nature physics

ARTICLES

https://doi.org/10.1038/s41567-020-01105-y

Quantum approximate optimization of non-planar graph problems on a planar superconducting processor

Matthew P. Harrigan[1]✉, Kevin J. Sung[1,2], Matthew Neeley[1], Kevin J. Satzinger[1], Frank Arute[1], Kunal Arya[1], Juan Atalaya[1], Joseph C. Bardin[1,3], Rami Barends[1], Sergio Boixo[1], Michael Broughton[1], Bob B. Buckley[1], David A. Buell[1], Brian Burkett[1], Nicholas Bushnell[1], Yu Chen[1], Zijun Chen[1], Ben Chiaro[1,4], Roberto Collins[1], William Courtney[1], Sean Demura[1], Andrew Dunsworth[1], Daniel Eppens[1], Austin Fowler[1], Brooks Foxen[1], Craig Gidney[1], Marissa Giustina[1], Rob Graff[1], Steve Habegger[1], Alan Ho[1], Sabrina Hong[1], Trent Huang[1], L. B. Ioffe[1], Sergei V. Isakov[1], Evan Jeffrey[1], Zhang Jiang[1], Cody Jones[1], Dvir Kafri[1], Kostyantyn Kechedzhi[1], Julian Kelly[1], Seon Kim[1], Paul V. Klimov[1], Alexander N. Korotkov[1,5], Fedor Kostritsa[1], David Landhuis[1], Pavel Laptev[1], Mike Lindmark[1], Martin Leib[6], Orion Martin[1], John M. Martinis[1,4], Jarrod R. McClean[1], Matt McEwen[1,4], Anthony Megrant[1], Xiao Mi[1], Masoud Mohseni[1], Wojciech Mruczkiewicz[1], Josh Mutus[1], Ofer Naaman[1], Charles Neill[1], Florian Neukart[6], Murphy Yuezhen Niu[1], Thomas E. O'Brien[1], Bryan O'Gorman[7,8], Eric Ostby[1], Andre Petukhov[1], Harald Putterman[1], Chris Quintana[1], Pedram Roushan[1], Nicholas C. Rubin[1], Daniel Sank[1], Andrea Skolik[6,9], Vadim Smelyanskiy[1], Doug Strain[1], Michael Streif[6,10], Marco Szalay[1], Amit Vainsencher[1], Theodore White[1], Z. Jamie Yao[1], Ping Yeh[1], Adam Zalcman[1], Leo Zhou[1,11], Hartmut Neven[1], Dave Bacon[1], Erik Lucero[1], Edward Farhi[1] and Ryan Babbush[1]✉

Faster algorithms for combinatorial optimization could prove transformative for diverse areas such as logistics, finance and machine learning. Accordingly, the possibility of quantum enhanced optimization has driven much interest in quantum technologies. Here we demonstrate the application of the Google Sycamore superconducting qubit quantum processor to combinatorial optimization problems with the quantum approximate optimization algorithm (QAOA). Like past QAOA experiments, we study performance for problems defined on the planar connectivity graph native to our hardware; however, we also apply the QAOA to the Sherrington–Kirkpatrick model and MaxCut, non-native problems that require extensive compilation to implement. For hardware-native problems, which are classically efficient to solve on average, we obtain an approximation ratio that is independent of problem size and observe that performance increases with circuit depth. For problems requiring compilation, performance decreases with problem size. Circuits involving several thousand gates still present an advantage over random guessing but not over some efficient classical algorithms. Our results suggest that it will be challenging to scale near-term implementations of the QAOA for problems on non-native graphs. As these graphs are closer to real-world instances, we suggest more emphasis should be placed on such problems when using the QAOA to benchmark quantum processors.

While the prospects for achieving quantum advantage with the quantum approximate optimization algorithm (QAOA) remain unclear, the algorithm prescribes a simple paradigm for optimization that makes it amenable to both analytical study and practical implementation[1–10]. Discrete optimization problems can be expressed as the minimization of a quadratic function of binary variables[11,12], and one can visualize these cost functions as graphs with binary variables as nodes and (weighted) edges connecting bits whose (weighted) products sum to the total cost function value. For most industrially relevant

[1]Google Research, Mountain View, CA, USA. [2]Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA. [3]Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA, USA. [4]Department of Physics, University of California, Santa Barbara, CA, USA. [5]Department of Electrical and Computer Engineering, University of California, Riverside, CA, USA. [6]Volkswagen Data:Lab, Munich, Germany. [7]NASA Ames Research Center, Moffett Field, CA, USA. [8]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA. [9]Leiden University, Leiden, Netherlands. [10]Department of Physics, Friedrich-Alexander University Erlangen-Nürnberg, Erlangen, Germany. [11]Department of Physics, Harvard University, Cambridge, MA, USA. ✉e-mail: mpharrigan@google.com; babbush@google.com

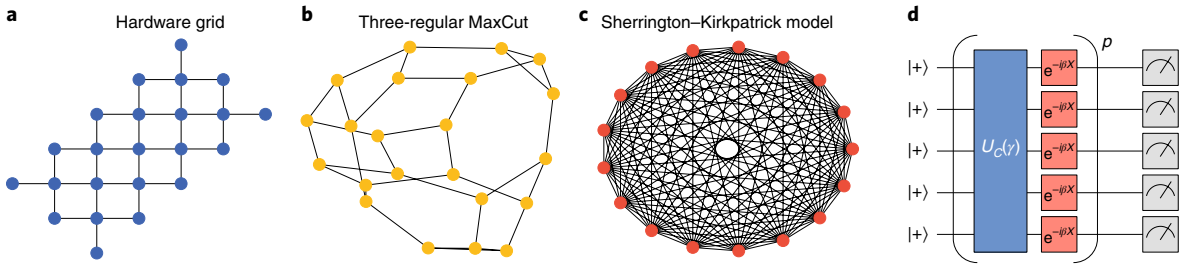

Fig. 1 | Problem families under study. a, Hardware grid problems with a graph matching the hardware connectivity of the 23 qubits used in this experiment. **b**, MaxCut on random three-regular graphs, with the largest instance depicted (22 qubits). **c**, The fully connected SK model shown at the largest size (17 qubits). **d**, QAOA uses p applications of problem and driver unitaries to approximate solutions to optimization problems.

problems, these graphs are non-planar and many ancilla would be required to embed them in (quasi-)planar graphs matching the qubit connectivity of most hardware platforms[13]. The challenge of realizing higher-dimensional problem graphs severely limits the applicability of scalable architectures for quantum annealing[14–16] and corresponds to increased circuit complexity in digital quantum algorithms for optimization such as QAOA.

This work builds on previous experimental demonstrations of QAOA on superconducting qubits[17–20], ion traps[21] and photonics systems[22], with a full comparison found in the Supplementary Information. The Google Sycamore superconducting quantum processor consists of a two-dimensional array of 54 transmon qubits[23] with each qubit tunably coupled to four nearest neighbours in a rectangular lattice. In this study, all device calibration was fully automated[24,25] and data were collected using a cloud interface to the platform programmed using Cirq[26]. Our experiment was restricted to 23 physical qubits of the larger Sycamore device, arranged as depicted in Fig. 1a. We are able to experimentally resolve increased performance with greater QAOA depth and apply the algorithm to cost functions on graphs that deviate markedly from our hardware connectivity. Owing to the low error rates of the Sycamore platform, the trade-off between the theoretical increase in quality of solutions with increasing decoherence due to noise is apparent for hardware-native problems as we scale the depth hyperparameter. We also apply the algorithm to non-native graph problems with their necessary compilation overhead and study the scaling of solution quality and problem size. Our results reveal that the performance of the QAOA is qualitatively different when applied to hardware-native graphs versus more complex graphs, highlighting the challenge of scaling QAOA to problems of industrial importance.

The shallowest depth version of the QAOA consists of the application of two unitary operators: the problem unitary (U_C) and the driver unitary (U_B). The first of these depends on the parameter γ and applies a phase to pairs of bits according to the problem-specific cost operator C:

$$C = \sum_{j<k} w_{jk} Z_j Z_k \tag{1}$$

$$U_C(\gamma) = e^{-i\gamma C} = \prod_{j<k} e^{-i\gamma w_{jk} Z_j Z_k}, \tag{2}$$

where we restrict our study to two-local cost operators with Z_j and Z_k denoting the Pauli Z operator on qubits j and k, respectively, and the w_{jk} corresponding to scalar weights with values $\{0, \pm 1\}$. Because the clauses act on at most two qubits, we are able to associate a graph with a given problem instance with weighted edges given by the w_{jk} adjacency matrix. The second unitary depends on the parameter β, is problem independent, and serves to drive transitions between bitstrings within the superposition state:

$$U_B(\beta) = e^{-i\beta B} = \prod_j e^{-i\beta X_j}, \quad B = \sum_j X_j \tag{3}$$

where X_j is the Pauli X operator on qubit j. Both operators can be implemented by sequentially evolving under each term of the product; specifically the problem unitary is applied with a sequence of two-body interactions while the driver unitary uses single-qubit rotations on each qubit. For higher-depth versions of the algorithm, the two unitaries are sequentially re-applied each with their own β and γ. The number of applications of the pair of unitaries is represented by the hyperparameter p giving parameter vectors $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$. For n qubits, we prepare the parameterized state

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = U_B(\beta_p) U_C(\gamma_p) \cdots U_B(\beta_1) U_C(\gamma_1) |+\rangle^{\otimes n}, \tag{4}$$

where $|+\rangle^{\otimes n}$ is the symmetric superposition of computational basis states. The algorithm is shown graphically in Fig. 1d.

Compilation and problem families

While the driver unitary U_B is straightforward to implement, care must be taken to compile the problem unitary U_C to the constraints of our superconducting processor. We approach compilation as two distinct steps: routing and gate synthesis. The need for routing arises when simulating U_C for a cost function whose graph is not a subgraph of our planar hardware connectivity. To simulate such U_C, we perform layers of swap gates that permute qubits such that all edges in the problem graph correspond to an edge in the hardware graph at least once, at which point the corresponding cost function terms can be implemented.

In this study, we consider three families of binary optimization problems typified by their graph representation. First, we study problem graphs that match the connectivity of our hardware, which we term 'hardware grid problems'. This family of problems is composed of random instances generated by sampling w_{jk} to be ± 1 for edges in the device topology or a subgraph thereof, as depicted in Fig. 1a. While formally NP-hard[12] (and thus, unlikely to be efficiently solvable in the worst case), problems defined on these graphs with couplings chosen in this fashion are known to be classically efficient to exactly solve on average[27]. However, we study these problems here as they are a simple example of a problem that does not require routing.

