Efficient Classical Computation of Quantum Mean Values for Shallow QAOA Circuits

Wei-Feng Zhuang,1,† Ya-Nan Pu,1,† Hong-Ze Xu,1 Xudan Chai,1 Yanwu Gu,1 Yunheng Ma,1 Shahid Qamar,1 Chen Qian,1 Peng Qian,1 Xiao Xiao,1 Meng-Jun Hu,1 and Dong E. Liu2,1,3,‡

1Beijing Academy of Quantum Information Sciences, Beijing 100193, China
2State Key Laboratory of Low Dimensional Quantum Physics, Department of Physics, Tsinghua University, Beijing, 100084, China
3Frontier Science Center for Quantum Information, Beijing 100184, China

(Dated: December 22, 2021)

The Quantum Approximate Optimization Algorithm (QAOA), which is a variational quantum algorithm, aims to give sub-optimal solutions of combinatorial optimization problems. It is widely believed that QAOA has the potential to demonstrate application-level quantum advantages in the noisy intermediate-scale quantum (NISQ) processors with shallow circuit depth. Since the core of QAOA is the computation of expectation values of the problem Hamiltonian, an important practical question is whether we can find an efficient classical algorithm to solve quantum mean value in the case of general shallow quantum circuits. Here, we present a novel graph decomposition based classical algorithm that scales linearly with the number of qubits for the shallow QAOA circuits in most optimization problems except for complete graph case. Numerical tests in Max-cut, graph coloring and Sherrington-Kirkpatrick model problems, compared to the state-of-the-art method, shows orders of magnitude performance improvement. Our results are not only important for the exploration of quantum advantages with QAOA, but also useful for the benchmarking of NISQ processors.

INTRODUCTION

The rapid development of quantum computing technologies in past decades has attracted a lot of interests from both academia and industrial community. In 2019, Google demonstrated the so-called “quantum supremacy” by using the 53-qubit superconducting quantum processor Sycamore [1]. Although the Google’s claim seems controversial now [2–4], stronger quantum advantage on the same problem has been verified by the USTC team with a higher quality 60-qubit superconducting quantum processor Zuchongzhi [5, 6]. These works imply that we have arrived the so-called noisy intermediate-scale quantum (NISQ) era [7], where the quantum processors contain about fifty to a few hundreds noisy qubits. Quantum error correction and fault tolerance could in principle support the quantum computation with arbitrary accuracy, but on the other hand, they require a much larger scale processor with an error rate in each physical element mitigated to an extremely low level. Unfortunately, due to high error rate of qubit operations, those NISQ processors are limited to shallow depth circuit without error corrections. In that case, a naturally important question arises: whether or not application-level quantum advantage can be demonstrated in NISQ processors.

Variational quantum algorithms (VQA), which includes variational quantum eigensolver (VQE), quantum machine learning (QML) and quantum approximate optimization algorithm (QAOA), has been widely believed as a promising approach for demonstrating NISQ applications, such as quantum chemistry [8–10], machine learning [11–13] and combinatorial optimization [14–16]. The basic idea of VQA is to estimate the cost function by sampling from parameterized quantum circuit and calls for classical optimizer to find iterative parameters until convergence condition is reached [17]. The bitstring as solution is obtained in the last stage by sampling from quantum circuit with the optimized parameters. The cost function is in general expectation value of a Hamiltonian that can be written as a linear combination of poly(n) Pauli operators, e.g., Ising Hamiltonian in the QAOA. Since the quantum circuit sampling has been shown classically intractable as the number of qubits increase [18–20], it is natural to ask whether or not quantum mean value of shallow depth circuit can be calculated efficiently by a classical computer. If the answer is positive, then NISQ processor is only necessary in the quantum sampling part of QAOA.

