structure identification. *Constraints* 26, 1 (Oct. 2021), 1–25. DOI: http://dx.doi.org/10.1007/s10601-020-09315-0

[11] Byron Tasseff, Tameem Albash, Zachary Morrell, Marc Vuffray, Andrey Y. Lokhov, Sidhant Misra, and Carleton Coffrin. 2022. On the Emerging Potential of Quantum Annealing Hardware for Combinatorial Optimization. Retrieved from http://dx.doi.org/10.48550/ARXIV.2210.04291

[12] M. R. Garey, D. S. Johnson, and L. Stockmeyer. 1976. Some simplified NP-complete graph problems. *Theor. Comput. Sci.* 1, 3 (Feb. 1976), 237–267. DOI: http://dx.doi.org/10.1016/0304-3975(76)90059-1

[13] Christos H. Papadimitriou and Mihalis Yannakakis. 1991. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.* 43, 3 (1991), 425–440. DOI: http://dx.doi.org/10.1016/0022-0000(91)90023-X

[14] Application-Oriented Performance Benchmarks for Quantum Computing. 2020. Retrieved from https://github.com/SRI-International/QC-App-Oriented-Benchmarks

[15] Thomas Lubinski, Sonika Johri, Paul Varosy, Jeremiah Coleman, Luning Zhao, Jason Necaise, Charles H. Baldwin, Karl Mayer, and Timothy Proctor. 2023. Application-oriented performance benchmarks for quantum computing. *IEEE Trans. Quant. Eng.* 4 (2023), 1–32. DOI: http://dx.doi.org/10.1109/TQE.2023.3253761

[16] Danylo Lykov, Ruslan Shaydulin, Yue Sun, Yuri Alexeev, and Marco Pistoia. 2023. Fast simulation of high-depth QAOA circuits. In *Proceedings of the Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W'23)*. ACM. DOI: http://dx.doi.org/10.1145/3624062.3624216

[17] Yingyue Zhu, Zewen Zhang, Bhuvanesh Sundar, Alaina M. Green, C. Huerta Alderete, Nhung H. Nguyen, Kaden R. A. Hazzard, and Norbert M. Linke. 2022. Multi-round QAOA and advanced mixers on a trapped-ion quantum computer. *Quantum Sci. Technol.* 8, 1 (Nov. 2022), 015007. DOI: http://dx.doi.org/10.1088/2058-9565/ac91ef

[18] Ritajit Majumdar, Dhiraj Madan, Debasmita Bhoumik, Dhinakaran Vinayagamurthy, Shesha Raghunathan, and Susmita Sur-Kolay. 2021. Optimizing Ansatz Design in QAOA for Max-cut. Retrieved from https://arxiv:quant-ph/2106.02812 https://arxiv.org/abs/2106.02812

[19] Rebekah Herrman, Phillip C. Lotshaw, James Ostrowski, Travis S. Humble, and George Siopsis. 2021. Multi-angle Quantum Approximate Optimization Algorithm. Retrieved from https://www.nature.com/articles/s41598-022-10555-8

[20] Rebekah Herrman. 2022. Relating the Multi-angle Quantum Approximate Optimization Algorithm and Continuous-time Quantum Walks on Dynamic Graphs. Retrieved from https://arxiv.org/abs/2209.00415

[21] Kaiyan Shi, Rebekah Herrman, Ruslan Shaydulin, Shouvanik Chakrabarti, Marco Pistoia, and Jeffrey Larson. 2022. Multiangle QAOA does not always need all its angles. In *Proceedings of the IEEE/ACM 7th Symposium on Edge Computing (SEC'22)*. IEEE. DOI: http://dx.doi.org/10.1109/sec54971.2022.00062

[22] Michelle Chalupnik, Hans Melo, Yuri Alexeev, and Alexey Galda. 2022. Augmenting QAOA Ansatz with Multiparameter Problem-Independent Layer. Retrieved from https://ieeexplore.ieee.org/document/9951267

[23] V. Vijendran, Aritra Das, Dax Enshan Koh, Syed M. Assad, and Ping Koy Lam. 2023. An Expressive Ansatz for Low-Depth Quantum Optimisation. Retrieved from https://arxiv.org/abs/2302.04479

