

# Unsupervised Machine Learning on a Hybrid Quantum Computer

J. S. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. Schuyler Fried, S. Hong, P. Karalekas, C. B. Osborn, A. Papageorge, E. C. Peterson, G. Prawiroatmodjo, N. Rubin, Colm A. Ryan, D. Scarabelli, M. Scheer, E. A. Sete, P. Sivarajah, Robert S. Smith, A. Staley, N. Tezak, W. J. Zeng, A. Hudson, Blake R. Johnson, M. Reagor, M. P. da Silva, and C. Rigetti

*Rigetti Computing, Inc., Berkeley, CA*

(Dated: December 18, 2017)

Machine learning techniques have led to broad adoption of a statistical model of computing. The statistical distributions natively available on quantum processors are a superset of those available classically. Harnessing this attribute has the potential to accelerate or otherwise improve machine learning relative to purely classical performance. A key challenge toward that goal is learning to hybridize classical computing resources and traditional learning techniques with the emerging capabilities of general purpose quantum processors. Here, we demonstrate such hybridization by training a 19-qubit gate model processor to solve a clustering problem, a foundational challenge in unsupervised learning. We use the quantum approximate optimization algorithm in conjunction with a gradient-free Bayesian optimization to train the quantum machine. This quantum/classical hybrid algorithm shows robustness to realistic noise, and we find evidence that classical optimization can be used to train around both coherent and incoherent imperfections.

## INTRODUCTION

The immense power of quantum computation is illustrated by flagship quantum algorithms that solve problems such as factoring [1] and linear systems of equations [2], amongst many others, much more efficiently than classical computers. The building of a quantum device with error rates well below the fault-tolerance threshold [3–6] poses a challenge to the implementation of these kinds of quantum algorithms on near-term devices. In recent years several new algorithms targeting these near-term devices have been proposed. These algorithms focus on short-depth parameterized quantum circuits, and use quantum computation as a subroutine embedded in a larger classical optimization loop. It has been shown that optimizing the performance of the quantum subroutine—by varying a few free parameters—allows for calculating binding energies in quantum chemistry [7–10], as well as solving some combinatorial [11, 12] and tensor network problems [13]. In this paper we choose to focus on an unsupervised machine learning task known as *clustering*, which we translate into a combinatorial optimization problem [14, 15] that can be solved by the quantum approximate optimization algorithm (QAOA) [11, 12]. We implement said algorithm on a 19-qubit computer using a flexible quantum programming platform [16, 17]. We show that our implementation of this algorithm finds the optimal solution to random problem instances with high probability, and that good approximate solutions are found in all investigated cases, even with relatively noisy gates. This robustness is enabled partially by using a Bayesian procedure [18, 19] in the classical optimization loop for the quantum circuit parameters.

## CLUSTERING

The particular unsupervised machine learning problem we focus on here is known as clustering [20, 21]. Clustering consists of assigning labels to elements of a dataset based only on how similar they are to each other—like objects will have the same label, unlike objects will have different labels. Mathematically, the dataset  $\mathcal{D}$  has elements  $\mathbf{x}_i$  for  $1 \leq i \leq n$ , where each element is a  $k$ -dimensional *feature vector* (a numerical representation of any object of interest: a photograph, an audio recording, etc.). In order to represent dissimilarity, we need to define a distance measure  $d(\mathbf{x}_i, \mathbf{x}_j)$  between two samples  $\mathbf{x}_i$  and  $\mathbf{x}_j$ . A familiar choice for a distance measure is the Euclidean distance, but specific applications may naturally lead to very different choices for the metric and the distance [22, 23].

Most common choices of distances allows us to calculate a matrix of distances between all points in  $\mathcal{O}(kn^2)$  steps, by simply calculating the distance between every possible pair of data samples. Let this matrix be denoted by  $C$  where  $C_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$ . This matrix can be interpreted as an adjacency matrix of a graph  $G$ , where each vertex represents an element of  $\mathcal{D}$  and  $C_{ij}$  is the *weight* of edge between vertices  $i$  and  $j$ . In general the matrix  $C$  will be dense leading to a fully connected graph on  $n$  vertices, but different choices of distance metrics along with coarse-graining can make this distance matrix sparse. In clustering, the main assumption is that distant points belong to different clusters; hence maximizing the overall sum of all weights (distances) between nodes with different labels represents a natural clustering algorithm. The mathematical formulation of this is a Maximum-Cut (MAXCUT) problem [15], defined as  $\text{MAXCUT}(G, C)$  for the dense graph  $G$  of the distance matrix  $C$ .

More precisely, the MAXCUT problem consists of an

undirected graph  $G = (V, E)$  with a set of vertices  $V$  and a set of edges  $E$  between those vertices. The weight  $w_{ij}$  of an edge between vertices  $i$  and  $j$  is a positive real number, with  $w_{ij} = 0$  if there is no edge between them. A *cut*  $\delta(S) \subset E$  is a set of edges that separates the vertices  $V$  into two disjoint sets  $S$  and  $\bar{S} = V \setminus S$ . The cost  $w(\delta(S))$  of a cut is defined as the sum of all weights of edges connecting vertices in  $S$  with vertices in  $\bar{S}$

$$w(\delta(S)) = \sum_{i \in S, j \in \bar{S}} w_{ij}. \quad (1)$$

The problem  $\text{MAXCUT}(G, w)$  is now easily formulated as an optimization objective

$$\text{MAXCUT}(G, w) = \max_{S \subset V} w(\delta(S)). \quad (2)$$

The MAXCUT problem is an example of the class of NP-complete problems [24], which are notoriously hard to solve. Many other combinatorial problems can be reduced to MAXCUT—e.g., machine scheduling [25], computer-aided design [26], traffic message management problems [27], image recognition [28], quadratic unconstrained optimization problems (QUBO) [15] and many more. One approach to solving MAXCUT is to construct a physical system—typically a set of interacting spin- $\frac{1}{2}$  particles [14]—whose lowest energy state encodes the solution to the problem, so that solving the problem is equivalent to finding the ground state of the system [29]. This is the approach we take here.

## QUANTUM APPROXIMATE OPTIMIZATION ALGORITHM

It is possible to find the ground state of interacting spin systems using an algorithm known as the *quantum approximate optimization algorithm* (QAOA) [11]. QAOA can be thought of as a heuristic to prepare a superposition of bit strings with probability amplitudes heavily concentrated around the solution. The encoding of the problem itself is given by a *cost Hamiltonian* (cf. S1)

$$\hat{H}_C = -\frac{1}{2} \sum_{i,j} C_{ij} (1 - \hat{\sigma}_i^z \hat{\sigma}_j^z), \quad (3)$$

and QAOA approximates the ground state by initially preparing the equal superposition of all bit strings, then iteratively applying a pair of unitary operations before measuring (see Fig. 1). For the  $i$ th iteration, we evolve the system with cost unitary  $\hat{U}_i = \exp(-i\gamma_i \hat{H}_C)$  for some angle  $\gamma_i$ , followed by the driver unitary  $\hat{V}_i = \exp(-i\beta_i \hat{H}_D)$  for some angle  $\beta_i$ , where the *driver Hamiltonian* is

$$\hat{H}_D = \sum_i \hat{\sigma}_i^x. \quad (4)$$

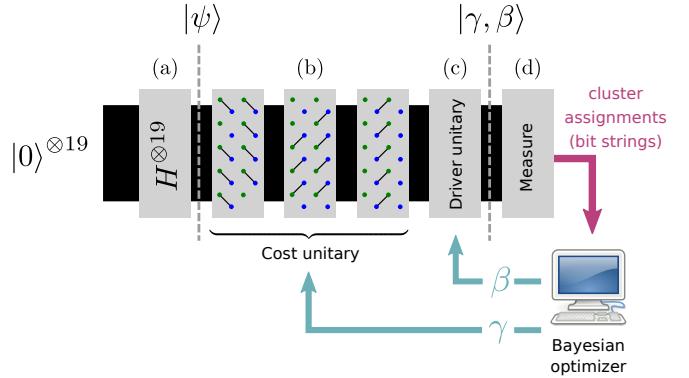


FIG. 1. Circuit diagram for the clustering experiment described in the main text <sup>a</sup>, which corresponds to a single iteration of QAOA to solve a MAXCUT problem. The algorithm proceeds by (a) initializing all qubits in the 0 state, and then applying Hadamard gates to each of the qubits. (b) Next, due to parallelism constraints, it applies three independent rounds of interactions of the form  $\exp(-i\gamma w_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z)$  (actual gate sequence not shown for brevity). The overall scaling  $\gamma$  for the interactions is chosen and varied by the optimization routine. (c) Single qubit  $\hat{\sigma}_x$  rotations are applied to each qubit, again with an overall rotation angle  $\beta$  chosen by the optimization routine. (d) Finally, all qubits are measured and fed into the classical optimization routine. Many shots are collected, and the best MAXCUT cost, along with the history of previous angles, is used to inform the next choice for  $(\beta, \gamma)$ .

<sup>a</sup> Computer icon is used under a Creative Commons license [30].