In a recent paper, Bravyi, Gosset and Movassagh show that there exist the classical algorithms scales linearly with the number of qubits for the special case of geometrically local two-dimensional quantum circuits [21]. Whether the quantum mean value problems can be solved efficiently on a classical computer in the case of general shallow circuits, however, remains a central open question. Here, we show that there does exist a classical algorithm capable of efficiently calculating quantum mean values in the case of general shallow QAOA circuits. The core of our algorithm is to divide a large QAOA instance into many independent small instances.
by using methods of graph decomposition and graph-to-circuit mapping. Since small instances can be handled independently, our algorithm is a natural parallel algorithm. Analysis and numerical tests show that the run time of our algorithm scales linearly with qubits number for most of optimization problems. For the special QAOA instances that its connections in problem Hamiltonian scales with size of instance, e.g. Sherrington-Kirkpatrick (S-K) model, though the run time scales exponentially, our algorithm still behaves much better than the state-of-the-art tensor network algorithm. Our results definitely show that current NISQ processors has no advantages in quantum mean value problem in QAOA.

FIG. 1. Example illustration of basic procedure of graph decomposition algorithm. a: The cost operator $\hat{C}$ is mapped to a weighed graph $G$, in which $w_{ij}$ and $h_k$ are the weight of edge $(ij)$ and bias of node $k$ respectively. According to the value of $p$, we only show $p = 1$ case here, the G can be decomposed into independent subgraphs that correspond to terms $(Z_i Z_j)_p$ and $(Z_k)_p$. The number of subgraphs equals to the number of individual terms in $\hat{C}$. b: There is an one-to-one correspondence between the subgraphs and the quantum circuits for computation of $(Z_i Z_j)_p$ and $(Z_k)_p$. Detail realization is given in Algorithm 1.

THE QUANTUM APPROXIMATE OPTIMIZATION ALGORITHM

In the seminal paper of Farhi, Goldstone, and Gutmann, the QAOA is proposed as a variational quantum algorithm to produce approximate solutions for combinatorial optimization (CO) problems [13]. Since then numerous research works on QAOA have been shown both theoretically [22][31] and experimentally [16][32][36]. Similar to quantum annealing (QA), in which CO problems are modeled as the form of Ising Hamiltonian, the QAOA also starts with Ising form as cost function. The cost function is quadratic and its general operator form reads

$$\hat{C} = \sum_{i,j} w_{ij} \hat{Z}_i \hat{Z}_j + \sum_k h_k \hat{Z}_k,$$

where $\hat{Z}_i$ are Pauli-Z operators with eigenvalues $\pm 1$, $w_{ij}$ and $h_k$ represent weights and bias respectively. If we denote computational basis vector as $|z\rangle$ with $z = z_1 z_2 \ldots z_n$ are $\{1, -1\}$ bitstrings, the $\hat{C}$ is diagonal in the computational basis vector $|z\rangle$ such that $\langle z | \hat{C} | z \rangle = C(z)$.

The bitstring $z$ that minimize (or maximize) $\langle z | \hat{C} | z \rangle$ is the optimal solution of optimization problems. Since most optimization problems are NP hard, there is still a lack of efficient algorithms. The QAOA, however, approximates the optimal solution by generating quantum state in quantum circuit that as close as possible to the optimal basis vector. This is done by introducing another operator $\hat{B} = \sum_i X_i$ and generating quantum state with $2p$ parameters as

$$|\tilde{\gamma}, \tilde{\beta} \rangle = \hat{U}_B(\beta_p) \hat{U}_C(\gamma_p) \cdots \hat{U}_B(\beta_1) \hat{U}_C(\gamma_1) |+\rangle,$$

where $\hat{U}_C(\gamma) = e^{-i \gamma \hat{C}}$, $\hat{U}_B(\beta) = e^{-i \beta \hat{B}}$, and the initial state $|+\rangle = \sum_z |z\rangle / \sqrt{2^n}$ be the uniform superposition of computational basis vectors. The expectation value of
cost operator $\hat{C}$

$$E_p(\vec{\gamma}, \vec{\beta}) = \langle \vec{\gamma}, \vec{\beta} | \hat{C} | \vec{\gamma}, \vec{\beta} \rangle$$  \hspace{1cm} (3)

is then optimized via outer-loop classical optimizer to find optimal parameters $\{\vec{\gamma}_\text{opt}, \vec{\beta}_\text{opt}\}$. Once the optimized state $|\vec{\gamma}_\text{opt}, \vec{\beta}_\text{opt}\rangle$ is obtained, we can perform sampling from the optimized quantum circuit and output the bit-string that minimized the $C(z)$ as approximate solution.

