Highlights

QAOA with fewer qubits: a coupling framework to solve larger-scale Max-Cut problem

Yiren Lu, Guojing Tian, Xiaoming Sun

- Proposed a reformulation of Max-Cut problem that is suitable for using quantum and classical algorithms jointly to solve larger-scale Max-Cut problems.
- Devised a framework that couples QAOA with a classical approximation algorithm to solve Max-Cut problem.
- Analytically confirmed that if QAOA has a provable higher approximation ratio than classical algorithms, then the quantum advantage can be extended to larger-scale problems.
- Numerically tested our framework by coupling QAOA with a heuristic algorithm and achieved better results than previous methods.

QAOA with fewer qubits: a coupling framework to solve larger-scale Max-Cut problem

Yiren Lu[a,b], Guojing Tian[a,b,1,], Xiaoming Sun[a,b]

[a]*State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China*
[b]*School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China, Beijing, 100049, China*

Abstract

Maximum cut (Max-Cut) problem is one of the most important combinatorial optimization problems because of its various applications in real life, and recently Quantum Approximate Optimization Algorithm (QAOA) has been widely employed to solve it. However, as the size of the problem increases, the number of qubits required will become larger. With the aim of saving qubits, we propose a coupling framework for designing QAOA circuits to solve larger-scale Max-Cut problems with conditional performance guarantee and nice numerical performance. To do this, we introduce a noval reformulation of the Max-Cut problem, and the problem will be partly solved using a classical approximation algorithm which is embedded into the QAOA circuit. Notably, our formalism makes it possible to derive an approximation guarantee theoretically assuming the approximation ratio of the classical algorithm and QAOA. The analytical results show that if QAOA has a provable quantum advantage, namely a better approximation ration than the classical algorithms, then the quantum advantage can be extended to problems with sizes much beyond available qubit numbers. Furthermore, we design a heuristic approach that fits in our framework and perform sufficient numerical experiments, where we solve Max-Cut on various 24-vertex Erdős-Rényi graphs. Our framework only consumes 18 qubits and achieves 0.9950 approximation ratio on average, which outperforms the previous methods showing 0.9778 (quantum algorithm using the same number of qubits) and 0.9643 (classical algorithm). The numerical

[1]corresponding author

results indicate our well-designed quantum-classical coupling framework gives satisfactory approximation ratios while reducing the qubit cost, allowing us to extend the potential quantum advantage of QAOA to larger-scale Max-Cut problems, which sheds light on more potential computing power of NISQ devices.

Keywords: variational quantum algorithm, quantum approximate optimization algorithm, Max-Cut, quantum resources

---

1. Introduction

The Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum-classical approach for combinatorial optimization problems, proposed by Farhi *et al.* [1]. Here, *hybrid* means that QAOA is composed of a parameterized circuit (ansatz) whose measurement outcome can be mapped to the value of the objective function, and the parameters are updated with a classical optimizer that maximizes (or minimizes) the expected value of the objective function. Delegating some computational processes to the classical domain grants QAOA a comparatively shallower circuit, making it more applicable to NISQ [2, 3, 4] devices. Also, as the circuit for QAOA is a parameterized one, it is easier to be modified and adapted to different types of quantum computers [5]. Moreover, the quantum gates required to implement a QAOA circuit are comparatively simple. For instance, the QAOA circuit for solving the maximum cut (Max-Cut) problem only comprises CNOT and single qubit rotation gates, which are much more friendly to physical realization than Toffoli gates, etc. Therefore, QAOA is more compatible with NISQ devices than other complex quantum algorithms such as Shor's algorithm [6], Grover's algorithm [7].

Since its introduction, QAOA has attracted considerable theoretical and experimental attention [8, 9, 10, 11, 12, 13]. Starting from the original paper of QAOA [1], Max-Cut problem has always been used as a good example to demonstrate QAOA's capabilities. Max-Cut problem is a classical NP-hard problem in graph theory and has applications in statistical physics and integrated circuit design. Furthermore, as a combinatorial optimization problem with simple constraints and objective function, solutions to Max-Cut problem can be potentially generalized to other hard optimization problems [14]. This paper also investigates how to solve the Max-Cut problem with QAOA. When using QAOA to solve Max-Cut problem on a graph,

the number of qubits required is usually equal to the number of vertices in this graph. But the number of available qubits seems to be constrained in the near future. First of all, the qubits are quite expensive currently, at a cost of tens of thousands of dollars per qubit [15, 16]. What's worse, the scaling of qubit count is a big challenge, as technologies used to realize quantum computers (ion traps, superconductors, etc) have strong limitations on the scaling of the number of qubits [17, 18, 19, 20]. Thus, it is a good question whether we can reduce the number of qubits required to deal with a specific problem instance. We thus works on this direction, and we propose a coupling framework to solve Max-Cut problem with fewer qubits.