In other words, QAOA prepares the state

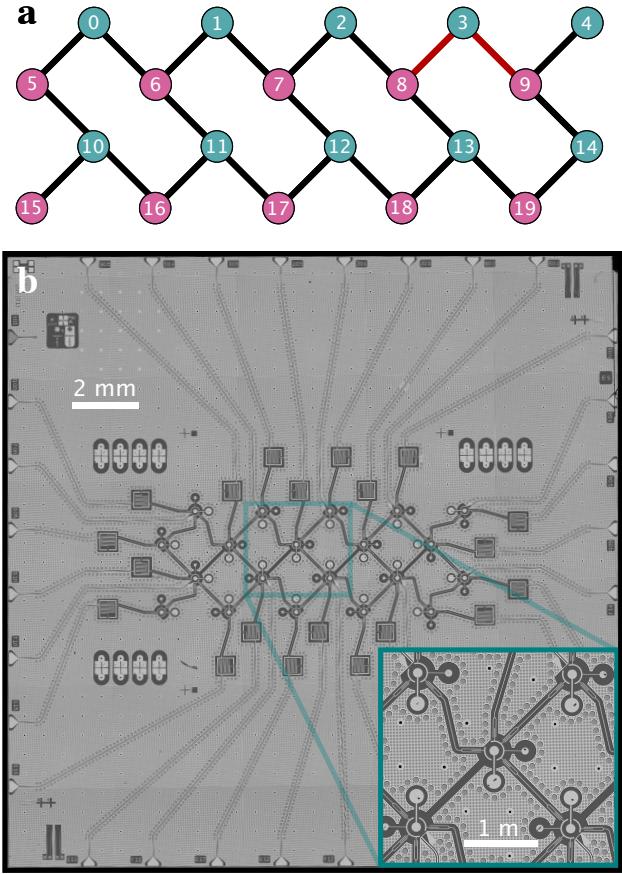
$$|\gamma, \beta\rangle = \hat{V}_p \hat{U}_p \cdots \hat{V}_1 \hat{U}_1 |\psi\rangle, \quad (5)$$

where  $p$  is the number of iterations,  $|\psi\rangle$  is the uniform superposition of all bit strings (with the same phase), and measures this state in the computational basis.

For the optimal choice of angles, the probability of sampling the bit string corresponding to the lowest energy state of  $\hat{H}_C$  increases monotonically with the number of iterations applied [11]. As the number of iterations approaches infinity, the probability of success approaches unity. However, QAOA does not specify how the optimal set of angles are found, although in some cases they can be derived analytically for a small number of iterations and for uniformly weighted graphs [11, 31].

## BAYESIAN OPTIMIZATION

Our strategy to find the (approximately) optimal bit-string for a weighted MAXCUT instance, corresponding to the solution of the clustering problem, uses Bayesian optimization over the QAOA angles  $(\gamma, \beta)$  [32, 33]. This framework constructs a prior distribution over functions, sequentially evaluates a black box objective—the MAXCUT instance in this case—and updates the prior accord-



**FIG. 2. Connectivity of Rigetti 19Q.** **a**, Chip schematic showing tunable transmons (teal circles) capacitively coupled to fixed-frequency transmons (pink circles). **b**, Optical chip image. Note that some couplers have been dropped to produce a lattice with three-fold, rather than four-fold, connectivity.

ing to Bayes' rule

$$p(f|y) \sim p(y|f)p(f), \quad (6)$$

where  $p(f|y)$  is the posterior distribution over function space given the observations  $y$ ,  $p(f)$  is the prior over function space, and  $p(y|f)$  is the likelihood of observing the values  $y$  given the model for  $f$ . With growing number of optimization steps (observations  $y$ ) the true black-box objective is increasingly well approximated. The trick lies in choosing the prior  $p(f)$  in a way that offers closed-form solutions for easy numerical updates, such as Gaussian processes, which assume a normal distribution as a prior over the function space [18](cf. S10). In the present case of QAOA, it should be noted that sampling at each step will generally lead to a non-trivial distribution of values when the state  $|\gamma, \beta\rangle$  is entangled or mixed. To fit this into the Bayesian Optimization framework we calculate the best observed sample and return this to the optimizer. Hence, the function  $f$  represents the value of the best sampled bit string at location  $\gamma, \beta$ . More generally, one could compute any statistic of the distribution (as detailed

in the appendix).

To avoid a random walk over the space of potential evaluation points, the Bayesian optimizer maximizes a utility function that can be calculated from the posterior distribution after each update. In this way, it intelligently chooses points to minimize the number of costly evaluations of the black box objective function (see the appendix for more details).

## THE QUANTUM PROCESSOR

We ran the QAOA optimizer on a quantum processor consisting of 20 superconducting transmon qubits [34] with fixed capacitive coupling in the lattice shown in Fig. 2. Qubits 0–4 and 10–14 are tunable while qubits 5–9 and 15–19 are fixed-frequency devices. The former have two Josephson junctions in an asymmetric SQUID geometry to provide roughly 1 GHz of frequency tunability, and flux-insensitive “sweet spots” [35] near  $\omega_{01}^{\max}/2\pi \approx 4.5$  GHz and  $\omega_{01}^{\min}/2\pi \approx 3.0$  GHz. These tunable qubits are coupled to bias lines for AC and DC flux delivery. Each qubit is capacitively coupled to a quasi-lumped element resonator for dispersive readout of the qubit state [36, 37]. Single-qubit control is effected by applying microwave drives at the resonator ports, and two-qubit gates are activated via RF drives on the flux bias lines, described below.

The device is fabricated on a high-resistivity silicon substrate with superconducting through-silicon via technology [38] to improve RF-isolation. The superconducting circuitry is realized with Aluminum ( $T_c \approx 1.2$  K), and patterned using a combination of optical and electron-beam lithography. Due to a fabrication defect, qubit 3 is not tunable, which prohibits operation of the 2-qubit parametric gate described below between qubits 3 and its neighbors (8 and 9). Consequently, we treat this as a 19-qubit processor.

In Rigetti 19Q, as we call our device, each tunable qubit is capacitively coupled to one-to-three fixed-frequency qubits. The DC flux biases are set close to zero flux such that each tunable qubit is at its maximum frequency  $\omega_T^{\max}$ . Two-qubit parametric CZ gates are activated in the  $|11\rangle \leftrightarrow |20\rangle$  and/or  $|11\rangle \leftrightarrow |02\rangle$  sub-manifolds by applying an RF flux pulse with amplitude  $A_0$ , frequency  $\omega_m$  and duration  $t_{CZ}$  to the tunable qubit [39–41]. For RF flux modulation about the qubit extremal frequency, the oscillation frequency is doubled to  $2\omega_m$  and the mean effective qubit frequency shifts to  $\bar{\omega}_T$ . Note that the frequency shift increases with larger flux pulse amplitude. The effective detuning between neighboring qubits becomes  $\Delta = \bar{\omega}_T - \omega_F$ .

The resonant condition for a CZ gate is achieved when  $\Delta = 2\omega_m - \eta_T$  or  $\Delta = 2\omega_m + \eta_F$ , where  $\eta_T, \eta_F$  are the anharmonicities of the tunable and fixed qubit, respectively. An effective rotation angle of  $2\pi$  on these transitions im-

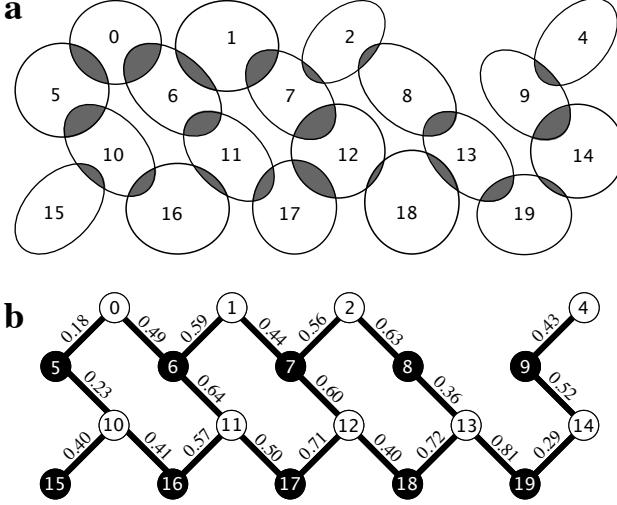


FIG. 3. **a**, The general form of the clustering problem instance solved on the 19Q chip, and **b**, the corresponding MAXCUT problem instance solved on the 19 qubit chip. The edge weights—corresponding to the overlap between neighbouring probability distributions on the plane—are chosen at random and the labels indicate the mapping to the corresponding qubit. The vertices are colored according to the solution to the problem.

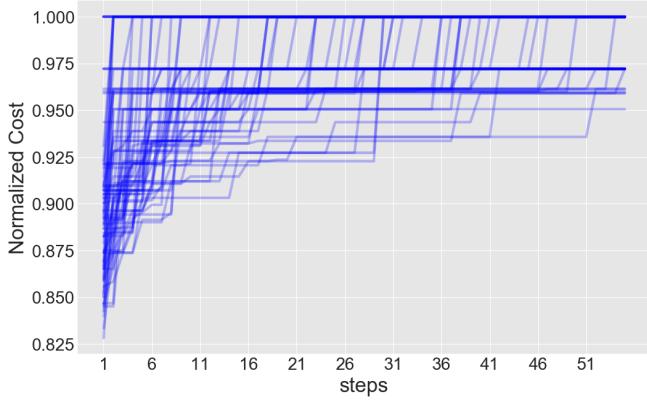


FIG. 4. Traces for the normalized MAXCUT cost for 83 independent runs of the algorithm on the 19Q chip for the fixed, but random, problem instances of Fig. 3. Notice that most traces reach the optimal value of 1 well before the cutoff at 55 steps.

parts a minus sign to the  $|11\rangle$  state, implementing an effective CZ gate. The time-scale of these entangling gates is in the range 100–250 ns. Due to finite bandwidth constraints of our control electronics, the applied flux pulse is shaped as a square pulse with linear rise and fall time of 30 ns.

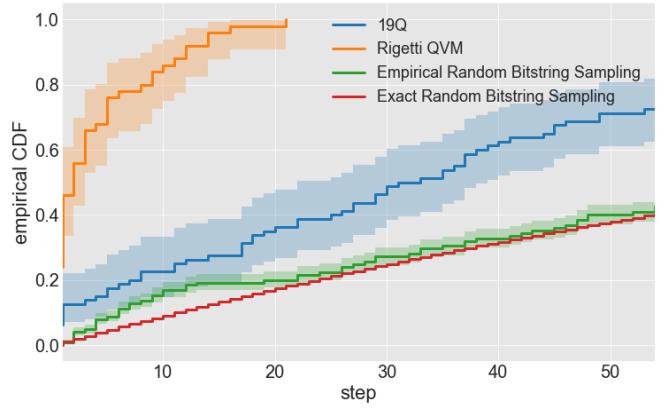


FIG. 5. The performance of our implementation of the clustering algorithm on the 19Q chip (blue) and a noiseless simulation through the Forest [42] quantum virtual machine (orange) can be compared to the performance of an algorithm that simply draws cluster assignments at random (red: Theoretical curve, green: empirical confirmation). It is clear that our algorithm generates the optimal assignment much more quickly than it would be expected by chance: the 95% confidence region for our empirical observations have very small overlap for the distribution given by random assignments. See appendix for more detailed statistics.