Next, we study instances of the MaxCut problem on three-regular graphs. This is a prototypical discrete optimization problem with a low, fixed node degree but a high dimension that cannot be trivially mapped to a planar architecture[28]. It more closely matches problems of industrial interest, and an example is shown in Fig. 1b. For MaxCut on degree-three graphs, there is a classical approximation algorithm that achieves an approximation ratio of 0.9326 (ref. [29]), and it is NP-hard to achieve $331/332 + \varepsilon \approx 0.997$ for every $\varepsilon > 0$ (ref. [30]).

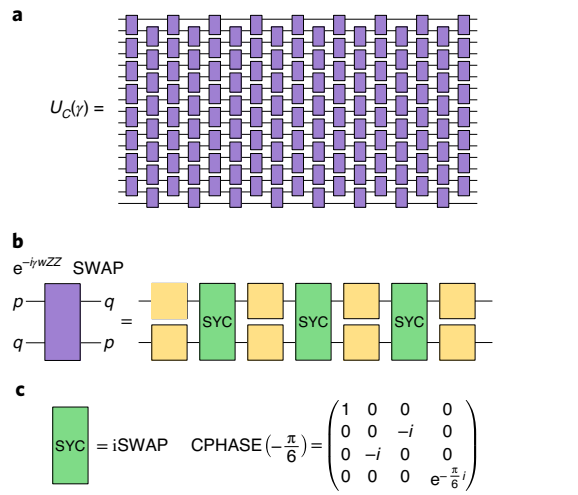

Fig. 2 | Circuits and compilation. a, The linear swap network can route a 17-qubit SK model problem unitary to n layers of nearest-neighbour two-qubit interactions. **b**, The $e^{-i\gamma wZZ}$ · SWAP interaction is a composite phasing and SWAP of example qubits p and q that can be synthesized from three applications of our hardware-native entangling SYC and single-qubit rotations (yellow boxes). **c**, The definition of the SYC gate.

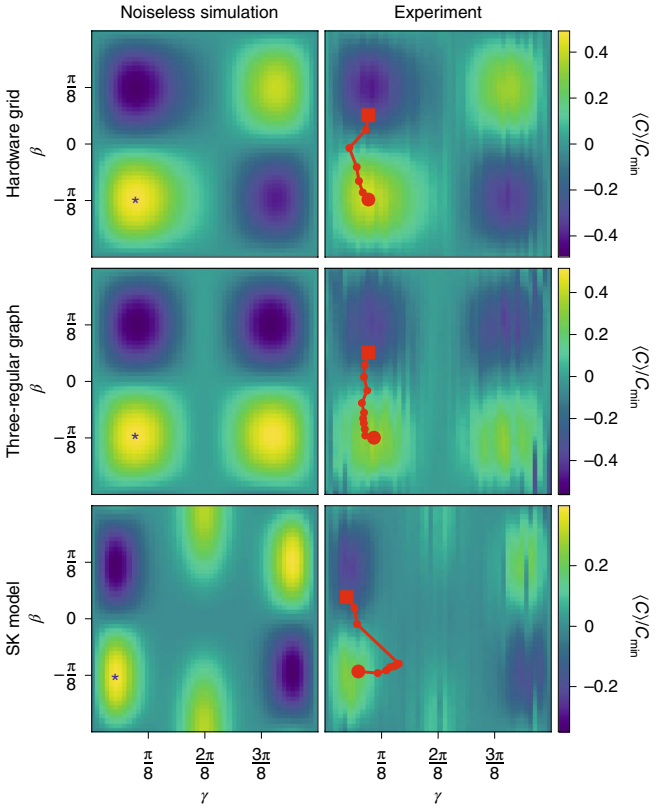

Fig. 3 | Simulated and experimental QAOA landscapes. Comparison of simulated (left) and experimental (right) $p=1$ landscapes shows a clear correspondence of landscape features. An overlaid optimization trace (red, initialized from square marker) demonstrates the ability of a classical optimizer to find optimal parameters. The blue star in each noiseless plot indicates the theoretical local optimum. Problem sizes are $n=23$, $n=14$ and $n=11$ for hardware grid, three-regular MaxCut and SK model, respectively.

For these problems, we use the routing functionality from the t|ket⟩ compiler to heuristically insert SWAP operations[31].

Finally, we study instances of the Sherrington–Kirkpatrick (SK) model[32], defined on the complete graph with w_{jk} randomly chosen to be ±1 as depicted in Fig. 1c. This is a canonical example of a frustrated spin glass (implying that large instances are probably classically inefficient to exactly solve on average[12,32,33]) and is most penalized by routing, which can be performed optimally using the linear swap networks discussed in ref. [34] and Fig. 2 at the cost of a linear increase in circuit depth. Thus, of our three example problems, the two requiring compilation are probably classically inefficient to solve on average whereas the native hardware graph is efficient to solve on average. But we emphasize that at the sizes studied in this paper, all of the problems are exactly solvable by classical algorithms.

The second compilation step is termed gate synthesis and involves decomposing arbitrary one- and two-qubit interactions into physical gates supported by the device (see, for example, Fig. 2b). The physical gates used in this experiment are arbitrary single-qubit rotations and a two-qubit entangling gate native to the Sycamore platform, which we call SYC and define in Fig. 2c. Through multiple applications of this gate with single-qubit rotations, we can synthesize arbitrary entangling gates with full compilation details in the Supplementary Information. The average two-qubit gate fidelities on this device were 99.4% as measured by cross-entropy benchmarking[23] and average readout fidelity was 95.9% per qubit.

Energy landscapes and optimization

QAOA is a variational quantum algorithm where circuit parameters $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ are optimized using a classical optimizer, but function evaluations are executed on a quantum processor[4,35,36]. First, one repeatedly constructs the state $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ with fixed parameters and samples bitstrings to estimate $\langle C\rangle \equiv \langle \boldsymbol{\gamma}, \boldsymbol{\beta}|C|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$. A classical 'outer loop' optimizer can then suggest new parameters to decrease the observed expectation value. Note that we normalize by the cost function's true minimum, so we are in fact maximizing $\langle C\rangle/C_{\min}$ ($C_{\min}$ is negative and hence minimizing $\langle C\rangle$ corresponds to maximizing $\langle C\rangle/C_{\min}$).

For $p=1$, we can visualize the cost function landscape as a function of the parameters $(\boldsymbol{\gamma}, \boldsymbol{\beta})=(\gamma_1, \beta_1)$ in a three-dimensional plot (where we drop the subscript 1 in axes labels). Comparison of simulated and empirical $p=1$ landscapes is a common qualitative diagnostic for the application of QAOA to real hardware[18–22]. For classical optimization to be successful, the quantum computer must provide accurate estimates of $\langle C\rangle$; otherwise, noise can overwhelm any signal and optimizations can fail to make progress. Hardware issues such as decoherence, crosstalk and coherent errors manifest as differences (damping, warping) from the ideal landscape.

Figure 3 contains simulated theoretical and experimental landscapes for selected instances of the three problem families evaluated on a grid of evenly spaced (γ, β) points. Each expectation value was estimated using 50,000 circuit repetitions with efficient post-processing to compensate for readout bias (Supplementary Information). The hardware grid problem shows clear peaks and valleys in the correct locations at the maximum size of our study, $n=23$. For the other two problems, performance degrades with increasing n so we show $n=14$ and $n=11$ for the three-regular graph and SK model, respectively. We highlight the correspondence between experimental and theoretical landscapes for these problems of considerably higher complexity. Previous experimental demonstrations have presented landscapes for a maximum of $n=20$ on a hardware-native interaction graph[21] and a maximum of $n=4$ for a fully connected problem like the SK model[19].

In Fig. 3, we also overlay a trace of the classical optimizer's path through parameter space as a red line. We used a classical optimizer called model gradient descent, which has been shown numerically to perform well with a small number of function evaluations

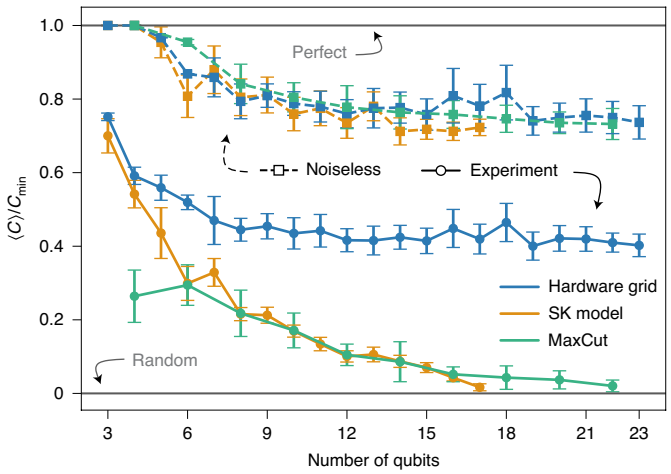

Fig. 4 | QAOA performance as a function of problem size *n*. Each size is the average over ten random instances (s.d. given by error bars). While hardware grid problems show *n*-independent noise, we observe that experimental SK model and MaxCut solutions approach those found by random guessing as *n* is increased. Note that due to the small problem sizes, many efficient classical algorithms could solve these instances exactly, giving a perfect approximation ratio. Therefore, we compare to random guessing as a baseline against a completely noisy quantum algorithm rather than as a point of comparison to classical algorithms.

by using a quadratic surrogate model of the objective function to estimate the gradient[37]. In this example, we initialized the parameter optimization from an intentionally bad parameter setting and observed that the optimizer was able to enter the vicinity of the optimum in ten iterations or fewer, with each iteration consisting of six energy evaluations of 25,000 shots each.

Performance scaling

Before fault tolerance, circuit executions are expected to degrade in fidelity as the number of gates is increased. Here we study the performance of the QAOA on our problems at different size n and depth p using the normalized observed cost function $\langle C\rangle/C_{\min}$ as an application-specific metric of performance. The normalization ensures that a value of 1 is perfect and 0 corresponds to random guessing. To distinguish the effects of noise from the quality of a classical outer-loop optimization, here we report results obtained from running circuits at the theoretically optimal parameter values (that is, starting from the β and γ parameters that were found to be optimal through classical simulation).

In Fig. 4 we observe that $\langle C\rangle/C_{\min}$ for hardware graph problems saturates at a value that is independent of n, which occurs despite the fact that circuit fidelity is decreasing with increasing n. This behaviour can be anticipated by moving to the Heisenberg operator formalism and considering an observable Z_iZ_j. The expectation value for this operator is conjugated by the problem unitary p times giving an expression for the expectation value of Z_iZ_j that only involves qubits that are at most p edges away from i and j. Thus, for fixed p, the error for a given term is asymptotically unaffected as n is increased. Non-local error channels could remove this property.

Compiled problems—namely SK model and three-regular MaxCut problems—result in deeper circuits extensive in the number of qubits. As the depth grows, there is a higher chance of an error occurring. The high degree of the SK model graph and the high effective degree of the MaxCut circuits after compilation means that these errors quickly propagate among all qubits and the quality of solutions can be approximately modelled as the result of a depolarizing channel, with further analysis in the Supplementary Information. Even on these challenging problems, we observe performance exceeding random guessing for problem sizes up to 17 bits, even with circuits of depth $p=3$. Note finally that despite circuits with materially fewer gates (although similar depth), performance on the MaxCut instances tracks performance on the SK model instances rather closely, further substantiating the circuit depth as a useful proxy for the performance of the QAOA.