The core of QAOA is the computation of quantum mean value $E_p(\vec{\gamma}, \vec{\beta})$, and there are two different implementations. The first one, which estimates the algorithm to realize efficient calculations for shallow depth $G$, surprisingly no specific algorithm implementation is given in the follow-up study. Besides, there are no further discussions about extending graph decomposition of Max-cut to general combinatorial optimization problems. Our contribution here is connecting Ising cost operator $\hat{C}$ of optimization problems to weighted graph representation $G$ and develop specific graph decomposition based algorithm to realize efficient calculations for shallow depth QAOA circuits.

### Algorithm 1 Graph Decomposition Algorithm

**Input:** $\hat{C}$, $p$, $\{\vec{\gamma}, \vec{\beta}\}$

**Output:** $E_p(\vec{\gamma}, \vec{\beta})$

1: weighted graph $G \leftarrow \hat{C}$
2: subgraph_set $\leftarrow \emptyset$, expectation_set $\leftarrow \emptyset$
3: $E_p(\vec{\gamma}, \vec{\beta}) \leftarrow 0$
4: for nodes and edges in $G$ do
5: elements $\leftarrow \{\text{nodes or edges}\}$
6: for $i \leftarrow 1$ to $p$ do
7: elements $\leftarrow \text{elements} \cup \text{adjacency nodes and edges}$
8: subgraph $\mathcal{G}(i, j, p) \text{ or } \mathcal{G}(k, p) \leftarrow \text{elements}$
9: subgraph_set $\leftarrow \text{subgraph} \cup \text{subgraph_set}$
10: for subgraph in subgraph_set do
11: qubits number $N \leftarrow \text{node numbers of subgraph}$
12: prepare $|+\rangle^N$ initial state
13: for $i \leftarrow 1$ to $p$ do
14: for edge in subgraph do
15: $(q_m, q_n) \leftarrow \text{nodes of edge}$
16: $w_{mn} \leftarrow \text{weight of edge}$
17: apply $RZZ(\gamma_i \times \omega_{mn})$ gate to $(q_m, q_n)$
18: for node in subgraph do
19: $q_m, h_k \leftarrow \text{node, weight of node}$
20: apply $RZ(\gamma_i \times h_k)$ gate to $q_m$
21: apply $RX(\beta_i \times h_k)$ gate to $q_m$
22: expectation $\leftarrow \text{calculating } \langle \hat{Z}_i \hat{Z}_j \rangle_p \text{ or } \langle \hat{Z}_k \rangle_p$
23: expectation_set $\leftarrow \text{expectation} \cup \text{expectation_set}$
24: for expectation in expectation_set do
25: $E_p(\vec{\gamma}, \vec{\beta}) \leftarrow \sum_{ij} w_{ij} \langle \hat{Z}_i \hat{Z}_j \rangle_p + \sum_k h_k \langle \hat{Z}_k \rangle_p$

The idea of our graph decomposition is based on two observations. The first one is that any factors in the operators $\hat{U}_h(\vec{\gamma})$, $\hat{U}_g(\vec{\beta})$ that do not involve qubits $i$ or $j$ commute through $\hat{Z}_i \hat{Z}_j$, $\hat{Z}_i$, and can thus be cancelled out without affecting the calculation of $\langle \hat{Z}_i \hat{Z}_j \rangle_p$, $\langle \hat{Z}_i \rangle_p$. The other one is that the cost operator $\hat{C}$ in Eq. (1) has one-to-one correspondence with the weighted graph $G$. The $\hat{Z}_i$ corresponds to the node $i$ in $G$, while $w_{ij}$ and $h_k$ are the weight of edge $(i, j)$ and the bias of node $k$ respectively. From the above two observations, it can be concluded that individual term like $\langle \hat{Z}_i \hat{Z}_j \rangle_p$ depends only on the subgraph $G(i, j, p)$, which involves nodes $i, j$ and other nodes connecting to them with no more than $p$ distance. Also the term like $\langle \hat{Z}_k \rangle_p$ depends on the subgraph $G(k, p)$. For each subgraph $G(i, j, p)$ or $G(k, p)$, there exists an one-to-one correspondence with the quantum circuits representation for the computation of $\langle \hat{Z}_i \hat{Z}_j \rangle_p$ or $\langle \hat{Z}_k \rangle_p$, in which the qubits number equals to nodes number in subgraphs. The edge $(i, j)$ with weight $w_{ij}$ in the subgraph corresponds to two-qubit gate $e^{-i \gamma_i w_{ij} \hat{Z}_i \hat{Z}_j}$ operating on qubits $i, j$. The node $k$ with bias $h_k$ in the subgraph corresponds to one-qubit operation $e^{-i \gamma_k h_k \hat{Z}_k}$.