## IMPLEMENTATION

We demonstrate the implementation of the proposed clustering algorithm on a problem with a cost Hamiltonian constructed to match the connectivity of Rigetti 19Q. Such a choice minimizes the circuit depth required for the algorithm while utilizing all available qubits. This problem corresponds to clustering an arrangement of overlapping probability distributions, whose dissimilarity is given by the Bhattacharyya coefficient [43]—a measure for distributional overlap. A cartoon of such an arrangement, and the corresponding graph for the weighted MAXCUT problem, is depicted in Fig. 3 [44]. This problem is solved using a single iteration ( $p = 1$ ) of QAOA, and using up to 55 steps of a Bayesian optimizer to choose the angles ( $\gamma, \beta$ ). This procedure is repeated several times to gather statistics about the number of steps necessary to reach the optimal answer.

Using the available interactions and local gate operations, we are able to implement the cost unitary in a circuit of depth corresponding to six CZ gates interspersed with single qubit operations and hence fitting well within a single coherence time of the qubits. This circuit depth is dictated by two independent factors. The first is the implementation of the cost unitary. Since all the terms in the cost Hamiltonian commute, it is possible to decompose the cost unitary into a separate unitary for each cost term. These cost terms, in turn, take the form  $\exp(-i\gamma w_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z)$  that does not directly correspond to one of the native gates in our quantum computer. However, they can be

implemented by the sequence requiring two interaction  $\text{CNOT}_{ij} \cdot \text{RZ}_j(\gamma w_{ij}) \cdot \text{CNOT}_{ij}$  [45]. The second factor is that the CNOT interactions cannot be applied simultaneously on overlapping sets of qubits. Given the connectivity of the graph for our problem (where some vertices have 3 neighbors), that means the cost terms must be broken down into 3 rounds of non-overlapping unitaries, each of which consists of two interactions, so that the overall circuit has a depth corresponding to 6 two-qubit gates interspersed with single qubit operations. Additional circuit compilation steps are taken to minimize the number of intermediate single-qubit operations, so the depth is ultimately dominated by the interaction.

The entire algorithm is implemented in Python, leveraging the `pyQuil` library [17] for describing parameterized quantum circuits in the quantum instruction language `Quil` [16], and run through the `Forest` platform [42] for controlling the quantum computer and accessing the data it generates. The Bayesian optimizer is provided by the open source package `BayesianOptimization`, also written in Python [19].

## RESULTS

The typical behaviour of the algorithm is illustrated in Fig. 4. Each trace corresponds to a separate run of the algorithm on a fixed, but random, choice for the graph weights. In particular, it corresponds to the best observed cost up to the current Bayesian optimization step, i.e. the historic best. The costs are normalized such that 1 corresponds to the optimal solution, which can be easily computed for the problem instance under consideration.

The run time for 55 Bayesian optimization steps with  $N = 2500$  measurements per step is approximately 10 minutes. This run time includes network latency in the communication between the quantum processor and the classical processor running the high-level hybrid algorithm, as well as many other factors such as circuit compilation and job scheduling. For the random problem instances chosen, the vast majority of the traces reach the optimum in fewer than 55 steps (only about 25% of runs did not reach the optimum within 55 steps, as illustrated in Fig. 5). Different random choices for the graph weights yield results that are qualitatively identical (see appendix for more details).

Quantitatively, we can compare how execution of the algorithm on the quantum computer compares with drawing a comparable number of bit strings uniformly at random, and remembering the ones with the highest cost. Random sampling of bit strings can be seen as a proxy for randomly walking the parameter space, which is expected to be a better strategy than grid-search [46]. In that case, the probability of obtaining the optimal cost should decrease exponentially with the number of points in the clustering problem, but increase monotonically

with the total number of samples. In Fig. 5 we compare the empirical distributions for a noiseless simulation of the algorithm (orange), the experimental implementation of the algorithm (blue), and the random sampling of bit strings with finite statistics (green), along with the analytical distribution for random sampling of bit strings (red)—the shaded regions correspond to 95% confidence intervals. Taking the null hypothesis to be the random sampling of bit strings, standard hypothesis testing methods based on the Kolmogorov-Smirnov statistic exclude the null hypothesis as an explanation for the behavior of the experimental implementation of the algorithm to a level higher than 99%. Similarly, the null hypothesis is excluded as an explanation for the behaviour of the noiseless simulation of the algorithm to a level higher than 99.99% (see the appendix for more details).

## CONCLUSION AND SUMMARY

We have used a 19-qubit quantum computer to demonstrate a hybrid quantum algorithm for clustering. This algorithm is based on the quantum approximate optimization algorithm, which in turn can be applied to a broad class of combinatorial optimization problems, ranging from image recognition to machine scheduling. Our implementation relies on Bayesian optimization of classical parameters within the quantum circuit, and we show that the algorithm reaches the optimal solution in many fewer steps than would otherwise be expected by drawing cluster assignments uniformly at random. This demonstration uses the largest number of qubits to-date for an algorithm in a gate-based quantum processor. Harder combinatorial problems (larger instances and more complex topologies) may be handled by larger number of iterations of the quantum approximate optimization algorithm, which would necessitate both higher gate fidelities and higher qubit counts. Given the complexity of solving combinatorial problems, relatively modest improvements on these fronts could lead to the demonstration of a hybrid algorithm that outperforms classical algorithms for the same problem.

## ACKNOWLEDGEMENTS

The algorithm was written in Python with `pyQuil` [17] and uses the `BayesianOptimization` package [19]. The data analysis was performed using code written in `NumPy` [47], `QuTiP` [48], and the figures were made with `matplotlib` [49] and `Inkscape` [50].

## CONTRIBUTIONS

J.S.O., N.R., and M.P.S. developed the theoretical proposal. J.S.O. and E.S.F. implemented the algorithm, and J.S.O. performed the data analysis. M.B., E.A.S., M.S., and A.B. designed the 20-qubit device. R.M., S.C., C.A.R., N.A., A.S., S.H., N.D., D.S., and P.S. brought up the experiment. C.B.O., A.P., B.B., P.K., G.P., N.T., M.R. developed the infrastructure for automatic recalibration. E.C.P., P.K., W.Z., and R.S.S. developed the compiler and QVM tools. J.S.O., M.P.S., B.R.J., M.R., R.M. wrote the manuscript. B.R.J., M.R., A.H., M.P.S., and C.R. were principal investigators of the effort.

- 
- [1] P. W. Shor, in *Proceedings 35th Annual Symposium on Foundations of Computer Science* (1994) pp. 124–134.
  - [2] A. W. Harrow, A. Hassidim, and S. Lloyd, *Phys. Rev. Lett.* **103**, 150502 (2009).
  - [3] P. W. Shor, in *Proceedings of 37th Conference on Foundations of Computer Science* (1996) pp. 56–65.
  - [4] E. Knill, R. Laflamme, and W. H. Zurek, *Science* **279**, 342 (1998), <http://science.science.org/content/279/5349/342.full.pdf>.
  - [5] D. Aharonov and M. Ben-Or, *SIAM Journal on Computing* **38**, 1207 (2008), <https://doi.org/10.1137/S0097539799359385>.
  - [6] P. Aliferis, D. Gottesman, and J. Preskill, *Quantum Info. Comput.* **6**, 97 (2006).
  - [7] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O’Brien, *Nature Communications* **5**, 4213 EP (2014).
  - [8] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, *New Journal of Physics* **18**, 023023 (2016).
  - [9] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, *Nature* **549**, 242 EP (2017).
  - [10] J. I. Colless, V. V. Ramasesh, D. Dahmen, M. S. Blok, J. R. McClean, J. Carter, W. A. de Jong, and I. Siddiqi, “Robust determination of molecular spectra on a quantum processor,” (2017), <arXiv:1707.06408>.
  - [11] E. Farhi, J. Goldstone, and S. Gutmann, <arXiv:1411.4028> (2014).
  - [12] S. Hadfield, Z. Wang, B. O’Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, “From the quantum approximate optimization algorithm to a quantum alternating operator ansatz,” (2017), <arXiv:1709.03489>.
  - [13] I. H. Kim and B. Swingle, “Robust entanglement renormalization on a noisy quantum computer,” (2017), <arXiv:1711.07500>.
  - [14] A. Lucas, *Frontiers in Physics* **2**, 5 (2014).
  - [15] S. Poljak and Z. Tuza, in *Combinatorial Optimization*, Vol. 20, edited by W. Cook, L. Lovász, and P. Seymour (American Mathematical Society, 1995).
  - [16] R. S. Smith, M. J. Curtis, and W. J. Zeng, “A practical quantum instruction set architecture,” (2016).
  - [17] Rigetti Computing, “pyquil,” <https://github.com/rigetticomputing/pyquil> (2016).
  - [18] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning* (MIT Press, 2006).
  - [19] F. Nogueira, “bayesian-optimization,” <https://github.com/fmfn/BayesianOptimization> (2014).
  - [20] A. K. Jain, M. N. Murty, and P. J. Flynn, *ACM Comput. Surv.* **31**, 264 (1999).
  - [21] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data* (Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988).
  - [22] A. S. Shirkhorshidi, S. Aghabozorgi, and T. Y. Wah, *PLOS ONE* **10**, 1 (2015).
  - [23] S. Boriah, V. Chandola, and V. Kumar, “Similarity measures for categorical data: A comparative evaluation,” in *Proceedings of the 2008 SIAM International Conference on Data Mining*, pp. 243–254, <http://pubs.siam.org/doi/pdf/10.1137/1.9781611972788.22>.
  - [24] R. M. Karp, in *Complexity of Computer Computations*, edited by R. E. Miller and J. W. Thatcher (New York: Plenum, 1972) pp. 85–103.
  - [25] B. Alidaee, G. A. Kochenberger, and A. Ahmadian, *International Journal of Systems Science* **25**, 401 (1994).
  - [26] J. Krarup and P. M. Pruzan, “Computer-aided layout design,” in *Mathematical Programming in Use*, edited by M. L. Balinski and C. Lemarechal (Springer Berlin Heidelberg, Berlin, Heidelberg, 1978) pp. 75–94.
  - [27] G. Gallo, P. L. Hammer, and B. Simeone, “Quadratic knapsack problems,” in *Combinatorial Optimization*, edited by M. W. Padberg (Springer Berlin Heidelberg, Berlin, Heidelberg, 1980) pp. 132–149.
  - [28] H. Neven, G. Rose, and W. G. Macready, <arXiv:0804.4457> (2008).
  - [29] Note that we can turn any maximization procedure into a minimization procedure by simply changing the sign of the objective function.
  - [30] GNOME icon artists, “gnome computer icon,” (2008), creative commons license.
  - [31] Z. Wang, S. Hadfield, Z. Jiang, and E. G. Rieffel, “The quantum approximation optimization algorithm for max-cut: A fermionic view,” (2017).
  - [32] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, *Proceedings of the IEEE* **104**, 148 (2015).
  - [33] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. M. A. Patwary, P. Prabhat, and R. P. Adams, in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML’15 (JMLR.org, 2015) pp. 2171–2180.
  - [34] J. Koch, T. Yu, J. M. Gambetta, A. A. Houck, D. I. Schuster, J. Majer, A. Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf, *Physical Review A* **76**, 042319 (2007).
  - [35] D. Vion, A. Aassime, A. Cottet, P. Joyez, H. Pothier, C. Urbina, D. Esteve, and M. H. Devoret, *Science* **296**, 886 (2002).
  - [36] A. Blais, R.-S. Huang, A. Wallraff, S. M. Girvin, and R. J. Schoelkopf, *Physical Review A* **69**, 062320 (2004).
  - [37] A. Blais, J. M. Gambetta, A. Wallraff, D. I. Schuster, S. M. Girvin, M. H. Devoret, and R. J. Schoelkopf, *Physical Review A* **75**, 032329 (2007).
  - [38] M. Vahidpour, W. O’Brien, J. T. Whyland, J. Angeles, J. Marshall, D. Scarabelli, G. Crossman, K. Yadav, Y. Mohan, C. Bui, V. Rawat, R. Renzas, N. Vodrahalli, A. Bestwick, and C. Rigetti, <arXiv:1708.02226> (2017).
  - [39] N. Didier, E. A. Sete, M. P. da Silva, and C. Rigetti, “Analytical modeling of parametrically-modulated transmon