[24] Marco Sciorilli, Lucas Borges, Taylor L. Patti, Diego García-Martín, Giancarlo Camilo, Anima Anandkumar, and Leandro Aolita. 2024. Towards Large-scale Quantum Optimization Solvers with Few Qubits. Retrieved from https://arxiv:quant-ph/2401.09421

[25] Daniel Beaulieu and Anh Pham. 2021. Evaluating Performance of Hybrid Quantum Optimization Algorithms for MAXCUT Clustering using IBM Runtime Environment. Retrieved from http://dx.doi.org/10.48550/ARXIV.2112.03199

[26] David Amaro, Carlo Modica, Matthias Rosenkranz, Mattia Fiorentini, Marcello Benedetti, and Michael Lubasch. 2022. Filtering variational quantum algorithms for combinatorial optimization. *Quantum Sci. Technol.* 7, 1 (Jan. 2022), 015021. DOI: http://dx.doi.org/10.1088/2058-9565/ac3e54

[27] Linghua Zhu, Ho Lun Tang, George S. Barron, F. A. Calderon-Vargas, Nicholas J. Mayhall, Edwin Barnes, and Sophia E. Economou. 2020. An Adaptive Quantum Approximate Optimization Algorithm for Solving Combinatorial Problems on a Quantum Computer. Retrieved from https://dx.doi.org/10.48550/ARXIV.2005.10258

[28] Johan Håstad. 2001. Some optimal inapproximability results. *J. ACM* 48, 4 (July 2001), 798–859. DOI: http://dx.doi.org/10.1145/502090.502098

[29] Fred Glover, Gary Kochenberger, and Yu Du. 2018. A Tutorial on Formulating and Using QUBO Models. Retrieved from http://dx.doi.org/10.48550/ARXIV.1811.11538

[30] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. 2020. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Phys. Rev. X* 10, 2 (June 2020). DOI: http://dx.doi.org/10.1103/physrevx.10.021067

[31] Andrew Lucas. 2014. Ising formulations of many NP problems. *Front. Phys.* 2 (2014). Retrieved from http://dx.doi.org/10.3389/fphy.2014.00005

[32] Francisco Barahona, Martin Grötschel, Michael Jünger, and Gerhard Reinelt. 1988. An application of combinatorial optimization to statistical physics and circuit layout design. *Oper. Res.* 36, 3 (June 1988), 493–513. DOI: http://dx.doi.org/10.1287/opre.36.3.493

[33] Bahram Alidaee, Gary A. Kochenberger, and Ahmad Ahmadian. 1994. 0-1 quadratic programming approach for optimum solutions of two scheduling problems. *Int. J. Syst. Sci.* 25, 2 (Feb. 1994), 401–408. DOI: http://dx.doi.org/10.1080/00207729408928968

[34] Cristina Bazgan and Zsolt Tuza. 2008. Combinatorial 5/6-approximation of max cut in graphs of maximum degree 3. *J. Discrete Algor.* 6, 3 (2008), 510–519. DOI: http://dx.doi.org/10.1016/j.jda.2007.02.002

[35] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. 2021. Variational quantum algorithms. *Nature Rev. Phys.* 3, 9 (Aug. 2021), 625–644. DOI: http://dx.doi.org/10.1038/s42254-021-00348-9

[36] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland. 2008. Randomized benchmarking of quantum gates. *Phys. Rev. A* 77 (Jan. 2008), 012307. Issue 1. DOI: http://dx.doi.org/10.1103/PhysRevA.77.012307

[37] Easwar Magesan, J. M. Gambetta, and Joseph Emerson. 2011. Scalable and robust randomized benchmarking of quantum processes. *Phys. Rev. Lett.* 106 (May 2011), 180504. Issue 18. DOI: http://dx.doi.org/10.1103/PhysRevLett.106.180504

[38] Robin Blume-Kohout, John King Gamble, Erik Nielsen, Kenneth Rudinger, Jonathan Mizrahi, Kevin Fortier, and Peter Maunz. 2017. Demonstration of qubit operations below a rigorous fault tolerance threshold with gate set tomography. *Nat. Commun.* 8 (Feb. 2017), 14485. DOI: http://dx.doi.org/10.1038/ncomms14485