In noiseless simulation, the quality of solutions is improved by increasing the depth hyperparameter p. However, the additional depth increases the probability of error on real hardware. We study this interplay in Fig. 5. Previously, improved performance with $p>1$ had only been experimentally demonstrated for an $n=2$ problem[20]. For larger problems ($n=20$), performance for $p=2$ was shown to be within error bars of the $p=1$ performance[21]. Figure 5 shows the p dependence averaged across all 130 instances where $n>10$. The mean finds its maximum at $p=3$, although the relatively flat dependence of performance on depth suggests that the experimental noise seems to nearly balance the increase in theoretical performance for this problem family. When we consider each instance individually and identify which value of p maximizes performance for that particular instance, performance is maximized at $p=3$ for over half of instances with a histogram of these per-instance maximal values is inset in Fig. 5. Our full dataset (available in the Supplementary Information) includes per-instance data at all settings of p.

Conclusion

Discrete optimization is an enticing application for near-term devices owing to both the potential value of solutions and the viability of heuristic low-depth algorithms such as the QAOA. While no existing quantum processors can outperform classical optimization heuristics, the application of popular methods such as the QAOA to prototypical problems can be used as a benchmark for comparing various hardware platforms.

Previous demonstrations of the QAOA have primarily optimized problems tailored to the hardware architecture at minimal depth. Using the Google 'Sycamore' platform, we explored these types of problem, which we termed hardware grid problems, and demonstrated robust performance at large numbers of qubits. We showed that the locations of maxima and minima in the $p=1$ diagnostic landscape match those from the theoretically computed surface, and that variational optimization can still find the optimum with noisy quantum objective function evaluation. We also applied the QAOA to various problem sizes using pre-computed parameters from noiseless simulation, and observed an n-independent noise effect on the approximation ratios for hardware grid problems. This is consistent with our theoretical understanding that the noise-induced degradation of each term in the objective function remains constant in the shallow-depth regime where correlations remain local. Furthermore, we report clear cases of performance maximization at $p=3$ for the QAOA owing to the low error rate of our hardware.

Most real-world instances of combinatorial optimization problems cannot be mapped to hardware-native topologies without additional resources. Instead, problems must be compiled by routing qubits with swap networks. This additional overhead can have a considerable impact on the algorithm's performance. We studied random instances of the fully connected SK model. Although we report non-negligible performance for large ($n=17$), deep ($p=3$) and complex (fully connected) problems, we see that performance degrades with problem size for such instances.

The promise of quantum enhanced optimization will continue to motivate the development of new quantum technology and algorithms. Nevertheless, for quantum optimization to compete with classical methods for real-world problems, it is necessary to push beyond contrived problems at low circuit depth. Our work demonstrates important progress in the implementation and performance

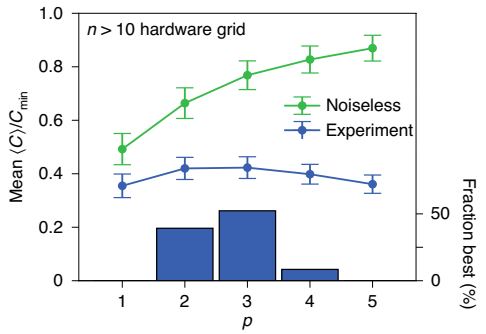

Fig. 5 | QAOA performance as a function of depth *p*. In an ideal simulation, increasing p increases the quality of solutions. For experimental hardware grid results, we observe increased performance for $p > 1$ both as measured by the mean over all instances (lines with s.d. error bars) and statistics of which p maximizes performance on a per-instance basis (histogram). At larger p, errors overwhelm the theoretical performance increase.

of quantum optimization algorithms on a real device, and underscores the challenges in applying these algorithms beyond those natively realized by hardware interaction graphs.

Online content

Any methods, additional references, Nature Research reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41567-020-01105-y.

Received: 8 April 2020; Accepted: 23 October 2020;
Published online: 04 February 2021

References

1. Farhi, E., Goldstone, J. & Gutmann, S. A quantum approximate optimization algorithm. Preprint at https://arxiv.org/abs/1411.4028 (2014).
2. Farhi, E., Goldstone, J. & Gutmann, S. A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem. Preprint at https://arxiv.org/abs/1412.6062 (2014).
3. Biswas, R. et al. A NASA perspective on quantum computing: opportunities and challenges. *Parallel Comput.* **64**, 81–98 (2017).
4. Wecker, D., Hastings, M. B. & Troyer, M. Training a quantum optimizer. *Phys. Rev. A* **94**, 022309 (2016).
5. Farhi, E. & Harrow, A. W. Quantum supremacy through the quantum approximate optimization algorithm. Preprint at https://arxiv.org/abs/1602.07674 (2016).
6. Jiang, Z., Rieffel, E. G. & Wang, Z. Near-optimal quantum circuit for Grover's unstructured search using a transverse field. *Phys. Rev. A* **95**, 062317 (2017).
7. Wang, Z., Hadfield, S., Jiang, Z. & Rieffel, E. G. Quantum approximate optimization algorithm for MaxCut: a fermionic view. *Phys. Rev. A* **97**, 022304 (2018).
8. Hadfield, S. et al. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *Algorithms* **12**, 34 (2017).
9. Lloyd, S. Quantum approximate optimization is computationally universal. Preprint at https://arxiv.org/abs/1812.11075 (2018).
10. Farhi, E., Goldstone, J., Gutmann, S. & Zhou, L. The quantum approximate optimization algorithm and the Sherrington–Kirkpatrick model at infinite size. Preprint at https://arxiv.org/abs/1910.08187 (2019).
11. Lucas, A. Ising formulations of many NP problems. *Front. Phys.* **2**, 5 (2014).
12. Barahona, F. On the computational complexity of Ising spin glass models. *J. Phys. A* **15**, 3241–3253 (1982).
13. Choi, V. Minor-embedding in adiabatic quantum computation: I. The parameter setting problem. *Quantum Inf. Process.* **7**, 193–209 (2008).
14. Kadowaki, T. & Nishimori, H. Quantum annealing in the transverse Ising model. *Phys. Rev. E* **58**, 5355–5363 (1998).
15. Farhi, E. et al. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science* **292**, 472–475 (2001).
16. Denchev, V. S. et al. What is the computational value of finite-range tunneling? *Phys. Rev. X* **6**, 031015 (2016).
17. Otterbach, J. S. et al. Unsupervised machine learning on a hybrid quantum computer. Preprint at https://arxiv.org/abs/1712.05771 (2017).
18. Willsch, M., Willsch, D., Jin, F., De Raedt, H. & Michielsen, K. Benchmarking the quantum approximate optimization algorithm. *Quantum Inf. Process.* **19**, 197 (2020).
19. Abrams, D. M., Didier, N., Johnson, B. R., da Silva, M. P. & Ryan, C. A. Implementation of XY entangling gates with a single calibrated pulse. *Nat. Electron.* **3**, 744–750 (2020).
20. Bengtsson, A. et al. Improved success probability with greater circuit depth for the quantum approximate optimization algorithm. *Phys. Rev. Appl.* **14**, 034010 (2020).
21. Pagano, G. et al. Quantum approximate optimization of the long-range Ising model with a trapped-ion quantum simulator. *Proc. Natl Acad. Sci. USA* **117**, 25396–25401 (2020).
22. Qiang, X. et al. Large-scale silicon quantum photonics implementing arbitrary two-qubit processing. *Nat. Photon.* **12**, 534–539 (2018).
23. Arute, F. et al. Quantum supremacy using a programmable superconducting processor. *Nature* **574**, 505–510 (2019).
24. Kelly, J., O'Malley, P., Neeley, M., Neven, H. & Martinis, J. M. Physical qubit calibration on a directed acyclic graph. Preprint at https://arxiv.org/abs/1803.03226 (2018).
25. Klimov, P. V., Kelly, J., Martinis, J. M. & Neven, H. The snake optimizer for learning quantum processor control parameters. Preprint at https://arxiv.org/abs/2006.04594 (2020).
26. The Cirq Developers Cirq: a python framework for creating, editing, and invoking noisy intermediate scale quantum (NISQ) circuits. *GitHub* https://github.com/quantumlib/Cirq (2020).
27. Rønnow, T. F. et al. Defining and detecting quantum speedup. *Science* **345**, 420–424 (2014).
28. Goemans, M. X. & Williamson, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.* **42**, 1115–1145 (1995).
29. Halperin, E., Livnat, D. & Zwick, U. Max Cut in cubic graphs. In *Proc. Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '02* 506–513 (Society for Industrial and Applied Mathematics, 2002).
30. Berman, P. & Karpinski, M. in *Automata, Languages and Programming* (eds Wiedermann, J. et al.) 200–209 (Springer, 1999).
31. Cowtan, A. et al. On the qubit routing problem. In *14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019), Vol. 135 of Leibniz International Proc. Informatics (LIPIcs)* (eds van Dam, W. & Mancinska, L.) 5:1–5:32 (Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019); http://drops.dagstuhl.de/opus/volltexte/2019/10397
32. Sherrington, D. & Kirkpatrick, S. Solvable model of a spin-glass. *Phys. Rev. Lett.* **35**, 1792–1796 (1975).
33. Montanari, A. Optimization of the Sherrington–Kirkpatrick Hamiltonian. Preprint at https://arxiv.org/abs/1812.10897 (2018).
34. Hirata, Y., Nakanishi, M., Yamashita, S. & Nakashima, Y. An efficient conversion of quantum circuits to a linear nearest neighbor architecture. *Quantum Inf. Comput.* **11**, 142–166 (2011).
35. Yang, Z.-C., Rahmani, A., Shabani, A., Neven, H. & Chamon, C. Optimizing variational quantum algorithms using Pontryagin's minimum principle. *Phys. Rev. X* **7**, 021027 (2017).
36. McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R. & Neven, H. Barren plateaus in quantum neural network training landscapes. *Nat. Commun.* **9**, 4812 (2018).
37. Sung, K. J. et al. Using models to improve optimizers for variational quantum algorithms. *Quantum Sci. Technol.* https://doi.org/10.1088/2058-9565/abb6d9 (2020).
38. Google AI Quantum and collaborators Recirq. *Zenodo* https://doi.org/10.5281/zenodo.3992332 (2020).
39. Google AI Quantum and collaborators. Sycamore QAOA experimental data. *figshare* https://figshare.com/articles/dataset/Sycamore_QAOA_experimental_data/12597590 (2020).

Data availability

Source data is available for this paper. The experimental data for this experiment is available from the Figshare repository[39].

