ARTICLE   OPEN

Sampling frequency thresholds for the quantum advantage of the quantum approximate optimization algorithm

Danylo Lykov [1,2✉], Jonathan Wurtz[3,4], Cody Poole[5], Mark Saffman [5,6], Tom Noel [7] and Yuri Alexeev [1]

We compare the performance of the Quantum Approximate Optimization Algorithm (QAOA) with state-of-the-art classical solvers Gurobi and MQLib to solve the MaxCut problem on 3-regular graphs. We identify the minimum noiseless sampling frequency and depth p required for a quantum device to outperform classical algorithms. There is potential for quantum advantage on hundreds of qubits and moderate depth with a sampling frequency of 10 kHz. We observe, however, that classical heuristic solvers are capable of producing high-quality approximate solutions in linear time complexity. In order to match this quality for large graph sizes N, a quantum device must support depth $p > 11$. Additionally, multi-shot QAOA is not efficient on large graphs, indicating that QAOA $p \leq 11$ does not scale with N. These results limit achieving quantum advantage for QAOA MaxCut on 3-regular graphs. Other problems, such as different graphs, weighted MaxCut, and 3-SAT, may be better suited for achieving quantum advantage on near-term quantum devices.

INTRODUCTION

Quantum computing promises enormous computational powers that can far outperform any classical computational capabilities[1]. In particular, certain problems can be solved much faster compared with classical computing, as demonstrated experimentally by Google for the task of sampling from a quantum state[2]. Thus, an important milestone[2] in quantum technology, so-called 'quantum supremacy', was achieved as defined by Preskill[3].

The next milestone, 'quantum advantage', where quantum devices solve useful problems faster than classical hardware, is more elusive and has arguably not yet been demonstrated. However, a recent study suggests a possibility of achieving a quantum advantage in runtime over specialized state-of-the-art heuristic algorithms to solve the Maximum Independent Set problem using Rydberg atom arrays[4]. Common classical solutions to several potential applications for near-future quantum computing are heuristic and do not have performance bounds. Thus, proving the advantage of quantum computers is far more challenging[5–7]. Providing an estimate of how quantum advantage over these classical solvers can be achieved is important for the community and is the subject of this paper.

Most of the useful quantum algorithms require large fault-tolerant quantum computers, which remain far in the future. In the near future, however, we can expect to have noisy intermediate-scale quantum (NISQ) devices[8]. In this context, variational quantum algorithms (VQAs) show the most promise[9] for the NISQ era, such as the variational quantum eigensolver (VQE)[10] and the Quantum Approximate Optimization Algorithm (QAOA)[11]. Researchers have shown remarkable interest in QAOA because it can be used to obtain approximate (i.e., valid but not optimal) solutions to a wide range of useful combinatorial optimization problems[4,12,13].

In opposition, powerful classical approximate and exact solvers have been developed to find good approximate solutions to combinatorial optimization problems. For example, a recent work by Guerreschi and Matsuura[5] compares the time to solution of QAOA vs. the classical combinatorial optimization suite AKMAX-SAT. The classical optimizer takes exponential time with a small prefactor, which leads to the conclusion that QAOA needs hundreds of qubits to be faster than classical. This analysis requires the classical optimizer to find an exact solution, while QAOA yields only approximate solutions. However, modern classical heuristic algorithms are able to return an approximate solution on demand. Allowing for worse-quality solutions makes these solvers extremely fast (on the order of milliseconds), suggesting that QAOA must also be fast to remain competitive. A valid comparison should consider both solution quality and time.

In this way, the locus of quantum advantage has two axes, as shown in Fig. 1: to reach advantage, a quantum algorithm must be both faster and return better solutions than a competing classical algorithm (green, top right). If the quantum version is slower and returns worse solutions (red, bottom left), there is clearly no advantage. However, two more regions are shown in the figure. If the QAOA returns better solutions more slowly than a classical algorithm (yellow, top left), then we can increase the running time for the classical version. It can try again and improve its solution with more time. This is a crucial mode to consider when assessing advantage: heuristic algorithms may always outperform quantum algorithms if the quantum time to solution is slow. Alternatively, QAOA may return worse solutions faster (yellow, bottom right), which may be useful for time-sensitive applications. In the same way, we may stop the classical algorithm earlier, and the classical solutions will become worse.

One must keep in mind that the reason for using a quantum algorithm is the scaling of its time to solution with the problem size N. Therefore, a strong quantum advantage claim should

[1]Computational Science Division, Argonne National Laboratory, 9700 S. Cass Ave., Lemont, IL 60439, USA. [2]Department of Computer Science, The University of Chicago, Chicago, IL 60637, USA. [3]Department of Physics and Astronomy, Tufts University, Medford, MA 02155, USA. [4]QuEra Computing Inc., Boston, MA 02135, USA. [5]Department of Physics, University of Wisconsin - Madison, Madison, WI 53706, USA. [6]ColdQuanta, Inc., 612 W. Main St., Madison, WI 53703, USA. [7]ColdQuanta, Inc., 3030 Sterling Circle, Boulder, CO 80301, USA. ✉email: dlykov@anl.gov
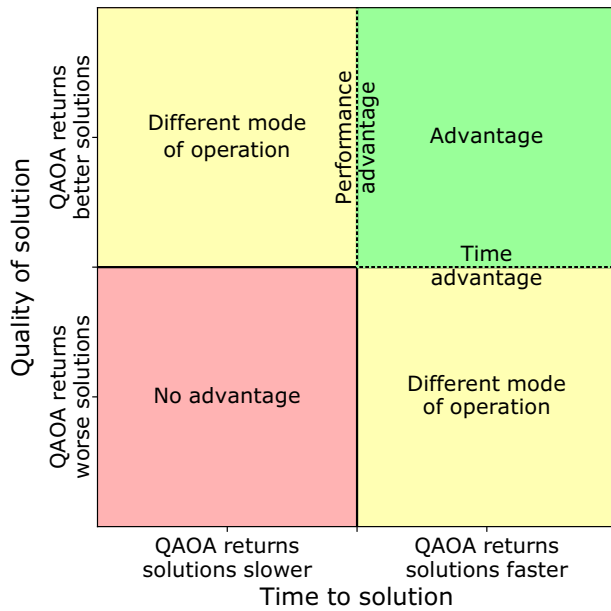
Fig. 1 **Locus of quantum advantage over classical algorithms.** A particular classical algorithm may return some solution to some ensemble of problems in time T_C (horizontal axis) with some quality C_C (vertical axis). Similarly, a quantum algorithm may return a different solution sampled in time T_Q, which may be faster (right) or slower (left) than classical, with a better (top) or worse (bottom) quality than classical. If QAOA returns better solutions faster than the classical ones, then there is a clear advantage (top right), and conversely, no advantage for worse solutions slower than the classical ones (bottom left).

Fig. 2 **Time required for a single-shot QAOA to match classical MaxCut algorithms.** The blue line shows the time for comparing with the Gurobi solver and using $p = 11$; the yellow line shows a comparison with the FLIP algorithm and $p = 6$. Each quantum device that runs MaxCut QAOA can be represented on this plot as a point, where the x-axis is the number of qubits and the y-axis is the time to solution. For any QAOA depth p, the quantum device should return at least one bitstring faster than the Y-value on this plot.

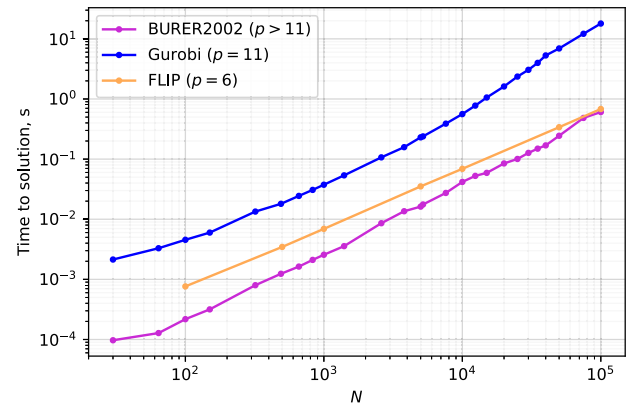demonstrate the superior performance of a quantum algorithm in the large-N limit.