[39] Jay M. Gambetta, A. D. Córcoles, Seth T. Merkel, Blake R. Johnson, John A. Smolin, Jerry M. Chow, Colm A. Ryan, Chad Rigetti, S. Poletto, Thomas A. Ohki, et al. 2012. Characterization of addressability by simultaneous randomized benchmarking. *Phys. Rev. Lett.* 109, 24 (2012), 240504. Retrieved from https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.109.240504

[40] Mohan Sarovar, Timothy Proctor, Kenneth Rudinger, Kevin Young, Erik Nielsen, and Robin Blume-Kohout. 2020. Detecting crosstalk errors in quantum information processors. *Quantum* 4 (2020), 321. Retrieved from https://quantum-journal.org/papers/q-2020-09-11-321/

[41] Timothy Proctor, Stefan Seritan, Kenneth Rudinger, Erik Nielsen, Robin Blume-Kohout, and Kevin Young. 2022. Scalable randomized benchmarking of quantum computers using mirror circuits. *Phys. Rev. Lett.* 129, 15 (Oct. 2022). DOI: http://dx.doi.org/10.1103/physrevlett.129.150502

[42] David C. McKay, Ian Hincks, Emily J. Pritchett, Malcolm Carroll, Luke C. G. Govia, and Seth T. Merkel. 2023. Benchmarking Quantum Processor Performance at Scale. Retrieved from https://arxiv.org/abs/2311.05933

[43] 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, 3 (Sep. 2019). DOI: http://dx.doi.org/10.1103/physreva.100.032328

[44] The Qiskit Team. 2021. Measuring Quantum Volume. Retrieved from https://qiskit.org/textbook/ch-quantum-hardware/measuring-quantum-volume.html

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

[46] Timothy Proctor, Kenneth Rudinger, Kevin Young, Erik Nielsen, and Robin Blume-Kohout. 2020. Measuring the Capabilities of Quantum Computers. Retrieved from https://arxiv:quant-ph/2008.11294

[47] Mark Johnson, Mohammad Amin, S. Gildert, Trevor Lanting, F. Hamze, N. Dickson, R. Harris, Andrew Berkley, Jan Johansson, Paul Bunyk, E. Chapple, C. Enderud, Jeremy Hilton, Kamran Karimi, E. Ladizinsky, Nicolas Ladizinsky, T. Oh, I. Perminov, C. Rich, and Geordie Rose. 2011. Quantum annealing with manufactured spins. *Nature* 473 (Sep. 2011), 194–198. DOI: http://dx.doi.org/10.1038/nature10012

[48] Alejandro Perdomo-Ortiz, Alexander Feldman, Asier Ozaeta, Sergei V. Isakov, Zheng Zhu, Bryan O'Gorman, Helmut G. Katzgraber, Alexander Diedrich, Hartmut Neven, Johan de Kleer, Brad Lackey, and Rupak Biswas. 2019. Readiness of quantum optimization machines for industrial applications. *Phys. Rev. Appl.* 12, 1 (July 2019). DOI: http://dx.doi.org/10.1103/physrevapplied.12.014004

[49] Bikas K. Chakrabarti and Sudip Mukherjee. 2022. Quantum Annealing and Computation. Retrieved from http://dx.doi.org/10.48550/ARXIV.2203.15839

[50] Neil G. Dickson, M. William Johnson, Mohammad H. S. Amin, R. Harris, F. Altomare, Andrew J. Berkley, Paul I. Bunyk, J. Cai, E. M. Chapple, P Chavez, Florentin Cioată, T Cirip, P Debuen, Marshall Drew-Brook, C. Enderud, S. Gildert, Firas Hamze, Jeremy P. Hilton, E. Hoskinson, Kamran Karimi, Eric Ladizinsky, Nicolas Ladizinsky, Trevor Lanting, Timothy Mahon, Richard Bryon Neufeld, Travis Oh, I. G. Perminov, C. P. Petroff, Anthony J. Przybysz, Chris Rich, P. Spear, Adi Tcaciuc, Murray C. Thom, Elena Tolkacheva, Sergey Uchaikin, J. Wang, A. B. Wilson, Zeeya Merali, and Geordie Rose. 2013. Thermally assisted quantum annealing of a 16-qubit problem. *Nature Commun.* 4 (2013), 1903.