Code availability

The code used in this experiment is available[38] with additional resources at https://github.com/quantumlib/ReCirq.

Acknowledgements

We thank the Cambridge Quantum Computing team for helpful correspondence about their t|ket⟩ compiler, which we used for routing of MaxCut problems. The VW team acknowledges support from the European Union's Horizon 2020 research and innovation programme under grant agreement number 828826 'Quromorphic'. We thank all other members of the Google Quantum team, as well as our executive sponsors. D.B. is a CIFAR Associate Fellow in the Quantum Information Science Program.

Author contributions

R. Babbush and E.F. designed the experiment. M.P.H. and K.J.S. led code development and data collection with assistance from non-Google collaborators. Z.J. and N.C.R. derived the gate synthesis used in compilation. The manuscript was written by M.P.H., R. Babbush, E.F. and K.J.S. Experiments were performed using cloud access to a quantum processor that was recently developed and fabricated by a large effort involving the entire Google Quantum AI team.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41567-020-01105-y.

Correspondence and requests for materials should be addressed to M.P.H. or R.B.

Peer review information *Nature Physics* thanks the anonymous reviewers for their contribution to the peer review of this work.

Reprints and permissions information is available at www.nature.com/reprints.

nature research

https://doi.org/10.1038/s41567-020-01105-y

Supplementary information

Quantum approximate optimization of non-planar graph problems on a planar superconducting processor

In the format provided by the authors and unedited

Supplementary Information for *Quantum Approximate Optimization of Non-Planar Graph Problems on a Planar Superconducting Processor*

Google AI Quantum Collaboration

(Dated: September 16, 2020)

S1. HARDWARE AND COMPILATION DETAILS

In this section, we discuss detailed compilation of the desired unitaries into the hardware native gateset, particularly the SYC gate defined in Figure 2c. The SYC gate is similar to the gate used in Arute *et al.* [1] but with the conditional phase tuned to be precisely $\pi/6$. A $\sqrt{\text{iSWAP}}$ gate is simultaneously calibrated and available but has a longer gate duration and requires additional (physical) Z rotations to match phases. The required interactions for this study are compiled to an equivalent number of SYC and $\sqrt{\text{iSWAP}}$, so SYC was used in all circuits. Single-qubit microwave pulses enact "Phased X" gates $\text{PhX}(\theta, \phi)$ (alternatively called XY rotations or the W gate) with $\phi = 0$ corresponding to $R_X(\theta)$ and $\phi = \frac{\pi}{2}$ corresponding to $R_Y(\theta)$ (up to global phase). Intermediate values of ϕ control the axis of rotation in the X-Y plane of the Bloch sphere.

Arbitrary single-qubit rotations can be applied by a $\text{PhX}(\theta, \phi)$ gate followed by a $R_Z(\vartheta)$ gate. As a compilation step, we merge adjacent single-qubit operations to be of this form. Therefore, our circuit is structured as a repeating sequence of: a layer of PhX gates; a layer of Z gates; and a layer of SYC gates. All Z rotations of the form $\exp[-i\theta Z]$ can be efficiently commuted through SYC and PhX to the end of the circuit and discarded. This leaves alternating layers of PhX and SYC gates. The overheads of compilation are summarized in Table S1.

Problem	Routing	Interaction	Synthesis
Hardware Grid	WESN	$e^{-i\gamma ZZ}$	2
MaxCut	Greedy	$e^{-i\gamma ZZ}$	2
MaxCut	Greedy	SWAP	3
SK Model	Swap Network	$e^{-i\gamma ZZ} \cdot$ SWAP	3

TABLE S1. Compilation details for the problems studied. "Routing" gives the strategy used for routing, "Interaction" gives the type of two-qubit gates which need to be compiled, and "Synthesis" gives the number of hardware native 2-qubit SYC gates required to realize the target interaction. "WESN" routing refers to planar activation of West, East, etc. links.

Compilation of $ZZ(\gamma)$. These interactions (used for Hardware Grid and MaxCut problems) can be compiled with 2 layers of SYC gates and 2+1 associated layers of single qubit PhX gates. We report the required number of single-qubit layers as 2+1 because the initial (or final) layer from one set of interactions can be merged into the final (initial) single qubit gate layer of the preceding (following) set of interactions. In general, the number of single qubit layers will be equivalent to the number of two-qubit gate layers with one additional single-qubit layer at the beginning of the circuit and one additional single-qubit layer at the end of the circuit. The explicit compilation of ZZ to SYC is available in Cirq and a proof can be found in the supplemental material of Ref. [1]. Here we reproduce the derivation in slightly different notation but following a similar motivation.

The SYC gate is an $\text{fSim}(\pi/2,\pi/6)$ which can be broken down into a CPHASE($\pi/6$), CZ, SWAP, and two S gates according to Figure S1. We analyze the KAK coefficients for a composite gate of two SYC gates sandwiching arbitrary

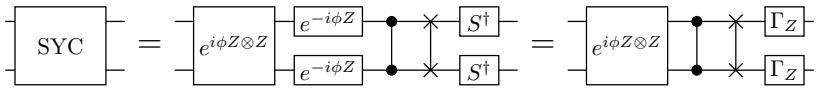

FIG. S1. Circuit decomposition of the SYC gate: $\Gamma_Z = S^\dagger e^{-i\phi Z} = e^{-i\phi Z} S^\dagger$ and $\phi = -\pi/24$, where two solid dots linked by a line represent the CZ gate and two crosses linked by a line represent the SWAP gate.

single qubit rotations, depicted in Figure S2, to determine the space of gates accessible with two SYC gates.

Any two qubit gate is locally equivalent to standard KAK form [2]. The coefficients in the KAK form is equivalent to the operator Schmidt coefficients of the 2-qubit unitary. To find the Schmidt coefficients, we introduce the matrix representation of 2-qubit gates in terms of Pauli operators, i.e., the jk-th matrix element equals to the corresponding

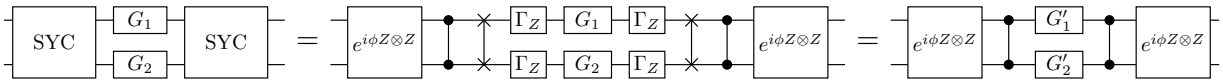

FIG. S2. Single-qubit gates sandwiched by two SYC gates: The Γ_Z gates map single-qubit operations to single-qubit operations

coefficient of the Pauli operator $P_j \otimes P_k$, where $P_{0,1,2,3} = I, X, Y, Z$,

$$O_M = \sum_{j,k=0}^{3} M_{jk} P_j \otimes P_k \,. \tag{1}$$

The Schmidt coefficients of O_M equal to the singular values of M. Any single-qubit gate $G'_{1,2}$ can be decomposed into the Z-X-Z rotations; the Z rotations commute with the CZ and the CPHASE, and they do not affect the Schmidt coefficients of the two-qubit operation defined in Figure S2. We neglect the Z rotations and simplify $G'_{1,2}$ to single-qubit X rotations

$$G'_1 = \cos\theta_1 I + i \sin\theta_1 X \,, \qquad G'_2 = \cos\theta_2 I + i \sin\theta_2 X \,. \tag{2}$$

The Pauli matrix representation of $G'_1 \otimes G'_2$ in Eq. (2) is

$$A = \begin{pmatrix} c_1 c_2 & i c_1 s_2 & 0 & 0 \\ i s_1 c_2 & -s_1 s_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} , \tag{3}$$

where $c_{1,2} = \cos\theta_{1,2}$ and $s_{1,2} = \sin\theta_{1,2}$. The rank of the matrix A is one, representing a product unitary. After being conjugated by the CZ gates, i.e, $O \mapsto \mathrm{CZ}\, O\, \mathrm{CZ}$, the matrix A becomes

$$A \mapsto B = \begin{pmatrix} c_1 c_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & i s_1 c_2 \\ 0 & 0 & -s_1 s_2 & 0 \\ 0 & i c_1 s_2 & 0 & 0 \end{pmatrix} , \tag{4}$$

where we use the relations for $O \mapsto \mathrm{CZ}\, O\, \mathrm{CZ}$,

$$X_1 X_2 \mapsto Y_1 Y_2 \,, \quad X_1 \mapsto X_1 Z_2 \,, \quad X_2 \mapsto Z_1 X_2 \,. \tag{5}$$

The CPHASE part in the SYC gate is

$$e^{i\phi Z\otimes Z} = \cos\phi\, I \otimes I + i \sin\phi\, Z \otimes Z \,, \tag{6}$$

where $\phi = -\pi/24$. An arbitrary operator O left and right multiplied by CPHASE part is expressed as

$$e^{i\phi Z\otimes Z} O e^{i\phi Z\otimes Z} = (\cos\phi)^2 O + \frac{i}{2} \sin(2\phi) \left(Z^{\otimes 2} O + O Z^{\otimes 2} \right) - (\sin\phi)^2 Z^{\otimes 2} O Z^{\otimes 2} \,. \tag{7}$$

Applying the operation $O \mapsto \frac{1}{2}(Z^{\otimes 2} O + O Z^{\otimes 2})$ to the operator B, we have

$$B \mapsto C = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & s_1 s_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_1 c_2 \end{pmatrix} . \tag{8}$$

Applying the operation $O \mapsto Z^{\otimes 2} O Z^{\otimes 2}$ to the operator B, we have

$$B \mapsto D = \begin{pmatrix} c_1 c_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -i s_1 c_2 \\ 0 & 0 & -s_1 s_2 & 0 \\ 0 & -i c_1 s_2 & 0 & 0 \end{pmatrix} . \tag{9}$$

The resulting two-qubit gate at the output of the circuit in Figure S2 takes the form

$$M = (\cos\phi)^2 B + i\sin(2\phi)\, C - (\sin\phi)^2 D\,. \tag{10}$$

Two singular values of M are $\cos(2\phi)c_1c_2$ and $\cos(2\phi)s_1s_2$ corresponding to the diagonal matrix elements $M_{0,0}$ and $M_{2,2}$, and the magnitudes of these two singular values are bounded by the angle ϕ. Consider the two-dimensional subspace of the matrix B, C, and D with the two known singular values removed

$$B \mapsto B' = \begin{pmatrix} 0 & is_1c_2 \\ ic_1s_2 & 0 \end{pmatrix}, \quad C \mapsto C' = \begin{pmatrix} s_1s_2 & 0 \\ 0 & c_1c_2 \end{pmatrix}, \quad D \mapsto D' = \begin{pmatrix} 0 & -is_1c_2 \\ -ic_1s_2 & 0 \end{pmatrix} \tag{11}$$