- qubits,” (2017), [arXiv:1706.06566](https://arxiv.org/abs/1706.06566).
- [40] S. Caldwell, N. Didier, C. A. Ryan, E. A. Sete, A. Hudson, P. Karalekas, R. Manenti, M. Reagor, M. P. da Silva, R. Sinclair, E. Acala, N. Alidoust, J. Angeles, A. Bestwick, M. Block, B. Bloom, A. Bradley, C. Bui, L. Capelluto, R. Chilcott, J. Cordova, G. Crossman, M. Curtis, S. Deshpande, T. E. Bouayadi, D. Girshovich, S. Hong, K. Kuang, M. Lenihan, T. Manning, A. Marchenkov, J. Marshall, R. Maydra, Y. Mohan, W. O’Brien, C. Osborn, J. Otterbach, A. Papageorge, J. P. Paquette, M. Pelstring, A. Polloreno, G. Prawiroatmodjo, V. Rawat, R. Renzas, N. Rubin, D. Russell, M. Rust, D. Scarabelli, M. Scheer, M. Selvanayagam, R. Smith, A. Staley, M. Suska, N. Tezak, D. C. Thompson, T. W. To, M. Vahidpour, N. Vodrahalli, T. Whyland, K. Yadav, W. Zeng, and C. Rigetti, “Parametrically activated entangling gates using transmon qubits,” (2017), [arXiv:1706.06562](https://arxiv.org/abs/1706.06562).
- [41] M. Reagor, C. B. Osborn, N. Tezak, A. Staley, G. Prawiroatmodjo, M. Scheer, N. Alidoust, E. A. Sete, N. Didier, M. P. da Silva, E. Acala, J. Angeles, A. Bestwick, M. Block, B. Bloom, A. Bradley, C. Bui, S. Caldwell, L. Capelluto, R. Chilcott, J. Cordova, G. Crossman, M. Curtis, S. Deshpande, T. E. Bouayadi, D. Girshovich, S. Hong, A. Hudson, P. Karalekas, K. Kuang, M. Lenihan, R. Manenti, T. Manning, J. Marshall, Y. Mohan, W. O’Brien, J. Otterbach, A. Papageorge, J. P. Paquette, M. Pelstring, A. Polloreno, V. Rawat, C. A. Ryan, R. Renzas, N. Rubin, D. Russell, M. Rust, D. Scarabelli, M. Selvanayagam, R. Sinclair, R. Smith, M. Suska, T. W. To, M. Vahidpour, N. Vodrahalli, T. Whyland, K. Yadav, W. Zeng, and C. T. Rigetti, “Demonstration of universal parametric entangling gates on a multi-qubit lattice,” (2017), [arXiv:1706.06570](https://arxiv.org/abs/1706.06570).
- [42] Rigetti Computing, “Forest,” <https://www.rigetti.com/forest> (2017).
- [43] A. Bhattacharyya, Bulletin of the Calcutta Mathematical Society , 99 (1943).
- [44] As the Bhattacharyya coefficient is not a traditional distance metric (it violates the triangle inequality) we should interpret clustering as the characteristic that distributions within the same cluster, i.e. the same label, have minimal—in our case zero—overlap. Phrased in this way the connection to VLSI design becomes obvious, where on sub-goal is to identify groups of objects with minimal overlap.
- [45] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, 2011).
- [46] J. Bergstra and Y. Bengio, *Journal of Machine Learning Research* **13**, 281 (2012).
- [47] S. van der Walt, S. C. Colbert, and G. Varoquaux, *Computing in Science Engineering* **13**, 22 (2011).
- [48] J. Johansson, P. Nation, and F. Nori, *Computer Physics Communications* **184**, 1234 (2013).
- [49] J. D. Hunter, *Computing In Science & Engineering* **9**, 90 (2007).
- [50] Inkscape Project, “Inkscape,” .
- [51] J. K. Blitzstein and J. Hwang, *Introduction to Probability* (CRC Press, Taylor & Francis Group, 2015).
- [52] We use the short-hand notation  $z_{i:j}$  to denote the collection of values  $\{z_i, \dots, z_j\}$ .
- [53] J. Snoek, H. Larochelle, and R. P. Adams, in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS’12 (Curran Associates Inc., USA, 2012) pp. 2951–2959.
- [54] J. Chung, P. Kannappan, C. Ng, and P. Sahoo, *Journal of Mathematical Analysis and Applications* **138**, 280 (1989).
- [55] G. B. Coleman and H. C. Andrews, *Proc IEEE* **67**, 773 (1979).
- [56] E. Magesan, M. Gambetta, and J. Emerson, *Phys Rev Lett* **106**, 180504 (2011).

---

## SUPPLEMENTARY INFORMATION

### Ising Hamiltonian of the Maxcut problem

Starting with the MAXCUT formulation (1) we can construct the Ising Hamiltonian connected to a given MAXCUT instance. To this end we note that we can lift a general graph  $G$  on  $n$ -nodes to a fully connected graph  $K_n$  by introducing missing edges and initializing their corresponding weights to zero. We assume that the weights  $w_{ij} = w_{ji}$  are symmetric, corresponding to an undirected graph and introduce Ising spin variables  $s_j \in \{-1, +1\}$  taking on value  $s_j = +1$  if  $v_j \in S$  and  $s_j = -1$  if  $v_j \in \bar{S}$ . With this we can express the cost of a cut as

$$\begin{aligned}
 w(\delta(S)) &= \sum_{i \in S, j \in \bar{S}} w_{ij} \\
 &= \frac{1}{2} \sum_{(i,j) \in \delta(S)} w_{ij} \\
 &= \frac{1}{4} \sum_{i,j \in V} w_{ij} - \frac{1}{4} \sum_{i,j \in V} w_{ij} s_i s_j \\
 &= \frac{1}{4} \sum_{i,j \in V} w_{ij} (1 - s_i s_j).
 \end{aligned} \tag{S1}$$

Identifying the spin variables with the spin operators of qubits yields the quantum analog of the weighted MAXCUT problem as

$$\hat{H}_C = -\frac{1}{2} \sum_{i,j \in V} w_{ij} (1 - \hat{\sigma}_i^z \hat{\sigma}_j^z) \tag{S2}$$

where we introduce the additional “–” sign to encode the optimal solution as the minimal energy state (as opposed to the maximal energy state from the MAXCUT prescription). In this sense we want to minimize  $\hat{H}_C$ .

### Detailed sampling procedure

The full optimization trace of running the Bayesian optimized MAXCUT clustering algorithm is shown in Fig. S1, where the abscissa shows the step count of the optimizer. Each violin in the top panel shows the kernel-density estimates (KDE) of the cost distribution associated with the sampled bit-strings at the corresponding step. The width reflects the frequency with which a given cost has been sampled, while the thick and thin line within each violin indicate the  $1\sigma$  and  $2\sigma$  intervals of the distribution, respectively. To indicate the extreme values we cut off the KDE at the extreme values of the sample. Finally the white dot at the center of the violins show the mean value of the sampled distribution. In the optimization procedure we return the largest value of the cost distribution. The middle panel shows the best sampled value at step  $i$  (red curve) corresponding to the extreme value of the distributions in the top panel, whereas the green curve is the mean value. The blue curve is the historic best value of the optimizer and shows the construction of the individual trace curves of Fig. 4. Finally the lowest panel shows the behavior of the Bayesian optimizer in choosing the next hyperparameter pair  $(\gamma, \beta)$ . The jumpiness of the angle choices is likely due to the noise in the 19Q chip and seems significantly reduced in a simulated experiment as seen in Fig. S2d.