This paper focuses on the MaxCut combinatorial optimization problem on 3-regular graphs for various problem sizes N. MaxCut is a popular benchmarking problem for QAOA because of its simplicity and straightforward implementation. We propose a fast fixed-angle approach to running QAOA that speeds up QAOA while preserving solution quality compared with slower conventional approaches. We evaluate the expectation value of noiseless QAOA solution quality using tensor network simulations on classical hardware. We then find the time required for classical solvers to match this expected QAOA solution quality. Surprisingly, we observe that even for the shortest possible time, the classical solution quality is above our QAOA solution quality for $p = 11$, our largest p with known performance. Therefore, we compensate for this difference in quality by using multishot QAOA and finding the number of samples K required to match the classical solution quality. K allows us to characterize quantum device parameters, such as sampling frequency, required for the quantum algorithm to match the classical solution quality.

RESULTS AND DISCUSSION

This section will outline the results and comparison between classical optimizers and QAOA. This has two halves: Section "Expected QAOA solution quality" outlines the results of the quantum algorithm and Section "Classical solution quality and time to solution" outlines the results of the classical competition.

Expected QAOA solution quality

The first algorithm is the quantum approximate optimization algorithm (QAOA), which uses a particular ansatz to generate approximate solutions through measurement. We evaluate QAOA

for two specific modes. The first is a single shot fixed angle QAOA, where a single solution is generated. This has the benefit of being very fast. The second generalization is multi-shot fixed angle QAOA, where many solutions are generated, and the best is kept. This has the benefit that the solution may be improved with increased run time.

In Section "QAOA performance," we find that one can put limits on the QAOA MaxCut performance even when the exact structure of a 3-regular graph is unknown using fixed angles. We have shown that for large N, the average cut fraction for QAOA solutions on 3-regular graphs converges to a fixed value f_{tree}. If memory limitations permit, we evaluate these values numerically using tensor network simulations. This gives us the average QAOA performance for any large N and $p \le 11$. To further strengthen the study of QAOA performance estimations, we verify that for the small N, the performance is close to the same value f_{tree}. We are able to numerically verify that for $p \le 4$ and small N, the typical cut fraction is close to f_{tree}, as shown in Fig. 6.

Combining the large-N theoretical analysis and small-N heuristic evidence, we are able to predict the average performance of QAOA on 3-regular graphs for $p \le 11$. We note that today's hardware can run QAOA up to $p \le 4$[4] and that for larger depths, the hardware noise prevents achieving better QAOA performance. Therefore, the $p \le 11$ constraint is not an important limitation for our analysis.

Classical solution quality and time to solution

The second ensemble of algorithms is classical heuristic or any-time algorithms. These algorithms have the property that they can be stopped mid-optimization and provide the best solution found so far. After a short time spent loading the instance, they find an initial 'zero-time' guess. Then, they explore the solution space and find incrementally better solutions until stopping with the best solution after a generally exponential amount of time. We experimentally evaluate the performance of the classical solvers Gurobi and MQLib using BURER2002 heuristic and FLIP in Section "Classical solvers". We observe that the zero-time performance, which is the quality of the fastest classical solution, is above the expected quality of QAOA $p = 11$, as shown in Fig. 3. The time to first solution scales almost linearly with size, as shown in Fig. 2. To compete with classical solvers, QAOA has to return better solutions faster.

Multi-shot QAOA

To improve the performance of QAOA, one can sample many bitstrings and then take the best one. This approach will work only if the dispersion of the cut fraction distribution is large, however. For example, if the dispersion is zero, measuring the ansatz state would return only bitstrings with a fixed cut value. By analyzing the correlations between the qubits in Section "QAOA performance", we show that the distribution of the cut fraction is a Gaussian with the standard deviation on the order of $1/\sqrt{N}$. The expectation value of the maximum of K samples is proportional to the standard deviation, as shown in Equation (7). This equation determines the performance of multishot QAOA. In the large N limit the standard deviation is small, and one might need to measure more samples in order to match the classical performance.

If we have the mean performance of a classical algorithm, we can estimate the number of samples K required for QAOA to match the classical performance. We denote the difference between classical and quantum expected cut fraction as $\Delta_p(t)$, which is a function of the running time of the classical algorithm. Moreover, it also depends on p since p determines QAOA expected performance. If $\Delta_p(t) < 0$, the performance of QAOA is better, and we need only a $K = 1$ sample. In order to provide an advantage, QAOA would have to measure this sample faster than the classical algorithm, as per Fig. 1. On the other hand, if $\Delta_p(t) > 0$, the classical expectation value is larger than the quantum one, and we have to perform multisample QAOA. We can find K by inverting Eq. (7). In order to match the classical algorithm, a quantum device should be able to run these K samples for no longer than t. We can therefore get the threshold sampling frequency.

$$v_p(t) = \frac{K}{t} = \frac{1}{t}\exp\left(\frac{N}{2\gamma_p^2}\Delta_p(t)^2\right) \qquad (1)$$

The scaling of $\Delta_p(t)$ with t is essential here since it determines at which point t we will have the smallest sampling frequency for advantage. We find that for BURER2002, the value of $\Delta(t)$ is the lowest for the smallest possible $t = t_0$, which is when a classical algorithm can produce its first solution. To provide the lower bound for QAOA, we consider t_0 as the most favorable point since a classical solution improves much faster with time than a multishot QAOA solution. This point is discussed in more detail in the Supplementary Methods.

Time t_0 is shown in Fig. 2 for different classical algorithms. We note that in the figure, the time scales polynomially with the number of nodes N. Figure 3 shows the mean cut fraction for the same classical algorithms, as well as the expectation value of QAOA at $p = 6, 11$. These two figures show that a simple linear-runtime FLIP algorithm is fast and gives a performance on par with $p = 6$ QAOA. In this case, $\Delta_6(t_0) < 0$, and we need to sample only a single bitstring. To obtain the $p = 6$ sampling frequency for advantage over the FLIP algorithm, one has to invert the time from Fig. 2. If the quantum device is not capable of running $p = 6$ with little noise, the quantum computer will have to do multishot QAOA. Note that any classical prepossessing for QAOA will be at least linear in the time since one must read the input and produce a quantum circuit. Therefore, for small $p < 6$, QAOA will not give a significant advantage: for any fast QAOA device, one needs a fast classical computer; one might just run the classical FLIP algorithm on it.

The Gurobi solver is able to achieve substantially better performance, and it slightly outperforms $p = 11$ QAOA. Moreover, the BURER2002 algorithm demonstrates even better solution quality than does Gurobi while being significantly faster. For both Gurobi and BURER2002, the $\Delta_{11}(t_0) > 0$, and we need to either perform multishot QAOA or increase p. Figure 4 shows the advantage sampling frequency $v_{11}(t_0)$ for the Gurobi and
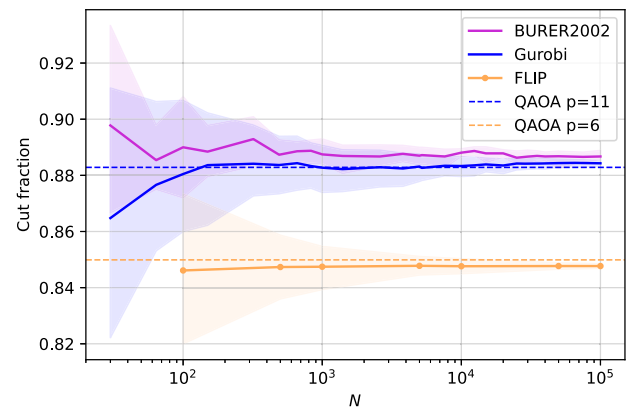