1.1. Related work

With the aim of solving Max-Cut with fewer qubits, efforts have been made in [21, 22], where the authors split the graph into several subgraphs and solve Max-Cut on the subgraphs independently, and then combine the cuts of subgraphs into a global cut. In the combination step, both methods only keep a *small* amount of candidate solutions, determined by their optimality inside the subgraph, and then try to select an optimal combination of the kept candidate solutions in subgraphs to maximize the overall cut. These methods allow for using an arbitrary small number of qubits for a given problem instance, by recursively partitioning the graph until the number of vertices is smaller than the number of available qubits. Moreover, for graphs innately composed of several dense subgraphs connected by a few edges, such splitting and combining strategy is reasonable for the cut is largely determined by edges inside the dense subgraphs, and the edges joining the subgraphs are relatively insignificant. But for general cases, Max-Cut is not a problem with much locality — a cut may not be so good in a subgraph, but can be a part of a nice solution for the whole graph. Therefore, due to the existence of such bold greedy reduction that discards locally bad solutions without much analysis, it is hard to mathematically derive a strong performance guarantee for such methods.

Considering the issues of previous works mentioned above, we propose a method that incorporates a classical approximation algorithm as part of the QAOA circuit, leaving some vertices and edges to be handled by classical computation. By combining quantum and classical computation, our method applies to more general graph instances and achieves better numerical results than previous works that only used quantum computation. In

addition to numerical results, we also derive a formula that lower bounds the performance guarantee with some mathematical analysis, assuming the approximation ratio of the classical algorithm and the quantum approximate optimization. Moreover, as we want to keep the overall method at least better than the classical one, we also analyze the portion of qubits that can be saved while this condition holds.

This paper is organized as follows. Section 2 reviews the definition and computational hardness of Max-Cut, along with the overall scheme of QAOA and the QAOA circuit for Max-Cut. Section 3 presents a general framework of our method and analyzes its performance. Section 4 discusses the implementation of the classical algorithm required in this framework and gives the simulation results compared with previous works, and Section 5 concludes with a summary of our results and some further discussions.

2. Preliminaries

This section reviews the definition of the maximum cut (Max-Cut) problem and the structure of QAOA, which are fundamental to the following parts.

2.1. Max-Cut problem

As a well-known NP-hard problem, Max-Cut problem is described as follows:

Problem: **Maximum Cut**
Input: An undirected graph $G = (V, E)$.
Output:

$$\max_{S \subset V} \sum_{u \in S} \sum_{v \in V - S} [(u, v) \in E], \tag{1}$$

where $(S, V - S)$ is a partition of V.

Note that the output can also be alternatively defined as

$$\max_{\sigma: V \to \{0,1\}} \sum_{(u,v) \in E} [\sigma(u) \neq \sigma(v)], \tag{2}$$

where σ is a binary coloring of vertices. Note that such a coloring can be represented using a binary string $s \in \{0,1\}^n$, where s_i denotes the color of the i-th vertex.

α_{opt}	Assumption
$\approx .878$ [26]	$P \neq NP$
(Goemans-Williamson algorithm)	*unique games conjecture*
$\approx .941$ [30, 31]	$P \neq NP$

Table 1: Approximation ratio upper bound, denoted by α_{opt}, for classical algorithms on Max-Cut.

No polynomial time algorithm solves Max-Cut accurately, so efficient algorithms proposed to solve Max-Cut problem only give approximate solutions [23, 24, 25, 26, 27, 28, 29]. Usually, these approaches have been benchmarked by the approximation ratio of obtained solutions, which is the value of the obtained solution divided by the optimal solution. In classical computation, the approximation ratio of any efficient algorithm is upper bounded under currently unfalsified complexity conjectures, see Table 1. However, it is worth mentioning that the upper bounds apply to the worst-case approximation ratio on any graph. In other words, for any efficient classical algorithm, there will be an infinite set of hard cases where it cannot do better than α_{opt}. But in average cases, classical algorithms usually give much better results (with an approximation ratio higher than .878). For example, the best-known algorithm for general cases, Goemans-Williamson algorithm, achieves no less than .878 on any graph, but on Erdős-Rényi random graphs, it usually shows over .9-approximation in experiments. So the reader might not be confused when seeing such cases, and when comparing a quantum algorithm with classical ones numerically, we need to do case-by-case comparisons, as indicated in [32]. Now we describe two classical algorithms which will be mentioned in the following sections.

