Quantum optimization using a 127-qubit gate-model IBM quantum computer can outperform quantum annealers for nontrivial binary optimization problems

Natasha Sachdeva,* Gavin S. Hartnett, Smarak Maity, Samuel Marsh, Yulun Wang, Adam Winick, Ryan Dougherty, Daniel Canuto, Michael Hush, Pranav S. Mundada, Christopher D. B. Bentley, Michael J. Biercuk, and Yuval Baum
Q-CTRL, Sydney, NSW Australia, and Los Angeles, CA USA

We introduce a comprehensive quantum solver for binary combinatorial optimization problems on gate-model quantum computers that outperforms any published alternative and consistently delivers correct solutions for problems with up to 127 qubits. We provide an overview of the internal workflow, describing the integration of a customized ansatz and variational update strategy, efficient error suppression in hardware execution, and an overhead-free post-processing to correct for bit-flip errors. We benchmark this solver on IBM quantum computers for several classically nontrivial unconstrained binary optimization problems—the entire optimization is conducted on hardware with no use of classical simulation or prior knowledge of the solution. First, we demonstrate the ability to correctly solve Max-Cut instances for random regular graphs with a variety of densities using up to 120 qubits, and whose topologies are not matched to device connectivity. These demonstrations are at least $4\times$ larger than previous successful implementations on a trapped-ion quantum computer and deliver up to $9\times$ increased likelihood of success for identical problem instances at the scale of 32 qubits. Next, we apply the solver to a higher-order binary optimization and successfully search for the ground state energy of a 127-qubit spin-glass model with linear, quadratic, and cubic interaction terms. Use of this new quantum solver increases the likelihood of finding the minimum energy by up to $\sim 1,500\times$ relative to published results using a DWave annealer, and can find the correct solution when the annealer fails. Furthermore, for both problem types, the Q-CTRL solver outperforms a heuristic local solver used to indicate the relative difficulty of the problems pursued. Overall, these results represent the largest quantum optimizations successfully solved on hardware to date, and demonstrate the first time a gate-model quantum computer has been able to outperform an annealer for a class of binary optimization problems.

I. INTRODUCTION

In recent years, the quantum computing industry has seen a significant acceleration in both hardware and software development. This progress is moving the industry beyond the noisy intermediate-scale quantum (NISQ) era [1]—where demonstrating conceptual capability was the primary focus—to the brink of delivering commercial and economic value for industry-relevant problems [2]. In this new so-called utility era, combinatorial optimization problems that are ubiquitous in logistics, finance, transportation, networking, and other industries [3] represent a key problem-class of interest for quantum solutions.

These problems are classically nontrivial, motivating the use of heuristics at relatively modest scales. Commonly used classical solvers, such as CPLEX [4] and Gurobi [5], are highly optimized and efficient engines that were developed and maintained over the last several decades (see Ref. [6] for more recent solvers). Nonetheless, these approaches are designed to find "good" solutions more efficiently than exhaustive search; consequently, they trade off run-time for solution quality. For medium-scale problem instances, they frequently struggle to find a favorable balance and the optimizations with an acceptable run-time can under perform the true optimal solutions by up to 50% for certain applications (see, for instance, challenges in periodic train scheduling [7]).

Quantum optimization algorithms have been identified as enabling an attractive near-term opportunity because quadratic unconstrained binary optimization (QUBO) problems, underlying key high-impact applications, are well-suited to quantum computers. Moreover, the combination of rapidly increasing problem complexity and the shortcomings of classical heuristics opens an opportunity to deliver commercial value with reasonably modest systems. Simple estimates suggest that it may be possible for quantum computers with 500–1000 functional qubits to outperform optimized classical heuristics for a range of industry-relevant problems, potentially with only modest performance increases above current hardware [8–12]. For such problems, both gate-model and annealer-based solvers provide a promising pathway towards useful optimization at relevant scales.

Gate-model quantum computers can execute QUBO problems and facilitate direct incorporation of higher-order optimization terms and constraints [13–18]. Unfortunately, the generality of the hardware opens many opportunities for errors which significantly curtail useful system size and performance. Recent experimental demonstrations have shown that clever algorithmic execution of the quantum approximate optimization algorithm (QAOA) can outperform "brute-force" random selection [19, 20], but still struggles to deliver high-quality solutions. In trapped-ion processors, successful QAOA implementations capable of returning optimal results have reached approximately 32 qubits [21], but have relied upon classical simulation to precalculate optimized circuits in place

* natasha.sachdeva@q-ctrl.com

of the hybrid-quantum-optimization loop. Those demonstrations also required high-depth QAOA circuits in order to find the optimal solution, typically failing for shallow-depth QAOA.

Quantum annealing (QA) is an efficient alternate technique for solving binary optimization problems [13, 22] because the limited functionality of the hardware reduces error pathways and simplifies device scaling. Gate-model quantum computers and quantum annealers share formal equivalence from the perspective of computational complexity [23], but various studies point to potential practical advantages for annealers on certain problem types [24, 25]. In multiple experiments, annealers have provided high-quality solutions for QUBO problems, and higher-order terms and/or constraints have also been included by leveraging their scaling advantages via the use of slack variables and penalty terms [26–28]. These systems have successfully been used to solve graph-optimization problems with hundreds of nodes, but show signs of optimality gaps at modest scales that challenge their relevance [29, 30]. In circumstances where head-to-head comparisons between gate-model quantum computers and quantum annealers have been performed [19, 29, 31], for all instances examined, the use of an annealer provided superior solutions. This has significantly contributed to growing concern as to whether NISQ-era gate-model quantum computers can deliver commercially relevant solutions [32].

In this work, we introduce a comprehensive quantum solver for binary combinatorial optimization problems on gate-model quantum computers that returns correct solutions up to 127-qubit implementations, surpassing any published alternative quantum solver. We provide an overview of the internal workflow, including a novel variational ansatz, efficient parametric compilation, automated error suppression in circuit execution, and classical post-processing to address bit-flip errors without adding execution overhead. We benchmark this solver on IBM quantum computers for several classically non-trivial unconstrained binary optimization problems and compare against published results using alternate hardware systems. First, we consider Max-Cut instances [33] for both sparse and dense random regular graphs that are not matched to device connectivity. For 3-regular graphs, the solver returns the correct Max-Cut value with unit probability over all instances tested up to 120 nodes (120 qubits). We provide a quantitative comparison to previously published trapped-ion results using identical problem instances [21] and demonstrate up to $9\times$ enhanced likelihood of finding the correct solution at the maximum scale previously achieved (32 qubits), despite previous efforts using deep $p \geq 10$ QAOA circuits pre-optimized using classical simulation. For denser graph connectivity, the solver finds the correct Max-Cut for 4-regular graphs up to at least 80 nodes, and for 7-regular graphs up to at least 50 nodes. Next, we apply the solver beyond quadratic optimization to search for the ground state energy of a 127-qubit spin-glass model with linear, quadratic, and cubic terms using instances defined in Ref. [28]. For this problem, we demonstrate the ability to find the correct energy minimum in four of six instances tested, including an instance where the annealer did not find the correct ground-state energy; for the two additional instances tested, solution approximation ratios exceed 99.5%. The solver increases the likelihood of finding the minimum energy by up to $\sim 1,500\times$ relative to the published annealer results for identical problem instances. Furthermore, for both problem types, the solver outperforms a heuristic local solver that applies a greedy optimization to a random distribution of candidate solutions; the optimized CPLEX classical solver provides the ground-truth. Overall, these results represent the largest quantum optimizations successfully performed to date for problems that challenge classical solvers. To the best of our knowledge, this work demonstrates the first time a gate-model quantum computer has been able to outperform a quantum annealer for binary optimization problems.

II. HARDWARE-AGNOSTIC EXECUTION-OPTIMIZED QUANTUM SOLVER

We introduce a fully abstracted end-to-end solver for unconstrained binary optimization problems over length-n bitstrings of the form

$$\min_{\boldsymbol{x}\in\{0,1\}^n} C(\boldsymbol{x}), \tag{1}$$

where the objective function $C(\boldsymbol{x})$ may be represented as a polynomial of the binary variables. The key elements of this solver are visually represented in Fig. 1. It builds upon on a commonly used quantum optimization algorithm, the Quantum Approximate Optimization Algorithm, which is a hybrid classical-quantum algorithm inspired by the adiabatic theorem to approximate solutions for combinatorial optimization problems [34].

The solver accepts a general unconstrained optimization problem as an input, defined as in Eq. 1: The user provides the optimizer a representation of the objective function that fully defines the problem, where both symbolic polynomial expressions and graph representations are accepted. After defining the optimization problem, we construct the variational circuit using additional parameterization that provides the flexibility to tune the input state of the solver, in addition to the conventional QAOA optimization parameters. We then compile this ansatz circuit using an error-aware, optimized compilation procedure described below. The parameterized circuits are iteratively submitted to the quantum hardware as part of the hybrid quantum-classical optimization loop in which the variational circuit parameters are optimized in order to minimize the cost function between circuit executions. Execution of each circuit invokes a comprehensive error-suppression pipeline in order to reduce gate-level and circuit-level errors [35]. Finally, we post-process the measured distributions to mitigate random bit-flip errors in a

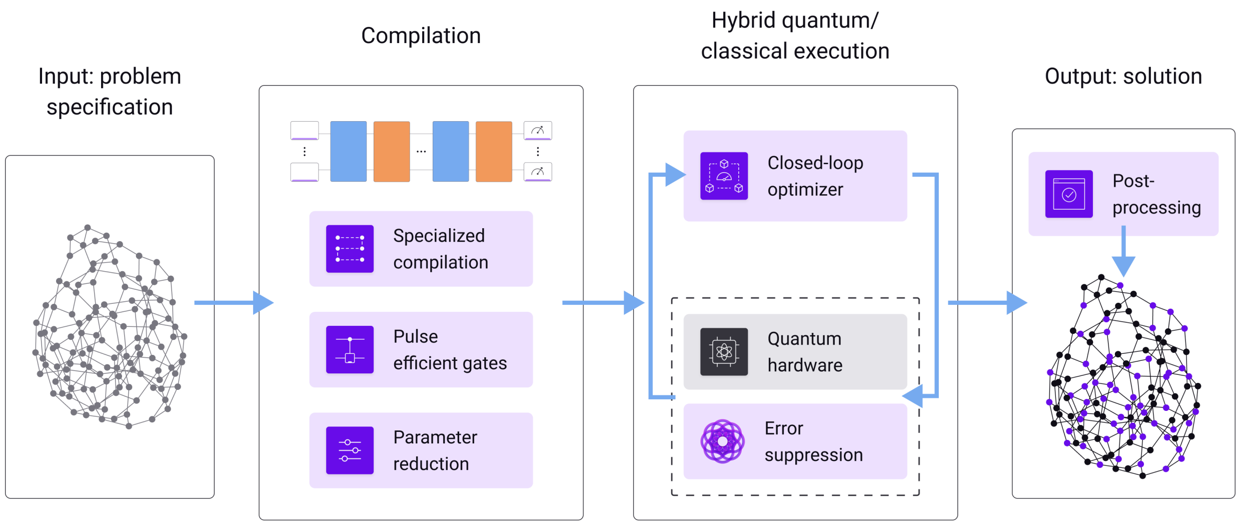

FIG. 1. The optimization pipeline behind the quantum solver. In the *input stage*, a user-defined representation of the optimization problem is provided to the solver. Next, we construct the variational ansatz that is employed during the optimization process. In the *compilation stage*, the variational circuit is compiled to produce an optimized parametric circuit. The compiler may leverage an enhanced gate set of pre-calibrated pulse-level instructions of efficient (fast) 2-qubit gates. Finally, the optimization parameters are initialized and then passed to the closed-loop optimizer. Depending on the number of circuit parameters, a parameter reduction procedure may take place. In the *hybrid execution stage*, the parameterized circuits are submitted to the quantum hardware and the variational circuit parameters are optimized via a hybrid quantum-classical optimization loop. In the *output stage*, after final post-processing, the solution to the optimization problem is returned to the user.

manner that involves no additional hardware execution and scales linearly with problem size. Beyond the initial problem definition, all processes are fully automated, eliminating the need for users to understand quantum circuit construction, quantum hardware characteristics, or error suppression and mitigation techniques. We review the key elements of the solver in detail below.