Fig. 3 Zero-time performance for graphs of different sizes N. The Y-value is the cut fraction obtained by running corresponding algorithms for the minimum possible time. This corresponds to the Y-value of the star marker in Fig. 5. Dashed lines show the expected QAOA performance for $p = 11$ (blue) and $p = 6$ (yellow). QAOA can outperform the FLIP algorithm at depth $p > 6$, while for Gurobi, it needs $p > 11$. Note that in order to claim an advantage, QAOA has to provide zero-time solutions in a faster time than FLIP or Gurobi does. These times are shown in Fig. 2.
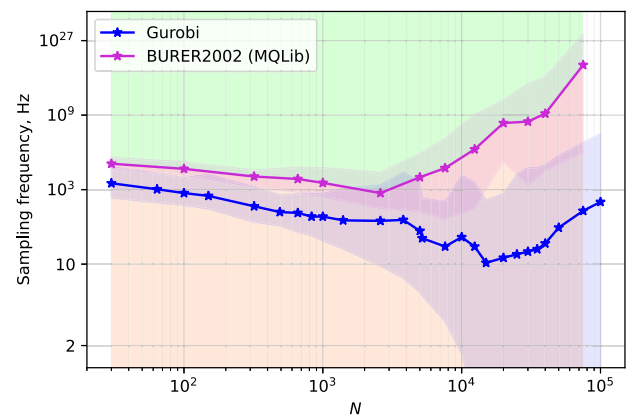
Fig. 4 Sampling frequency required to achieve MaxCut advantage using QAOA $p = 11$. The shaded area around the solid lines corresponds to 90–10 percentiles over 100 seeds for Gurobi and 20 seeds for BURER2002. The background shading represents a comparison of a quantum computer with the BURER2002 solver corresponding to modes in Fig. 1. Each quantum device can be represented on this plot as a point, where the x-axis is the number of qubits, and the y-axis is the time to solution. Depending on the region where the point lands, there are different results of comparisons. QAOA becomes inefficient for large N when sampling frequency starts to grow exponentially with N.

BURER2002 algorithms; note that the vertical axis is doubly exponential.

The sampling frequency is a result of two factors that work in opposite directions. On the one hand, the time to solution for a classical algorithm grows with N, and hence v drops. On the other hand, the standard deviation of distribution vanishes as $1/\sqrt{N}$, and therefore the number of samples K grows exponentially. There is an optimal size N for which the sampling frequency is minimal. This analysis shows that there is a possibility for an advantage with multi-shot QAOA for moderate sizes of $N = 100\ldots10\,000$, for which a sampling frequency of ≈ 10kHz is required. These frequencies are very sensitive to the difference in solution quality, and for $p \geq 12$, a different presentation is needed if one quantum sample is expected to give better than classical
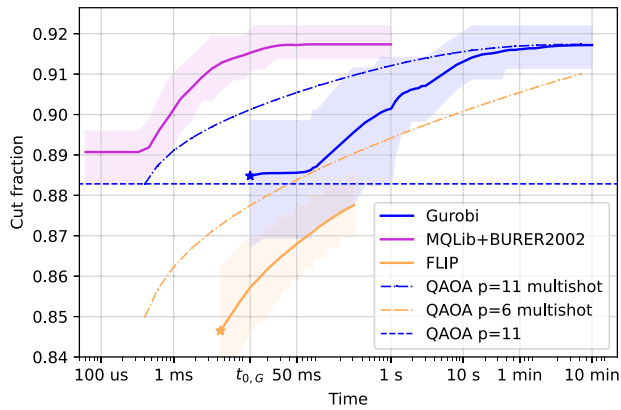
Fig. 5 Evolution of cut fraction value in the process of running the classical algorithms solving 3-regular MaxCut with $N = 256$. The shaded area shows a 90–10 percentiles interval, and the solid line shows the mean cut fraction over 100 graphs. The dashed lines show the expectation value of single-shot QAOA for $p = 6, 11$, and the dash-dotted lines show the expected performance for multishot QAOA given a sampling rate of 5 kHz. Note that for this $N = 256$, the multi-shot QAOA with $p = 6$ can compete with Gurobi at 50 ms. However, the slope of the multi-shot line will decrease for larger N, reducing the utility of the multi-shot QAOA.

solution quality. This is discussed in more detail in Supplementary Methods.

For large N, as expected, we see a rapid growth of sampling frequency, which indicates that QAOA does not scale for larger graph sizes unless we go to a higher depth $p > 11$. The color shading shows correspondence with Fig. 1. If the quantum device is able to run $p \geq 11$ and its sampling frequency and the number of qubits N corresponds to the green area, we have a quantum advantage. Otherwise, the quantum device belongs to the red area, and there is no advantage.

It is important to note the effect of classical parallelization on our results. Despite giving more resources to the classical side, parallel computing is unlikely to help it. To understand this, one has to think about how parallelization would change the performance profile, as shown in Fig. 5. The time to the first classical solution is usually bound from below by preparation tasks such as reading the graph, which is inherently serial. Thus, parallelization will not reduce t_0 and is, in fact, likely to increase it due to communication overhead. Instead, it will increase the slope of the solution quality curve, helping classical algorithms to compete in the convergence regime.

Discussion

As shown in Fig. 1, to achieve quantum advantage, QAOA must return better solutions faster than the competing classical algorithm. This puts stringent requirements on the speed of QAOA, which previously may have gone unevaluated. If QAOA returns a solution more slowly, the competing classical algorithm may 'try again' to improve its solution, as is the case for anytime optimizers such as the Gurobi solver. The simplest way to improve the speed of QAOA is to reduce the number of queries to the quantum device, which we propose in our fixed-angle QAOA approach. This implementation forgoes the variational optimization step and uses solution concentration, reducing the number of samples to order 1 instead of order 100,000. Even with these improvements, however, the space of quantum advantage may be difficult to access.

Our work demonstrates that with a quantum computer of ≈100 qubits, QAOA can be competitive with classical MaxCut solvers if the time to solution is shorter than 100 μs and the depth of the

QAOA circuit is $p \geq 6$. Note that this time to solution must include all parts of the computation, including state preparation, gate execution, and measurement. Depending on the parallelization of the architecture, there may be a quadratic time overhead. However, the required speed of the quantum device grows exponentially with N. Even if an experiment shows an advantage for intermediate N and $p \leq 11$, the advantage will be lost on larger problems regardless of the quantum sampling rate. Thus, in order to be fully competitive with classical MaxCut solvers, quantum computers have to increase solution quality, for instance, by using $p \geq 12$. Notably, $p = 12$ is required but not sufficient for achieving advantage: the end goal is obtaining a cut fraction better than ≥ 0.885 for large N, including overcoming other challenges of quantum devices such as noise.

These results lead us to conclude that for 3-regular graphs (perhaps all regular graphs), achieving quantum advantage on NISQ devices may be difficult. For example, the fidelity requirements to achieve quantum advantage are well above the characteristics of NISQ devices.

We note that improved versions of QAOA exist, where the initial state is replaced with a preoptimized state[14] or the mixer operator is adapted to improve performance[15,16]. One also can use information from classical solvers to generate a better ansatz state[17]. These algorithms have further potential to compete against classical MaxCut algorithms. Also, more general problems, such as weighted MaxCut, maximum independent set, and 3-SAT, may be necessary in order to find problem instances suitable for achieving quantum advantage.

When comparing with classical algorithms, one must record the complete time to solution from the circuit configuration to the measured state. This parameter may be used in the extension of the notion of quantum volume, which is customarily used for quantum device characterization. Our work shows that QAOA MaxCut does not scale with graph size for at least up to $p \leq 11$, thus putting the quantum advantage for this problem away from the NISQ era.

METHODS

Both classical solvers and QAOA return a bitstring as a solution to the MaxCut problem. To compare the algorithms, we must decide on a metric to use to measure the quality of the solution. A common metric for QAOA and many classical algorithms is the approximation ratio, which is defined as the ratio of the cut value (as defined in Eq. (3)) of the solution divided by the optimal (i.e., maximum possible) cut value for the given graph. This metric is hard to evaluate heuristically for large N since we do not know the optimal solution. We, therefore, use the cut fraction as the metric for solution quality, which is the cut value divided by the number of edges.