Naive random algorithm. For a graph G, we independently assign each vertex a random color. This results in a random algorithm with $\frac{1}{2}$ approximation ratio, namely the expected output is at least half of the optimal solution.

Local search. A local search algorithm starts from an arbitrary feasible solution to the problem and incrementally modifies the solution until reaching a local optimum. For the case of Max-Cut, a local search algorithm might

look like Algorithm 2. In the pseudo code, $S \triangle \{u\}$ means symmetric difference of S and $\{u\}$, i.e., removing u from S if $u \in S$ or add u to S if $u \notin S$. The procedure is bound to stop in polynomial iterations and guarantees to give a solution with at least $\frac{1}{2}$ approximation ratio. Though local search

Algorithm 1: Naive Max-Cut algorithm

Data: A graph $G = (V, E)$.
Result: A cut that achieves expected $\frac{1}{2}$-approximation.
$S \leftarrow \emptyset$;
for $u \in V$ **do**
  u is added to S with $\frac{1}{2}$ probability;
end
return $(S, V - S)$;

Algorithm 2: Local search for Max-Cut

Data: A graph $G = (V, E)$.
Result: A cut that achieves $\frac{1}{2}$-approximation.
$S \leftarrow S_0$; `/* initial cut solution */`
while $\exists u, \text{CUT}(S \triangle \{u\}) > \text{CUT}(S)$ **do**
  $S \leftarrow S \triangle \{u\}$;
end
return $(S, V - S)$;

does not theoretically ensure a high approximation ratio, it does demonstrate good performance in experiments. Moreover, it is quite simple and scalable, so in later sections, we implement our heuristic method on top of it.

2.2. QAOA

Besides the classical algorithms for Max-Cut problem, some QAOA-based quantum algorithms have been introduced recently. QAOA is a quantum-classical hybrid method for combinatorial optimization, solving problems that look as follows [1]:

$$\text{Given } C : \{0,1\}^n \mapsto \mathbb{R}, \text{ find } \max_{s \in \{0,1\}^n} C(s).$$

This cost function C can be encoded into a problem Hamiltonian

$$H_C = \sum_{s \in \{0,1\}^n} C(s) |s\rangle\langle s|, \tag{3}$$

and in QAOA, the combinatorial optimization problems are reformulated into approximately finding the ground state and ground energy of $-H_C$, which is done with a hybrid optimization procedure. Firstly, prepare the parameterized unitary transformation, which can generate the following state when operating on $|+\rangle^{\otimes n}$, i.e.,

$$|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle = \left(\prod_{k=1}^{p} \exp(-\mathrm{i}\beta_k H_B) \exp(-\mathrm{i}\gamma_k H_C) \right) |+\rangle^{\otimes n}, \tag{4}$$

with $H_B = \sum_{i=1}^{n} X_i$, p being the number of layers for QAOA and $\boldsymbol{\beta}, \boldsymbol{\gamma} \in [0, 2\pi)^p$ being parameter vectors. Secondly, update the parameters $\boldsymbol{\beta}, \boldsymbol{\gamma}$ using a classical optimizer to maximize the expected measurement outcome,

$$(\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*) = \underset{\boldsymbol{\beta}, \boldsymbol{\gamma}}{\operatorname{argmax}} \langle \boldsymbol{\beta}, \boldsymbol{\gamma} | H_C | \boldsymbol{\beta}, \boldsymbol{\gamma} \rangle. \tag{5}$$

Thus, the approximation ratio will be

$$\frac{\langle \boldsymbol{\beta}^*, \boldsymbol{\gamma}^* | H_C | \boldsymbol{\beta}^*, \boldsymbol{\gamma}^* \rangle - \min_s C(s)}{\max_s C(s) - \min_s C(s)}. \tag{6}$$

Before explaining how to realize the above process in the language of quantum circuits, it is necessary to review some elementary quantum gates we will use in our paper, including 1-qubit gates R_X (rotation along X-axis), R_Z (rotation along Z-axis), H (Hadamard gate),

$$R_X(\theta) = \begin{pmatrix} \cos(\theta/2) & -\mathrm{i}\sin(\theta/2) \\ -\mathrm{i}\sin(\theta/2) & \cos(\theta/2) \end{pmatrix},$$
$$R_Z(\theta) = \begin{pmatrix} \mathrm{e}^{-\mathrm{i}(\theta/2)} & 0 \\ 0 & \mathrm{e}^{\mathrm{i}(\theta/2)} \end{pmatrix},$$
$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

(where $\theta \in \mathbb{R}$ is a parameter denoting rotation angle) and 2-qubit controlled-NOT gate along with its circuit notation Figure 1(a)

$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Now we show how to construct the unitaries above required by QAOA. The $\exp(-\mathrm{i}\beta_k H_B)$ part is straightforward as it is simply a layer of $R_X(\beta_k)$ gates. Now we consider $\exp(-\mathrm{i}\gamma_k H_C)$. When solving Max-Cut problem with QAOA, we use an $s \in \{0,1\}^n$ to encode the coloring of vertices, and $C(s)$ would be the number of edges whose endpoints are colored differently.