As we can see, there is some variability in the mean value as well as the width of the distributions. At certain  $(\beta, \gamma)$  points we do indeed sample large cost values from the distribution corresponding to (approximate) solutions of the randomly chosen MAXCUT problem instance.

### Clustering on a fully connected graph

To demonstrate the clustering properties of the algorithm we simulate a larger MAXCUT problem instance on the Quantum Virtual Machine (QVM). We construct the distance matrix shown in Fig. S2a resulting from the Euclidean distance between 20 random points in  $\mathbb{R}^2$  as shown in Fig. S2b. The corresponding graph is fully connected and the label assignment corresponding to its MAXCUT solution is shown in Fig. S2b. It is worth pointing out that this is a

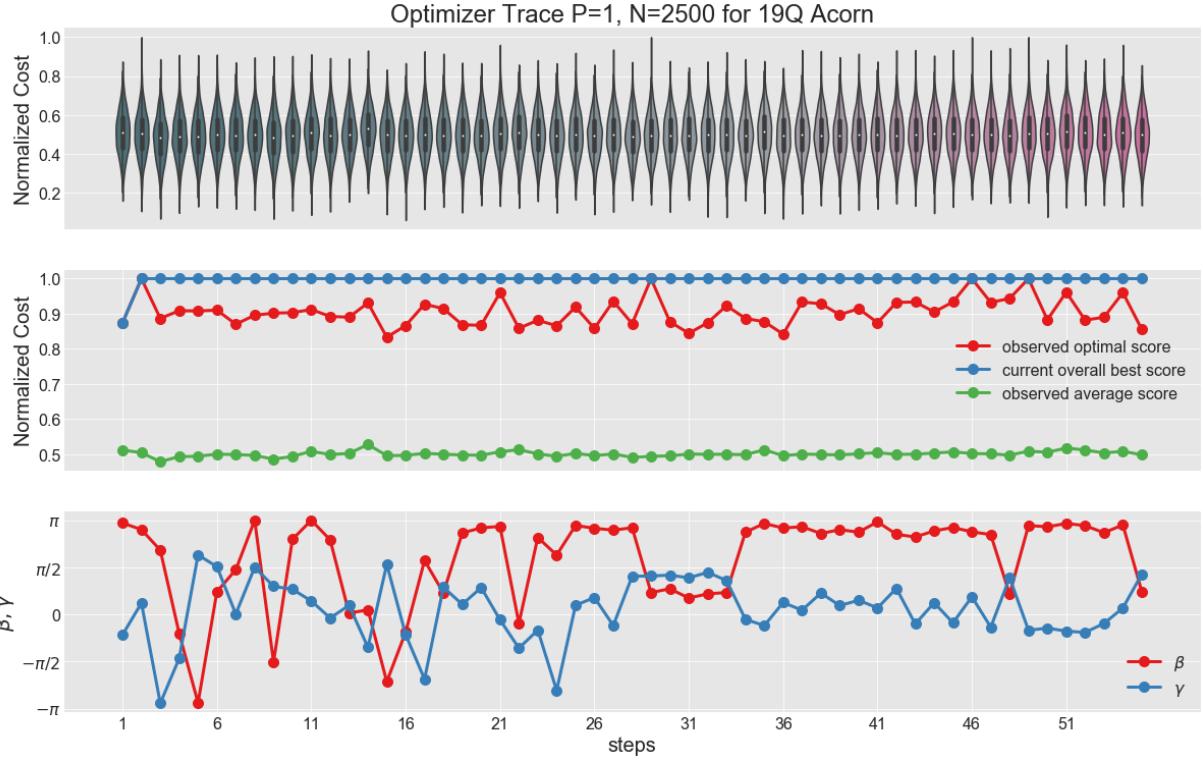


FIG. S1. Trace of the Bayesian Optimization of a  $p = 1$  step QAOA procedure for the 19 qubit MAXCUT problem instance with random weights as discussed in the main text. Each violin contains 2500 samples drawn from the QPU and is cut off at its observed extreme values. We normalized the plot to indicate the best possible value. Note that this is intractable in general (i.e., it requires knowledge of the true optimum, which is hard to obtain). Detailed descriptions are in the text.

bipartite graph and has only two equally optimal solutions, the one shown and the exact opposite coloring. Hence randomly sampling bit-strings only has a chance of  $2/2^{20} \approx 2 \cdot 10^{-6}$  of finding an optimal solution, meaning we would have to sample on the order of  $2^{19}$  bit-strings to find the correct answer with significant success probability. The corresponding optimization trace is shown in Fig. S2c. Each violin contains  $N = 250$  samples, and hence we sample only  $250/2^{20} \approx 0.02$  of the full state space at each point corresponding to a chance  $250 \cdot 2/2^{20} \approx 4.7 \cdot 10^{-4}$  to sample the right bit-string. This corresponds to a  $100\times$  improvement of the sampling procedure given a correctly prepared

---

#### Algorithm 1: MAXCUT bi-clustering

---

**Data:** Dataset  $\mathcal{D}$  of points  $p_i, i = 1, \dots, N$   
**Result:** bi-clustering assignments for the dataset  $\mathcal{D}$  into  $S$  and  $\bar{S}$

```

for  $i \leftarrow 1$  to  $N$  do
  for  $j \leftarrow 1$  to  $N$  do
    |  $C_{ij} \leftarrow \text{Distance}(p_i, p_j)$ 
  end
end
for  $i \leftarrow 1$  to  $N$  do
  if  $\text{bitString}[i] == 1$  then
    |  $\mathcal{D}_S.append(p_i)$ 
  else
    |  $\mathcal{D}_{\bar{S}}.append(p_0)$ 
  end
end
return  $\mathcal{D}_S, \mathcal{D}_{\bar{S}}$ ;

```

---

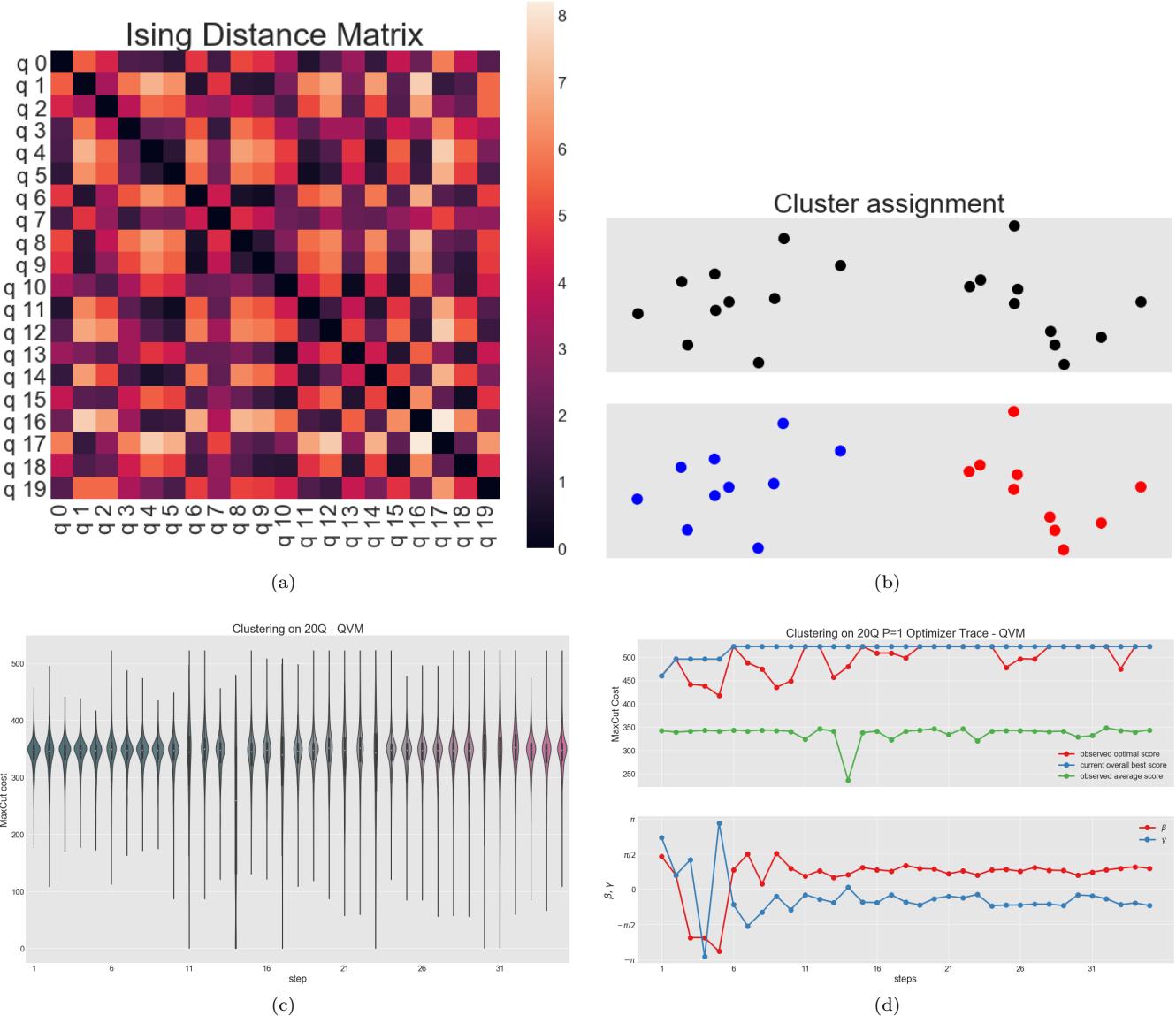


FIG. S2. (a) Euclidean distance matrix for a sample of 20 points as shown in Fig. S2b before labels have been assigned. This matrix will be used as the adjacency matrix for the MAXCUT clustering algorithm. (b) Random sample of 20 two-dimensional points forming two visually distinguishable clusters. The color assignment is the result of the clustering algorithm 1. Calculating the mutual Euclidean distances for all points gives rise to the distance matrix shown in Fig. S2a. (c) QAOA optimization trace for a fully connected 20 node graph corresponding to the distance matrix in Fig. S2a. The parameters are  $p = 1$  and each violin contains  $N = 250$  samples, i.e. we sample  $250/2^{20} \approx 0.02$  of the whole state space. This demonstrates that the algorithm is capable of finding good solutions even for a non-trivial instance. (d) QAOA optimization trace for a fully connected 20 node graph corresponding to the distance matrix in Fig. S2a. The parameters are  $p = 1$  and each violin contains  $N = 250$  samples. This demonstrates that the algorithm is capable of finding good solutions even for a non-trivial instance.