The Pauli representation matrix in the reduced space is

$$M' = (\cos\phi)^2 B' + i\sin(2\phi)\, C' - (\sin\phi)^2 D' \tag{12}$$

$$= i\begin{pmatrix} \sin(2\phi)s_1s_2 & s_1c_2 \\ c_1s_2 & \sin(2\phi)c_1c_2 \end{pmatrix} = ic_1c_2\begin{pmatrix} \sin(2\phi)t_1t_2 & t_1 \\ t_2 & \sin(2\phi) \end{pmatrix}. \tag{13}$$

To calculate the singular values of a 2×2 matrix

$$M_{\boldsymbol{\alpha}} = \alpha_0 I + \alpha_1 X + \alpha_2 Y + \alpha_3 Z\,, \tag{14}$$

we used the formula

$$\sigma_\pm = \sqrt{\eta \pm \sqrt{\eta^2 - \xi^2}}\,, \tag{15}$$

where $\eta = |\alpha_0|^2 + |\alpha_1|^2 + |\alpha_2|^2 + |\alpha_3|^2$ and $\xi = |\alpha_0^2 - \alpha_1^2 - \alpha_2^2 - \alpha_3^2|$. For matrix M', we have,

$$\eta = \frac{1}{2}\sum_{j,k} |M'_{jk}|^2 \tag{16}$$

$$= \frac{1}{2}\left(\sin(2\phi)^2 s_1^2 s_2^2 + s_1^2 c_2^2 + c_1^2 s_2^2 + \sin(2\phi)^2 c_1^2 c_2^2\right) \tag{17}$$

$$= \frac{1}{2} - \frac{1}{2}\cos(2\phi)^2\left(s_1^2 s_2^2 + c_1^2 c_2^2\right). \tag{18}$$

and

$$\xi = \frac{1}{4}\left|\sin(2\phi)^2\left(s_1s_2 + c_1c_2\right)^2 - \left(s_1c_2 + c_1s_2\right)^2 + \left(s_1c_2 - c_1s_2\right)^2 - \sin(2\phi)^2\left(s_1s_2 - c_1c_2\right)^2\right| \tag{19}$$

$$= \cos(2\phi)^2\left|s_1s_2c_1c_2\right|. \tag{20}$$

We have solved all the four singular values of the 2-qubit unitary at the output of Figure S1,

$$\lambda_0 = |\cos(2\phi)c_1c_2|,\ \ \lambda_1 = |\cos(2\phi)s_1s_2|,\ \ \lambda_2 = \sqrt{\eta + \sqrt{\eta^2 - \xi^2}},\ \ \lambda_3 = \sqrt{\eta - \sqrt{\eta^2 - \xi^2}}. \tag{21}$$

For the case $s_1 = 0$ and $c_1 = 1$, we have $\lambda_1 = \lambda_3 = 0$ and the other two singular values

$$\lambda_0 = |\cos(2\phi)c_2| \in [0,\ \cos(2\phi)]\,, \quad \lambda_2 = \sqrt{2\eta} = \sqrt{1 - \cos(2\phi)^2 c_2^2}\,. \tag{22}$$

Since $\cos(2\phi) \simeq 0.966 > 1/\sqrt{2}$, we can implement any CPHASE gate using only two SYC gates. This is achieved by matching the Schmidt coefficients of $e^{-i\theta ZZ/2}$ to λ_0 and λ_2. If $|\cos(\theta)| > \cos(2\phi)$ then we can reset $c_{1,2}$ and $s_{1,2}$ appropriately to select out the other pair of singular values.

Compilation of swap. A SWAP gate requires three applications of SYC and is used for the 3-regular MaxCut problem circuits. The SWAP gate was numerically compiled by optimizing the angles of the circuit in Figure S3 to match the KAK interaction coefficients for the SWAP gate.

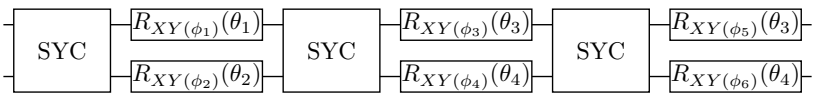

FIG. S3. Circuit used to match the KAK coefficients of the SWAP gate. The $R_{XY}(\phi)(\theta)$ is a rotation of θ around an axis in the XY-plane defined by ϕ. This is implemented in Cirq as a PhasedXPow gate.

After the the angles in the circuit depicted in Figure S3 are determined to match the KAK coefficient of the swap gate we add single qubit rotations to make the circuit fully equivalent to SWAP.

Compilation of $e^{-i\gamma w ZZ} \cdot$ swap. This composite interaction can be effected with three applications of SYC and is used for SK-model circuits. The SYC gate KAK coefficients are $(\pi/4, \pi/4, \pi/24)$ which is locally equivalent to a CPHASE$(\pi/4 - \pi/24)$ followed by a SWAP. Therefore, to implement a $ZZ(\gamma)$ followed by a swap we need to apply a single SYC gate followed by the composite CPHASE$(\gamma - \pi/24 + \pi/4)$. The total composite gate now involves 3 SYC gates, a single Rx gate and two Rz gates.

Scheduling of Hardware Grid gates. An efficient planar graph edge-coloring can be used to schedule as many simultaneous ZZ interactions as possible. We activate links on the graph in the following order: 1) horizontal edges starting from even nodes; 2) horizontal edges starting from odd nodes; 3) vertical edges starting from even nodes; 4) vertical edges starting from odd nodes. Viewed as cardinal directions and choosing an even node as the central point this corresponds to a west, east, south, north (W, E, S, N) activation sequence.

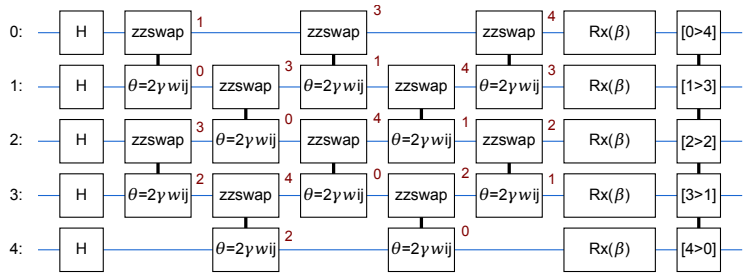

FIG. S4. $p = 1$ swap network for a 5-qubit SK-model. Physical qubits are indicated by horizontal lines and logical node indices are indicated by red numbers. The network effects all-to-all logical interactions with nearest-neighbor interactions in depth n.

Fully Connected Swap Network. All-to-all interactions can be implemented optimally with a swap network in which pairs of linear-nearest-neighbor qubits are repeatedly interacted and swapped. Crucially, the required interactions SWAP and $e^{-i\gamma ZZ}$ between all pairs all mutually commute so we are free to re-order all two-qubit interactions to minimize compiled circuit depth. After n applications of layers of $e^{-i\gamma w ZZ} \cdot$ SWAP interactions (alternating between even and odd qubits), every qubit has been involved in a ZZ interaction with every other qubit and logical qubit indices have been reversed. This can be viewed as a (parallel) bubble sort algorithm initialized with a reverse-sorted list of logical qubit indices. An example at $n = 5$ is shown in Figure S4. If p is even, two applications of the swap network return qubit indices to their original mapping. Otherwise, post-processing can reverse the measured bitstrings.

The swap network requires linear connectivity. On the 23-qubit subgraph of the Sycamore device used for this experiment, this limits us to a maximum size of $n = 17$ for the SK model, shown in Figure S5.

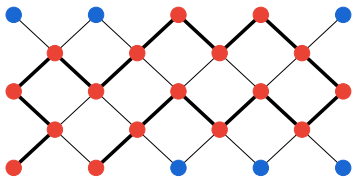

FIG. S5. The largest line one can embed on the 23-qubit device is of length 17.

Distance between compiled problems and hardware grid. There are several ways of quantifying the distance between two graphs. In our case, the only relevant one is how much the performance degrades by embedding a graph into the planar grid representing our hardware connectivity. We found that the depth of the compilation is a good correlate of this, and so optimized for that. Finding the minimum-depth compilation (in the routing framework) is likely NP-hard, though this has not been proven, so the relevant distance metric becomes the lowest depth of a compilation that we can efficiently find. (A few related problems have been shown to be NP-hard, e.g., Maslov *et al.* [3], Botea *et al.* [4].) Spectral sparsification has been used [5] to find an initial assignment of logical to physical qubits. That approach, while generalizable to 2D, is primarily focused on 1D hardware graphs. More importantly, as with the routing problems, these metrics may also provide good heuristics and approximations for the depth, but are not the same.

S2. PRIOR WORK

Reference	Date	Problem topology	$\Delta(G)$	n	p	Optimization
Otterbach *et al.* [6]	2017-12	Hardware	3	19	1	Yes
Qiang *et al.* [7]	2018-08	Hardware	1	2	1	No
Pagano *et al.* [8]	2019-06	Hardware[1] (system 1)	n	12, 20	1	Yes
		Hardware[1] (system 2)	n	20–40	1–2[(2)]	No
Willsch *et al.* [9]	2019-07	Hardware	3	8	1	No
Abrams *et al.* [10]	2019-12	Ring	2	4	1	No
		Fully-connected	n			No
Bengtsson *et al.* [11]	2019-12	Hardware	1	2	1, 2	Yes
This work		Hardware	4	2–23	1–5	Yes
		3-regular	3	4–22	1–3	Yes
		Fully-connected	n	3–17	1–3	Yes

TABLE S2. An overview of experimental demonstrations of QAOA. Although each work generally frames the algorithm in terms of a combinatorial optimization problem (2SAT, Exact Cover, etc.), we classify problems based on their topology, maximum degree of the problem graph $\Delta(G)$, the number of qubits n and the depth of the algorithm p. These attributes give a rough view of the difficulty of a particular instance. We indicate whether variational optimization of parameters was demonstrated. [1]In superconducting processors, "Hardware" topologies are 2-local planar lattices. In ion trap processors, hardware-native topologies are long range couplings of the form $J_{ij} \approx J_0/|i-j|^\alpha$. [2]$p = 2$ only for $n = 20$.

Prior work has included experimental demonstration of the QAOA. The referenced works often include additional results, but we focus specifically on the sections dealing with experimental implementation of the algorithm.

Otterbach *et al.* [6] demonstrated a Bayesian optimization of p = 1 parameters on a 19-bit hardware-native Ising graph using a Rigetti superconducting qubit processor. The authors compared the cumulative probability of finding the lowest energy bitstring over the course of the optimization to binomial coin flips and showed performance from the device exceeding random guessing. The problem topology involved a roughly hexagonal tessellation. The problems were related to a restricted form of two-class clustering.

Qiang *et al.* [7] demonstrated a $n = 2$, $p = 1$ QAOA landscape on their photonic quantum processor. They presented three instances of the two-bit problem, which was framed as Max2Xor. The color scale for the landscapes was re-scaled for experimental values. They demonstrated high probability of obtaining the correct bitstrings.