We give a simple example in Fig. 1 to illustrate the basic procedure of graph decomposition algorithm. The size of subgraphs is determined by $p$ and degree of connection of $G$, which is usually independent of $n$ unless the $G$ is a complete graph. The weighted graphs of optimization
problems are sparse or medium density in most cases, making the size of subgraphs not so big for small $p$. For example, in the case of 3-regular Max-cut, the subgraphs for $p=1$ contains only 6 qubits which can be easily simulated classically, regardless of the size of the problem. Since the computational complexity of graph decomposition is $o(n)$, our algorithm scales linearly with sparse density optimization problems in the case of shallow circuits. The size of subgraphs grows as $p$ increases, which increase the run time for calculation of each individual term in Eq. (4). Suppose that the weighted graph $G$ has an average degree $d = l/n$ with $l$, $n$ are total number of edges and nodes respectively. If $d$ keeps constant as $n$ grows, e.g., 3-regular Max-cut, then the size of subgraphs is constant for arbitrary finite $p$, which implies the time complexity of our algorithm be $C(d, p)o(n)$ with $C(d, p)$ is a constant depending on the value of $(d, p)$. For large $d$ or $p$, the value of $C(d, p)$ may large enough that beyond current classical simulation but the resource requirement is independent of $n$. For special case that $d$ depends on $n$, e.g., $d = n-1$ in S-K model, $C(d, p)$ now depends on $n$ with exponential complexity in general for current classical algorithms. For most optimization problems, $d$ is constant or $o(\log n)$, the run time of graph decomposition algorithm scales linearly or polynomially with nodes number $n$.

**NUMERICAL RESULTS**

The pseudo code implementation of graph decomposition algorithm is given in Algorithm 1. The input is the cost operator $\hat{C}$ and $p$, while the output is the expectation value $E_p(\vec{\gamma}, \vec{\beta})$. The first step is convert the $\hat{C}$ to corresponding weighted graph $G$, and then decompose the $G$ starting from nodes and edges of $G$ according to the value of $p$. The total number of subgraphs is the sum of numbers of nodes and edges. For each subgraph $G(i, j, p)$ or $G(k, p)$, it can be mapped to a corresponding QAOA quantum circuit for the calculation of expectation value $\langle \hat{Z}_i, \hat{Z}_j \rangle_p$ or $\langle \hat{Z}_k \rangle_p$. The simulation of quantum circuits can be implemented by kinds of method, e.g. statevector or tensor contraction. At last the $E_p(\vec{\gamma}, \vec{\beta})$ is obtained by adding every terms $\langle \hat{Z}_i, \hat{Z}_j \rangle_p$ or $\langle \hat{Z}_k \rangle_p$. Based on Algorithm 1, we have developed a quantum software package named QCover [38] to help finding optimal parameters of
shallow QAOA circuits more faster than existing software.

To test our algorithm, we perform numerical experiments on three classical optimization problems, i.e. 3-regular Max-cut, graph coloring and S-K model by comparing the QCover with the IBM Qiskit [39] and the Quimb [40]. The IBM Qiskit uses statevector method to realize QAOA simulation, while the Quimb takes the method of tensor network contraction. The most important part of tensor method is to search the optimal contraction path. The Quimb provides many heuristic algorithms to determine contraction path and we choose the best one in our instance tests. The engine of the QCover to compute subgraphs could be statevector or tensor, depending on the characteristics of subgraphs. For example, statevector engine is more efficient than tensor one in the case of 3-regular Max-cut with $p = 1$, while it is better to use tensor engine for $p \geq 2$. The QCover can automatically choose the best engine to complete the computation. The cost operator of Max-cut and S-K are given with the same form