[51] T. Lanting, A. J. Przybysz, A. Yu. Smirnov, F. M. Spedalieri, M. H. Amin, A. J. Berkley, R. Harris, F. Altomare, S. Boixo, P. Bunyk, N. Dickson, C. Enderud, J. P. Hilton, E. Hoskinson, M. W. Johnson, E. Ladizinsky, N. Ladizinsky, R. Neufeld, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, S. Uchaikin, A. B. Wilson, and G. Rose. 2014. Entanglement in a quantum annealing processor. *Phys. Rev. X* 4, 2 (May 2014). DOI: http://dx.doi.org/10.1103/physrevx.4.021041

[52] Tristan Zaborniak and Rogério de Sousa. 2021. Benchmarking hamiltonian noise in the D-Wave quantum annealer. *IEEE Trans. Quantum Eng.* 2 (2021), 1–6. DOI: http://dx.doi.org/10.1109/TQE.2021.3050449

[53] Jon Nelson, Marc Vuffray, Andrey Y. Lokhov, and Carleton Coffrin. 2021. Single-qubit fidelity assessment of quantum annealing hardware. *IEEE Trans. Quantum Eng.* 2 (2021), 1–10. DOI: http://dx.doi.org/10.1109/TQE.2021.3092710

[54] Marc Vuffray, Carleton Coffrin, Yaroslav A. Kharkov, and Andrey Y. Lokhov. 2022. Programmable quantum annealers as noisy gibbs samplers. *PRX Quantum* 3 (Apr. 2022), 020317. Issue 2. DOI: http://dx.doi.org/10.1103/PRXQuantum.3.020317

[55] Daniel C. Murphy and Kenneth R. Brown. 2019. Controlling error orientation to improve quantum algorithm success rates. *Phys. Rev. A* 99, 3 (2019), 032318. Retrieved from https://journals.aps.org/pra/abstract/10.1103/PhysRevA.99.032318

[56] James King, Sheir Yarkoni, Mayssam M. Nevisi, Jeremy P. Hilton, and Catherine C. McGeoch. 2015. Benchmarking a Quantum Annealing Processor with the Time-to-target Metric. Retrieved from http://dx.doi.org/10.48550/ARXIV.1508.05087

[57] David Subires, Fernando J. Gómez-Ruiz, Antonia Ruiz-García, Daniel Alonso, and Adolfo del Campo. 2022. Benchmarking quantum annealing dynamics: The spin-vector Langevin model. *Phys. Rev. Res.* 4, 2 (May 2022). DOI: http://dx.doi.org/10.1103/physrevresearch.4.023104

[58] Antika Sinha. 2022. Development of Research Network on Quantum Annealing Computation and Information using Google Scholar Data. Retrieved from http://dx.doi.org/10.48550/ARXIV.2206.02176

[59] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science* 220, 4598 (1983), 671–680. DOI: http://dx.doi.org/10.1126/science.220.4598.671

[60] Charlie J. Geyer. 1991. Parallel tempering. In *Proceedings of the Computing Science and Statistics Proceedings of the 23rd Symposium on the Interface*, E. M. Keramidas and S. M. Kaufman (Eds.). American Statistical Association, New York, NY, 156.

[61] Zheng Zhu, Andrew J. Ochoa, and Helmut G. Katzgraber. 2015. Efficient cluster algorithm for spin glasses in any space dimension. *Phys. Rev. Lett.* 115 (Aug. 2015), 077201. Issue 7. DOI: http://dx.doi.org/10.1103/PhysRevLett.115.077201 arXiv:1501.05630

[62] Carleton Coffrin, Harsha Nagarajan, and Russell Bent. 2019. Evaluating ising processing units with integer programming. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, Louis-Martin Rousseau and Kostas Stergiou (Eds.). Springer International Publishing, Cham, 163–181.