Pagano *et al.* [8] demonstrated application of the QAOA with two ion trap quantum processors, called "system 1" and "system 2". The problems were of the form $J_{ij} \approx J_0/|i-j|^\alpha$ with α close to unity. This corresponds to an antiferromagnetic 1D chain. This problem is fully connected, but is spiritually similar to the hardware native planar graphs studied in superconducting architectures in the sense that the cost function cannot be programmed and is easily solvable at any system size. A landscape is shown for $n = 20$ from system 1. Optimization traces are shown for $n = 12$ and $n = 20$ on system 1. On system 2, performance was demonstrated at optimal parameters for $n = \{20, 25, 30, 35, 40\}$. Additionally, a partial $p = 2$ grid search was performed on system 2. Nine discrete choices for $(\gamma_1, \beta_1, \beta_2)$ were selected and then a scan over γ_2 was reported for each choice. Finally, on system 2, performance was compared between $p = 1$ and $p = 2$ at $n = 20$, giving a ratio of $(93.8 \pm 0.4)\%$ versus $(93.9 \pm 0.3)\%$, respectively.

Willsch *et al.* [9] demonstrated an application of the QAOA via IBM's Quantum Experience cloud service on the 16Q Melbourne device. The 8-bit problem studied was framed as 2SAT and had a topology matching the device with maximum node degree of 3. A landscape with re-scaled color map was compared to the theoretical landscape.

Abrams *et al.* [10] implemented QAOA on two types of problems; each with two compilation strategies. The 4-bit problems had a ring topology and a fully-connected topology. While a 4-qubit ring would fit on the Rigetti superconducting device, they implemented both problems using only linear connectivity with the introduction of SWAPs. In one compilation strategy, they used CZ as the gate-synthesis target. In the other, they used both CZ and iSWAP. The color bars were re-scaled for the experimental data.

Bengtsson *et al.* [11] ran 2-bit QAOA instances on their superconducting architecture at $p = 1$ and $p = 2$. They show four $p = 1$ landscapes and demonstrate optimization for $n = 2, p = 2$. They observed that increasing circuit depth to $p = 2$ increases the probability of observing the correct bitstring.

S3. READOUT CORRECTION

The experimentally measured expectation values plotted in Figure 3 were adjusted with a procedure used to compensate for qubit readout error. We model readout error as a classical bit-flip error channel that changes the measurement result of qubit i from 0 to 1 with probability $p_{0,i}$ and from 1 to 0 with probability $p_{1,i}$. Under the effect of this error channel, a measurement of a single qubit in the computational basis is described by the following positive operator-valued measure (POVM) elements (we drop the subscript i here for clarity):

$$\tilde{\Pi}_0 = (1 - p_0)\Pi_0 + p_1\Pi_1 \tag{23}$$

$$\tilde{\Pi}_1 = p_0\Pi_0 + (1 - p_1)\Pi_1, \tag{24}$$

where $\Pi_0 = |0\rangle\langle 0|$, $\Pi_1 = |1\rangle\langle 1|$. The uncorrected Z observable can be written as

$$\tilde{Z} = \tilde{\Pi}_0 - \tilde{\Pi}_1 = (p_1 - p_0)I + (1 - p_1 - p_0)Z. \tag{25}$$

Solving for Z, we have

$$Z = \frac{\tilde{Z} - (p_1 - p_0)I}{1 - p_1 - p_0}. \tag{26}$$

For our problems we are interested in the two-qubit observable Z_iZ_j, so the corrected observable is

$$Z_iZ_j = \frac{\tilde{Z}_i - (p_{1,i} - p_{0,i})I}{1 - p_{1,i} - p_{0,i}} \cdot \frac{\tilde{Z}_j - (p_{1,j} - p_{0,j})I}{1 - p_{1,j} - p_{0,j}}. \tag{27}$$

This expression tells us how to adjust the measured observable to compensate for the readout error. In the above analysis, we can replace p_0 and p_1 by their average $(p_0 + p_1)/2$ if we perform measurements in the following way: for half of the measurements, apply a layer of X gates immediately before measuring, and then flip the measurement results. In this case, the corrected observable is

$$Z_iZ_j = \tilde{Z}_i\tilde{Z}_j \cdot \frac{1}{1 - p_{1,i} - p_{0,i}} \cdot \frac{1}{1 - p_{1,j} - p_{0,j}}. \tag{28}$$

We estimated the value of $p_{0,i}$ on the device by preparing and measuring the qubit in the $|0\rangle$ state 1,000,000 times and counting how often a 1 was measured; $p_{1,i}$ was estimated in the same way but by preparing the $|1\rangle$ state instead of the zero state. This estimation was performed periodically during the data collection for Figure 3 to account for drift following automated calibration.

We measure each qubit via the state-dependent dispersive shift they induce on their corresponding harmonic readout resonator as described in Arute *et al.* [1] supplementary information section III. We interrogate the readout resonator frequency with an appropriately calibrated microwave pulse (e.g. a frequency, power, and duration). When demodulated, the readout signal produces a 'cloud' of In-phase and Quadrature (IQ) Voltage points which are used to train an out state descrimator. Often, we find that optimal single-qubit calibrations extend to the case of simultaneous readout, but this is not always the case. For example, due to the Stark shift induced by photons in readout resonators, new frequency collisions may be introduced that are not present in the isolated readout case. Similarly, the combined power of a multiplexed readout pulses may exceed the saturation power of our parametric amplifier.

At the time of the primary data collection for this experiment, all automated calibration routines were performed with each qubit in isolation. Subsequently, a calibration which optimizes qubit detunings during readout was implemented to mitigate these correlated readout errors caused by frequency collisions. Figure S6 shows $|0\rangle$ and $|1\rangle$ state errors for *simultaneous* readout of all 23 qubits (which are used to correct $\langle ZZ\rangle$ observables) both as they were during primary data taking for Figure 3 (top) and after implementing the improved readout detuning calibration (bottom). During primary data collection, the median isolated readout error was 4.4% as measured during the previous automated calibration. The discrepancy between these figures and the calibration values shown in Figure S6, top can be attributed to drift since the automated system calibration in addition to the simultaneity effects described above.

Data presented in Figure 4 and Figure 5 was taken on a different date with median isolated readout error as 4.1% as reported in the main text. Readout correction was not used for these two figures.

While automated calibrations will continue to improve, drift will likely remain an inevitability when controlling qubits with analog signals. As such, we expect the readout corrections employed here will continue to provide utility for end-users of cloud-accessible devices. In general, there will always be a difference between a hands-on calibration conducted by an experimental physicist and automated calibration for a cloud-accessible device, and we look forward

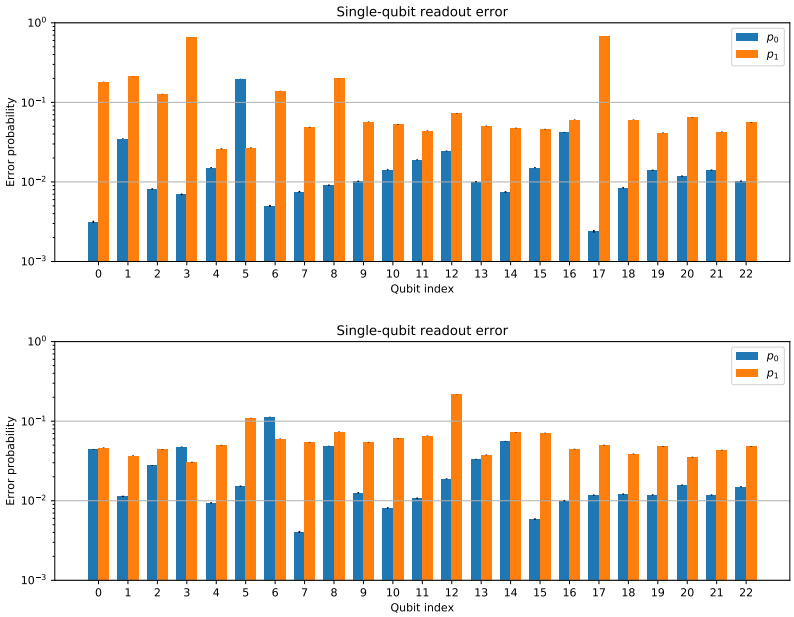

FIG. S6. (Top) Marginalized error probabilities $p_{0,i}$ and $p_{1,i}$ for simultaneous readout of all qubits from a representative calibration used to correct Figure 3 for readout error. (Bottom) Values for typical marginalized simultaneous readout error probabilities after the implementation of an improved automated calibration routine. Error bars (barely visible) represent a 95% confidence interval.

to future research ideas being "productionized" to make them accessible to a wide audience of algorithms researchers. Even in this instance there is still an imperfect abstraction: if one is interested in reading only a subset of all available qubits, higher performance can be obtained by doing a highly-tailored calibration; but we expect that the vast majority of cases will be served better by the new calibration routines.

S4. OPTIMIZER DETAILS

In this section, we describe the classical optimization algorithm that we used to obtain the optimization results presented in Figure 3. The algorithm is a variant of gradient descent which we call "Model Gradient Descent". In each iteration of the algorithm, several points are randomly chosen from the vicinity of the current iterate. The objective function is evaluated at these points, and a quadratic model is fit to the graph of these points and previously evaluated points in the vicinity using least-squares regression. The gradient of this quadratic model is then used as a surrogate for the true gradient, and the algorithm descends in the corresponding direction. Our implementation includes hyperparameters that determine the rate of descent, the radius of the vicinity from which points are sampled (the sample radius), the number of points to sample, and optionally, whether and how quickly the rate of descent and the sample radius should decay as the algorithm proceeds. Pseudocode is given in Algorithm 1.

Algorithm 1 Model Gradient Descent

Input: Initial point x_0, learning rate γ, sample radius δ, sample number k, rate decay exponent α, stability constant A, sample radius decay exponent ξ, tolerance ε, maximum evaluations n

Initialize a list L
Let $x \leftarrow x_0$
Let $m \leftarrow 0$
while (#function evaluations so far) + k does not exceed n **do**
    Add the tuple $(x, f(x))$ to the list L
    Let $\delta' \leftarrow \delta/(m+1)^{\xi}$
    Sample k points uniformly at random from the δ'-neighborhood of x; Call the resulting set S
    for each x' in S **do**
        Add $(x', f(x'))$ to L
    end for
    Initialize a list L'
    for each tuple (x', y') in L **do**
        if $|x' - x| < \delta'$ **then**
            Add (x', y') to L'
        end if
    end for
    Fit a quadratic model to the points in L' using least squares linear regression with polynomial features
    Let g be the gradient of the quadratic model evaluated at x
    Let $\gamma' = \gamma/(m+1+A)^{\alpha}$
    if $\gamma' \cdot |g| < \varepsilon$ **then**
        return x
    end if
    Let $x \leftarrow x - \gamma' \cdot g$
    Let $m \leftarrow m + 1$
end while
return x

S5. SUPPORTING PLOTS FOR PERFORMANCE AT OPTIMAL ANGLES

A. Analysis of Noise

There are two relevant mechanisms when considering the difference in performance between problems. One is the propagation of faults through the circuit and the other is fidelity decay due to circuit depth. A single fault on low-degree problems (Hardware Grid and 3-regular MaxCut, with degree four and three, respectively) can only propagate to terms p edges away from the original location of the fault, irrespective of the total number of qubits. However, if compilation results in circuits extensive in the system size, the probability of a fault increases. For the SK-model, the degree of the problem is extensive in system size so both the propensity for fault propagation as well as the probability of faults grows with n. Additionally, compilation of the 3-regular problems onto the hardware topology introduces SWAPs, which can propagate faults through nodes which would otherwise not be adjacent in the problem graph.