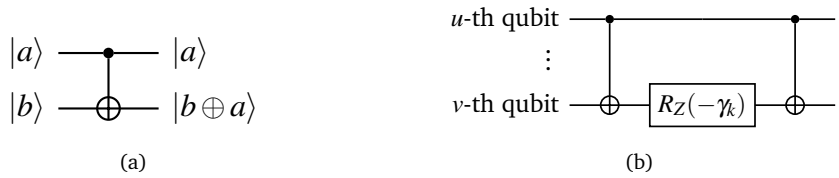


Figure 1: (a) Circuit notation of CNOT gate, which means the target qubit (marked with $\bigoplus$) is flipped iff the control qubit (marked with $\bullet$) is in state $|1\rangle$. (b) Adding a phase factor depending on whether two qubits u, v share the same value.

And for such $C(s)$, $\exp(-\mathrm{i}\gamma_k H_C)$ can be implemented by applying the circuit shown in Figure 1(b) for all edges (u,v). Apparently, the circuit in Figure 1(b) add phase factors depending on whether x-th and the y-th qubits are in different states, and the phase factors thus accumulate to the whole cut size (up to a global phase factor). Actually, Max-Cut problem has an equivalent Ising formulation

$$\sum_{(u,v)\in E} Z_u Z_v. \tag{7}$$

The authors of [1] have proved that the approximation ratio approaches 1 when p tends to infinity. But with moderate circuit depth, QAOA hasn't been proved theoretically to achieve a better approximation ratio than best-known classical algorithms. However, in numerical experiments, QAOA does seem to show nice performance in shallow circuit depth. For example, Crooks [32] shows for Erdős-Rényi graphs with up to 17 vertices, QAOA significantly outperforms Goemans-Williamson algorithm when the hyper-parameter p grows to 8, and the approximation ratio is very close to 1 when p is set 32. Therefore, it is promising to achieve quantum advantage with QAOA, namely, obtain better cut solutions on certain problem instances than efficient classical algorithms. In the following sections, we describe our coupling framework which embeds a classical approximation algorithm with QAOA which will be able to use fewer qubits to solve Max-Cut problem. We demonstrate its performance theoretically and numerically. Due to the good numerical performance of QAOA, our framework indeed gives good results in numerical experiments and outperforms currently the best-known classical algorithm.

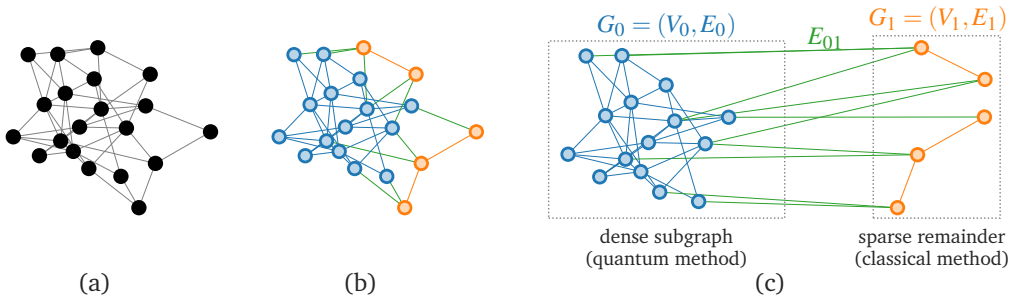


Figure 2: An example of graph splitting. (a) The original graph. (b) Select a dense graph indicated by blue vertices. (c) Split the graph and use different methods on different parts. It is clear from (c) that the blue subgraph is comparatively denser.

3. Theoretical results

Despite the proof of quantum advantage for the original QAOA referred in [1] being hard, for a new coupling framework below we can give some theoretical results assuming the approximation ratio of QAOA is known. Firstly we give a reformulation of Max-Cut and a new variant of the Max-Cut that leads to our new framework. Then we present details of our framework and give some performance analysis.

3.1. Framework description

When given a Max-Cut problem, if we use QAOA to solve it, the number of qubits required is equal to the number of vertices in the graph. Therefore, to solve it with fewer qubits, we need to reduce the number of vertices in the graph by selecting a subgraph and solving Max-Cut with QAOA on it, and for the remaining part we use classical algorithms. To formalize this process, we define the notations in Table 2. An exemplary demonstration of this process can be seen in Figure 2.