distribution as compared to just sampling from a uniform distribution. We can see that due to the fully connected nature of the graph the variation in the mean is not significant for  $p = 1$  steps in the QAOA iteration (see Fig. S2d for more details). However, there is significant variations in the standard deviation of the sampled distributions with only a few of them allowing access to the optimal value with so few samples. A better view of the optimizer trace is shown in Fig. S2d where we plot the average and best observed cost at each  $(\beta, \gamma)$ -pair in addition to the overall best value observed at the time a new point is evaluated. We can see that the optimizer slowly improves its best value and that it increasingly samples from distributions with large standard deviations. The clustering steps are described in pseudo-code by algorithm 1

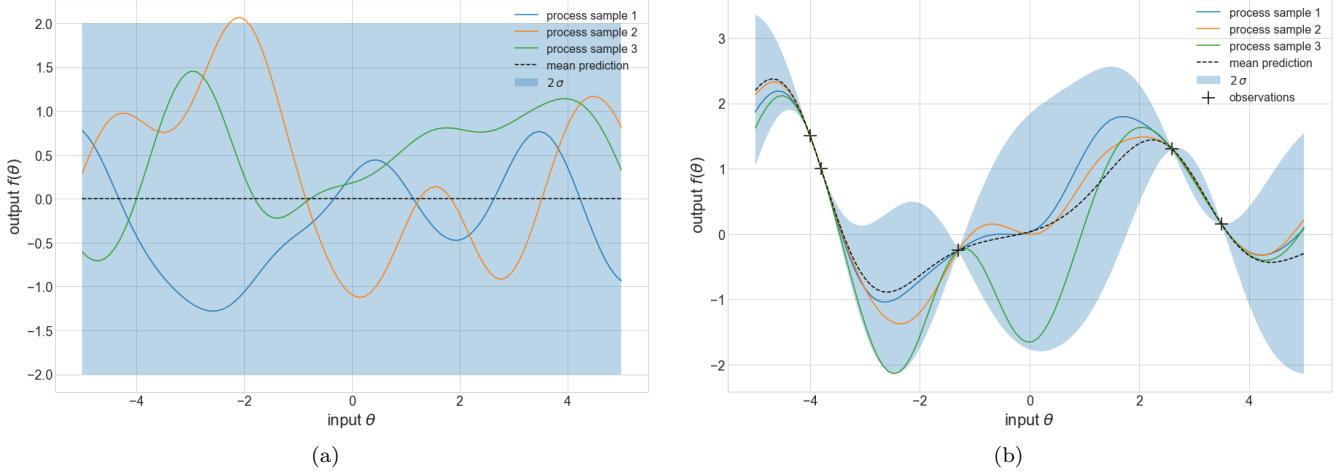


FIG. S3. Update process of a Bayesian Optimization procedure. In both plots the blue area indicate a  $2\sigma$  interval of the GP distribution. (a) GP distribution prior; without any updates we can draw any of the three curves which more or less are within the two-standard deviation of the prior. (b) Evaluating the true value of the random variable will *clamp* the GP distribution at the observation points, forcing us to update the priors according to Bayes' rule. Drawing values from the posterior GP distribution will hence have a reduced overall uncertainty and adjusted mean value.

### Construction of the optimization functional

We can interpret the cost associated with the Ising Hamiltonian  $\hat{H}_C$  corresponding to a classical combinatorial optimization problem (COP) as a function

$$f : \mathbb{Z}_2^n \times [0, 2\pi)^{2p} \rightarrow \mathbb{R}, f(x; \theta) \mapsto \langle x | \hat{H}_C | x \rangle, \quad (\text{S3})$$

where  $|x\rangle \in \mathbb{Z}_2^n$  is a classic bit-string sampled from the distribution  $\mathcal{D}_x$  with PDF  $p(x; \theta) = |\psi(x; \theta)|^2$  prepared by running the QAOA procedure as described in the main text and  $\theta = (\gamma, \beta)$ . We can now identify the bit-string variable  $|x\rangle$  as a random variable  $X \sim \mathcal{D}_x(\theta)$  drawn from the parameterized distribution  $\mathcal{D}_x(\theta) \subseteq \mathbb{Z}_2^n$  and consequently can interpret the function  $f$  as a random variable  $F \sim \mathcal{D}_f(\theta) \subseteq \mathbb{R}$  by functional composition with a PDF  $p_f(v; \theta)$  and CDF  $C_f(v; \theta)$ .

In order to optimize the distribution  $\mathcal{D}_f(\theta)$  for sampling (nearly) optimal bit-strings we need to define an optimization objective. To this end, we consider the  $j$ -th order statistics  $F_{(j)}$  [51] of an i.i.d. sample of  $N$  experiments  $\{F_i\}_{i=1\dots N}$

$$\begin{aligned} F_{(1)} &= \min\{F_1, \dots, F_N\} \\ F_{(2)} &= \min\{F_1, \dots, F_N\} \setminus \{F_{(1)}\} \\ &\vdots \\ F_{(N-1)} &= \max\{F_1, \dots, F_N\} \setminus \{F_{(N)}\} \\ F_{(N)} &= \max\{F_1, \dots, F_N\}. \end{aligned} \quad (\text{S4})$$

By definition we have  $F_{(1)} \leq F_{(2)} \leq \dots \leq F_{(N)}$ . Notice that the  $F_{(j)}$  are random variables as well, but due to the ordering relation are not i.i.d. anymore. We can use the order statistic to define the  $k$ -th percentile ( $k = 1, \dots, 100$ ) as the CDF of  $F_{(\lfloor 100j/N \rfloor = k)}$ . The optimization routine can now be used to optimize the expectation values of any order statistic at hand. To this end we need to compute the PDF of these order statistics [51] according to

$$p_{F_{(j)}}(v; \theta) = N \binom{N-1}{j-1} p_f(v; \theta) C_f(v; \theta)^{j-1} (1 - C_f(v; \theta))^{N-j} \quad (\text{S5})$$

In practice we are mostly interested in optimizing the extreme value statistic, i.e. minimizing the first-order or maximizing the  $N$ -th order statistic. The expectation values of these can be computed as

---


$$s_1(\theta) := \langle F_{(1)}(\theta) \rangle = N \int dv v p_f(v; \theta) (1 - C_f(v; \theta))^{N-1} \quad (\text{S6})$$

and

$$s_N(\theta) := \langle F_{(N)}(\theta) \rangle = N \int dv v p_f(v; \theta) C_f(v; \theta)^{N-1} \quad (\text{S7})$$

Note that this approach also enables us to estimate the uncertainty of these random variables, giving quality estimates of the sample. Despite looking horribly complicated, those values can readily be computed numerically from a set of samples of the distribution  $\mathcal{D}_f(\theta)$ . A pseudo-code representation of the statistics calculation is given in algorithm 2.

#### Gaussian Process description of the extreme value optimization

The extreme value functions  $s_j(\theta) : [0, 2\pi]^{2p} \rightarrow \mathbb{R}, j = 1, N$  are generally not analytically known and typically expensive to evaluate. Hence we have access to  $s(\theta)$  only through evaluating it on a set of  $m$  points  $\theta_{1:m}$  [52] with corresponding variables  $v_i = s(\theta_i)$  and noisy observations  $y_{1:m}$ . Note that we drop the subscript  $j$  to denote the distinction between minimal and maximal value functions as the approach is identical for either of them. To make use of Bayesian optimization techniques [32, 53] we assume that the variables  $\mathbf{v} = v_{1:m}$  are jointly Gaussian and that the observations  $y_{1:m}$  are normally distributed given  $\mathbf{v}$ , completing the construction of a Gaussian process (GP)[18]. The distinction between the variable  $v_i$  and its observation  $y_i$  is important given that the expectation values S6 and S7 are subject to sampling noise, due to finite samples, but also due to finite gate and readout fidelities and other experimental realities, and hence cannot be known exactly. We describe the GP using the moments of a multivariate normal distribution

$$m(\theta) = \mathbb{E}[v(\theta)] \quad (\text{S8})$$

$$k(\theta, \theta') = \mathbb{E}[(v(\theta) - \mu(\theta))(v(\theta') - \mu(\theta'))] \quad (\text{S9})$$

and introduce the notation

$$v(\theta) \sim \mathcal{GP}(m(\theta), k(\theta, \theta')) \quad (\text{S10})$$

This result summarizes the basic assumption that the extreme value statistic of *the cost of best sampled bit-string* follows a Gaussian distribution as a function of the parameters  $\theta$ . While this might not be true, it proves to be a good choice for the unbiased prior in practice. To use the GP we need to specify the kernel function  $k(\theta, \theta')$ , which specifies a measure of correlation between two observation points  $\theta$  and  $\theta'$ . Following [53] we use the Matérn-2.5 kernel in the following, but there are many alternative choices [18, 32].

The basic idea of Bayesian Optimization is to draw new observations  $y_*$  at points  $\theta_*$  of the GP random variable  $v$  and iteratively use Bayes' rule to condition the prior of  $v$  on  $y_*$ . This conditioning is particularly simple for GPs as it can be performed analytically using the properties of multivariate normal distributions.