$$\hat{C} = \sum_{i \neq j} w_{ij} \hat{Z}_i \hat{Z}_j,$$

where $w_{ij} = 1$ for Max-cut and $w_{ij}$ is randomly chosen to be $\pm 1$ for S-K model. For graph coloring problem, we first model it with quadratic unconstrained binary optimization (QUBO) form and then transformed it into Ising cost form $\hat{C}$ [41]. The QCover has an instance library to automatically generate the weighted graph $G$ corresponding to $\hat{C}$ for above three problems. The weighted graph $G$ of S-K model is a complete graph [32], while it is constant density graph for Max-cut or graph coloring problem [41]. The size of subgraphs for complete graph equals to $G$ even for $p = 1$ case, which constrains the calculations for small size. For constant density graph, the size of subgraphs depend only on the density and the value of $p$, which makes us capable of dealing with large size instances given that the size of subgraphs is within the scope of computing power. The Fig. 2 shows the test results in which horizontal axis represents the node numbers (or qubit numbers) of weighted graph $G$ and longitudinal axis represents the time to complete calculation of $E_p(\vec{\gamma}, \vec{\beta})$ with $(\vec{\gamma}, \vec{\beta})$ is given randomly. For each data, we show the average time value of 5 independent tests. The QCover scales linearly with the node numbers in Max-cut and graph coloring problems as shown in sub-figures (a), (b) and (c) of Fig. 2, while the qiskit and Quimb behave exponentially. For S-K model result shown in the sub-figure (d) of Fig. 2, the QCover also has better performance but scales exponentially due to the size of subgraphs equal to the node numbers. It should be noted here that all data from the QCover are obtained from the personal computer run without using parallel computing. Since the subgraphs obtained from weighted graph can be handled individually with different computing cores, the QCover supports high performance parallel computing and better speed-up can be achieved.

**DISCUSSION AND CONCLUSION**

Although we have shown quantum mean values for shallow QAOA circuits can be efficiently solved by classical computers, it remains an open question for other VQAs, e.g., VQE and QML. Besides, the sampling in the last stage of QAOA, which has been shown classically intractable, still requires NISQ processor. Our algorithm and software, however, can be used as a powerful classical-assisted tool to help finding and realizing the possible optimization problems that is suitable to demonstrate application-level quantum advantage in NISQ processor. It is also helpful of our algorithm for the verification and benchmarking of NISQ computers. Since QAOA deals with Ising Hamiltonian, our algorithm has the potential to assist realizing fast approximate ground state preparation of arbitrary long-range Ising-type Hamiltonian [34].

In summary, we have present a graph decomposition algorithm to realize efficient classical computation of quantum mean values for shallow QAOA circuits. For most optimization problems, the run time of our algorithm scales linearly with the instance size in the case of shallow circuits. When combined with high-performance parallel computing in the next stage, the performance of our algorithm can be further significantly improved. Our algorithm and the related software Qcover, when used with NISQ processor, may accelerate the demonstration of application-level quantum advantage.

**Acknowledgments:** The authors would like to thank Bai-Ting Liu, Xingyao Wu, Diqing Chen and Yuxuan Wang for valuable discussions. This work is supported by the Beijing Academy of Quantum Information Sciences. Data availability: All codes and data can be found in GitHub https://github.com/BAQIS-Quantum/Qcover.

[1] F. Arute et al., Quantum supremacy using a programmable superconducting processor, Nature 574, 505-510 (2019).
[2] Cupjin Huang et al., Classical simulation of quantum supremacy circuits, arXiv:2005.06787 (2020).
[3] Feng Pan and Pan Zhang. Simulating the sycamore quantum supremacy circuits, arXiv:2103.03074 (2021).
[4] Yong Liu et al., Closing the “Quantum Supremacy” Gap: Achieving Real-Time Simulation of a Random Quantum Circuit Using a New Sunway Supercomputer, arXiv:2110.14502 (2021).
[5] Yulin Wu et al., Strong Quantum Computational Advantage Using a Superconducting Quantum Processor, Phys. Rev. Lett. 127, 180501 (2021).