Enhanced variational ansatz: After receiving the user-defined problem, the solver constructs the variational ansatz for the optimization routine. We aim to overcome the challenge of limited accessible circuit depth in commercial quantum processors, as this curtails the achievable QAOA repetition level, denoted p, well before the value $p > 11$ anticipated to be required to achieve definitive quantum advantage [36] in a conventioanl implementation. We follow previous literature to address both scalability and generalizability [37] through use of a modified QAOA ansatz and update algorithm.

The standard QAOA ansatz starts with an initial n-qubit equal superposition state, established by application of a Hadamard gate to each qubit, followed by alternating layers of unitary transformations corresponding to the cost and mixer Hamiltonians. The cost Hamiltonian, H_C, encodes the optimization problem to be solved and the corresponding unitary transformation, U_C, is parameterized by variational parameters γ_i, where $i = 1, \ldots, p$. The mixer Hamiltonian, H_M, facilitates transitions between states in order to explore the solution space, and the mixing Unitary U_M for each layer is tuned with the variational parameters β_i.

Our modified QAOA ansatz, shown in Fig. 2, substitutes the typical equal-superposition starting state with a new input state generated using arbitrary rotation operators $R_y(\theta_j)$, where $j = 1, \ldots, n$ and θ_j varies individually for each qubit. (The standard QAOA ansatz is recovered in the case where all θ_j are set to $\pi/2$, producing the uniform superposition state.) This approach builds on related work, but constitutes a departure from warm-start QAOA methods [38–41] in which the initial state remains unchanged throughout the circuit optimization. Instead,

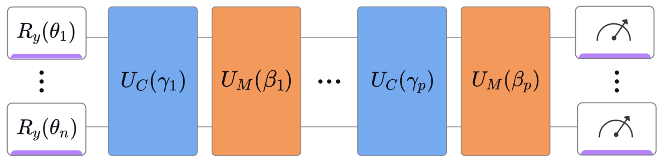

FIG. 2. The ansatz we employ for solving unconstrained optimization problems. The initial state is parameterized angles θ_j, for $j = 1, .., n$, followed by the standard QAOA repetition pattern of a cost-unitary and a mixing-unitary, defined by the γ_i, β_i parameters for $i = 1, ..., p$. This ansatz has a total of $n + 2p$ parameters where n is the number of qubits and p the number of QAOA blocks.

our ansatz starts from the zero-information point in the first cycle of the classical loop, and the initial state parameters are updated iteratively alongside $\{\boldsymbol{\gamma}, \boldsymbol{\beta}\}$. We emphasize that the additional parameterization in this ansatz and parameter update strategy is meant to enable the solver to converge on correct solutions without the need for large p, hence mitigating practical concerns around the impact of incoherent errors and limited circuit depths encountered on today's machines.

Efficient parametric compilation: Our solver employs an efficient form of parametric compilation in order to reduce execution overheads—a practical rather than fundamental consideration that can have substantial impact on the achieved solution quality. During operation of the solver, the parameters of the executed quantum circuits are updated in real-time based on feedback from the classical optimizer during execution of the hybrid loop. This procedure repeats itself anywhere from tens of times for small problems to thousands of times for more complex problems. Compilation overhead causes effective dead-time in which the hardware system is not being sampled, introducing sensitivity to drifts on the timescale of the optimization loop's execution. Accordingly, inefficient compilation introduces additional sensitivity to errors, the cumulative effect of which can impair convergence.

Our parametric compilation strategy incurs the initial compilation overhead only once, followed by real-time updates of the circuit parameters during the execution of the hybrid loop to minimize dead-time. Successful implementation requires careful consideration of the fact that symmetries and circuit identities that apply to specific parameters do not hold in general [42]. Additionally, while the available native gate set is universal, commonly used ansatzes such as that in Fig. 2 may be compiled more efficiently by enriching the available gate set. In particular, our compiler can leverage pulse-level instructions of pre-calibrated gates to realize faster and higher quality arbitrary two-qubit rotations [43].

Automated error suppression in hardware execution: The solver uses an automated error-suppression pipeline [35] during hardware execution of each parameterized circuit. This strategy combines several key components beyond efficient compilation: layout selection [44], dynamical decoupling embedding for simultaneous crosstalk and dephasing suppression, automated AI-driven gate-waveform replacement, and readout-error mitigation. We emphasize that the form of readout-error mitigation employed here scales as $\mathcal{O}(1)$ with qubit count and therefore does not introduce any additional overhead. This pipeline has been shown to deliver $> 1000\times$ performance enhancement in algorithmic benchmarks on IBM hardware [35, 45].

Automated classical optimization loop: The solver optimizes parameters by executing circuits directly on quantum hardware, forsaking any form of classical simulation or parameter pre-computation generally employed to simplify the optimization procedure (c.f. Ref. [21]). We therefore focus extensively on efficiently executing the classical optimization loop used to iteratively tune the variational QAOA parameters $\{\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{\theta}\}$ between circuit executions. The classical procedure maintains responsibility for the parameter update step and is carefully constructed, but all supporting measurements and inputs are derived from circuit execution on quantum hardware.

We implement the classical component of the optimization loop using a covariance matrix adaptation evolution strategy (CMA-ES); it is a fast, stochastic, and derivative-free method used for non-convex numerical optimization problems [46]. We select Conditional Value-at-Risk (CVaR) as the objective function, as it has been shown to lead to faster convergence for combinatorial optimization problems [47].

To maximize the efficiency of quantum-resource utilization, we use multiple parameter update strategies in the optimization procedure. The number of variational parameters (the $2p$ parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$) is reduced by employing the Fourier parameterization introduced in [48], although for the problems presented in this work $p = 1$ was sufficient to find the known correct solutions. Additionally, the $\boldsymbol{\theta}$ angles are initialized to $\pi/2$ and are updated every few iterations according to the information collected during the previous optimization steps; this procedure takes advantage of the aggregate bitstring distributions observed in the prior steps to define a favorable starting superposition for the circuit in the subsequent iterations. During the execution of the hybrid loop, the $\boldsymbol{\theta}$ angles trend towards the extreme values 0 or π (placing the qubits in the state $|0\rangle$ or $|1\rangle$, respectively) and the variational parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ both become very small as the initial superposition state converges toward the solution. $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are initialized randomly; this method does not require problem-specific parameter initialization in order to converge toward high-quality solutions [49].

Classical post-processing: Uncorrelated bitflip errors in the measured output distribution significantly impact solution quality. These errors can come from a variety of sources, commonly dominated by device T_1 noise and measurement errors. For large-scale problems beyond 100 qubits, eliminating these errors completely during circuit execution is unrealistic with contemporary devices. However, these errors may be addressed by a purely classical post-processing step.

We implement an $\mathcal{O}(n)$ naïve greedy optimization on the solution bitstrings returned from measuring the final, optimized variational circuit. This may be done efficiently given that the cost function C is local—evaluating the cost difference $C(\boldsymbol{x}') - C(\boldsymbol{x})$ for $\boldsymbol{x}, \boldsymbol{x}'$ differing by a single bitflip involves a constant number of steps. With no additional measurements or circuit modifications required, this procedure randomly traverses the measured solution bitstring and greedily flips bits if they improve the bitstring cost. This approach continues until no improvement is realized from successive bitflips, and we cap the number of full traversals of the bitstring to five passes. This implementation is not problem-dependent and can be applied to any unconstrained binary combinatorial optimization problem. Importantly, it requires no additional execution overhead in increased shots on hardware. The use of greedy classical optimization in conjunction with QAOA has been explored in Refs. [50–52].

III. HARDWARE IMPLEMENTATION

We test the performance of the solver via execution on 127-qubit IBM quantum computers *ibm_ brisbane* and *ibm_ sherbrooke*. In all cases, the circuit executions within the full optimization loop are executed on the quantum device, without any use of quantum simulators or classical solvers in the background. The solver operation is not tuned or modified for the specific problems we implement and the results we show faithfully represent the quality of expected solutions for any optimization problem in the same complexity class as the ones we optimized.

We evaluate solver performance by determining a series of metrics which highlight the quality of the output solutions relative to the ground truth, and the likelihood of achieving the correct solution. In our tests, the ground truth is determined using the classical solver CPLEX [4].

First, we evaluate the approximation ratio; for a solution defined over binary variables as $\boldsymbol{x} \in \{0,1\}^n$, the approximation ratio is

$$\mathrm{AR}(\boldsymbol{x}) = \frac{C(\boldsymbol{x}) - C_{\max}}{C_{\min} - C_{\max}}, \tag{2}$$

where $C(\boldsymbol{x})$ is the objective function, $C_{\min}$ and $C_{\max}$ are the true minimum and maximum energies of the model. An AR = 100% corresponds to the solver finding a solution matching the ground truth; good solvers return solutions with AR $\sim$ 100%.

In addition, we provide simple metrics allowing quantitative comparison of solution quality by including the raw likelihood that the top solution found by the solver is returned. This metric is particularly relevant for cases where the achieved AR = 100%, i.e., the top solution is the correct solution, as it provides additional information on the stability of the optimization process and the relative correctness of each sampling of the optimized circuit. Any circuit, when sampled, will return a distribution of outputs that ideally skews towards the ground truth; a high-quality solver will deliver a large overlap of outputs with the ground truth while a weaker solver may find the ground-truth solution only rarely if at all.

When available, we compare the quality of solutions returned from our solver against published results derived from a commercial DWave annealer and a Quantinuum trapped-ion gate-model quantum computer. In such instances, the problems we execute are defined identically to those executed in the relevant references. In addition, to provide insight into the relative challenge of the test problems we explore, we also include a classical local solver in our benchmarks. A summary of all results achieved with comparisons against previously published literature appears in Table I. Additional information including estimates of execution time and cost is included for completeness in the Appendix.