---

#### Algorithm 2: Statistics Calculation of Bit-String Distribution

---

```

Data: QAOA angle parameters  $\theta$ 
Result: value of a statistic  $s(\theta)$  sampled from  $\mathcal{D}_f(\theta)$ 

CostValues  $\leftarrow$  empty List;
for  $i \leftarrow 1$  to  $N$  do
| bitString  $\leftarrow$  Sample( $\mathcal{D}_x(\theta)$ );
| cost  $\leftarrow$  Cost(bitString);
| CostValues.append(cost);
end
// calculate the statistic of interest
 $s(\theta) \leftarrow$  Statistic(CostValues);
return  $s(\theta);$ 
```

---

To account for imperfect/noisy evaluations of the true underlying function we simply need to adjust the update rule for the GP mean and kernels. This can also be done analytically and hence be directly applied to our case of numerically sampled extreme values statistics. For exhaustive details of the update rules, kernels and general GP properties see Ref. [18]. It should be noted that the updated rules of Gaussian kernels require a matrix inversion which scales as  $\mathcal{O}(m^3)$  and hence can become prohibitively expensive when the number  $m$  of samples becomes large. Improving this scaling is an active area of research and early promising results such as Deep Networks for Global Optimization (DNGO) [33] provide surrogate methods for Gaussian processes with linear scaling.

So far we have only been concerned with updating the GP to reflect the incorporation of new knowledge. To close the optimization loop we need a procedure to select the next sampling point  $\theta_*$ . Choosing a point at random will essentially induce a random walk over the optimization parameter domain and might not be very efficient [46] (though this is still better than a grid search). A nice improvement over this random walk is offered by Bayesian framework itself: due to the constantly updating the posterior distribution we can estimate the areas of highest uncertainty and consequently chose the next point accordingly. However, this might bring the optimizer far away from the optimal point by trying to minimize the global uncertainty. To prevent the optimizer from drifting off we need a way to balance its tendency to explore areas of high uncertainty with exploiting the search around the currently best known value. This procedure is encapsulated in the acquisition function  $\alpha(\theta; \mathcal{D}_m)$  where  $\mathcal{D}_m = \{(\theta_i, y_i)\}_{i=1,\dots,m}$  is the set of observations up to iteration  $m$  [32]. Since the posterior is Gaussian for every point  $\theta$  there are many analytic ways to construct an acquisition function. Here, we use the *Upper Confidence Bound* (UCB) metric, which can be calculated as

$$\alpha_{\text{UCB}}(\theta; \mathcal{D}_m) = \mu_m(\theta) + \beta_m \sigma_m(\theta) \quad (\text{S11})$$

where  $\beta_m$  is a hyperparameter controlling the explore-exploit behavior of the optimizer and  $\mu_m(\theta), \sigma_m(\theta)$  are the mean and variance of the Gaussian of the posterior GP restricted to point  $\theta$ . Maximizing the acquisition function of all values  $\theta$  at each iteration step yields the next point for sampling from the unknown function  $s$ . For more details see Ref. [32]. A pseudo-code representation of the Bayesian Optimization routine is given in algorithm 3.

### Comparison to Random Sampling

To demonstrate the applicability of the results beyond a single problem instance we ran the simulations on 5 randomly chosen problem instances over a fourteen hour window on 19Q architecture. We recorded the optimization traces (cf. Fig. S4a) and calculated the empirical CDF (eCDF) for the time-to-optimum, i.e. the number of steps before the optimizer reached the optimal value, as seen in Figs. 5, S4b. Note that we can estimate the optimal value easily for the problem at hand. We compared the empirical CDF (eCDF) to the CDF of a random sampling procedure that follows a Bernoulli distribution  $\mathcal{B}(N, p)$  with a success probability  $p = 2/2^{19}$  and  $N = N_{\text{steps}} N_{\text{shots}}$  samples. The additional factor 2 in the success probability is due to the inversion symmetry of the solution, i.e. there are two equivalent solutions which minimize the cost and are related to each other by simply inverting each bit-assignment. The CDF for the Bernoulli random variable can then be easily written as:

---

#### Algorithm 3: Bayesian Optimization of QAOA Extreme Value Statistics

---

```

Data: statistics function  $f_s$  and parameter range  $\mathcal{R}_\theta$ 
Result: optimal statistic  $f_s(\theta_{\text{opt}})$ 

GP_Dist  $\leftarrow$  InitPrior;
bestVal  $\leftarrow$  void;
for  $i \leftarrow 1$  to N_iter do
    nextTheta  $\leftarrow$  SampleNextPoint(GP_Dist);
    // calc. statistic with alg. 2
    curVal  $\leftarrow$  Statistic(nextTheta);
    if curVal > bestVal then
        bestVal  $\leftarrow$  curVal
    end
    GP_Dist  $\leftarrow$  Update(GP_Dist, curVal, nextTheta)
end
return bestVal;
```

---

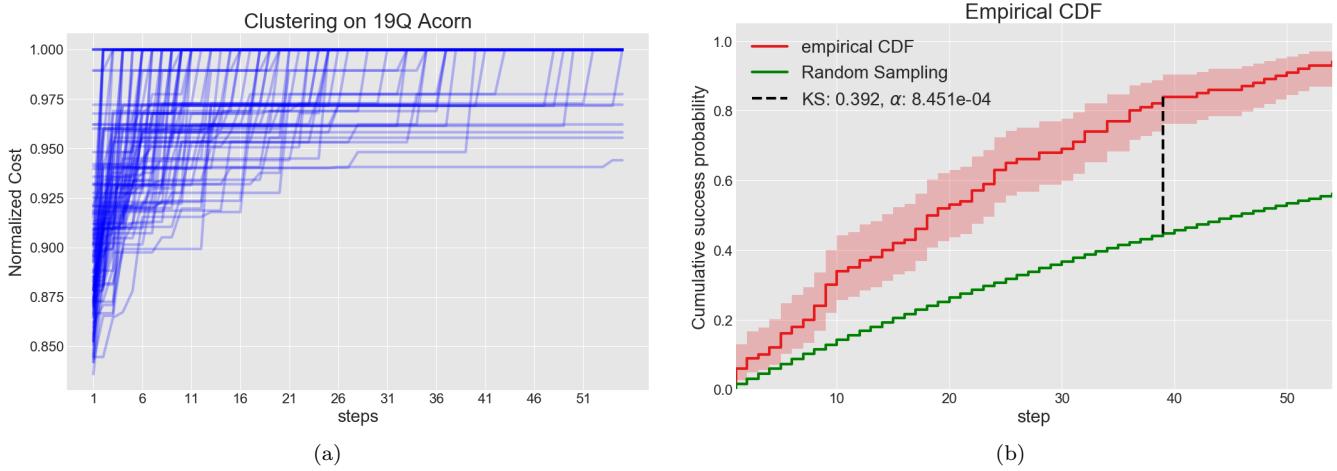


FIG. S4. (a) Traces for the normalized MAXCUT cost for 83 independent runs of the algorithm on the 19Q chip for a fixed random problem instances of Fig. 3. Notice that most traces reach the optimal cost well before the cutoff at 55 steps. (b) The performance of our implementation of the clustering algorithm (red) can be compared to the performance of an algorithm that simply draws cluster assignments at random (green). It is clear that our algorithm generates the optimal assignment much more quickly than it would be expected by chance: the 95% confidence region for our empirical observations have very small overlap for the distribution given by random assignments, and the Kolmogorov-Smirnov statistic indicates we can reject the null hypothesis of random assignments at a level higher than 99.9%.

TABLE S1. Kolmogorov-Smirnov statistics and significance values for the CDF shown in Fig. 5. All values are calculated with respect to the exact random sampling CDF

	eCDF	KS	$\alpha$
empirical random bitstring sampling (Fig. 5)	0.077		1.559
Rigetti-QVM (Fig. 5)	0.838	$1.273 \cdot 10^{-7}$	
19Q single instance (Fig. 5)	0.339	$1.339 \cdot 10^{-2}$	
19Q randomized instances (Fig. S4b)	0.392	$8.4451 \cdot 10^{-4}$	

$$P(\text{success after } k \text{ steps}) = 1 - (1-p)^{k \cdot N_{\text{shots}}} \quad (\text{S12})$$

To compare the eCDF (red curve in Fig. S4b) to the random sampling CDF (green curve) we calculate the Kolmogorov-Smirnov statistic between two eCDFs

$$KS_{n,m} = \sup_x |F_{1,n}(x) - F_{2,m}(x)| \quad (\text{S13})$$

where  $F_{1,n}$  is the first eCDF with  $n$  points and  $F_{2,m}$  is the second one with  $m$  points. Given the eCDFs of Fig. S4b we find  $KS_{23,55} \approx 0.392$ . We can calculate the significance level  $\alpha$  by inverting the prescription for rejection of the Null-Hypothesis  $H_0$ , i.e. the two eCDFs result from the same underlying distribution function:

$$KS_{n,m} \geq c(\alpha) \sqrt{\frac{n+m}{nm}} \quad (\text{S14})$$

where  $c(\alpha) = \sqrt{-0.5 \log(\alpha/2)}$ . Plugging in the empirical KS statistic we find that  $H_0$  can be rejected with a probability  $p = 1 - \alpha$  with  $\alpha = 8.451 \cdot 10^{-4}$ . We also calculated the KS statistic for the curves in the main body of the text summarized in Table. S1