[63] Itay Hen, Joshua Job, Tameem Albash, Troels F. Rønnow, Matthias Troyer, and Daniel A. Lidar. 2015. Probing for quantum speedup in spin-glass problems with planted solutions. *Phys. Rev. A* 92, 4 (Oct. 2015), 042325. DOI: http://dx.doi.org/10.1103/PhysRevA.92.042325

[64] Dilina Perera, Inimfon Akpabio, Firas Hamze, Salvatore Mandra, Nathan Rose, Maliheh Aramon, and Helmut G. Katzgraber. 2020. Chook—A Comprehensive Suite for Generating Binary Optimization Problems with Planted Solutions. Retrieved from http://dx.doi.org/10.48550/ARXIV.2005.14344

[65] Matthew Kowalsky, Tameem Albash, Itay Hen, and Daniel A. Lidar. 2021. 3-Regular 3-XORSAT planted solutions benchmark of classical and quantum heuristic optimizers. *Quantum Sci. Technol.* 7 025008 (2022), (2021). DOI: http://dx.doi.org/10.1088/2058-9565/ac4d1b

[66] Ang Li, Samuel Stein, Sriram Krishnamoorthy, and James Ang. 2020. QASMBench: A Low-level QASM Benchmark Suite for NISQ Evaluation and Simulation. Retrieved from http://dx.doi.org/10.48550/ARXIV.2005.13018

[67] Teague Tomesh, Pranav Gokhale, Victory Omole, Gokul Subramanian Ravi, Kaitlin N. Smith, Joshua Viszlai, Xin-Chuan Wu, Nikos Hardavellas, Margaret R. Martonosi, and Frederic T. Chong. 2022. SupermarQ: A Scalable Quantum Benchmark Suite. Retrieved from http://dx.doi.org/10.48550/ARXIV.2202.11045

[68] Huub Donkers, Koen Mesman, Zaid Al-Ars, and Matthias Möller. 2022. QPack Scores: Quantitative Performance Metrics for Application-oriented Quantum Computer Benchmarking. Retrieved from http://dx.doi.org/10.48550/ARXIV.2205.12142

[69] Jernej Rudi Finžgar, Philipp Ross, Leonhard Hölscher, Johannes Klepsch, and Andre Luckow. 2022. QUARK: A Framework for Quantum Computing Application Benchmarking. Retrieved from http://dx.doi.org/10.48550/ARXIV.2202.03028

[70] Ward van der Schoot, Daan Leermakers, Robert Wezeman, Niels Neumann, and Frank Phillipson. 2022. Evaluating the Q-score of quantum annealers. In *Proceedings of the IEEE International Conference on Quantum Software (QSW'22)*. IEEE. DOI: http://dx.doi.org/10.1109/qsw55613.2022.00017

[71] Blake Johnson and Ismael Faro. 2021. IBM Quantum Delivers 120x Speedup of Quantum Workloads with Qiskit Runtime. Retrieved from https://research.ibm.com/blog/120x-quantum-speedup?lnk=ushpv18re2

[72] K. Bertels, A. Sarkar, T. Hubregtsen, M. Serrao, A. A. Mouedenne, A. Yadav, A. Krol, and I. Ashraf. 2020. Quantum computer architecture: Towards full-stack quantum accelerators. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE'20)*. DOI: http://dx.doi.org/10.23919/date48585.2020.9116502

[73] Matthias Möller and Cornelis Vuik. 2017. On the impact of quantum computing technology on future developments in high-performance scientific computing. *Ethics Info Technol* 19, 4 (Aug. 2017), 253–269. DOI: http://dx.doi.org/10.1007/s10676-017-9438-0

[74] Yudong Cao and Timothy Hirzel. 2020. Quantum Acceleration in 2020. Retrieved from https://www.infoq.com/articles/quantum-acceleration-2020

[75] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak 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 Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, 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 (2019), 505–510. DOI: http://dx.doi.org/10.1038/s41586-019-1666-5

[76] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. 2020. Quantum computational advantage using photons. *Science* 370, 6523 (2020), 1460–1463. DOI: http://dx.doi.org/10.1126/science.abe8770

[77] Juneseo Lee, Alicia B. Magann, Herschel A. Rabitz, and Christian Arenz. 2021. Progress toward favorable landscapes in quantum combinatorial optimization. *Phys. Rev. A* 104, 3 (Sep. 2021). DOI: http://dx.doi.org/10.1103/physreva.104.032401

[78] Jonathan Ward, Johannes Otterbach, Gavin Crooks, Nicholas Rubin, and Marcus da Silva. 2018. QAOA performance benchmarks using max-cut. In *APS March Meeting Abstracts*, Vol. 2018. R15.007.