We analyze the algorithms on an ensemble of problem instances. Some instances may give an advantage, while others may not. We, therefore, analyze ensemble advantage, which compares the average solution quality over the ensemble. The set of 3-regular graphs is extremely large for large graph size N, so for classical heuristic algorithms, we evaluate the performance on a subset of graphs. We then look at the mean of the cut fraction over the ensemble, which is the statistical approximation of the mean of the cut fraction over all 3-regular graphs.

QAOA methodology

Usually, QAOA is thought of as a hybrid algorithm, where a quantum-classical outer loop optimizes the angles γ, β through a repeated query to the quantum device by a classical optimizer. Depending on the noise, this process may require hundreds or thousands of queries in order to find optimal angles, which slows the computation. To our knowledge, no comprehensive work

exists on exactly how many queries may be required to find such angles. It has been numerically observed[6,18], however, that for small graph size $N = 12$ and $p = 4$, classical noise-free optimizers may find good angles in approximately 100 steps, which can be larger for higher N and p. Each step may need order 10^3 bitstring queries to average out shot noise and find expectation values for an optimizer, and thus seeking global angles may require approximately 100,000 queries to the simulator. The angles are then used for preparing an ansatz state, which is, in turn, measured (potentially multiple times) to obtain a solution. Assuming a sampling rate of 1 kHz, this approach implies a QAOA solution of approximately 100 s.

Recent results, however, suggest that angles may be precomputed on a classical device[19] or transferred from other similar graphs[20]. Further research analytically finds optimal angles for $p \leq 20$ and $d \to \infty$ for all large-girth d-regular graphs but does not give angles for finite d[21]. Going a step further, recent work finds that evaluating regular graphs at particular fixed angles has good performance on all problem instances[22]. These precomputed or fixed angles allow the outer loop to be bypassed, finding close to optimal results in a single shot. In this way, a 1000 Hz QAOA solution can be found in milliseconds, a speedup of several orders of magnitude.

For this reason, we study the prospect of quantum advantage in the context of fixed-angle QAOA. For d-regular graphs, there exist particular fixed angles with universally good performance[23]. Additionally, as will be shown in Section "Single-shot QAOA Sampling", one can reasonably expect that sampling a single bitstring from the fixed-angle QAOA will yield a solution with a cut fraction close to the expected value.

The crucial property of the fixed-angle single-shot approach is that it is guaranteed to work for any graph size N. On the other hand, angle optimization could be less productive for large N, and the multiple-shot (measuring the QAOA ansatz multiple times) approach is less productive for large N, as shown in Section "Multi-shot QAOA Sampling". Moreover, the quality of the solution scales with a depth of $\sqrt{p}$[23], which is faster than with the number of samples $\sqrt{\log K}$, instructing us to resort to multishot QAOA only if a larger p is unreachable. Thus, the fixed-angle single-shot QAOA can robustly speed up finding a good approximate solution from the order of seconds to milliseconds, a necessity for advantage over state-of-the-art anytime heuristic classical solvers, which can get good or exact solutions in approximately milliseconds. Crucially, single-shot QAOA quality of solution can be maintained for all sizes N at fixed depth p, which can mean constant time scaling for particularly capable quantum devices.

To simulate the expectation value of the cost function for QAOA, we employ a classical quantum circuit simulation algorithm QTensor[24–26]. This algorithm is based on tensor network contraction and is described in more detail in Supplementary Methods. Using this approach, one can simulate expectation values on a classical computer, even for circuits with millions of qubits.

Classical solvers

Two main types of classical MaxCut algorithms exist approximate algorithms and heuristic solvers. Approximate algorithms guarantee a certain quality of solution for any problem instance. Such algorithms[27,28] also provide polynomial-time scaling. Heuristic solvers[29,30] are usually based on branch-and-bound methods[31] that use branch pruning and heuristic rules for variable and value order. These heuristics are usually designed to run well on graphs that are common in practical use cases. Heuristic solvers typically return better solutions than approximate solvers, but they provide no guarantee of the quality of the solution.

The comparison of QAOA with classical solvers thus requires making choices of measures that depend on the context of comparison. From a theoretical point of view, guaranteed

performance is more important; in contrast, from an applied point of view, heuristic performance is the measure of choice. A previous work[22] demonstrates that QAOA provides better performance guarantees than does the Goemans–Williamson algorithm[28]. In this paper, we compare against heuristic algorithms since such a comparison is more relevant for real-world problems. On the other hand, the performance of classical solvers reported in this paper can depend on a particular problem instance.

We evaluate two classical algorithms using a single node of Argonne's Skylake testbed; the processor used is an Intel Xeon Platinum 8180M CPU @ 2.50 GHz with 768 GB of RAM.

The first algorithm we study is the Gurobi solver[29], which is a combination of many heuristic algorithms. We evaluate Gurobi with an improved configuration based on communication with Gurobi support (https://support.gurobi.com/hc/en-us/community/posts/4403570181137-Worse-performance-for-smaller-problem). We use `Symmetry=0` and `PreQLinearize=2` in our improved configuration. As further tweaks and hardware resources may increase the speed, the results here serve as a characteristic lower bound on Gurobi performance rather than a true guarantee. We run Gurobi on 100 random-regular graphs for each size N and allow each optimization to run for 30 min. During the algorithm runtime, we collect information about the process, in particular, the quality of the best-known solution. In this way, we obtain a performance profile of the algorithm that shows the relation between the solution quality and the running time. An example of such a performance profile for $N = 256$ is shown in Fig. 5. Gurobi was configured to use only a single CPU to avoid interference in runtime between different Gurobi optimization runs for different problem instances. In order to speed up the collection of the statistics, 55 problem instances were executed in parallel.

The second algorithm is MQLib[30], which is implemented in C++ and uses a variety of different heuristics for solving MaxCut and QUBO problems. We chose the BURER2002 heuristic since, in our experiments, it performs the best for MaxCut on random regular graphs. Despite using a single thread, this algorithm is much faster than Gurobi; thus, we run it for 1 s. In the same way as with Gurobi, we collect the performance profile of this algorithm.

While QAOA and Gurobi can be used as general-purpose combinatorial optimization algorithms, this algorithm is designed to solve MaxCut problems only, and the heuristic was picked that demonstrated the best performance on the graphs we considered. In this way, we use Gurobi as a worst-case classical solver, which is capable of solving the same problems as QAOA can. Moreover, Gurobi is a well-established commercial tool that is widely used in industry. Note, however, that we use QAOA fixed angles that are optimized specifically for 3-regular graphs, and one can argue that our fixed-angle QAOA is an algorithm designed for 3-regular MaxCut. For this reason, we also consider the best-case MQLib +BURER2002 classical algorithm, which is designed for MaxCut, and we choose the heuristic that performs best on 3-regular graphs.

QAOA performance

Two aspects are involved in comparing the performance of algorithms, as outlined in Fig. 1: time to solution and quality of solution. In this section, we evaluate the performance of single-shot fixed-angle QAOA. As discussed in the introduction, the time to solution is a crucial part, and for QAOA is dependent on the initialization time and the number of rounds of sampling. Single-shot fixed-angle QAOA involves only a single round of sampling, so the time to solution can be extremely fast, with initialization time potentially becoming the limiting factor. This initialization time is bound by the speed of classical computers, which perform calibration and device control. Naturally, if one is able to achieve greater initialization speed by using better classical computers, the

same computers can be used to improve the speed of solving MaxCut classically. Therefore, it is also important to consider the time scaling of both quantum initialization and classical runtime.

The quality of the QAOA solution is the other part of the performance. The discussion below evaluates this feature by using subgraph decompositions and QAOA typicality, including a justification of single-shot sampling.