Now suppose we select a subgraph of size n_0. Once we fix a coloring $s_0 \in \{0,1\}^{n_0}$ of V_0, the cut size in E_0 is fixed, and we denote it with $\mathrm{CUT}_0(s_0)$. Now, choosing different coloring $s_1 \in \{0,1\}^{n_1}$ of V_1 will result in different cut sizes in the remaining edges of the graph (or formally, $E - E_0$), defined as

$$\mathrm{REM}(s_0, s_1) = \mathrm{CUT}_{01}(s_0, s_1) + \mathrm{CUT}_1(s_1), \tag{8}$$

where $\mathrm{CUT}_{01}(s_0, s_1)$ denotes the number of cut edges in E_{01}, and $\mathrm{CUT}_1(s_1)$ denotes the number of cut edges in E_1. For each s_0, there will be an optimal

Symbol	Meaning
$G_0 = (V_0, E_0)$	the subgraph we select
$V_1 = V \setminus V_0$	the remaining vertices
$n_1 = \lvert V_1 \rvert$	the number of remaining vertices
$G_1 = (V_1, E_1)$	the induced subgraph by V_1
$E_{01} = E \setminus (E_0 \cup E_1)$	the edges between V_0 and V_1

Table 2: Notations for the graph splitting.

s_1 that maximizes the size of the cut in the remaining edges of the graph, and we define this maximum as

$$\textsc{opt-rem}(s_0) = \max_{s_1 \in \{0,1\}^{n_1}} \textsc{rem}(s_0, s_1). \tag{9}$$

With the definitions above, Max-Cut problem is now reformulated into

$$\textsc{Max-Cut}(G) = \max_{s_0 \in \{0,1\}^{n_0}} \left\{ \textsc{cut}_0(s_0) + \textsc{opt-rem}(s_0) \right\}. \tag{10}$$

Our framework will rely on a classical algorithm that approximately computes $\textsc{opt-rem}(s_0)$, which will be used in the QAOA circuit to solve (10). In the following text, we will use $\textsc{opt-rem'}(s_0)$ to denote the approximate solution of $\textsc{opt-rem}(s_0)$, and suppose we have access to the classical algorithm as an oracle shown below (more discussions on this classical oracle are given in Section 4),

$$O_{\textsc{opt-rem'}} : |s_0\rangle|0\rangle^{\otimes \ell} \mapsto |s_0\rangle \left| \textsc{opt-rem'}(s_0) \right\rangle. \tag{11}$$

Here, we need to store the largest possible $\textsc{opt-rem'}(s_0)$ on the second ℓ-qubit register, thus ℓ should be set $\left\lceil \log\left(|E| - |E_0|\right) \right\rceil \in O(\log n)$.

Remember in QAOA, to optimize $\max_{s \in \{0,1\}^n} C(s)$, we need to construct the problem unitary, which implements

$$|s\rangle \mapsto \mathrm{e}^{-\mathrm{i}\gamma_k \cdot C(s)} |s\rangle. \tag{12}$$

Accordingly, here, a problem unitary for (10) would be composed of two steps:

$$|s_0\rangle \xmapsto{\text{step 1}} \mathrm{e}^{-\mathrm{i}\gamma_k \cdot \textsc{cut}_0(s_0)} |s_0\rangle \tag{13}$$

$$\xmapsto{\text{step 2}} \mathrm{e}^{-\mathrm{i}\gamma_k \cdot \textsc{opt-rem'}(s_0)} \cdot \mathrm{e}^{-\mathrm{i}\gamma_k \cdot \textsc{cut}_0(s_0)} |s_0\rangle. \tag{14}$$

Step 1. The unitary $|s_0\rangle \mapsto \mathrm{e}^{-\mathrm{i}\gamma_k \cdot \mathrm{CUT}_0(s_0)}|s_0\rangle$ is the same as that in the original QAOA circuit for Max-Cut, and can be implemented by applying the circuit Figure 1(b) for all $(u,v) \in E_0$.

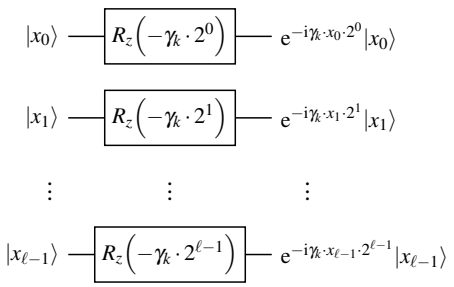


Figure 3: The implementation of phase oracle.