[79] Gavin E. Crooks. 2018. Performance of the Quantum Approximate Optimization Algorithm on the Maximum Cut Problem. Retrieved from http://dx.doi.org/10.48550/ARXIV.1811.08419

[80] Byron Tasseff, Tameem Albash, Zachary Morrell, Marc Vuffray, Andrey Y. Lokhov, Sidhant Misra, and Carleton Coffrin. 2022. On the Emerging Potential of Quantum Annealing Hardware for Combinatorial Optimization. Retrieved from http://dx.doi.org/10.48550/ARXIV.2210.04291

[81] Hristo N. Djidjev, Guillaume Chapuis, Georg Hahn, and Guillaume Rizk. 2018. Efficient Combinatorial Optimization Using Quantum Annealing. Retrieved from http://dx.doi.org/10.48550/ARXIV.1801.08653

[82] Tameem Albash and Daniel A. Lidar. 2018. Demonstration of a scaling advantage for a quantum annealer over simulated annealing. *Phys. Rev. X* 8, 3 (July 2018). DOI: http://dx.doi.org/10.1103/physrevx.8.031016

[83] S. Ebadi, A. Keesling, M. Cain, T. T. Wang, H. Levine, D. Bluvstein, G. Semeghini, A. Omran, J.-G. Liu, R. Samajdar, X.-Z. Luo, B. Nash, X. Gao, B. Barak, E. Farhi, S. Sachdev, N. Gemelke, L. Zhou, S. Choi, H. Pichler, S.-T. Wang, M. Greiner, V. Vuletić, and M. D. Lukin. 2022. Quantum optimization of maximum independent set using Rydberg atom arrays. *Science* 376, 6598 (2022), 1209–1215. DOI: http://dx.doi.org/10.1126/science.abo6587

[84] Troels F. Rønnow, Zhihui Wang, Joshua Job, Sergio Boixo, Sergei V. Isakov, David Wecker, John M. Martinis, Daniel A. Lidar, and Matthias Troyer. 2014. Defining and detecting quantum speedup. *Science* 345, 6195 (July 2014), 420–424. DOI: http://dx.doi.org/10.1126/science.1252319

[85] Salvatore Mandrà and Helmut G. Katzgraber. 2018. A deceptive step towards quantum speedup detection. *Quantum Sci. Technol.* 3, 4 (July 2018), 04LT01. DOI: http://dx.doi.org/10.1088/2058-9565/aac8b2

[86] NetworkX. 2023. NetworkX—Network Analysis in Python. Retrieved from https://networkx.org/

[87] Gurobi. 2023. Gurobi Optimization. Retrieved from https://www.gurobi.com/

[88] Madita Willsch, Dennis Willsch, Fengping Jin, Hans De Raedt, and Kristel Michielsen. 2020. Benchmarking the quantum approximate optimization algorithm. *Quantum Info. Process.* 19, 7 (July 2020), 197. DOI: http://dx.doi.org/10.1007/s11128-020-02692-8

[89] Panagiotis Kl Barkoutsos, Giacomo Nannicini, Anton Robert, Ivano Tavernelli, and Stefan Woerner. 2020. Improving variational quantum optimization using CVaR. *Quantum* 4 (Apr. 2020), 256. DOI: http://dx.doi.org/10.22331/q-2020-04-20-256

[90] Li Li, Minjie Fan, Marc Coram, Patrick Riley, and Stefan Leichenauer. 2020. Quantum optimization with a novel gibbs objective function and ansatz architecture search. *Phys. Rev. Res.* 2, 2 (Apr. 2020), 023074. DOI: http://dx.doi.org/10.1103/PhysRevResearch.2.023074

[91] 2023. IBM Cloud Qiskit Runtime. Retrieved from https://cloud.ibm.com/quantum. Accessed 2023-05-15.