QAOA is a variational ansatz algorithm structured to provide solutions to combinatorial optimization problems. The ansatz is constructed as p repeated applications of an objective $\hat{C}$ and mixing $\hat{B}$ unitary:

$$|\gamma, \beta\rangle = e^{-i\beta_p \hat{B}} e^{-i\gamma_p \hat{C}} (\cdots) e^{-i\beta_1 \hat{B}} e^{-i\gamma_1 \hat{C}} |+\rangle, \qquad (2)$$

where $\hat{B}$ is a sum over Pauli X operators $\hat{B} = \sum_i^N \hat{\sigma}_x^i$. A common problem instance is MaxCut, which strives to bipartition the vertices of some graph $\mathcal{G}$ such that the maximum number of edges have vertices in opposite sets. Each such edge is considered to be cut by the bipartition. This may be captured in the objective function

$$\hat{C} = \frac{1}{2} \sum_{\langle ij \rangle \in \mathcal{G}} (1 - \hat{\sigma}_z^i \hat{\sigma}_z^j), \qquad (3)$$

whose eigenstates are bipartitions in the Z basis, with eigenvalues that count the number of cut edges. To get the solution to the optimization problem, one prepares the ansatz state $|\gamma, \beta\rangle$ on a quantum device and then measures the state. The measured bitstring is the solution output from the algorithm.

While QAOA is guaranteed to converge to the exact solution in the $p \to \infty$ limit in accordance with the adiabatic theorem[11,32], today's hardware is limited to low depths $p \sim 1$ to 5 because of the noise and decoherence effects inherent to the NISQ era.

A useful tool for analyzing the performance of QAOA is the fact that QAOA is local[11,12]: the entanglement between any two qubits at a distance of $\geq 2p$ steps from each other is strictly zero. For a similar reason, the expectation value of a particular edge $\langle ij \rangle$

$$f_{\langle ij \rangle} = \frac{1}{2} \langle \gamma, \beta | 1 - \hat{\sigma}_z^i \hat{\sigma}_z^j | \gamma, \beta \rangle \qquad (4)$$

depends only on the structure of the graph within p steps of edge $\langle ij \rangle$. Regular graphs have a finite number of such local structures (also known as subgraphs)[22], and so the expectation value of the objective function can be rewritten as a sum over subgraphs

$$\langle \hat{C} \rangle = \sum_{\text{subgraphs} \lambda} M_\lambda(\mathcal{G}) f_\lambda. \qquad (5)$$

Here, λ indexes the different possible subgraphs of depth p for a d regular graph, $M_\lambda(\mathcal{G})$ counts the number of each subgraph λ for a particular graph $\mathcal{G}$, and f_λ is the expectation value of the subgraph (e.g., Eq. (4)). For example, if there are no cycles $\leq 2p + 1$, only one subgraph (the tree subgraph) contributes to the sum.

With this tool, we may ask and answer the following question: What is the typical performance of single-shot fixed-angle QAOA evaluated over some ensemble of graphs? Here, performance is characterized as the typical (average) fraction of edges cut by a bitstring solution returned by a single sample of fixed-angle QAOA, averaged over all graphs in the particular ensemble.

For our study, we choose the ensemble of 3-regular graphs on N vertices. Different ensembles, characterized by different connectivity d and size N, may have different QAOA performance[33,34].

Using the structure of the random regular graphs, we can put bounds on the cut fraction by bounding the number of different subgraphs and evaluating the number of large cycles. These bounds become tighter for $N \longrightarrow \infty$ and fixed p since the majority of subgraphs become trees and 1-cycle graphs. We describe this analysis in detail in Supplemental methods, which shows that the QAOA cut fraction will equal the expectation value on the tree subgraph, which may be used as a 'with high probability' (WHP)
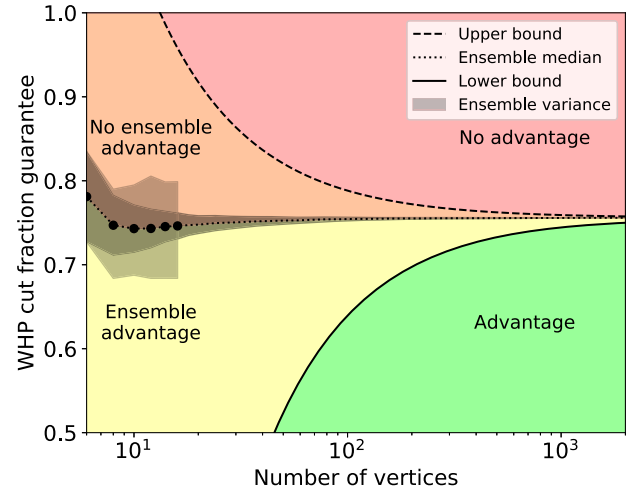
Fig. 6 $p = 2$ QAOA cut fraction guarantees under different assumptions. Dashed and solid lines plot with high probability the lower and upper bounds on cut fractions, respectively, assuming only graph theoretic typicality on the number of subgraphs. Dotted plots are the ensemble median over an ensemble of 3 regular graphs; for $N \leq 16$ (dots); this includes all graphs, while for $N > 16$, this is an ensemble of 1000 graphs for each size. We used 32 sizes between 16 and 256. Dark black fill plots the variance in the cut fraction over the ensemble, and light black fill plots the extremal values over the ensemble. The median serves as a proxy of performance, assuming QAOA typicality. Given a particular cut from a classical solver, there may be different regions of advantage, shown by the four colors and discussed in the text.

proxy of performance. Furthermore, using a subgraph counting argument, we may count the number of tree subgraphs to find an upper and lower WHP bound on the cut fraction for smaller graphs. These bounds are shown as the boundaries of the red and green regions in Fig. 6.

QAOA ensemble estimates

A more straightforward but less rigorous characterization of QAOA performance is simply to evaluate fixed-angle QAOA on a subsample of graphs in the ensemble. The results of such an analysis require an assumption not on the particular combinatorial graph structure of ensembles but instead on the typicality of expectation values on subgraphs. This is an assumption on the structure of QAOA and allows an extension of typical cut fractions from the large N limit where most subgraphs are trees to a small N limit where typically a very small fraction of subgraphs are trees.

Figure 6 plots the ensemble-averaged cut fraction for $p = 2$ and various sizes of graphs. For $N \leq 16$, the ensemble includes every 3-regular graph (4681 in total). For each size of $N > 16$, we evaluate fixed-angle QAOA on 1000 3-regular graphs drawn at random from the ensemble of all 3-regular graphs for each size $N \in (16, 256)$. Note that because the evaluation is done at fixed angles, it may be done with minimal quantum calculation by a decomposition into subgraphs, then looking up the subgraph expectation value f_λ from[22]. This approach is also described in more detail in[35]. In this way, expectation values can be computed as fast as an isomorphism check.

From Fig. 6, we observe that the median cut fraction across the ensemble appears to concentrate around that of the tree subgraph value, even for ensembles where the typical graph is too small to include many tree subgraphs. Additionally, the variance (dark fill) reduces as N increases, consistent with the fact that for larger N, there are fewer kinds of subgraphs with non-negligible frequency. Furthermore, the absolute range (light fill), which plots the largest and smallest expectation value across the

ensemble, is consistently small. While the data for the absolute range exists here only for $N \leq 16$ because of complete sampling of the ensemble, 0ne can reasonably expect that these absolute ranges extend for all N, suggesting that the absolute best performance of $p = 2$ QAOA on 3-regular graphs is around ≈ 0.8.

We numerically observe across a range of p (not shown) that these behaviors persist: the typical cut fraction is approximately equal to that of the tree subgraph value $f_{\text{p-tree}}$ even in the limit where no subgraph is a tree. This suggests that the typical subgraph expectation value $f_\lambda \approx f_{\text{p-tree}}$, and only an atypical number of subgraphs have expectation values that diverge from the tree value. With this observation, we may use the value $f_{\text{p-tree}}$ as a proxy for the average cut fraction of fixed-angle QAOA.