Step 2. With the oracle defined in (11), we can implement the required phase oracle

$$|s_0\rangle\left|\text{OPT-REM'}(s_0)\right\rangle \mapsto \mathrm{e}^{-\mathrm{i}\gamma_k \cdot \text{OPT-REM'}(s_0)}|s_0\rangle\left|\text{OPT-REM'}(s_0)\right\rangle,$$

using only rotation gates, as shown in Figure 3. And the overall circuit requires $n_0 + \left\lceil \log\left(|E| - |E_0|\right)\right\rceil = n_0 + O(\log n)$ qubits.

Finally, we discuss what kind of subgraph should be selected and solved using QAOA. When solving the Max-Cut problem on a graph, clearly a denser subgraph tends to have a larger impact on the overall solution. Moreover, generally speaking, Max-Cut on dense graphs is harder than it is on sparse graphs. This intuition is supported by the following facts:

- Planar graph, being sparse, is the most well-known class of graphs where Max-Cut can be solved in polynomial time [33].
- There are parameterized and exact algorithms for Max-Cut whose running times are positively correlated with graph density. For example, $2^{(1-(2/\Delta))n} \cdot \mathrm{poly}(n)$ where Δ is the maximum degree of all vertices [34], $2^k \cdot \mathrm{poly}(n)$ where k is the number of crossings in a graph's drawing (meaning that when a graph is drawn, there will be k crossings between the edges)[35]. The denser the graph is, the longer time it takes for these algorithms to find its Max-Cut.

Based on all the observations above, given a graph instance, we will select a dense subgraph and use QAOA to deal with it and classical heuristics to handle the remaining part. This can help us overcome the difficult part of the Max-Cut with quantum computation and tackle the easy part with classical computation, focusing our quantum resources on the hard core of the problem.

3.2. Performance analysis

Next, we will analyze the performance of the above framework in terms of approximation ratio and the size of the chosen subgraph (which determines the number of saved qubits).

Suppose the classical oracle achieves α_{C}-approximation ratio $(0 < \alpha_{\mathrm{C}} \leq 1)$, that is, for all choices of s, it is guaranteed to output a solution $\textsc{opt-rem'}(s)$ such that

$$\frac{\textsc{opt-rem'}(s)}{\textsc{opt-rem}(s)} \geq \alpha_{\mathrm{C}}, \tag{15}$$

or equivalently

$$\frac{\textsc{opt-rem'}(s)}{\textsc{opt-rem}(s)} = \alpha_{\mathrm{C}} + \varepsilon_s \tag{16}$$

with $\varepsilon_s \geq 0$. As the classical oracle is unable to find the exact solution, the QAOA procedure in our framework might not be able to discover the optimal solution, as it now tries to solve the following optimization problem:

$$\max_s \{a_s + (\alpha_{\mathrm{C}} + \varepsilon_s) b_s\}, \tag{17}$$

where $a_s = \textsc{cut}_0(s), b_s = \textsc{opt-rem}(s)$. Suppose the QAOA part solves (17) in α_{Q}-approximation ratio $(0 < \alpha_{\mathrm{Q}} \leq 1)$ and produces expected answer A_{Q}, which means

$$\frac{A_{\mathrm{Q}}}{\max_s \{a_s + (\alpha_{\mathrm{C}} + \varepsilon_s) b_s\}} \geq \alpha_{\mathrm{Q}}, \tag{18}$$

then we have

Lemma 1 (Approximation ratio lower bound)**.** *When we use coupling framework with α_Q-approx QAOA and α_C-approx classical algorithm, the approximation ratio for the overall method α_{CQ} satisfies*

$$\alpha_{CQ} \geq \alpha_Q \cdot \left(1 + (\alpha_C - 1) \frac{\max_s \{b_s\}}{\max_s \{a_s + b_s\}}\right), \tag{19}$$

where $a_s = \textsc{cut}_0(s), b_s = \textsc{opt-rem}(s)$.

Proof.

$$\alpha_{\mathsf{CQ}} = \frac{A_{\mathsf{Q}}}{\max_s\{a_s + b_s\}} \tag{20}$$

$$= \frac{A_{\mathsf{Q}}}{\max\{a_s + (\alpha_{\mathsf{C}} + \varepsilon_s) b_s\}} \cdot \frac{\max\{a_s + (\alpha_{\mathsf{C}} + \varepsilon_s) b_s\}}{\max_s\{a_s + b_s\}} \tag{21}$$

$$\geq \alpha_{\mathsf{Q}} \cdot \frac{\max\{a_s + (\alpha_{\mathsf{C}} + \varepsilon_s) b_s\}}{\max_s\{a_s + b_s\}} \tag{22}$$