[92] Endre Boros, Peter L. Hammer, and Gabriel Tavares. 2007. Local search heuristics for quadratic unconstrained binary optimization (QUBO). *J. Heuristics* 13, 2 (2007), 99–132.

[93] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. 2014. The unconstrained binary quadratic programming problem: A survey. *J. Comb. Optimiz.* 28, 1 (2014), 58–81.

[94] M. Born and V. Fock. 1928. Beweis des Adiabatensatzes. *Zeitschrift für Physik* 51, 3 (1928), 165–180. DOI: http://dx.doi.org/10.1007/BF01343193

[95] Tosio Kato. 1950. On the adiabatic theorem of quantum mechanics. *J. Phys. Soc. Japan* 5, 6 (1950), 435–439. DOI: http://dx.doi.org/10.1143/JPSJ.5.435

[96] Sabine Jansen, Mary-Beth Ruskai, and Ruedi Seiler. 2007. Bounds for the adiabatic approximation with applications to quantum computation. *J. Math. Phys.* 48, 10 (2007), 102111. DOI: http://dx.doi.org/10.1063/1.2798382 arXiv:quant-ph/0603175

[97] Vicky Choi. 2008. Minor-embedding in adiabatic quantum computation: I. The parameter setting problem. *Quantum Info. Process.* 7 (2008), 193–209.

[98] Andrew Wack, Hanhee Paik, Ali Javadi-Abhari, Petar Jurcevic, Ismael Faro, Jay M. Gambetta, and Blake R. Johnson. 2021. Quality, Speed, and Scale: Three Key Attributes to Measure the Performance of Near-term Quantum Computers. Retrieved from http://dx.doi.org/10.48550/ARXIV.2110.14108

[99] Lennart Bittel and Martin Kliesch. 2021. Training variational quantum algorithms is NP-Hard. *Phys. Rev. Lett.* 127, 12 (Sept. 2021), 120502. DOI: http://dx.doi.org/10.1103/PhysRevLett.127.120502

[100] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. 2018. Barren plateaus in quantum neural network training landscapes. *Nature Commun.* 9, 1 (Nov. 2018), 4812. DOI: http://dx.doi.org/10.1038/s41467-018-07090-4

[101] M. Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J. Coles. 2021. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Commun.* 12, 1 (Mar. 2021), 1791. DOI: http://dx.doi.org/10.1038/s41467-021-21728-w

[102] Samson Wang, Enrico Fontana, M. Cerezo, Kunal Sharma, Akira Sone, Lukasz Cincio, and Patrick J. Coles. 2021. Noise-induced barren plateaus in variational quantum algorithms. *Nature Commun.* 12, 1 (Nov. 2021), 6961. DOI: http://dx.doi.org/10.1038/s41467-021-27045-6

[103] Carlos Ortiz Marrero, Mária Kieferová, and Nathan Wiebe. 2021. Entanglement induced barren plateaus. Retrieved from https://arXiv:2010.15968. DOI: http://dx.doi.org/10.48550/arXiv.2010.15968

[104] Jonathan Wurtz and Danylo Lykov. 2021. Fixed-angle conjectures for the quantum approximate optimization algorithm on regular MaxCut graphs. *Phys. Rev. A* 104, 5 (Nov. 2021), 052419. DOI: http://dx.doi.org/10.1103/PhysRevA.104.052419

[105] V. Akshay, D. Rabinovich, E. Campos, and J. Biamonte. 2021. Parameter concentrations in quantum approximate optimization. *Phys. Rev. A* 104, 1 (July 2021), L010401. DOI: http://dx.doi.org/10.1103/PhysRevA.104.L010401

[106] David Bernal Neira, Davide Venturelli, Filip Wudarski, and Eleanor Rieffel. 2022. Benchmarking the operation of quantum heuristics and ising machines: Scoring parameter setting strategies on real world optimization applications. In *APS March Meeting Abstracts*, Vol. 2022. F38–005.

[107] Stochastic Benchmark. 2022. Retrieved from https://github.com/usra-riacs/stochastic-benchmark

Received 22 March 2023; revised 30 January 2024; accepted 1 July 2024