These analyses yield four different regimes for advantage vs. classical algorithms, shown in Fig. 6. If a classical algorithm yields small cut fractions for large graphs (green, bottom right), then there is an advantage in a strong sense. Based only on graph combinatorics, with a high probability, most of the edges participate in a few cycles, and thus the cut fraction is almost guaranteed to be around the tree value, larger than the classical solver. Conversely, if the classical algorithm yields large cut fractions for large graphs (red, top right), there is no advantage in the strong sense: QAOA will yield, for example, only ~0.756 for $p = 2$ because most edges see no global structure. This analysis emphasizes that of[12], which suggests that QAOA needs to 'see' the whole graph in order to get reasonable performance.

Two additional performance regimes for small graphs exist, where QAOA can reasonably see the whole graph. If a classical algorithm yields small cut fractions for small graphs (yellow, bottom left), then there is an advantage in a weak sense, which we call the 'ensemble advantage'. Based on QAOA concentration, there is at least a 50% chance that the QAOA result on a particular graph will yield a better cut fraction than will the classical algorithm; assuming that the variance in the cut fraction is small, this is a 'with high probability' statement. Conversely, if the classical algorithm yields large cut fractions for small graphs (orange, top left), there is no advantage in a weak sense. Assuming QAOA concentration, the cut fraction will be smaller than the classical value, and for some classical cut fractions there are no graphs with advantage (e.g., >0.8 for $p = 2$).

Based on these numerical results, we may use the expectation value of the tree subgraph $f_{\text{p-tree}}$ as a high-probability proxy for typical fixed-angle QAOA performance on regular graphs. For large N, this result is validated by graph-theoretic bounds counting the typical number of tree subgraphs in a typical graph. For small N, this result is validated by fixed-angle QAOA evaluation on a large ensemble of graphs.

Single-shot QAOA Sampling

A crucial element of single-shot fixed-angle QAOA is that the typical bitstring measured from the QAOA ansatz has a cut value similar to the average. This fact was originally observed by Farhi et al. in the original QAOA proposal[11]: because of the strict locality of QAOA, vertices a distance more than >2p steps from each other have a ZZ correlation of strictly zero. Thus, for large graphs with a width >2p, by the central limit theorem, the cut fraction concentrates to a Gaussian with a standard deviation of order $\frac{1}{\sqrt{N}}$ around the mean. As the variance grows sublinearly in N, the values concentrate at the mean, and thus with high probability measuring a single sample of QAOA will yield a solution with a cut value close to the average.

However, this result is limited in scope for larger depths p because it imposes no requirements on the strength of correlations for vertices within distance $\leq 2p$. Therefore, here we strengthen the argument of Farhi et al. and show that these concentration results may persist even in the limit of large depth p and small graphs N. We formalize these results by evaluating the
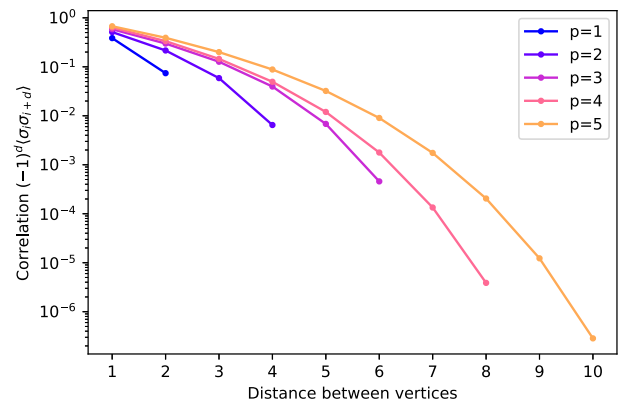
Fig. 7 Long-range antiferromagnetic correlation coefficient on the 3-regular Bethe lattice, which is a proxy for an $N \to \infty$ typical 3-regular graph. Horizontal indexes the distance between two vertices. QAOA is strictly local, which implies that no correlations exist between vertices a distance >2p away. As shown here, however, these correlations are exponentially decaying with distance. This suggests that even if the QAOA 'sees the whole graph', one can use the central limit theorem to argue that the distribution of QAOA performance is Gaussian with the standard deviation of $\propto 1/\sqrt{N}$.

ZZ correlations of vertices within $2p$ steps, as shown in Fig. 7. Expectation values are computed on the 3-regular Bethe lattice, which has no cycles and thus can be considered the $N \to \infty$ typical limit. Instead of computing the nearest-neighbor correlation function, the x-axis computes the correlation function between vertices a certain distance apart. For distance 1, the correlations are that of the objective function $f_{\text{p-tree}}$. Additionally, for distance >2p, the correlations are strictly zero in accordance with the strict locality of QAOA. For distance $\leq 2p$, the correlations are exponentially decaying with distance. Consequently, even for vertices within the lightcone of QAOA, the correlation is small; and so by the central limit theorem, the distribution will be Gaussian. This result holds because the probability of having a cycle of fixed size converges to 0 as $N \to \infty$. In other words, we know that with $N \to \infty$, we will have a Gaussian cost distribution with standard deviation $\propto \frac{1}{\sqrt{N}}$.

When considering small N graphs, ones that have cycles of length $\leq 2p + 1$, we can reasonably extend the argument of Section "QAOA ensemble estimates" on the typicality of subgraph expectation values. Under this typicality argument, the correlations between close vertices are still exponentially decaying with distance, even though the subgraph may not be a tree and there are multiple short paths between vertices. Thus, for all graphs, by the central limit theorem, the distribution of solutions concentrates as a Gaussian with a standard deviation of order $\frac{1}{\sqrt{N}}$ around the mean. By extension, with a probability ~50%, any single measurement will yield a bitstring with a cut value greater than the average. These results of cut distributions have been found heuristically in[36].

The results are a full characterization of the fixed-angle single-shot QAOA on 3-regular graphs. Given a typical graph sampled from the ensemble of all regular graphs, the typical cut fraction from level p QAOA will be about that of the expectation value of the p-tree $f_{\text{p-tree}}$. The distribution of bitstrings is concentrated as a Gaussian of sub-extensive variance around the mean, indicating that one can find a solution with quality greater than the mean with order 1 samples. Furthermore, because the fixed angles bypass the hybrid optimization loop, the number of queries to the quantum simulator is reduced by orders of magnitude, yielding solutions on potentially millisecond timescales.

Mult-shot QAOA sampling

In the preceding section, we demonstrated that the standard deviation of MaxCut cost distribution falls as $1/\sqrt{N}$, which deems impractical the usage of multiple shots for large graphs. However, it is worth verifying more precisely its effect on the QAOA performance. The multiple-shot QAOA involves measuring the bitstring from the same ansatz state and then picking the bitstring with the best cost. To evaluate such an approach, we need to find the expectation value for the best bitstring over K measurements.

As shown above, the distribution of cost for each measured bitstring is Gaussian, $p(x) = G(\frac{x-\mu_p}{\sigma_N})$. We define a new random variable ξ which is the cost of the best of K bitstrings. The cumulative distribution function (CDF) of the best of K bitstrings is $F_K(\xi)$, and $F_1(\xi)$ is the CDF of a normal distribution. The probability density for ξ is

$$p_K(\xi) = \frac{d}{d\xi}F_K(\xi) = \frac{d}{d\xi}F_1^K(\xi) = KF_1^{K-1}(\xi)p(\xi), \tag{6}$$

where $F_1(\xi) = \int_{-\infty}^{\xi} p(x)dx$ and F_1^K is the ordinary exponentiation. The expectation value for ξ can be found by $E_K = \int_{-\infty}^{\infty} dx\, xp_K(x)$. While the analytical expression for the integral can be extensive, a good upper bound exists for it: $E_K \leq \sigma\sqrt{2\log K} + \mu$.

Combined with the $1/\sqrt{N}$ scaling of the standard deviation, we can obtain a bound on improvement in cut fraction from sampling K times:

$$\Delta = \gamma_p\sqrt{\frac{2}{N}\log K}, \tag{7}$$

where γ_p is a scaling parameter. The value Δ is the difference in solution quality for multishot and single-shot QAOA. Essentially it determines the utility of using multishot QAOA. We can determine the scaling constant γ_p by classically simulating the distribution of the cost value in the ansatz state. We perform these simulations using QTensor for an ensemble of graphs with $N \leq 26$ to obtain $\gamma_6 = 0.1926$ and $\gamma_{11} = 0.1284$.