A. Max-Cut

The Max-Cut problem is a classic problem in combinatorial optimization and graph theory. It involves partitioning the vertices of a graph into two distinct subsets such that the number of edges between the two subsets is maximized. There is a formal equivalence between Max-Cut and QUBO problems—any Max-Cut instance may be mapped to a QUBO instance and vice versa [54]. Max-Cut on k-regular graphs, meaning graphs where each node is connected to exactly k other nodes, are NP-hard problems in the case where $k > 2$ and have real-world significance, particularly in networking [33, 55]. For arbitrary graphs, this problem is also challenging because it does not exploit native symmetries or connectivity of the underlying quantum hardware in use. Accordingly, Max-Cut on random 3-regular graphs is a standard benchmark problem used to test the efficacy of quantum algorithms against classical counterparts by being both a simple and nontrivial example of a QUBO problem.

Max-Cut can be framed as a minimization of an objective function of the form

$$C(\boldsymbol{z}) = -\frac{1}{2}\sum_{i<j} A_{ij}\left(1 - z_i z_j\right) . \tag{3}$$

Here, we have expressed the cost function in terms of the Ising spin variables $\boldsymbol{z} = 2\,\boldsymbol{x} - 1 \in \{-1,1\}^n$, and A is the $n \times n$ adjacency matrix of the graph ($A_{ij} \neq 0$ if $(i,j) \in G$, otherwise $A_{ij} = 0$), with n is the number of nodes. We distinguish between unweighted graphs where $A_{ij} = 1$ if $(i,j) \in G$, and weighted graphs where $A_{ij} = w_{ij} \in \mathbb{R}$ if $(i,j) \in G$. For all problems we present, the number of nodes is equivalent to the number of qubits used on the device. Candidate partitions of the graph are then represented by length-n bitstrings (entries are zero or one depending on which subgraph each node is assigned to), and each graph node is assigned to a single qubit. Given a partition, the cut value is calculated straightforwardly as the number of edges between the two groups.

In Fig. 3(a-b), we demonstrate the performance of our solver on two example Max-Cut problems: An unweighted 120-node 3-regular graph and a weighted 80-node 4-regular graph (see Table I for summary results on these and additional instances). In these figures, we plot the histogram of cut values calculated from the bitstring solutions returned by the solver, output as measurements of the optimal circuit. A good solution approaches and ideally overlaps the actual maximum value derived from CPLEX (denoted by a dashed vertical red line) towards the right side of the plot.

Our performance baseline is set by sampling bitstrings uniformly at random (light gray histogram), akin to a

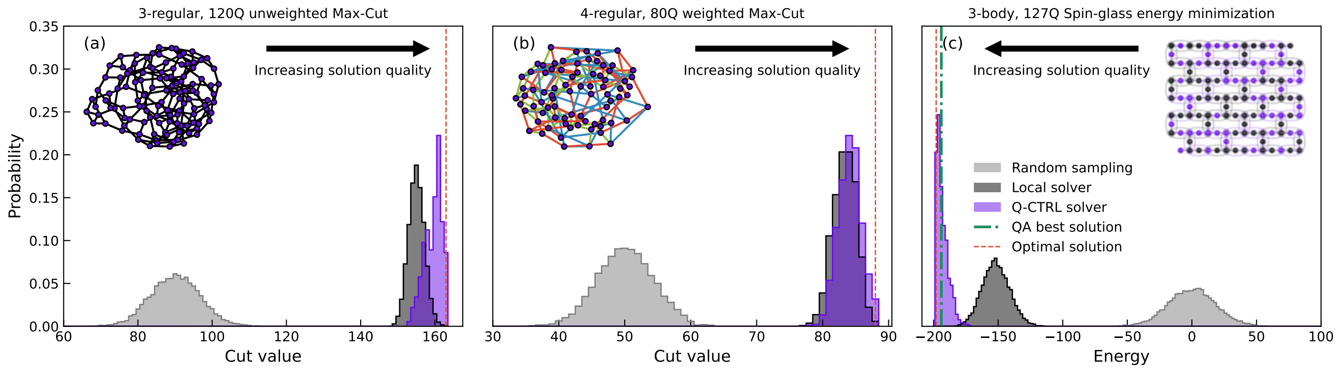

FIG. 3. Performance of the quantum solver on three combinatorial optimization problems. (a) A Max-Cut instance for a 3-regular unweighted random graph with 120 nodes. The plot shows the distribution of cut values for 12k configurations sampled from the optimal circuit found by the Q-CTRL quantum solver, executed on *ibm_ brisbane*. The true maximum cut value of 163 is successfully attained by the solver. The random search and classical local solver distributions show the result of 12k configurations sampled uniformly at random and 12k local minimum configurations. (b) A Max-Cut instance for a 4-regular weighted random graph with 80 nodes. The weight of each edge was randomly chosen from four possible values $[1/4, 2/4, 3/4, 1]$, represented by different edge colors. Shown, cut values for 10k configurations sampled from the optimal circuit found by the quantum solver, executed on *ibm_ brisbane*. The true maximum cut value of 88 is successfully attained by the solver. (c) Energy minimization of a 127-node high-order spin-glass model with parameters from *ibm_ washington* instance 5 as defined in Ref. [53]. The plot shows the energy distribution for 15k configurations sampled from the optimal circuit found by the quantum solver, executed on *ibm_ sherbrooke*, and for 15k configurations sampled uniformly at random (brute-force) and 15k local minimum configuration (local solver). The green vertical dash-dotted line indicates the minimum energy found by a DWave quantum annealer (QA) for this specific instance [19] out of $1,328,000$ samples (annealing processes). The lowest and highest energies of this model instance (the energy band edges) are known to be -198 and 192, respectively. As in (a,b), the true ground state was attained by the solver.

random brute-force scan through the solution set. In all cases, the number of samples is taken to be the same as the samples of the optimal QAOA circuit. We choose this baseline as an indication of the difficulty of the problem and also because, as shown later, a naive QAOA implementation using standard Qiskit compilers returns results indistinguishable from random selection (see Fig. 4 for detail).

Executing the entire quantum-solver workflow articulated in Fig. 1 results in a dramatic shift of the output histogram to large cut values, and produces solutions which overlap the known maximum cut value. Up to 120 qubits we find that not only does the Q-CTRL quantum solver's distribution overlap the maximum cut value, but it is asymmetrically skewed towards this value resulting in a high likelihood of achieving the correct answer. The quantum solver also provides superior solutions in these cases to those found using a classical local solver shown in the dark gray histogram (see Appendix for implementation details). For the 120-qubit 3-regular graph shown in Fig. 3a, the quantum solver delivers the correct result with 8.6% likelihood while the local solver returns it with 0.017% likelihood.

In Table I, we show additional results across different Max-Cut problems with higher-density graphs, up to 7-regular graphs. In all executions attempted, and across all specific problem instances within each class, the Q-CTRL quantum solver achieves AR = 100%. In these cases, the solver also uniformly outperforms the classical local solver in solution likelihood, and matches the CPLEX ground-truth solutions. Additional data across the problem instances explored appear in the Appendix.

These results demonstrate superior performance to previously published results on a trapped-ion quantum processor with 32 qubits from Quantinuum. Because we do not have access to the same trapped-ion device, we cannot comment on relative base-hardware performance had the Q-CTRL pipeline been implemented; rather we only make comparisons to the results published to date. Table I provides a direct comparison of executions of the Q-CTRL quantum solver on IBM hardware using the same problem instances presented in [21] for smaller graphs up to 32 nodes. The trapped-ion demonstrations failed to obtain a maximum cut for $k = 3$ and $n \approx 32$ nodes [21] using a $p = 1$ implementation of QAOA, whereas our solver achieves AR = 100% across all instances for $p = 1$. In execution of higher-depth circuits with $p = 10$ and $p = 11$, the trapped-ion device returned the maximum cut, but with up to 9× lower likelihood than our solver; across these problem instances our solver finds the maximum cut with likelihood of ∼82–94%, while on Quantinuum the range was ∼10–18%. Perhaps more importantly, the trapped-ion demonstrations were limited to hardware execution of a single circuit using pre-optimized $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ parameters attained via classical numeric simulation. Our solver, by comparison, exclusively runs full hardware execution of the QAOA optimization loop.

To the best of our knowledge these results represent the largest nontrivial Max-Cut problems successfully executed on quantum hardware across a range of problem instances and graph densities.

B. HOBO Ising spin glass

Next, we consider an optimization problem designed to find the ground-state energy of a random-bond 127-qubit Ising model possessing cubic terms. This is an example of unconstrained Higher-Order Binary Optimization (HOBO), known to be considerably more complicated to solve than conventional QUBO problems [13–16].

In our benchmarking, we follow the same model as introduced in [58]. In particular, we consider the model with 1-, 2-, and 3-body interaction terms tailored to connectivity of the IBM Eagle-class heavy-hex devices [58] (this device topology includes the systems used in our experiments). Denoting the heavy-hex connectivity graph as $G = (V, E)$, the cost function for this problem is

$$C(\boldsymbol{z}) = \sum_{i\in|V|} a_i z_i + \sum_{(i,j)\in E} b_{i,j} z_i z_j + \sum_{(i,j,k)\in W} c_{i,j,k} z_i z_j z_k \,. \tag{4}$$

The 1-, 2-, and 3-body couplings are given by a, b, c, respectively, and W is a particular set of 3-tuples that corresponds to 3-body interaction terms that can be efficiently implemented on IBM hardware. Overall, the objective function contains 127 linear terms, 142 quadratic terms and 69 cubic terms. The values of the 338 coefficients are randomly chosen to be ± 1. We refer to different choices of these coefficients as model instances, and use identical instances and notation as in [58] for the problems defined on the *ibm_ washington* device.

In Fig. 3c we show the performance of our solver for "instance 5" of the model accessed from Ref. [53] (see the Appendix for the remaining instances). Here, the distribution produced by the quantum solver is narrowly concentrated around the edge of the energy band (in this case solutions at the left edge of the plot are best), producing high quality ground-state estimates that match the known ground state energy. As above, this solution outperforms all candidate solutions returned by random sampling. As for Max-Cut, we also benchmark against a local solver, showing that our solver consistently outperforms the local solver. We provide further detail in Table I where we highlight the performance of the Q-CTRL solver over multiple problem instances reproduced from [19]. The worst-case approximation ratio returned for the selected problem instances tested was AR = 99.5%.

Through the introduction of additional variables and interaction terms in the energy, this HOBO problem may be converted to a QUBO problem, allowing for execution on quantum annealers [13]. The original 127-node HOBO in Eq. 4 can be written as a 265-node QUBO with 625 quadratic terms and 69 penalty constants. The nonlocal nature of the additional quadratic terms and the additional energy scales introduced by the multiple penalty terms render the extended QUBO problem challenging for quantum annealers [59]. For that reason, in order to benchmark this problem using an annealer, the authors in [19] developed a dedicated order-reduction scheme that was tailored for the specific problem and for the specific QA device used in that work.