$$\geq \alpha_{\mathsf{Q}} \cdot \frac{\max_s\{a_s + \alpha_{\mathsf{C}} b_s\}}{\max_s\{a_s + b_s\}} \tag{23}$$

$$\geq \alpha_{\mathsf{Q}} \cdot \frac{\max_s\{a_s + b_s\} + \min_s\{(\alpha_{\mathsf{C}} - 1) b_s\}}{\max_s\{a_s + b_s\}} \tag{24}$$

$$= \alpha_{\mathsf{Q}} \cdot \left(1 + (\alpha_{\mathsf{C}} - 1) \frac{\max_s\{b_s\}}{\max_s\{a_s + b_s\}}\right), \tag{25}$$

where (24) is due to $\alpha_{\mathsf{C}} - 1$ being negative so the max operation flips to min. □

As the framework embeds a α_{C}-approximation algorithm as a subprogram of an α_{Q}-approximation algorithm, the overall approximation ratio would be lower than α_{Q} (otherwise we magically save the number of qubits with no cost). But as a basic requirement, we need to ensure the overall framework achieves better performance than the classical algorithm (otherwise we can just use the classical algorithm to solve Max-Cut and dump the quantum part). The following lemma gives a sufficient condition to ensure that requirement, namely, $\alpha_{\mathsf{CQ}} > \alpha_{\mathsf{C}}$.

Lemma 2 (Sufficient condition for quantum advantage)**.** *If we select a dense subgraph of n_0 vertices satisfying*

$$\frac{n_0(n_0-1)}{n(n-1)} > 1 - \frac{1}{2}\left(\frac{1}{\alpha_Q} + \frac{1}{1-\alpha_C} - \frac{1}{\alpha_Q(1-\alpha_C)}\right), \tag{26}$$

then $\alpha_{CQ} > \alpha_C$.

To prove the lemma above, we need the following proposition.

Proposition 1 (Number of edges in a random subgraph)**.** *For a graph $G = (V,E)$, when we select a random subgraph $G_0 = (V_0, E_0)$ with $|V_0| = n_0$, we have*

$$\mathbb{E}(|E_0|) = |E| \cdot \frac{n_0 \cdot (n_0 - 1)}{n \cdot (n-1)}. \tag{27}$$

Proof. There are $\binom{n}{n_0}$ subgraphs sized n_0.

For each edge, it is included in a subgraph if and only if the subgraph contains its two endpoints, so it is included in $\binom{n-2}{n_0-2}$ subgraphs. So the total number of edges in the $\binom{n}{n_0}$ subgraphs mentioned above is

$$|E| \binom{n-2}{n_0-2}, \tag{28}$$

giving

$$\mathbb{E}(|E_0|) = \frac{|E|\binom{n-2}{n_0-2}}{\binom{n}{n_0}} = |E| \cdot \frac{n_0(n_0-1)}{n(n-1)}. \tag{29}$$

□

Now we give the proof of Lemma 2.

Proof. (of Lemma 2) To have the right hand side of (25) no less than α_{C}, we need

$$\frac{\max_s\{b_s\}}{\max_s\{a_s + b_s\}} < \frac{\alpha_{\mathsf{Q}} - \alpha_{\mathsf{C}}}{\alpha_{\mathsf{Q}}(1-\alpha_{\mathsf{C}})} = \frac{1}{\alpha_{\mathsf{Q}}} + \frac{1}{1-\alpha_{\mathsf{C}}} - \frac{1}{\alpha_{\mathsf{Q}}(1-\alpha_{\mathsf{C}})}.$$

Meanwhile, we have

$$\max_s\{b_s\} \leq |E - E_0|, \tag{30}$$

$$\max_s\{a_s + b_s\} = \textsc{Max-Cut}(G) \geq \frac{|E|}{2}, \tag{31}$$

$$\Rightarrow \frac{\max_s\{b_s\}}{\max_s\{a_s+b_s\}} \leq \frac{|E-E_0|}{\frac{|E|}{2}}, \tag{32}$$

so to ensure $\alpha_{\mathsf{CQ}} > \alpha_{\mathsf{C}}$, it suffices to satisfy

$$\frac{|E-E_0|}{|E|} < \frac{1}{2}\left(\frac{1}{\alpha_{\mathsf{Q}}} + \frac{1}{1-\alpha_{\mathsf{C}}} - \frac{1}{\alpha_{\mathsf{Q}}(1-\alpha_{\mathsf{C}})}\right), \tag{33}$$

that is,

$$\frac{|E_0|}{|E|} > 1 - \frac{1}{2}\left(\frac{1}{\alpha_{\mathsf{Q}}} + \frac{1}{1-\alpha_{\mathsf{C}}} - \frac{1}{\alpha_{\mathsf{Q}}(1-\alpha_{\mathsf{C}})}\right). \tag{34}$$