It is also worthwhile to verify the $1/\sqrt{N}$ scaling, by calculating γ_p for various N. We can do so for smaller $p = 3$ and graph sizes $N \leq 256$. We calculate the standard deviation by $\Delta C = \sqrt{\langle C^2 \rangle - \langle C \rangle^2}$ and evaluate the $\langle C^2 \rangle$ using QTensor. This evaluation gives large light cones for large p; the largest that we were able to simulate is $p = 3$. From the deviations ΔC, we can obtain values for γ_3. We find that for all N, the values stay within 5% of the average overall N. This shows that they do not depend on N, which in turn signifies that the $1/\sqrt{N}$ scaling is a valid model. The results of the numerical simulation of the standard deviation are discussed in more detail in the Supplementary Methods.

To compare multishot QAOA with classical solvers, we plot the expected performance of multishot QAOA in Fig. 5 as dash-dotted lines. We assume that a quantum device is able to sample at the 5kHz rate. Today's hardware is able to run up to $p = 5$ and achieve the 5 kHz sampling rate[37]. Notably, the sampling frequency of modern quantum computers is bound not by gate duration but by qubit preparation and measurement.

For small N, reasonable improvement can be achieved by using a few samples. For example, for $N = 256$ with $p = 6$ and just $K = 200$ shots, QAOA can perform as well as single-shot $p = 11$ QAOA. For large N, however, too many samples are required to obtain substantial improvement for multishot QAOA to be practical.

Classical performance

To compare the QAOA algorithm with its classical counterparts, we choose state-of-the-art algorithms that solve a similar spectrum of problems as QAOA, and we evaluate the time to solution and solution quality. Here, we compare two algorithms: Gurobi and MQLib+BURER2002. Both are anytime heuristic algorithms that can provide an approximate solution at an arbitrary time. For these algorithms, we collect the 'performance profiles'—the dependence of solution quality on time spent finding the solution. We also evaluate the performance of a simple MaxCut algorithm FLIP. This algorithm has a proven linear time scaling with input size. It returns a single solution after a short time. To obtain a better FLIP solution, one may run the algorithm several times and take the best solution, similar to the multishot QAOA.

Both algorithms have to read the input and perform some initialization steps to output any solution. This initialization step determines the minimum time required for getting the initial solution—a 'first guess' of the algorithm. This time is the leftmost point of the performance profile marked with a star in Fig. 5. We call this time t_0 and the corresponding solution quality 'zero-time performance'.

We observe two important results.

1. Zero-time performance is constant with N and is comparable to that of $p = 11$ QAOA, as shown in Fig. 3, where solid lines show classical performance and dashed lines show QAOA performance.
2. t_0 scales as a low-degree polynomial in N, as shown in Fig. 2. The y-axis is t_0 for several classical algorithms.

Since the zero-time performance is slightly above the expected QAOA performance at $p = 11$, we focus on analyzing this zero-time regime. In the following subsections, we discuss the performance of the classical algorithms and then proceed to the comparison with QAOA.

Performance of Gurobi solver

In our classical experiments, as mentioned in Section "Classical solvers", we collect the solution quality with respect to time for multiple N and graph instances. An example of averaged solution quality evolution is shown in Fig. 5 for an ensemble of 256 vertex 3-regular graphs. Between times 0 and $t_{0,G}$, the Gurobi algorithm goes through some initialization and quickly finds some naive approximate solution. Next, the first incumbent solution is generated, which will be improved in further runtime. Notably, for the first 50 ms, no significant improvement in solution quality is found. After that, the solution quality starts to rise and slowly converges to the optimal value of ~0.92.

It is important to appreciate that Gurobi is more than just a heuristic solver: in addition to the incumbent solution, it always returns an upper bound on the optimal cost. When the upper bound and the cost for the incumbent solution match, the optimal solution is found. It is likely that Gurobi spends a large portion of its runtime on proving the optimality by lowering the upper bound. This emphasizes that we use Gurobi as a worst-case classical solver.

Notably, the x-axis of Fig. 5 is logarithmic: the lower and upper bounds eventually converge after exponential time with a small prefactor, ending the program and yielding the exact solution. Additionally, the typical upper and lower bounds of the cut fraction of the best solution are close to 1. Even after approximately 10 s for a 256-vertex graph, the algorithm returns cut fractions with very high quality ~0.92, far better than intermediate-depth QAOA.

The zero-time performance of Gurobi for $N = 256$ corresponds to the Y-value of the star marker in Fig. 5. We plot this value for various N in Fig. 3. As shown in the figure, zero-time performance goes up and reaches a constant value of ~0.882 at N ~100. Even for large graphs of $N = 10^5$, the solution quality stays at the same level.

Such solution quality is returned after time $t_{0,G}$, which we plot in Fig. 2 for various N. For example, for a 1000-node graph, it will take ~40 ms to return the first solution. Evidently, this time scales as a low-degree polynomial with N. This shows that Gurobi can consistently return solutions of quality ~0.882 in polynomial time.

Performance of MQLib + BURER2002 and FLIP algorithms

The MQLib algorithm with the BURER2002 heuristic shows significantly better performance, which is expected since it is specific to MaxCut. As shown in Fig. 5 for $N = 256$ and in Fig. 2 for various N, the speed of this algorithm is much better compared with Gurobi's. Moreover, t_0 for MQLib also scales as a low-degree polynomial, and for 1000 nodes MQLib can return a solution in 2 ms. The zero-time performance shows the same constant behavior, and the value of the constant is slightly higher than that of Gurobi, as shown in Fig. 3.

While for Gurobi and MQLib, we find the time scaling heuristically, the FLIP algorithm is known to have linear time scaling. Our implementation in Python, shows speed comparable to that of MQLib and solution quality comparable to QAOA $p = 6$. We use this algorithm as a demonstration that a linear-time algorithm can give a constant performance for large N, averaged over multiple graph instances. The supplementary material uses the following research[38–52] to provide more details on corresponding questions.

DATA AVAILABILITY

The code, figures, and datasets generated during the current study are available in a public repository https://github.com/danlkv/quantum-classical-time-maxcut. See the README.md file for the details on the contents of the repository.