Fig. 3c includes the reported best solution found using quantum annealing [19] (vertical green dashed line), demonstrating that the Q-CTRL quantum solver executed on a gate-model quantum computer can exceed the performance of a quantum annealer via the quality of the best result returned. Further insight is gleaned through examination of full results as detailed in Table I; for this problem instance our solver sampled the ground state 388 times and identified 83 unique states that belong to the ground state manifold, while out of 1,328,000 samples, the quantum annealer failed to find any state that belongs either to the ground state or the first excited state manifolds, and sampled only 5 states from the second excited state manifold.

In Ref. [19], quantum annealing was benchmarked for ten different problem instances. The instance above was the only one where the ground state was not sampled at all. Yet, in other instances where AR = 100% for both the quantum solver and the annealer, success on the annealer was marginal. For example, in two instances the correct ground state was measured only once out of 1,328,000 samples, corresponding to success likelihood of $\sim 7.5 \times 10^{-7}$ (see instances 0 and 3 in Table I). By contrast for these same instances the quantum solver sampled the solution with likelihood of $\sim 1.3 \times 10^{-4}$ and ~ 0.118. In order to compare conservatively, we ignore any circumstances in which one of the 95 circuits sampled during the optimization process prior to finding the optimal (96 total circuits) returns the correct answer; over all executions the quantum solver is therefore up to over 1,500× more likely to achieve the correct ground state compared to the quantum annealer solution.

To the best of our knowledge these data in aggregate demonstrate for the first time that QAOA executed on a gate-model quantum computer can outperform a quantum annealer for a binary optimization problem.

IV. PERFORMANCE ANALYSIS

The results presented above indicate that it is possible to achieve useful and correct results from quantum optimization executed at full device scale on a gate-model superconducting quantum computer. The ability to consistently solve complex problems at large scales, as we demonstrated, results from the combination of performance enhancements delivered via various parts of the solver pipeline. The exact role and contribution of each subroutine strongly depends on the optimization problem; below we include experimental comparisons in order to provide qualitative insights regarding the role of each component.

Max-Cut

Global		**Q-CTRL (IBM Quantum)** Full hybrid optimization				**Local solver**			**Quantinuum (H2-1)** Classically optimized in simulation		
Instance	Min/Max	Max	AR (%)	Likelihood	Device	Max	AR (%)	Likelihood	Max	AR (%)	Likelihood
(28, 3, 102, u)	0 / 40	**40**	**100**	9.44E-1	S	**40**	**100**	6.20E-1	**40**	**100**	1.78E-1
(30, 3, 264, u)	0 / 43	**43**	**100**	9.18E-1	S	**43**	**100**	4.68E-1	**43**	**100**	1.04E-1
(32, 3, 7, u)	0 / 46	**46**	**100**	8.25E-1	S	**46**	**100**	4.06E-1	**46**	**100**	1.27E-1
(80, 3, 68, u)	0 / 106	**106**	**100**	1.38E-1	B	**106**	**100**	1.00E-4			
(100, 3, 12, u)	0 / 135	**135**	**100**	1.21E-1	B	**135**	**100**	1.00E-4			
(120, 3, 8, u)	0 / 163	**163**	**100**	8.59E-2	B	**163**	**100**	1.67E-4			
(50, 6, 28, w)	0 / 73.2	**73.2**	**100**	8.10E-2	S	**73.2**	**100**	3.67E-2		Not applicable	
(50, 7, 27, w)	0 / 86.25	**86.25**	**100**	7.71E-2	S	**86.25**	**100**	3.67E-2			
(60, 5, 36, w)	0 / 76.25	**76.25**	**100**	1.01E-1	S	**76.25**	**100**	1.57E-2			
(70, 4, 75, w)	0 / 83.75	**83.75**	**100**	6.16E-2	S	**83.75**	**100**	2.20E-3			
(80, 4, 46, w)	0 / 88.0	**88.0**	**100**	2.08E-2	S	**88.0**	**100**	2.90E-3			

Spin Glass

Global		**Q-CTRL (IBM Quantum)** Full hybrid optimization				**Local solver**			**DWave (Pegasus)** Grid search		
Instance	Min/Max	Min	AR (%)	Likelihood	Device	Min	AR (%)	Likelihood	Min	AR (%)	Likelihood
0	-200 / 192	**-200**	**100**	1.34E-4	S	-188	96.9	6.67E-5	**-200**	**100**	7.53E-7
3	-198 / 184	**-198**	**100**	1.18E-1	S	-190	97.9	6.67E-5	**-198**	**100**	7.53E-7
5	-198 / 192	**-198**	**100**	2.03E-1	S	-188	97.4	2.67E-4	-194	99.5	3.77E-6
10	-202 / 190	**-202**	**100**	1.92E-2	B	-194	98.0	6.67E-5			
11	-180 / 196	-178	99.5	1.69E-1	B	-170	97.3	6.67E-5		No data	
69	-190 / 212	-188	99.5	1.75E-1	B	-180	97.5	6.67E-5			

TABLE I. Comparison between different solvers on a variety of (top) Max-Cut problems and (bottom) Spin glass energy minimization problems. Different Max-Cut problems are specified by four parameters $(n, d, s, u/w)$, where n is the number of graph nodes, k, the degree of the regular graph, s, the seed of the random graph and u/w indicating if the graph is unweighted (u) or weighted (w). The seeds for the first three problems define the problems accessed from Ref. [56]. The other Max-Cut problems are defined by the seeds used to generate them with NetworkX's random regular graph generator [57]. The spin-glass problems are specified by the instance numbers as defined in [53] for the *ibm_washington* problems. For a faithful comparison, we follow their QAOA circuit construction approach for these problems [58]. Exact global minimum and maximum values were evaluated using CPLEX. For each solver we show the best objective found (top solution), its approximation ratio (AR), and the observed top solution likelihood estimated as the ratio of the number of times it was sampled and the total number of samples. The Q-CTRL solver data indicates the top solution found out of N_{shots} used for sampling the optimal circuit. For all spin-glass problems $N_{\text{shots}} = 15\text{k}$, where for the Max-Cut problems the number of shots varies between 6–12k depending on the number of nodes (see appendix for more experimental details). The IBM device used for each problem is indicated as B for *ibm_brisbane* and S for *ibm_sherbrooke*. The local solver data indicates the top solution found out of N_{shots} local minima found by a greedy algorithm applied to a random distribution of candidate solutions (see appendix for implementation details). The QA data, taken from [19], indicates the top solution found out of $1,328,000$ samples collected from two Pegasus-class DWave devices. The Quantinuum data, taken from [21], is based on 1024 samples of a pre-optimized QAOA circuit, executed on the H2-1 device, where for each problem the best performing p, out of $p \leq 11$, is shown.

The core differentiation of the quantum solver relative to its classical counterparts is the information extracted during the quantum circuit execution. Therefore, the ability to faithfully execute circuits on the quantum hardware is directly translated to the value the quantum solver provides. We now show the essential role played by *error suppression* in ensuring high-quality outputs from the quantum solver. These build on previous demonstrations that the QAOA landscape could show up to 200× greater structural similarity to the ideal when comparing default circuit implementation to that realized in the Q-CTRL solver [35].

Fig. 4(a-c) shows the distribution of sampled circuit outputs for an example 40-qubit Max-Cut problem in which we isolate the hardware execution by removal of key parts of the pipeline including the modified QAOA ansatz and output post-processing. We compare the Q-CTRL solver's hardware execution to the default execution achieved using Qiskit and to random sampling. These data indicate that default circuit execution without additional error-suppression technology is indistinguishable from random sampling, and thus the quantum nature of the optimization plays no useful role. By contrast, the use of error-suppression leads to a separation of the

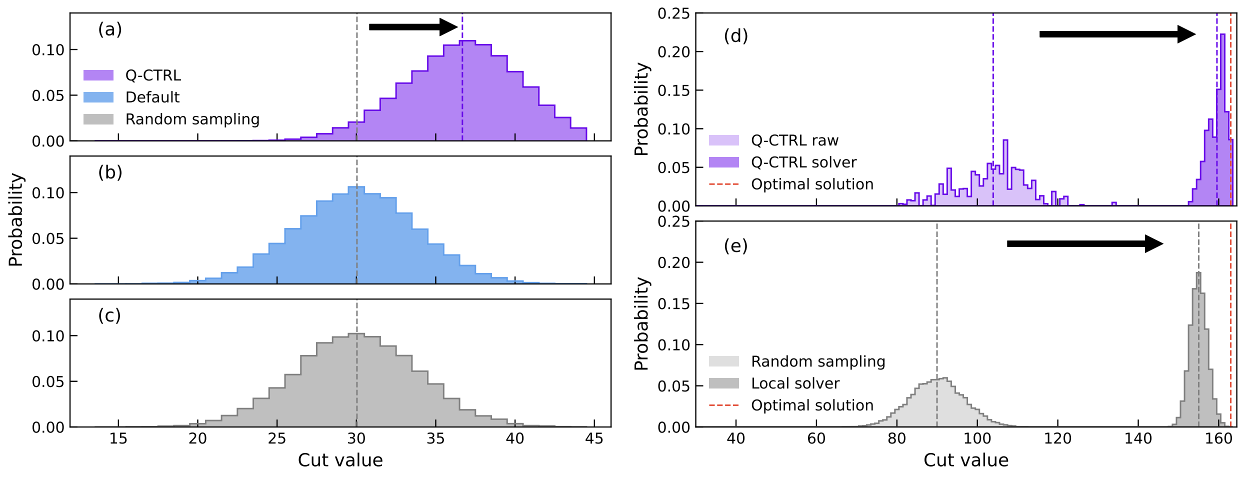

FIG. 4. (a-c) The role of error suppression. The distribution of cut values obtained by sampling 16,384 configurations from a $p = 1$ QAOA circuit for a Max-Cut instance for a randomly generated 3-regular graph with 40 nodes. The circuit was executed on the *ibm_sherbrooke* device utilizing two performance management schemes, (a) using Q-CTRL error suppression pipeline and (b) Qiskit version 0.44, with optimization level 3 and resilience level 1. In all executions, the equal weights superposition state served as an initial state, optimal QAOA parameters were used and the data shown does not include post processing. (c) A reference distribution corresponding to uniformly sampling bitstrings at random. The purple and gray dashed lines indicate the mean of the Q-CTRL and the reference distribution. (d-e) The role of post processing. (d) The distributions of cut values obtained by sampling 12k configurations from the optimal circuit obtained by our pipeline, before and after the application of post-processing, for a Max-Cut instance on a randomly generated 3-regular graph with 120 nodes. (e) For reference, we show the distributions of cut values obtained by 12k random sampling and by applying post-processing on each of these 12k configurations.

distributions towards larger cut values, indicating more useful outputs from sampling the quantum circuit. This behaviour is consistently observed as the scale and complexity of the problem increases.

Next, we compare the quantum solver with and without post-processing to the classical local solver in order to disambiguate benefits coming from the quantum optimization and classical *post-processing*. In Fig. 4(d-e), we show the performance of the quantum and local solvers with and without post-processing for the optimization problem defined by the 120-qubit Max-Cut instance appearing in Fig. 3.