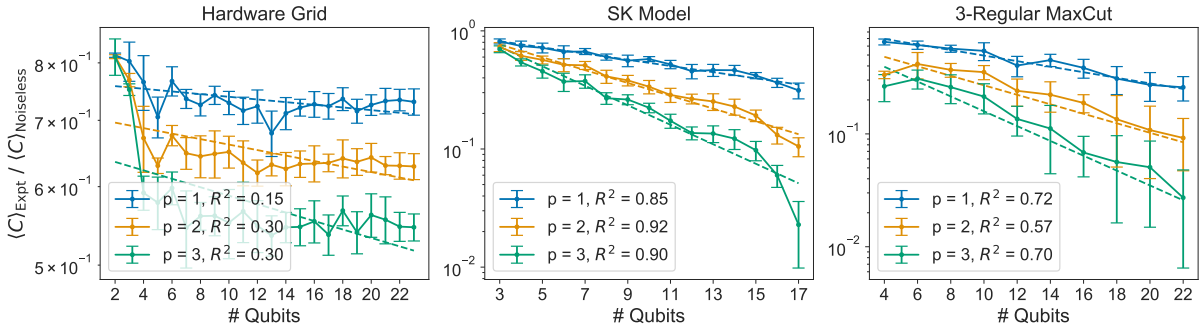

FIG. S7. An exponential model compatible with a depolarizing error channel reasonably models the performance of compiled SK Model and 3-Regular MaxCut problems because their circuits are extensive in system size and faults are rapidly mixed. This error model is a poor fit for Hardware Grid problems due to the low degree of the problem graph and simple compilation.

To probe these two effects, we fit a global depolarizing channel to the results for the three problems. A global depolarizing channel results in the mixed state

$$\rho = f_c|\psi\rangle\langle\psi| + \frac{1-f_c}{d}\mathbf{I}$$

where $|\psi\rangle$ is the noiseless QAOA state, $\mathbf{I}$ is the n-qubit identity matrix, and f_c is the total circuit fidelity. $\mathrm{Tr}(\mathbf{I}C) = 0$ because of the ZZ structure of the cost function, so the experimental objective function is simply a scaled version of the noiseless version, $\langle C\rangle_{\mathrm{Expt}} = f_c\langle C\rangle_{\mathrm{Noiseless}}$. We perform a linear regression on $f_c = f^n \times f_0 \leftrightarrow \log(f_c) = n\log(f) + \log(f_0)$ where $\log(f)$ and $\log(f_0)$ are fittable parameters physically corresponding to a per-qubit fidelity and a qubit-independent offset. For the Hardware Grid, a depolarizing model is inappropriate, as the limited fault propagation and fixed circuit depth yield a largely n-independent noise signature. The exponential decay expected from a global depolarizing channel reasonably fits both the SK model and MaxCut results. We note that the fit is considerably stronger for the high-degree SK model where faults are rapidly mixed.

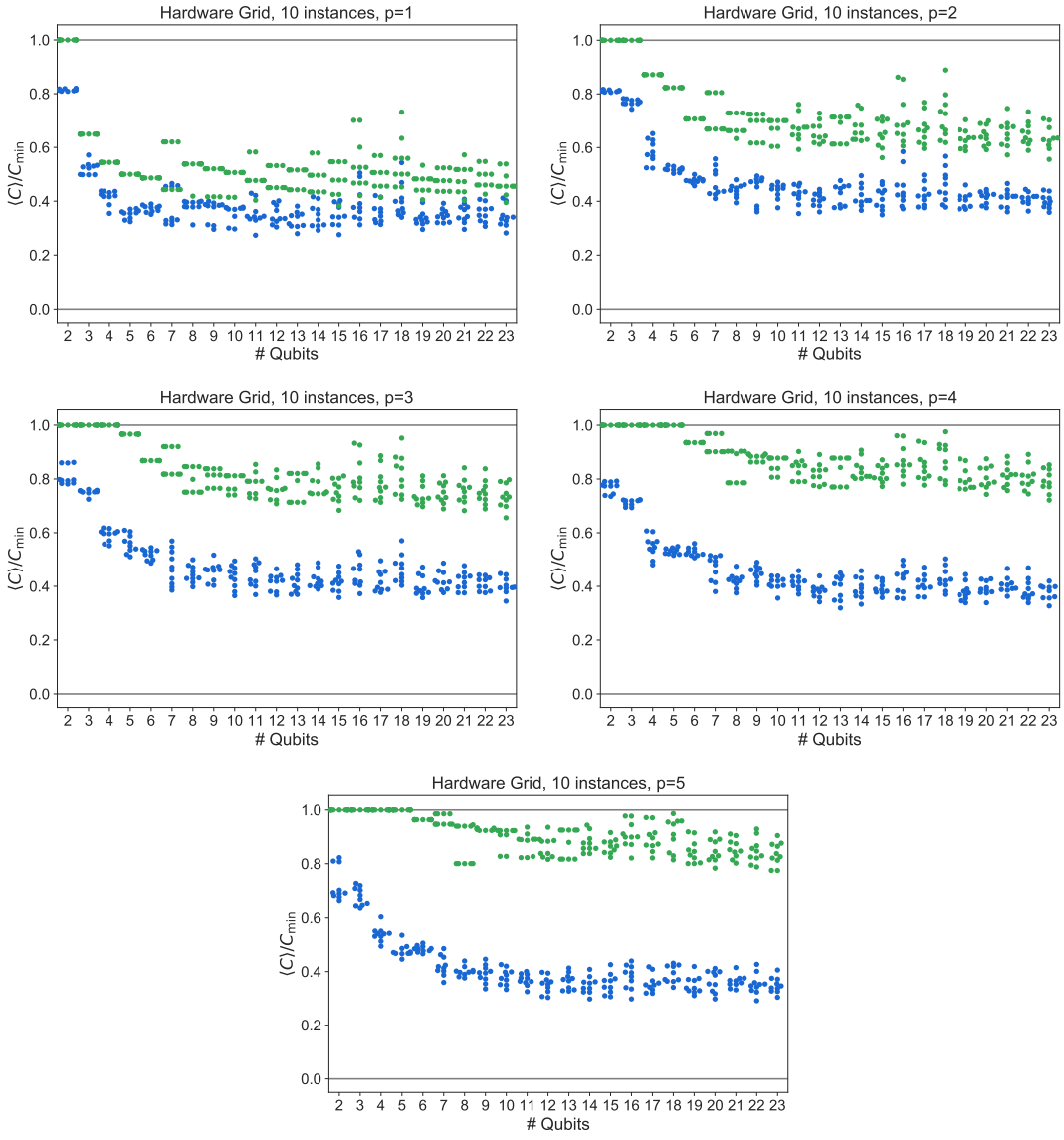

FIG. S8. Performance of QAOA at $p \in [1, 5]$ and $n \in [2, 23]$ over random instantiations of couplings as described in the main text. Points have been perturbed along the x-axis to avoid overlap. **Green:** Noiseless **Blue:** Experimental

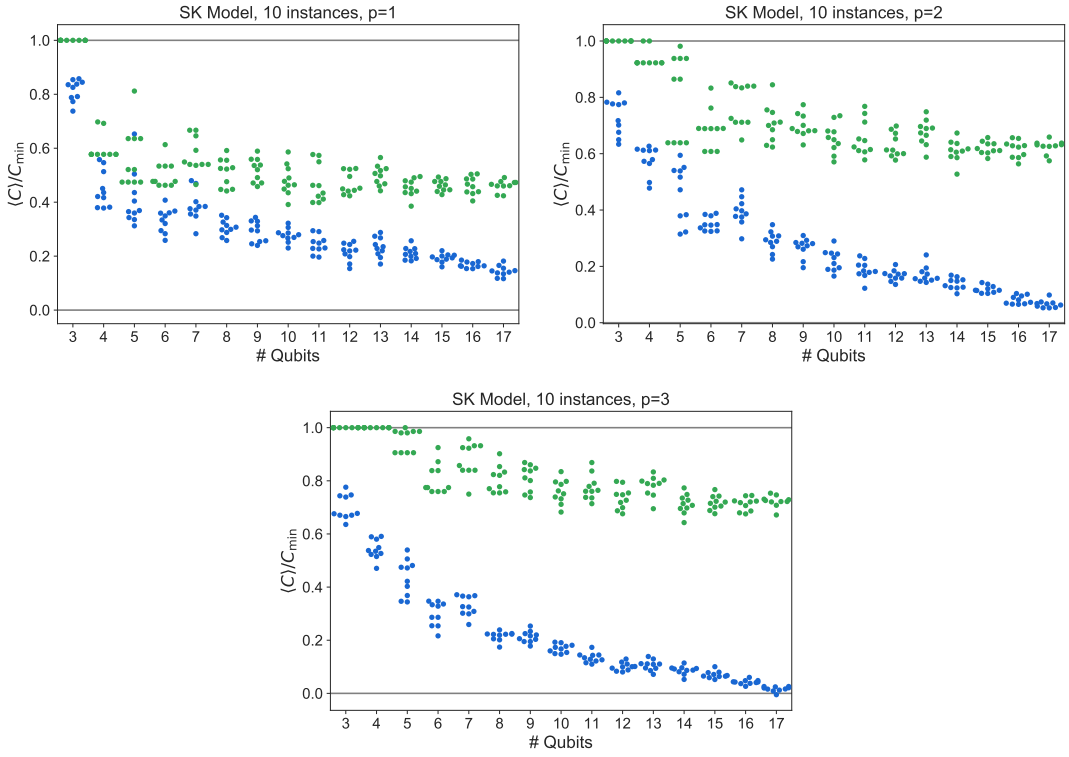

FIG. S9. Performance of QAOA at $p \in [1, 3]$ and $n \in [3, 17]$ over random SK model instances as described in the main text. Points have been perturbed along the x-axis to avoid overlap. **Green:** Noiseless **Blue:** Experimental