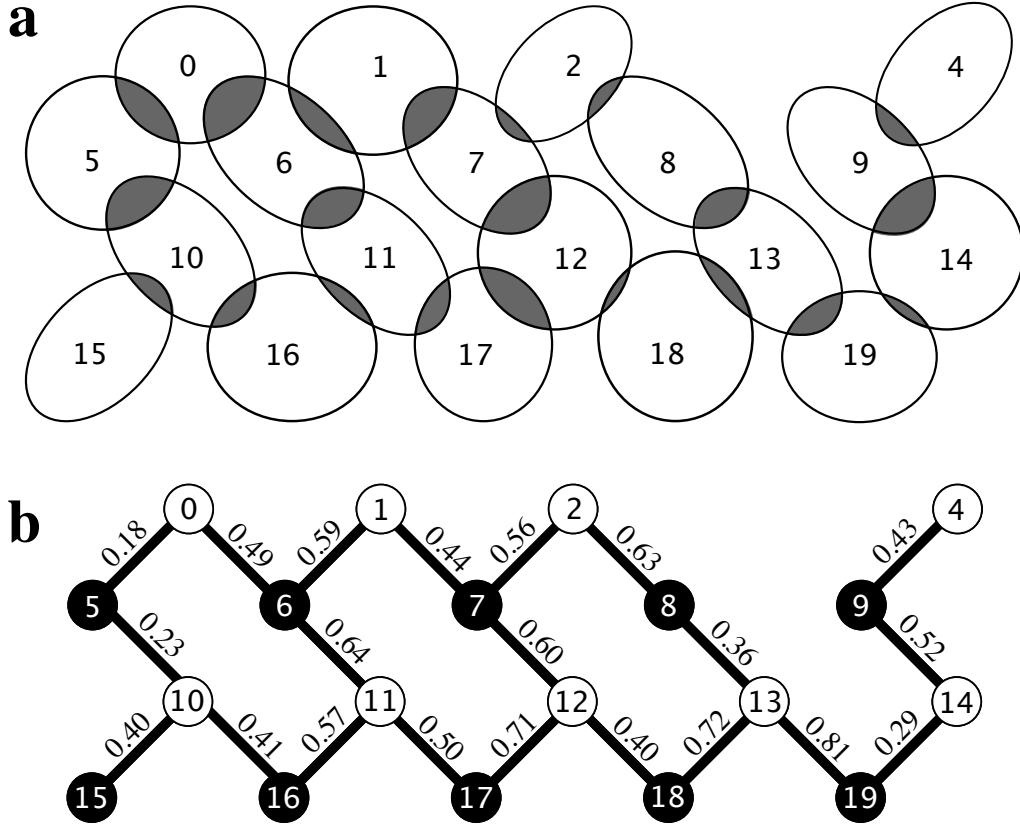


FIG. S5. Pictorial representation of a set of compact distributions in  $\mathbb{R}^2$ . Calculating the mutual Bhattacharyya coefficient between the distributions leads to a graph that maps to the Rigetti 19Q architecture with low overhead. Applying the proposed clustering algorithm will find assignments for sets of the least overlapping distributions. For more details see the body of the appendix.

#### Example of a clustering application for the 19Q architecture

To motivate the clustering application we look at Fig. S5. It shows 19 distribution with compact support in  $\mathbb{R}^2$  with finite overlaps. We can define a similarity metric between these distributions using the Bhattacharyya coefficient [43, 54, 55]

$$b : \mathcal{D}_{\mathbb{R}^2} \times \mathcal{D}_{\mathbb{R}^2} \longrightarrow \mathbb{R}, \quad b(p, q) = \int \sqrt{p(x)q(x)} d^2x \quad (\text{S15})$$

where we use  $\mathcal{D}_{\mathbb{R}^2}$  to denote the space of compact-support distributions over  $\mathbb{R}^2$ . Since the Bhattacharyya coefficient is not a typical distance metric –it does not fulfill the triangle inequality–, we use the following procedure: Using the example of Fig. S5 we can identify the individual distributions with a given qubit and calculate the overlap metric with all other distributions. With this we can construct a graph  $G = (V, E)$  where the vertices correspond to identifiers of the individual distributions and the edges to overlaps in these distributions where the weights are given by the Bhattacharyya coefficient. In the case of Fig. S5 this will lead to a graph that has a low overhead when mapped to the Rigetti 19Q connectivity and enables us to run the clustering algorithm on the quantum hardware. To make this translation we need to remove the self-similarity between the distributions, corresponding to self-cycles in the graph. It should be noted that clustering in this context means to identify a set of distribution that are *as dissimilar as possible*, i.e. have as little overlap as possible.

#### Detailed device parameters

Tables S2 and S3 summarize the main performance parameters of Rigetti 19Q. Single-qubit gate fidelities are estimated with standard randomized benchmarking protocols [56] with 25 random Clifford gate sequences of lengths

TABLE S2. **Rigetti 19Q performance parameters** — All of the parameters listed in this table have been measured at base temperature  $T \approx 10$  mK. The reported  $T_1$ 's and  $T_2^*$ 's are averaged values over 10 measurements acquired at  $\omega_{01}^{\max}$ . The errors indicate the standard deviation of the averaged value. Note that these estimates fluctuate in time due to multiple factors.

	$\omega_r^{\max}/2\pi$	$\omega_{01}^{\max}/2\pi$	$\eta/2\pi$	$T_1$	$T_2^*$	$\mathcal{F}_{1q}$	$\mathcal{F}_{RO}$
	MHz	MHz	MHz	$\mu s$	$\mu s$		
0	5592	4386	-208	<b>15.2</b> $\pm$ 2.5	<b>7.2</b> $\pm$ 0.7	0.9815	0.938
1	5703	4292	-210	<b>17.6</b> $\pm$ 1.7	<b>7.7</b> $\pm$ 1.4	0.9907	0.958
2	5599	4221	-142	<b>18.2</b> $\pm$ 1.1	<b>10.8</b> $\pm$ 0.6	0.9813	0.970
3	5708	3829	-224	<b>31.0</b> $\pm$ 2.6	<b>16.8</b> $\pm$ 0.8	0.9908	0.886
4	5633	4372	-220	<b>23.0</b> $\pm$ 0.5	<b>5.2</b> $\pm$ 0.2	0.9887	0.953
5	5178	3690	-224	<b>22.2</b> $\pm$ 2.1	<b>11.1</b> $\pm$ 1.0	0.9645	0.965
6	5356	3809	-208	<b>26.8</b> $\pm$ 2.5	<b>26.8</b> $\pm$ 2.5	0.9905	0.840
7	5164	3531	-216	<b>29.4</b> $\pm$ 3.8	<b>13.0</b> $\pm$ 1.2	0.9916	0.925
8	5367	3707	-208	<b>24.5</b> $\pm$ 2.8	<b>13.8</b> $\pm$ 0.4	0.9869	0.947
9	5201	3690	-214	<b>20.8</b> $\pm$ 6.2	<b>11.1</b> $\pm$ 0.7	0.9934	0.927
10	5801	4595	-194	<b>17.1</b> $\pm$ 1.2	<b>10.6</b> $\pm$ 0.5	0.9916	0.942
11	5511	4275	-204	<b>16.9</b> $\pm$ 2.0	<b>4.9</b> $\pm$ 1.0	0.9901	0.900
12	5825	4600	-194	<b>8.2</b> $\pm$ 0.9	<b>10.9</b> $\pm$ 1.4	0.9902	0.942
13	5523	4434	-196	<b>18.7</b> $\pm$ 2.0	<b>12.7</b> $\pm$ 0.4	0.9933	0.921
14	5848	4552	-204	<b>13.9</b> $\pm$ 2.2	<b>9.4</b> $\pm$ 0.7	0.9916	0.947
15	5093	3733	-230	<b>20.8</b> $\pm$ 3.1	<b>7.3</b> $\pm$ 0.4	0.9852	0.970
16	5298	3854	-218	<b>16.7</b> $\pm$ 1.2	<b>7.5</b> $\pm$ 0.5	0.9906	0.948
17	5097	3574	-226	<b>24.0</b> $\pm$ 4.2	<b>8.4</b> $\pm$ 0.4	0.9895	0.921
18	5301	3877	-216	<b>16.9</b> $\pm$ 2.9	<b>12.9</b> $\pm$ 1.3	0.9496	0.930
19	5108	3574	-228	<b>24.7</b> $\pm$ 2.8	<b>9.8</b> $\pm$ 0.8	0.9942	0.930

$l \in \{2, 4, 8, 16, 32, 64, 128\}$ . Readout fidelity is given by the assignment fidelity  $\mathcal{F}_{RO} = [p(0|0) + p(1|1)]/2$ , where  $p(b|a)$  is the probability of measuring the qubit in state  $b$  when prepared in state  $a$ . Two-qubit gate fidelities are estimated with quantum process tomography [45] with preparation and measurement rotations  $\{\mathbb{I}, R_x(\pi/2), R_y(\pi/2), R_z(\pi)\}$ . The reported process fidelity  $\mathcal{F}_{2q}$  indicates the average fidelity between the ideal process and the measured process imposing complete positivity and trace preservation constraints. We further averaged over the extracted  $\mathcal{F}_{2q}$  from four separate tomography experiments. Qubit-qubit coupling strengths are extracted from Ramsey experiments with and without  $\pi$ -pulses on neighboring qubits.

TABLE S3. **Rigetti 19Q two-qubit gate parameters and performance** — These parameters refer to two-qubit interactions of Rigetti 19Q. Qubit 3 is not tunable and for this reason parameters related to the pairs 3 – 8, 3 – 9 are not included.

	$A_0$	$f_m$	$t_{CZ}$	$\mathcal{F}_{2q}$
	$\Phi/\Phi_0$	MHz	ns	
0 – 5	0.27	94.5	168	0.936
0 – 6	0.36	123.9	197	0.889
1 – 6	0.37	137.1	173	0.888
1 – 7	0.59	137.9	179	0.919
2 – 7	0.62	87.4	160	0.817
2 – 8	0.23	55.6	189	0.906
4 – 9	0.43	183.6	122	0.854
5 – 10	0.60	152.9	145	0.870
6 – 11	0.38	142.4	180	0.838
7 – 12	0.60	241.9	214	0.870
8 – 13	0.40	152.0	185	0.881
9 – 14	0.62	130.8	139	0.872
10 – 15	0.53	142.1	154	0.854
10 – 16	0.43	170.3	180	0.838
11 – 16	0.38	160.6	155	0.891
11 – 17	0.29	85.7	207	0.844
12 – 17	0.36	177.1	184	0.876
12 – 18	0.28	113.9	203	0.886
13 – 18	0.24	66.2	152	0.936
13 – 19	0.62	109.6	181	0.921
14 – 19	0.59	188.1	142	0.797