REFERENCES

1. Alexeev, Y. et al. Quantum computer systems for scientific discovery. PRX Quantum https://doi.org/10.1103/prxquantum.2.017001 (2021).
2. Arute, F. et al. Quantum supremacy using a programmable superconducting processor. Nature 574, 505–510 (2019).
3. Preskill, J. Quantum computing and the entanglement frontier. Preprint at arXiv http://arxiv.org/abs/1203.5813 (2012).
4. Ebadi, S. et al. Quantum optimization of maximum independent set using Rydberg atom arrays. Science 376, 1209–1215 (2022).
5. Guerreschi, G. G. & Matsuura, A. Y. QAOA for Max-Cut requires hundreds of qubits for quantum speed-up. Sci. Rep. 9, 6903 (2019).
6. Zhou, L., Wang, S.-T., Choi, S., Pichler, H. & Lukin, M. D. Quantum approximate optimization algorithm: performance, mechanism, and implementation on near-term devices. Phys. Rev. X 10, 021067 (2020).
7. Serret, M. F., Marchand, B. & Ayral, T. Solving optimization problems with Rydberg analog quantum computers: Realistic requirements for quantum advantage using noisy simulation and classical benchmarks. Phys. Rev. A 102, 052617 (2020).
8. Preskill, J. Quantum computing in the NISQ era and beyond. Quantum 2, 79 (2018).
9. Cerezo, M. et al. Variational quantum algorithms. Nat. Rev. Phys. 3, 625–644 (2021).
10. Peruzzo, A. et al. A variational eigenvalue solver on a photonic quantum processor. Nat. Commun. 5, 1–7 (2014).
11. Farhi, E., Goldstone, J. & Gutmann, S. A quantum approximate optimization algorithm. Preprint at arXiv http://arxiv.org/abs/1411.4028 (2014).
12. Farhi, E., Gamarnik, D. & Gutmann, S. The quantum approximate optimization algorithm needs to see the whole graph: a typical case. Preprint at arXiv http://arxiv.org/abs/2004.09002 (2020).
13. Chatterjee, T., Mohtashim, S. I. & Kundu, A. On the variational perspectives to the graph isomorphism problem. Preprint at arXiv https://doi.org/10.48550/arXiv.2111.09821 (2021).
14. Egger, D. J., Mareček, J. & Woerner, S. Warm-starting quantum optimization. Quantum 5, 479 (2021).
15. Zhu, L. et al. Adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer. Phys. Rev. Res. 4, 033029 (2022).
16. Govia, L. C. G., Poole, C., Saffman, M. & Krovi, H. K., Freedom of the mixer rotation axis improves performance in the quantum approximate optimization algorithm. Phys. Rev. A https://doi.org/10.1103/physreva.104.062428 (2021)
17. Wurtz, J. & Love, P. Classically optimal variational quantum algorithms. IEEE Trans. Quantum Eng. 2, 1–7 (2021).
18. Shaydulin, R., Safro, I. & Larson, J., Multistart methods for quantum approximate optimization. in 2019 IEEE High Performance Extreme Computing Conference (HPEC) https://doi.org/10.1109/hpec.2019.8916288 (2019).
19. Streif, M. & Leib, M. Training the quantum approximate optimization algorithm without access to a quantum processing unit. Quantum Sci. Technol. 5, 034008 (2020).
20. Galda, A., Liu, X., Lykov, D., Alexeev, Y. & Safro, I. Transferability of optimal QAOA parameters between random graphs. Preprint at arXiv http://arxiv.org/abs/2106.07531 (2021).
21. Basso, J., Farhi, E., Marwaha, K., Villalonga, B. & Zhou, L. The quantum approximate optimization algorithm at high depth for MaxCut on large-girth regular graphs and the Sherrington-Kirkpatrick model. Leibniz Int. Proc. Inform. 232, 7:1–7:21 (2022).
22. Wurtz, J. & Love, P. MaxCut quantum approximate optimization algorithm performance guarantees for $p > 1$. Phys. Rev. A 103, 042612 (2021).
23. Wurtz, J. & Lykov, D. Fixed-angle conjectures for the quantum approximate optimization algorithm on regular MaxCut graphs. Phys. Rev. A 104, 052419 (2021).
24. Lykov, D., Schutski, R., Galda, A., Vinokur, V. & Alexeev, Y. Tensor network quantum simulator with step-dependent parallelization. in 2022 IEEE International Conference on Quantum Computing and Engineering (QCE) https://doi.org/10.1109/qce53715.2022.00081 (2022).
25. Lykov, D. & Alexeev, Y. Importance of diagonal gates in tensor network simulations. in 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 447–452 https://doi.org/10.1109/ISVLSI51109.2021.00088 (2021).
26. Lykov, D. et al. Performance evaluation and acceleration of the QTensor quantum circuit simulator on GPUs. in 2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS) https://doi.org/10.1109/qcs54837.2021.00007 (2021).
27. Halperin, E., Livnat, D. & Zwick, U. Max cut in cubic graphs. J. Algorithms 53, 169–185 (2004).
28. Goemans, M. X. & Williamson, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM 42, 1115–1145 (1995).
29. LLC Gurobi Optimization. Gurobi optimizer reference manual. https://www.gurobi.com (2021a).
30. Dunning, I., Gupta, S. & Silberholz, J. What works best when? a systematic evaluation of heuristics for max-cut and QUBO. INFORMS J. Comput. 30, 608–624 (2018).
31. LLC Gurobi Optimization. Mixed integer programming basics. https://www.gurobi.com/resource/mip-basics/ (2021b).
32. Wurtz, J. & Love, P. J. Counterdiabaticity and the quantum approximate optimization algorithm. Quantum 6, 635 (2022).
33. Herrman, R. et al. Impact of graph structures for QAOA on MaxCut. Quantum Inf. Process. https://doi.org/10.1007/s11128-021-03232-8 (2021).
34. Shaydulin, R., Marwaha, K., Wurtz, J. & Lotshaw, P. C. QAOAKit: a toolkit for reproducible study, application, and verification of the QAOA", 2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS) https://doi.org/10.1109/qcs54837.2021.00011 (2021).
35. Shaydulin, R. & Wild, S. M. Exploiting symmetry reduces the cost of training qaoa. IEEE Trans. Quantum Eng. 2, 1–9 (2021).
36. Larkin, J., Jonsson, M., Justice, D. & Guerreschi, G. G. Evaluation of QAOA based on the approximation ratio of individual samples. Quantum Sci. Technol. 7, 045014 (2022).
37. Harrigan, M. et al. Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. Nat. Phys. 17, 332–336 (2021).
38. Häner, T. & Steiger, D. S. 0.5 petabyte simulation of a 45-qubit quantum circuit. in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis SC '17 https://doi.org/10.1145/3126908.3126947 (2017).
39. Wu, X.-C. et al. Full-state quantum circuit simulation by using data compression. in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis https://doi.org/10.1145/3295500.3356155 (2019).
40. Lykov, D. QTensor. https://github.com/danlkv/qtensor (2021).
41. Liu, M. et al. Embedding Learning in Hybrid Quantum-Classical Neural Networks. 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), 79–86 (2022). https://doi.org/10.1109/QCE53715.2022.00026.

42. Dechter, R. Bucket elimination: a unifying framework for probabilistic inference. in *Learning in Graphical Models* (Springer Netherlands, 1998) https://doi.org/10.1007/978-94-011-5014-9_4 pp. 75–104.

43. Harvey, D. J. & Wood, D. R. The treewidth of line graphs. *J. Comb. Theory Ser. B* **132**, 157–179 (2018).

44. Markov, I. L. & Shi, Y. Simulating quantum computation by contracting tensor networks. *Siam. J. Sci. Comput.* **38**, 963–981 (2008).

45. Schutski, R., Lykov, D. & Oseledets, I. Adaptive algorithm for quantum circuit simulation. *Phys. Rev. A* **101**, 042335 (2020).

46. Gray, J. & Kourtis, S. Hyper-optimized tensor network contraction. *Quantum* **5**, 410 (2021).

47. Lee, J., Magann, A. B., Rabitz, H. A. & Arenz, C. Progress toward favorable landscapes in quantum combinatorial optimization. *Phys. Rev. A* **104**, 032401 (2021).

48. Elsässer, R. & Tscheuschner, T. Settling the complexity of local max-cut (almost) completely. in *Automata, Languages and Programming* (Springer Berlin Heidelberg, 2011) https://doi.org/10.1007/978-3-642-22006-7_15 pp. 171–182.

49. Alaoui, A. E., Montanari, A. & Sellke, M. Local algorithms for maximum cut and minimum bisection on locally treelike regular graphs of large degree. Preprint at *arXiv* http://arxiv.org/abs/2111.06813 (2021).

50. Barak, B. & Marwaha, K. Classical algorithms and quantum limitations for maximum cut on high-girth graphs. https://doi.org/10.4230/LIPICS.ITCS.2022.14 (2022).

51. Wormald, N. C. The asymptotic distribution of short cycles in random regular graphs. *J. Comb. Theory Ser. B* **31**, 168–182 (1981).

52. McKay, B. D., Wormald, N. C. & Wysocka, B. Short cycles in random regular graphs. *Electron. J. Comb.* **11**, R66 (2004).

AUTHOR CONTRIBUTIONS

D.L. and J.W. performed and analyzed the experiments and wrote the main text. C.P. generated the FLIP data. M.S., T.N, and Y.A. edited the paper.

COMPETING INTERESTS

T.N. and M.S. are equity holders of and employed by ColdQuanta, a quantum technology company. J.W. is a small equity holder of and employed by QuEra Computing. The remaining authors declare no competing interests.

ADDITIONAL INFORMATION

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41534-023-00718-4.

Correspondence and requests for materials should be addressed to Danylo Lykov.

Reprints and permission information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.