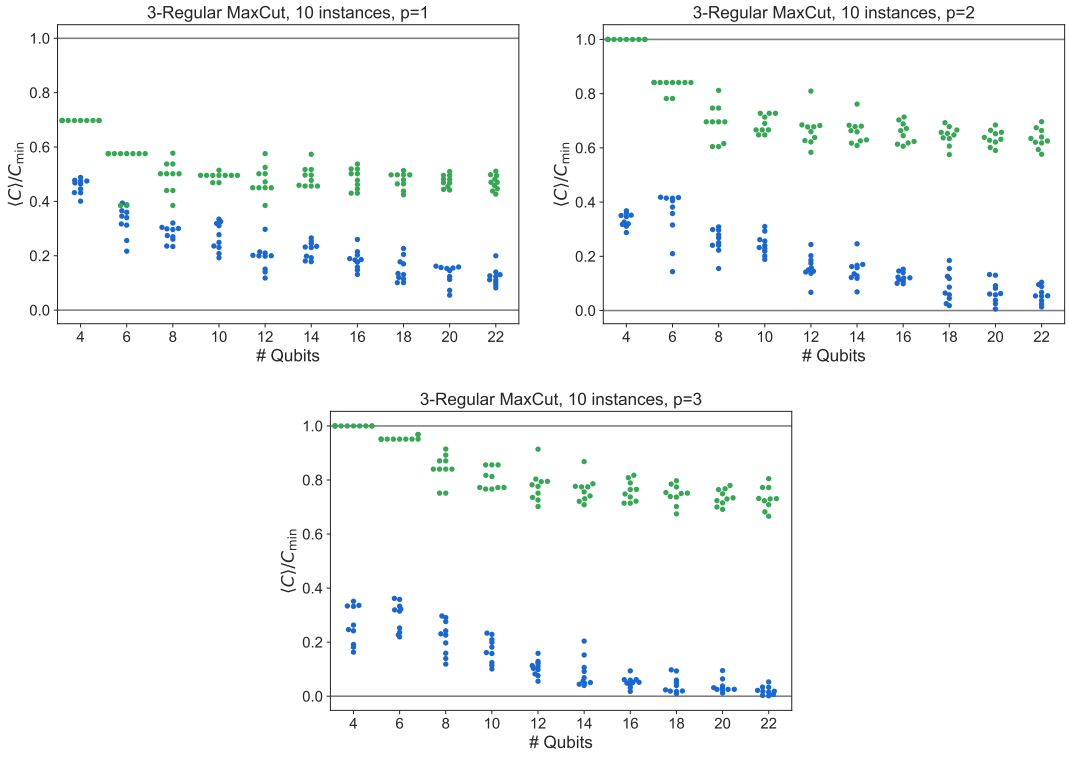

FIG. S10. Performance of QAOA at $p \in [1, 3]$ and $n \in [4, 22]$ over random 3-regular MaxCut problems as described in the main text. Points have been perturbed along the x-axis to avoid overlap. k-regular graphs must satisfy $n \geq k + 1$ and nk must be even, hence only even n are considered here. **Green:** Noiseless **Blue:** Experimental

S6. SUPPORTING PLOTS FOR OPTIMIZATION TRACES

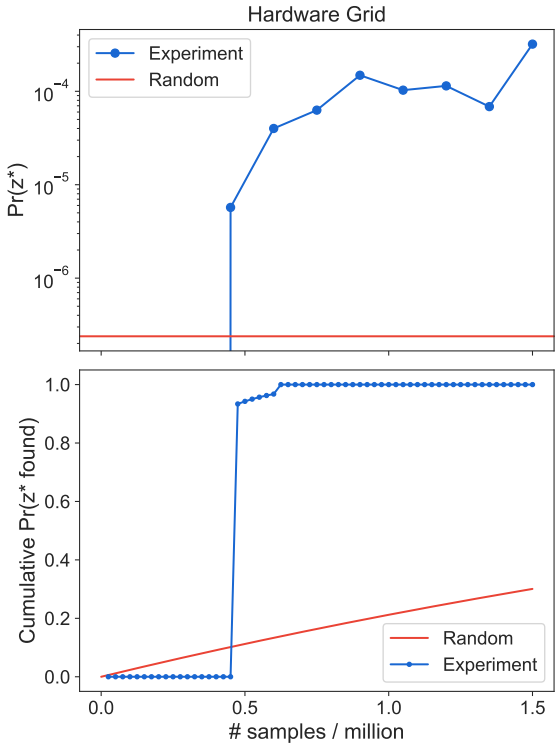

FIG. S11. During experimental optimization of angles, we use six executions of 25,000 samples to estimate the gradient and propose new angles. (top) We estimate the probability of finding a correct bitstring $\Pr(z*)$ by combining the data for the six gradient evaluations as well as an additional execution of 25,000 repetitions at the new angles for a total of 175,000 bitstrings. Measuring these low-probability events requires many samples for an accurate estimate, which is why we mainly focus on the more robust quantity $\langle C \rangle$ in the main text. Points below the plot correspond to regions of parameter space where an optimal bitstring was not observed at all. (bottom) The cumulative probability of seeing an optimal bitstring given the number of samples taken thus far. While random sampling on this 23-bit problem can find a ground state bitstring with roughly 20% chance, it is worthwhile to use the samples to move into a better region of parameter space where the cumulative probability of finding a ground state bitstring quickly jumps to nearly 100%.

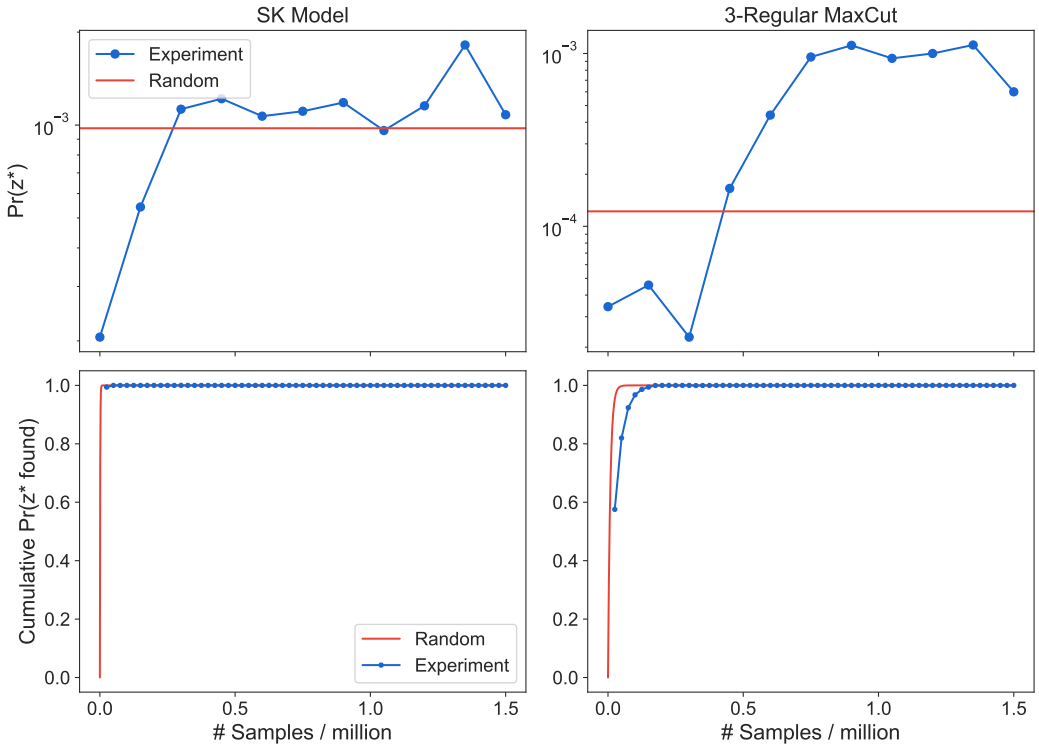

FIG. S12. For the smaller problems, random guessing is an effective strategy compared to the initial optimization points (deliberately chosen to be far from optimal). Once the optimizer makes progress, QAOA is also able to find the optimal bitstrings rapidly.

[1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, Quantum supremacy using a programmable superconducting processor, Nature **574**, 505 (2019).

[2] J. Zhang, J. Vala, S. Sastry, and K. B. Whaley, Geometric theory of nonlocal two-qubit operations, Phys. Rev. A **67**, 042313 (2003).

[3] D. Maslov, S. M. Falconer, and M. Mosca, Quantum circuit placement, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **27**, 752 (2008).

[4] A. Botea, A. Kishimoto, and R. Marinescu, On the complexity of quantum circuit compilation, in *Eleventh Annual Symposium on Combinatorial Search* (2018).

[5] J. X. Lin, E. R. Anschuetz, and A. W. Harrow, Using spectral graph theory to map qubits onto connectivity-limited devices, arXiv:1910.11489 [quant-ph] (2019).

[6] J. S. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. S. Fried, S. Hong, P. Karalekas, C. B. Osborn, A. Papageorge, E. C. Peterson, G. Prawiroatmodjo, N. Rubin, C. A. Ryan, D. Scarabelli, M. Scheer, E. A. Sete, P. Sivarajah, R. S. Smith, A. Staley, N. Tezak, W. J. Zeng, A. Hudson, B. R. Johnson, M. Reagor, M. P. d. Silva, and C. Rigetti, Unsupervised machine learning on a hybrid quantum computer, arXiv:1712.05771 [quant-ph] (2017).

[7] X. Qiang, X. Zhou, J. Wang, C. M. Wilkes, T. Loke, S. O'Gara, L. Kling, G. D. Marshall, R. Santagati, T. C. Ralph, J. B. Wang, J. L. O'Brien, M. G. Thompson, and J. C. F. Matthews, Large-scale silicon quantum photonics implementing arbitrary two-qubit processing, Nature Photonics **12**, 534 (2018).

[8] G. Pagano, A. Bapat, P. Becker, K. S. Collins, A. De, P. W. Hess, H. B. Kaplan, A. Kyprianidis, W. L. Tan, C. Baldwin, L. T. Brady, A. Deshpande, F. Liu, S. Jordan, A. V. Gorshkov, and C. Monroe, Quantum approximate optimization with

a trapped-ion quantum simulator, arXiv:1906.02700 [quant-ph] (2019).
[9] M. Willsch, D. Willsch, F. Jin, H. De Raedt, and K. Michielsen, Benchmarking the quantum approximate optimization algorithm, Quantum Information Processing **19**, 197 (2020).
[10] D. M. Abrams, N. Didier, B. R. Johnson, M. P. d. Silva, and C. A. Ryan, Implementation of the XY interaction family with calibration of a single pulse, arXiv:1912.04424 [quant-ph] (2019).
[11] A. Bengtsson, P. Vikstål, C. Warren, M. Svensson, X. Gu, A. F. Kockum, P. Krantz, C. Križan, D. Shiri, I.-M. Svensson, G. Tancredi, G. Johansson, P. Delsing, G. Ferrini, and J. Bylander, Improved success probability with greater circuit depth for the quantum approximate optimization algorithm, Phys. Rev. Applied **14**, 034010 (2020).