In both the quantum and classical cases, post-processing plays a significant role. The quantum solver exhibits a shift of its distribution to large cuts and achieves the correct Max-Cut solution with $\sim 8.5\%$ likelihood (with post-processing AR $\sim 82 \to 100\%$). The local solver (which includes postprocessing) also finds the correct solution and sees a shift towards higher AR than simple random search, but with $\sim 500\times$ lower likelihood than the quantum solver. This is visible from the fact that only the right-hand tail of the local solver's distribution overlaps the true maximum cut value, whereas the quantum solver's distribution is skewed towards the correct answer. Refs. [60, 61] show that the AR achieved through a greedy classical optimization algorithm alone for Max-Cut is expected to be 0.5. Nonetheless, as a post-processing step, it enables our solver to achieve much faster convergence to high-quality solutions and reduces the sensitivity of the algorithm to uncorrelated random bitflip errors. We also use this greedy algorithm as part of our $\boldsymbol{\theta}$ update strategy.

Finally, we discuss the role of the *enhanced variational ansatz* and associated *parametric compiler* employed in the quantum solver. The ability to individually vary the angles $\boldsymbol{\theta}$ allows the ansatz to encode a broader range of potential solutions and permits a reduction in the number of QAOA layers needed to converge on high-quality solutions. This implementation does not rely on any prior information or use classical solvers to construct the initial state, in contrast to previously published warm-start QAOA methods. Instead, our approach adopts a richer parameterization to facilitate a pathway to the solution while keeping the quantum circuits relatively shallow. Nonetheless, even in the shallow limit, the number of two-qubit gates in each QAOA block increases dramatically with the connectivity and scale of the problem.

The corollary to implementing the enhanced variational ansatz is the need to ensure efficient compilation for large-scale circuits. The *parametric compiler* we have developed effectively reduces both circuit gate counts and duration compared to other commercially available parametric compilers [42]. In the Appendix we compare the quality of compilation, measured in terms of the number of two-qubit entangling gates required for key sample problems. These include $p = 2$ QAOA circuits of n-node Max-Cut problems over random 3-regular graphs and a more generic parametric circuit structure [62, 63]. Such structures are likely to appear for constrained optimization where the circuit ansatz may include non trivial mixing layers or more dense parameterization. The demonstrated compiler efficiency enables the greater performance of the optimization routine described above without incurring any substantive penalty in noise-susceptibility as might be expected by inclusion of the enhanced variational ansatz.

V. CONCLUSION

In this manuscript we have introduced and benchmarked an end-to-end quantum solver for binary optimization problems that *successfully* solves classically nontrivial problems at the scale of $\gtrsim 120$ qubits – the largest reported to date. The methods we introduce consistently found the known optimal solution for Max-Cut and HOBO spin-glass problems; worst-case problem instances delivered approximation ratios $\geq 99.5\%$. Moreover, when compared against the open literature for identical problem instances, these results demonstrate a factor of up to $9\times$ enhancement in the likelihood of finding the correct solution over the best published results using a trapped-ion gate-model machine and over $1{,}500\times$ enhancement vs published results on an annealer. This metric is material in that it translates directly into required runtime and hence user cost. The new quantum solver was able to handle a broad range of problem classes and topologies, and relied exclusively on hardware execution without any classical simulation or precomputation involved.

The results presented in this work demonstrate that an appropriately constructed quantum solver executed on an IBM gate-model quantum computer can outperform published outputs from quantum annealers for nontrivial binary optimization problems. Such a statement warrants careful qualification. First, we do not comment on the potential for different relative performance among gate-model machines should our quantum solver be implemented on alternative hardware. Next, as we relied upon published problem instances and implementations for quantum annealers we do not exclude the possibility of crafting better solutions in detailed implementation. We note that the higher-order binary optimization implemented by the authors of [19] was highly tailored to the problem and carefully constructed to efficiently use the available hardware resources in mapping a HOBO problem to the QUBO problem class efficiently addressed by annealers. Nonetheless it contains no optimality proof and an alternate implementation could deliver superior annealer results. Further, our results do not preclude the existence of alternate problem statements in which annealers retain advantages over gate-model machines, or that individual hardware systems may undergo periodic upgrades [64] to deliver improved performance relative to the reference results used here. Overall, despite these qualifications we believe these results now challenge the heretofore practically correct assertion that annealers would consistently outperform gate-model machines for quantum optimization problems.

These results also suggest the potential for significant advantages in future scaling of problem size and complexity leveraging the gate-model architecture. This is because the use of this architecture directly maintains the richness and flexibility to generalize to a broad class of problems beyond QUBO, as we show in the problems treated here. For the avoidance of any doubt we do not claim quantum advantage over or even practical equivalence with the highly efficient classical CPLEX solver. Nonetheless, the success of the quantum solver we have tested at full device scale provides a potential line of sight to solving much larger future problems which do challenge classical heuristics. Higher complexity problems are likely to lead to a higher gate count in the required circuits, but we believe increasing hardware scale and performance shows a potential path to quantum advantage in quantum optimization. We look forward to continued innovation in this space, including in novel alternate optimization algorithms that could be combined with our solver [65].

Note in preparation

While finalizing the preparation of this manuscript, two recent demonstrations of enhanced quantum optimization using gate-model quantum computers were published. First, [66] suggests signatures of quantum advantage on QAOA problems up to 109 qubits. The problems treated were defined on a simpler linear topology and hardware implementations were not successful finding the correct solutions above 30 nodes. Further, the achieved values $\mathrm{AR} < 80\%$ suggest that classical local solvers may be able to outperform these results (see Table II). Results in [65] present an alternative method to QAOA, and deliver a roughly 30% enhancement over a standard implementation in simulation. Hardware execution of a pure QUBO problem at 100-qubit scale achieved $\mathrm{AR} \sim 65\%$.

Acknowledgments

The authors are grateful to IBM Quantum for the provision of device access as well as access to the CPLEX classical optimization engine. We thank Andreas Bärtschi, John Golden, Stephan Eidenbenz, Elijah Pelofske, and Reuben Tate for useful discussions and the sharing of data from their previous study. We also thank Ruslan Shaydulin for sharing data and IBM device access used during early development of this work. Finally, we are grateful to all other colleagues at Q-CTRL whose technical, product engineering, and design work has supported the results presented in this paper.

[1] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum **2**, 79 (2018).

[2] Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. Van Den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme, *et al.*, Evidence for the utility of quantum computing before fault tolerance, Nature **618**, 500 (2023).

[3] *Combinatorial Optimization and Applications: 11th International Conference, COCOA 2017, Shanghai, China, December 16-18, 2017, Proceedings, Part I* (Springer-Verlag, Berlin, Heidelberg, 2017).

[4] IBM Corporation, IBM ILOG CPLEX Optimization Studio (2024).

[5] Gurobi Optimization, LLC, Gurobi Optimizer Reference Manual (2023).

[6] D. Rehfeldt, T. Koch, and Y. Shinano, Faster exact solution of sparse maxcut and QUBO problems, Mathematical Programming Computation **15**, 445 (2023).

[7] R. Borndörfer, N. Lindner, and S. Roth, A concurrent approach to the periodic event scheduling problem, Journal of Rail Transport Planning & Management **15**, 100175 (2020).

[8] S. Boulebnane and A. Montanaro, Solving boolean satisfiability problems with the quantum approximate optimization algorithm (2022), arXiv:2208.06909 [quant-ph].

[9] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, A unified solution framework for multi-attribute vehicle routing problems, European Journal of Operational Research **234**, 658 (2014).

[10] G. G. Guerreschi and A. Y. Matsuura, QAOA for max-cut requires hundreds of qubits for quantum speed-up, Scientific Reports **9**, 6903 (2019).

[11] R. Shaydulin, C. Li, S. Chakrabarti, M. DeCross, D. Herman, N. Kumar, J. Larson, D. Lykov, P. Minssen, Y. Sun, Y. Alexeev, J. M. Dreiling, J. P. Gaebler, T. M. Gatterman, J. A. Gerber, K. Gilmore, D. Gresh, N. Hewitt, C. V. Horst, S. Hu, J. Johansen, M. Matheny, T. Mengle, M. Mills, S. A. Moses, B. Neyenhuis, P. Siegfried, R. Yalovetzky, and M. Pistoia, Evidence of scaling advantage for the quantum approximate optimization algorithm on a classically intractable problem, Science Advances **10**, eadm6761 (2024), https://www.science.org/doi/pdf/10.1126/sciadv.adm6761.

[12] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, and A. Summer, A review on quantum approximate optimization algorithm and its variants, Physics Reports **1068**, 1 (2024).

[13] S. Yarkoni, E. Raponi, T. Bäck, and S. Schmitt, Quantum annealing for industry applications: introduction and review, Reports on Progress in Physics **85**, 104001 (2022).

[14] Z. Deng, X. Wang, and B. Dong, Quantum computing for future real-time building hvac controls, Applied Energy **334**, 120621 (2023).

[15] Z. Verchère, S. Elloumi, and A. Simonetto, Optimizing variational circuits for higher-order binary optimization (2023), arXiv:2307.16756 [quant-ph].

[16] A. Glos, A. Krawiec, and Z. Zimborás, Space-efficient binary optimization for variational quantum computing, npj Quantum Information **8**, 39 (2022).

[17] P. Niroula, R. Shaydulin, R. Yalovetzky, P. Minssen, D. Herman, S. Hu, and M. Pistoia, Constrained quantum optimization for extractive summarization on a trapped-ion quantum computer, Scientific Reports **12**, 17171 (2022).

[18] F. G. Fuchs, K. O. Lye, H. Møll Nilsen, A. J. Stasik, and G. Sartor, Constraint preserving mixers for the quantum approximate optimization algorithm, Algorithms **15**, 202 (2022).

[19] E. Pelofske, A. Bärtschi, and S. Eidenbenz, Short-depth QAOA circuits and quantum annealing on higher-order ising models, npj Quantum Information **10**, 30 (2024).

[20] D. Herman, R. Shaydulin, Y. Sun, S. Chakrabarti, S. Hu, P. Minssen, A. Rattew, R. Yalovetzky, and M. Pistoia, Constrained optimization via quantum zeno dynamics, Communications Physics **6**, 219 (2023).

[21] R. Shaydulin and M. Pistoia, QAOA with $n \cdot p \geq 200$ (2023), arXiv:2303.02064 [quant-ph].

[22] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, Perspectives of quantum annealing: methods and implementations, Reports on Progress in Physics **83**, 054401 (2020).

[23] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, Adiabatic quantum computation is equivalent to standard quantum computation, SIAM Journal on Computing **37**, 166 (2007).

[24] S. Zbinden, A. Bärtschi, H. Djidjev, and S. Eidenbenz, Embedding algorithms for quantum annealers with chimera and pegasus connection topologies, in *High Performance Computing*, edited by P. Sadayappan, B. L. Chamberlain, G. Juckeland, and H. Ltaief (Springer International Publishing, Cham, 2020) pp. 187–206.

[25] A. Lopez-Bezanilla, A. D. King, C. Nisoli, and A. Saxena, Quantum fluctuations drive nonmonotonic correlations in a qubit lattice, Nature Commun. **15**, 589 (2024).

[26] M. Jünger, E. Lobe, P. Mutzel, G. Reinelt, F. Rendl, G. Rinaldi, and T. Stollenwerk, Quantum annealing versus digital computing: An experimental comparison, ACM J. Exp. Algorithmics **26**, 10.1145/3459606 (2021).

[27] B. Tasseff, T. Albash, Z. Morrell, M. Vuffray, A. Y. Lokhov, S. Misra, and C. Coffrin, On the emerging potential of quantum annealing hardware for combinatorial optimization (2022), arXiv:2210.04291 [math.OC].

[28] E. Pelofske, A. Bärtschi, and S. Eidenbenz, Quantum annealing vs. QAOA: 127 qubit higher-order ising problems on nisq computers, in *International Conference on High Performance Computing* (Springer, 2023) pp. 240–258.

[29] T. Lubinski, C. Coffrin, C. McGeoch, P. Sathe, J. Apanavicius, and D. E. B. Neira, Optimization applications as quantum performance benchmarks (2024), arXiv:2302.02278 [quant-ph].

[30] E. Pelofske, Comparing three generations of d-wave quantum annealers for minor embedded combinatorial optimization problems (2023), arXiv:2301.03009 [quant-ph].

[31] N. Mohseni, T. Morstyn, C. O. Meara, D. Bucher, J. Nüßlein, and G. Cortiana, A competitive showcase of quantum versus classical algorithms in energy coalition formation (2024), arXiv:2405.11917 [quant-ph].

[32] J. Preskill, Q2B 2023, Crossing the Quantum Chasm: From NISQ to Fault Tolerance.

[33] C. W. Commander, Maximum cut problem, max-cutmaximum cut problem, max-cut, in *Encyclopedia of Optimization*, edited by C. A. Floudas and P. M. Pardalos (Springer US, Boston, MA, 2009) pp. 1991–1999.

[34] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, arXiv preprint arXiv:1411.4028 (2014).

[35] P. S. Mundada, A. Barbosa, S. Maity, Y. Wang, T. Merkh, T. Stace, F. Nielson, A. R. Carvalho, M. Hush, M. J. Biercuk, *et al.*, Experimental benchmarking of an automated deterministic error-suppression workflow for quantum algorithms, Physical Review Applied **20**, 024034 (2023).

[36] D. Lykov, J. Wurtz, C. Poole, M. Saffman, T. Noel, and Y. Alexeev, Sampling frequency thresholds for the quantum advantage of the quantum approximate optimization algorithm, npj Quantum Information **9**, 73 (2023).

[37] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, and A. Summer, A review on quantum approximate optimization algorithm and its variants, Physics Reports **1068**, 1 (2024).

[38] D. J. Egger, J. Mareček, and S. Woerner, Warm-starting quantum optimization, Quantum **5**, 479 (2021).

[39] R. Tate, J. Moondra, B. Gard, G. Mohler, and S. Gupta, Warm-started QAOA with custom mixers provably converges and computationally beats goemans-williamson's max-cut at low circuit depths, Quantum **7**, 1121 (2023).

[40] F. Truger, J. Barzen, M. Bechtold, M. Beisel, F. Leymann, A. Mandl, and V. Yussupov, Warm-starting and quantum computing: A systematic mapping study, arXiv preprint arXiv:2303.06133 (2023).

[41] R. Tate, M. Farhadi, C. Herold, G. Mohler, and S. Gupta, Bridging classical and quantum with SDP initialized warm-starts for QAOA, ACM Transactions on Quantum Computing **4**, 1 (2023).

[42] Q-CTRL, Q-CTRL blogs: Avoiding an unexpected roadblock in quantum computing - compilation (2023), publication in preperation.

[43] Q-CTRL, In preparation, a detailed description of a fast, full device, pulse-level calibration procedure of gates beyond the native gate set (2023).

[44] G. S. Hartnett, A. Barbosa, P. S. Mundada, M. Hush, M. J. Biercuk, and Y. Baum, Learning to rank quantum circuits for hardware-optimized performance enhancement, arXiv preprint arXiv:2404.06535 (2024).

[45] Q-CTRL, Q-CTRL Documentation: Algorithmic Benchmarks (2023).

[46] A. Auger and N. Hansen, A restart cma evolution strategy with increasing population size, in *2005 IEEE Congress on Evolutionary Computation*, Vol. 2 (2005) pp. 1769–1776 Vol. 2.

[47] P. K. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli, and S. Woerner, Improving variational quantum optimization using CVaR, Quantum **4**, 256 (2020).

[48] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices, Phys. Rev. X **10**, 021067 (2020).

[49] V. Akshay, D. Rabinovich, E. Campos, and J. Biamonte, Parameter concentrations in quantum approximate optimization, Phys. Rev. A **104**, L010401 (2021).

[50] W. Dam, K. Eldefrawy, N. Genise, and N. Parham, Quantum optimization heuristics with an application to knapsack problems, in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE Computer Society, Los Alamitos, CA, USA, 2021) pp. 160–170.

[51] L. Caha, A. Kliesch, and R. Koenig, Twisted hybrid algorithms for combinatorial optimization, Quantum Science and Technology **7**, 045013 (2022).

[52] S. H. Sack, R. A. Medina, R. Kueng, and M. Serbyn, Recursive greedy initialization of the quantum approximate optimization algorithm with guaranteed improvement, Phys. Rev. A **107**, 062404 (2023).

[53] E. Pelofske, QAOA vs. QA, `https://github.com/lanl/QAOA_vs_QA` (2024).

[54] F. Barahona, M. Jünger, and G. Reinelt, Experiments in quadratic 0–1 programming, Mathematical programming **44**, 127 (1989).

[55] B. R. Jumaa and A. S. Al-Jilawi, Solving max-cut optimization problem, Journal of Physics: Conference Series **1591**, 012051 (2020).

[56] R. Shaydulin and M. Pistoia, *Data from "QAOA with $N\backslash cdotp\backslash geq 200$"* (2023).

[57] A. A. Hagberg, D. A. Schult, and P. J. Swart, Exploring network structure, dynamics, and function using networkx, in *Proceedings of the 7th Python in Science Conference*, edited by G. Varoquaux, T. Vaught, and J. Millman (Pasadena, CA USA, 2008) pp. 11 – 15.

[58] E. Pelofske, A. Bärtschi, L. Cincio, J. Golden, and S. Eidenbenz, Scaling whole-chip QAOA for higher-order ising spin glass models on heavy-hex graphs, arXiv preprint arXiv:2312.00997 (2023).

[59] E. Valiante, M. Hernandez, A. Barzegar, and H. G. Katzgraber, Computational overhead of locality reduction in binary optimization problems, Computer Physics Communications **269**, 108102 (2021).

[60] C. Mathieu and W. Schudy, Yet another algorithm for dense max cut: go greedy. (2008) pp. 176–182.

[61] K. P. Costello, A. Shapira, and P. Tetali, Randomized greedy: new variants of some classic approximation algorithms, in *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11 (Society for Industrial and Applied Mathematics, USA, 2011) p. 647–655.

[62] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature **567**, 209 (2019).

[63] IBM Corporation, IBM documentation.

[64] DWave, DWave systems.

[65] A. G. Cadavid, A. Dalal, A. Simen, E. Solano, and N. N. Hegade, Bias-field digitized counterdiabatic quantum optimization (2024), arXiv:2405.13898 [quant-ph].

[66] J. A. Montanez-Barrera and K. Michielsen, Towards a universal qaoa protocol: Evidence of quantum advantage in solving combinatorial optimization problems (2024), arXiv:2405.09169 [quant-ph].

[67] A. Quantum, Quantinuum provider.

Appendix A: Resources and run time analysis

In previous sections we analyzed the quality of the quantum solver compared to other solvers and previous demonstrations. Run time (cost) is another crucial metric for utility assessment. In this section, we provide details regarding the execution time and cost required for generating the data presented in this work. In Table I, we show results from Quantinuum and Dwave as have appeared in previous publications. Using publicly available information, we approximate the runtime and cost required to produce the results shown.

Quantum solver (IBM) – The optimization scheme we detail in the main text is designed for fast convergence. Our method batches six circuits at each optimization step in order to reduce overhead due to API and data transfer latency. The classical compute time between each two steps includes, for each of the six circuits, the application of measurement error mitigation, analysis of the data for cost extraction, the determination of the next set of circuit parameters, parameter binding and transpilation. At this work, the classical processing is performed sequentially for the six circuits and requires under 10 seconds, which totals to under two minutes throughout the full optimization loop. A straightforward parallelization of this step can effectively render the classical compute time negligible.

The smallest problems we solved (28–32 nodes Max-Cut) implemented 12 optimization steps, which totals to 72 circuits, each executed with 6,000 shots to a total of 432,000 shots throughout the full optimization process. IBM's reported repetition time is $\sim 4,000$ shots per second, resulting in an ideal QPU time of 108 seconds. In practice, the measured QPU time of these jobs is about 150 seconds. Additional API latency, overheads, and queue times on the hardware provider side exists, and typically match the total QPU time (about 150 seconds), leading to a total wall-clock optimization time of about six minutes. Based on published IBM pricing, execution of the full optimization loop for this problem required \$400 USD to run.

The larger problems we consider use 16 optimization steps, totalling 96 circuits, each with $8,000 - 15,000$ shots leading to a total of $768,000 - 1,440,000$ shots. Processes with the maximum number of shots require 7–8 minutes of QPU time and approximately 20 minutes of total wall-clock time. Cost for such executions ranges $\sim \$1,120 - \$1,280$ USD.

Quantum annealers (Dwave) – Based on Refs. [29, 30], the dominant component of QPU time for QA is the anneal time which can be tuned between 0.5–2000 μs. In general, longer anneal times show better stability; in Ref. [30], the best anneal times were identified to be between 1000–2000 μs. On top of the anneal time, readout time of 250 μs and a programming time of 160 μs should be added for each sample.

The QA data in Table I, for three instances of the spin-glass problem, is taken from Ref. [19], where a total of $1,328,000$ samples where used for each problem. Assuming a sample time of 1000–1500 μs corresponds to 22–33 minutes of QA time.

The total optimization time 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. We have no information regarding the total execution time or cost for this system.

Previous demonstration on Ion based QC – Ref. [21], tested QAOA executions on Quantinuum H2-1 device. The results in Table I represent a single circuit execution rather than a full iterative optimization process. We do not have information regarding typical execution times.

We can estimate the cost of running such a circuit using available pricing information in [67]; we are not aware whether this published pricing is indicative of commercial terms available to Quantinuum customers. A 32-node 3-regular graph has 48 edges, hence, assuming arbitrary 2-qubit rotations are considered as basis gates and all to all connectivity, such a $p = 10$ QAOA circuit includes 480 2-qubit gates, 362 single-qubit gates and 64 reset/measurement operations (a requirement of higher-depth QAOA with $p > 1$ as implemented in [21] would require a proportionate growth in circuit resources). With 1,024 shots, such a circuit consumes 1,128 H-System Quantum Credits (HQCs). With a standard subscription to H2 devices [67], one HQC is equivalent to \$13.5 USD, making the total cost of a single circuit execution $> \$15,000$ USD.

We do not know whether full convergence of the optimization process on hardware would be possible as executed or whether if successful the convergence rate would differ from the implementation we describe above. However, assuming convergence equivalent to that achieved in the tests we have performed, full loop optimization over an equivalent 72 circuits would require $\$1,080,000$ USD.

Appendix B: Classical post-processing and local solver

Classical post-processing: We use a naïve greedy optimization for the classical post-processing pass. Considering a bit flip to be a local move, bitstrings for which no local move improves the cost can be considered to be a local minimum of the problem. Greedy optimization corresponds to flipping single bits to improve the cost until a local minimum is reached, and no further improvement is possible. Pseudocode of the greedy optimization procedure is presented in Algorithm 2. There, Shuffle(N) randomly shuffles the bitstring indices $i = 0, ..., n-1$, Flip(x, i) modifies bitstring x by flipping the i-th bit. We limit the number of full traversals of the bitstring to a maximum of five passes.

The motivation for this pass is that the QAOA circuit produces a nontrivial bitstring distribution with support on bitstrings in the basin of good local minima. Device noise and measurement errors can be expected to perturb the bitstrings by local moves, but even a few erroneous bit flips can significantly decrease the solution quality. Greedy optimization corrects for this. The greedy post-processing scales as $\mathcal{O}(n)$ and therefore does not introduce any additional overhead.

Algorithm1 LocalSolver

```
Input: sample size $N$, cost function $C$, passes $K$
Output: sample of local minima
  Initialize $\boldsymbol{x}_i$ as $N$ randomly sampled bitstrings
  for $i$ in $1:N$ do
    $c \leftarrow C(\boldsymbol{x}_i)$
    $\boldsymbol{x}_{\text{best}} \leftarrow \boldsymbol{x}'$
    for $k = 1$ to $K$ do
      $\boldsymbol{x}' \leftarrow \text{GreedyPass}(\boldsymbol{x}_i)$
      if $C(\boldsymbol{x}') < c$ then
        $c \leftarrow C(\boldsymbol{x}')$
        $\boldsymbol{x}_{\text{best}} \leftarrow \boldsymbol{x}'$
      end if
    end for
    $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_{\text{best}}$
  end for
  return: $\{\boldsymbol{x}_i\}_{i=1,\ldots,N}$
```

Algorithm2 GreedyPass

```
Input: length-$n$ bitstring $\boldsymbol{x}$, cost function $C$
Output: local minima bitstring
  while $\boldsymbol{x}$ not local minima of $C$ do
    for $j$ in Shuffle($n$) do
      $\boldsymbol{x}' = \text{Flip}(\boldsymbol{x}, j)$
      if $C(\boldsymbol{x}') < C(\boldsymbol{x})$ then
        $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
      end if
    end for
  end while
  return: $\boldsymbol{x}$
```

Local solver: We implement a purely classical "local solver" that samples bitstrings uniformly at random and applies a greedy post-processing pass to generate local minima of the cost function (see Fig. 4e). Here, a local minimum is defined to be a bitstring whose cost cannot be improved by local moves, in this case single bitflips. Pseudocode for the local solver is presented in Algorithm 1. After first generating a set of random bitstrings, a greedy pass (Algorithm 2) produces a local minimum by greedily flipping any bits that result in an improved cost (in Algorithm 2, Flip$(\boldsymbol{x}, j)$ flips the j-th bit of bitstring $\boldsymbol{x}$). Importantly, the order in which the local moves are considered is randomized (as indicated by the Shuffle(n) term in the pseudocode). The greedy pass terminates when no cost-improving local moves exist. (In practice, the greedy pass terminates after a few bitflips, however, to avoid pathological edge cases, the pass is constrained through early stopping to no more than 5 full traversals of the bitstring.) Since a given bitstring may be connected by cost-reducing local moves to multiple local minima, the local solver applies the greedy pass K times for each bitstring, each time using a different random seed. Since the order in which bits (spins) are traversed affects the outcome, each seed can lead to a different local minimum. The best of the K minima produced is taken to be the post-processed bitstring. Throughout this work we set $K = 5$.

We note that such a local solver may be thought of as a zero-temperature version of Simulated Annealing (SA). In SA, similar sweeps are performed, however non-greedy moves are accepted with some probability that gradually approaches zero towards the end of the annealing process. Typical SA uses 500–5000 sweeps for each initial state (typically 100–1000 such states are considered).

Appendix C: Parametric compilation analysis

The *parametric compiler* we have developed effectively reduces both circuit gate counts and duration compared to other commercially available parametric compilers. Here, we compare the quality of compilation, measured in terms of the number of two-qubit entangling gates required for circuit execution. We treat two extremal problems which bound the expected envelope of circuit structures to be encountered in generic optimization problems.

In Fig. 5(a-b) show $p = 2$ QAOA circuits of n-node Max-Cut problems over 40 different random 3-regular graphs, similar to the problems we show in the main text. We observed a $\sim 10\%$ reduction in the number of 2-qubit gates for each circuit which amounts to a reduction of ~ 300 gates at the maximum tested 120 nodes (qubits).

In Fig. 5(c-d) we consider a more generic parametric circuit structure [62, 63]. In this example, we use circuits with fully entangling layers, resulting in deep circuits. Such structures are likely to appear for constrained optimization where the circuit ansatz may include non trivial mixing layers or more dense parameterization. We observed $> 25\%$ reduction in the number of 2-qubit gates which amounts to a reduction of $\sim 4{,}000$ gates for 80-qubit circuits.

Appendix D: Additional data

In this section we provide additional data about the different problem instance tested. Comprehensive information about each problem execution is presented in Table II.

There we provide for each problem instance the number of optimization steps used (each optimization step contains six circuits), and the number of shots (samples) used for each circuit. Each quantum circuit execution during the optimization process uses the same number of shots. The information regarding best solution found and likelihood always refer to the optimal circuit (minimum cost) only. Additional information on each problem instance includes: the number of times the top solution was sampled, the number of unique optimal configurations found, the mean value of the objective function across all samples and the approximation ratio (AR) of the mean.

In the main text we present the AR with respect to the best solution identified by the solver. Another common convention evaluates the AR using the mean value over all samples. In particular, this convention is informative for small problems where the number of samples – either of the optimal circuit or throughout the full optimization process – is not negligible compared to the number of possible configurations. None of problems we consider have this property; we provide here additional information about the mean values for completeness.

In Figs. 7,6, we present further information for selected problem instance from Table I. Fig. 7 shows six spin-glass problem instances and Fig. 6 six Max-Cut instances.

In both, the left column shows the objective function (ground state energy or maximum cut value) distribution for N (given for each problem in Table II) sampled configurations produced by three different approaches: sampling configurations uniformly at random (Brute force), a local solver that applies the greedy optimization of Algorithm 2 to randomly-sampled configurations (Local solver), and the Q-CTRL solver. The true ground-state energy (red dashed line) is calculated using CPLEX. The right column shows the Cumulative Distribution Function (CDF) of the approximation ratio for the same data as in the adjacent panel. The vertical dashed lines indicate the maximal AR found by each solver. The x-axis value indicates which fraction of the sampled configurations (out of N) has an AR equal or larger than the value on the y-axis.

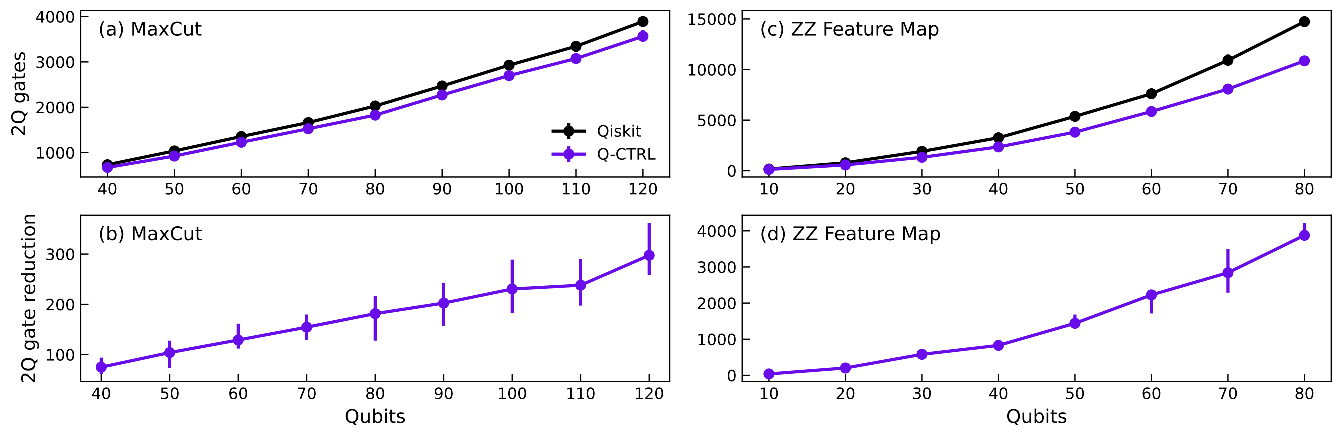

FIG. 5. Compiler benchmarks for sparse Max-Cut and fully entangling ZZ-feature-map problems. (a-b) Compiler-performance measured as the the median number of $2Q$ entangling gates in compiled circuits over 40 random 3-regular graphs for Max-Cut as a function of qubit number. Data compare our parametric compiler to Qiskit level 3 optimization. (a) shows the the total number of gates in the circuits and (b) the difference (gain) in the 2-qubit gate count between the two compilers. (c-d) Similar analysis for the ZZ feature map construction [63] with fully entangled layers. Similar to (a-b) we compare our and Qiskit level compiler showing (c) the total number of 2-qubit gates and (d) the gain.

Max-Cut													
Global			**Q-CTRL (IBM Quantum)** Full hybrid optimization						**Local solver**				
Instance	Min/Max	Shots	Max	Mean	Mean AR	Count	Unique	Total shots	Max	Mean	Mean AR	Count	Unique
(28, 3, 102, u)	0 / 40	6k	40	39.82	0.995	5,661	2	432k	40	38.82	0.971	3,718	4
(30, 3, 264, u)	0 / 43	6k	43	42.75	0.994	5,511	1	432k	43	41.43	0.963	2,810	2
(32, 3, 7, u)	0 / 46	6k	46	45.42	0.987	4,949	1	432k	46	43.93	0.955	2,433	2
(30, 7, 54, u)	0 / 78	6k	78	77.80	0.997	5,294	10	576k	78	77.65	0.995	4,781	10
(80, 3, 68, u)	0 / 106	10k	106	104.23	0.983	1,383	1	1.20M	106	101.79	0.960	1	1
(100, 3, 12, u)	0 / 135	10k	135	132.18	0.979	1,214	2	1.20M	135	128.86	0.955	1	1
(120, 3, 8, u)	0 / 163	12k	163	159.55	0.979	1,031	10	1.15M	163	155.04	0.951	2	2
(50, 6, 28, w)	0 / 73.20	10k	73.20	71.43	0.976	810	4	960k	73.20	71.11	0.971	367	4
(50, 7, 27, w)	0 / 86.25	10k	86.25	84.39	0.978	771	2	960k	86.25	84.04	0.974	367	2
(60, 5, 36, w)	0 / 76.25	10k	76.25	74.01	0.971	1,013	5	960k	76.25	73.22	0.960	157	6
(70, 4, 75, w)	0 / 83.75	10k	83.75	80.58	0.962	616	6	960k	83.75	78.96	0.943	22	11
(80, 4, 46, w)	0 / 88.00	10k	88.00	84.01	0.955	208	12	960k	88.00	83.23	0.946	29	12

Spin glass													
Global			**Q-CTRL (IBM Quantum)** Full hybrid optimization						**Local solver**				
Instance	Min/Max	Shots	Min	Mean	Mean AR	Count	Unique	Total shots	Min	Mean	Mean AR	Count	Unique
0	-200 / 192	15k	-200	-188.43	0.970	2	1	1.44M	-188	-149.98	0.872	1	1
3	-198 / 184	15k	-198	-190.37	0.980	1,768	33	1.44M	-190	-152.63	0.881	1	1
5	-198 / 192	15k	-198	-193.15	0.988	3,045	129	1.44M	-188	-151.75	0.881	4	4
10	-202 / 190	15k	-202	-193.42	0.978	288	7	1.44M	-194	-146.14	0.858	1	1
11	-180 / 196	15k	-178	-173.25	0.982	2,530	86	1.44M	-170	-142.98	0.902	1	1
69	-190 / 212	15k	-188	-182.97	0.983	2,630	27	1.80M	-180	-148.52	0.897	1	1

TABLE II. Additional information to Table I in the main text regarding the Q-CTRL solver and the Local solver. The set of problems we show here is identical. Both the ideal Min/Max values and the ones found by each solver are shown in the main text and are presented here for completeness. *Shots* represents the number times the optimal circuit, found by the quantum solver, was sampled and the number of initial configurations used by the local solver, *Mean* represents the averaged objective value (max cut or min energy) across all samples (shots), *Mean AR* is the approximation ratio of the mean value, *Count* is the number of times the top solution, found by each solver, was sampled, *Unique* is the number of unique optimal configurations each solver found and *Total shots* is the total amount of shots used by the quantum solver throughout the full optimization process.

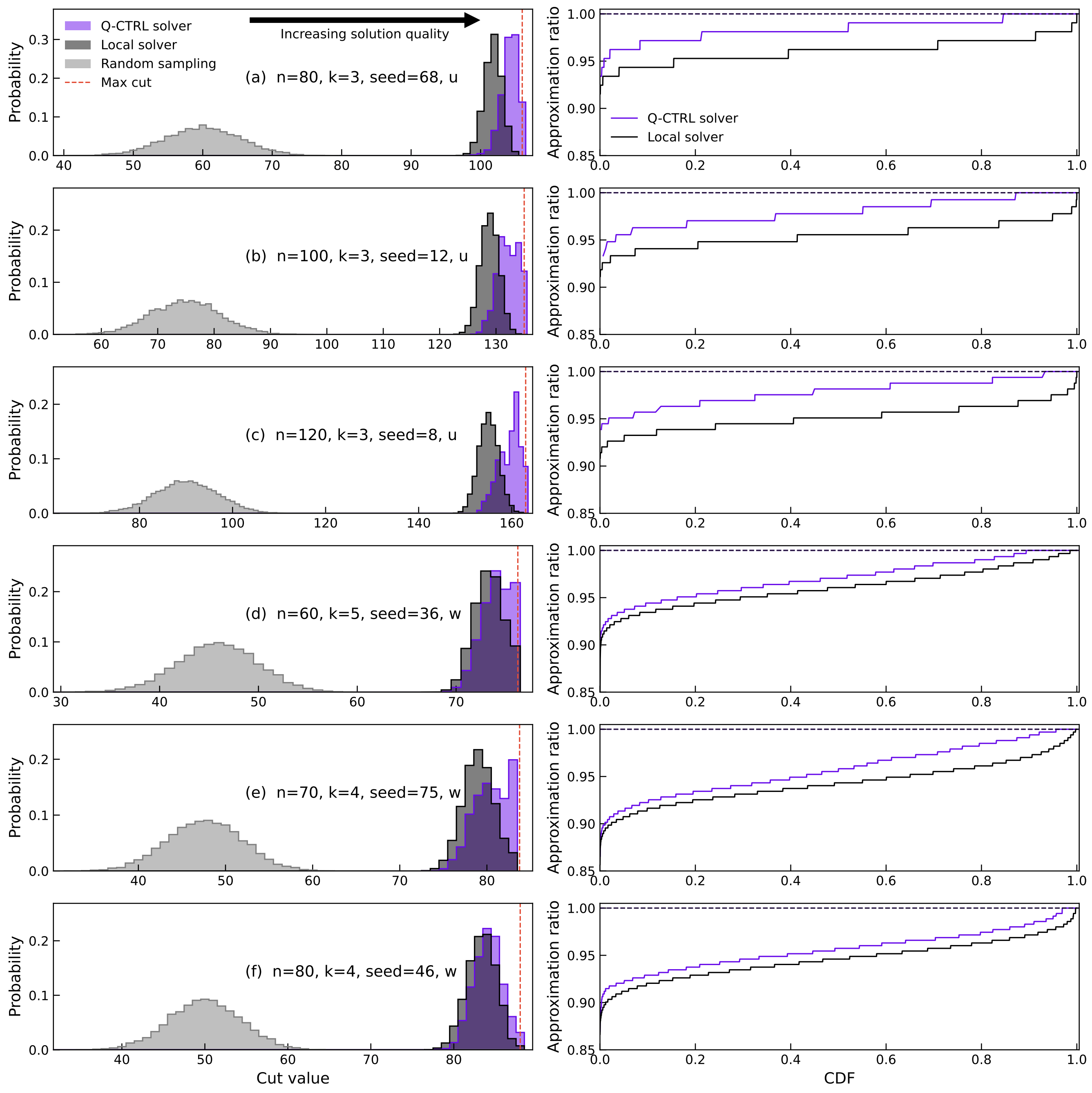

FIG. 6. Distribution of optimized-circuit outcomes for various Max-Cut problems. (Left column) The cut value distribution for $N = 6k - 12k$ sampled configurations (see the 'Shots' column in Table II) produced by three different approaches: sampling configurations uniformly at random (Brute force), a local solver that applies the greedy optimization of Algorithm 2 to randomly-sampled configurations (Local solver), and the Q-CTRL solver. The true maximum cut value (red dashed line) is calculated using CPLEX. (Right column) The Cumulative Distribution Function (CDF) of the approximation ratio for the same data as in the adjacent panel. The vertical dashed lines indicate the maximal AR found by each solver (the correct solution corresponds to $AR = 1$).

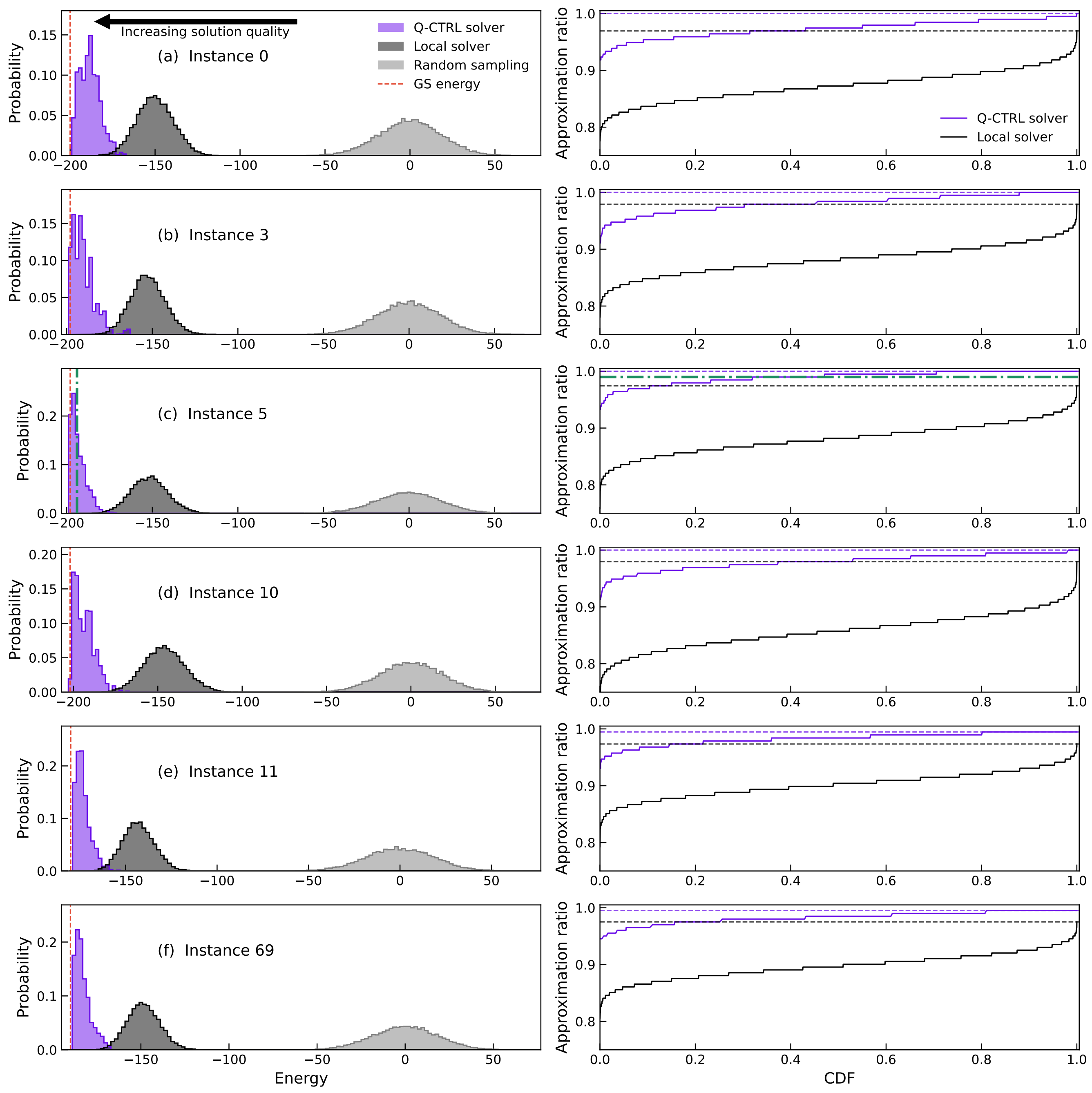

FIG. 7. Distribution of optimized-circuit outcomes for various spin-glass problem instances. (Left column) The energy distribution for 15,000 sampled configurations produced by three different approaches: sampling configurations uniformly at random (Brute force), a local solver that applies the greedy optimization of Algorithm 2 to randomly-sampled configurations (Local solver), and the Q-CTRL solver. The true ground-state energy (red dashed line) is calculated using CPLEX. (Right column) The Cumulative Distribution Function (CDF) of the approximation ratio (AR) for the same data as in the adjacent panel. The vertical dashed lines indicate the maximal AR found by each solver (the correct solution corresponds to $AR = 1$).