From Proposition 1, we can obtain a lower bound for $|E_0|/|E|$ that is related to n_0, because since we have the expected number of edges for a random subgraph with n_0 vertices, we can assert there are comparatively denser subgraphs with at least $\mathbb{E}(|E_0|)$ edges. We can find such a subgraph easily, then we have

$$|E_0| \geq |E| \cdot \frac{n_0(n_0-1)}{n(n-1)} \Rightarrow \frac{|E_0|}{|E|} \geq \frac{n_0(n_0-1)}{n(n-1)}. \tag{35}$$

Thus, to keep $\alpha_{\mathsf{CQ}} > \alpha_{\mathsf{C}}$ to remain advantage over the classical algorithm, it suffices to ensure

$$\frac{n_0(n_0-1)}{n(n-1)} > 1 - \frac{1}{2}\left(\frac{1}{\alpha_{\mathsf{Q}}} + \frac{1}{1-\alpha_{\mathsf{C}}} - \frac{1}{\alpha_{\mathsf{Q}}(1-\alpha_{\mathsf{C}})}\right). \tag{36}$$

□

We observe the right-hand side of (26) is considerably small, see Table 3. For example, if the classical approximation ratio $\alpha_{\mathsf{C}} = 0.89$ and the

α_{Q} \ α_{C}	0.85	0.87	0.89	0.91	0.93	0.95	0.97	0.99
0.85	1.0							
0.87	0.92	1.0						
0.89	0.85	0.91	1.0					
0.91	0.78	0.83	0.9	1.0				
0.93	0.71	0.75	0.8	0.88	1.0			
0.95	0.65	0.68	0.71	0.77	0.85	1.0		
0.97	0.59	0.6	0.63	0.66	0.71	0.79	1.0	
0.99	0.53	0.53	0.54	0.55	0.57	0.6	0.66	1.0

Table 3: The value of $1 - \frac{1}{2}\left(\frac{1}{\alpha_{\mathsf{Q}}} + \frac{1}{1-\alpha_{\mathsf{C}}} - \frac{1}{\alpha_{\mathsf{Q}}(1-\alpha_{\mathsf{C}})}\right)$.

QAOA approximation ratio $\alpha_{\mathsf{Q}} = 0.95$, then $1 - \frac{1}{2}\left(\frac{1}{\alpha_{\mathsf{Q}}} + \frac{1}{1-\alpha_{\mathsf{C}}} - \frac{1}{\alpha_{\mathsf{Q}}(1-\alpha_{\mathsf{C}})}\right) = 0.71$ and n_0/n approaches 85% as n scales up, which means we can save up 15% qubits. This result will help us extend the power of quantum optimization to graphs whose sizes are larger than the quantum computer we have, see the following illustrative plot Figure 4. Given a certain quantum

device of n_0 qubits which is used to solve Max-Cut on larger graphs, the approximation ratio will drop bit by bit as the graph size increases, until it reaches the point where the approximation ratio is no longer better than the classical algorithm. This point, denoted by $\max n$, is the maximum problem size that can be solved with this device using our framework. And Δn, the difference between $\max n$ and n_0, is the maximum amount of qubits that can be saved.

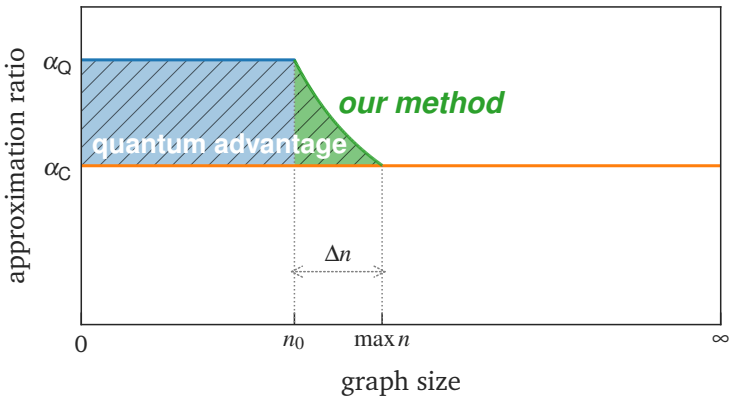


Figure 4: An illustration of the effect of our method.

A trivial case: $\alpha_C = \frac{1}{2}$

There is a trivial approach that takes the spirit of the naive algorithm mentioned in Algorithm 1, which achieves $\frac{1}{2}$-approximation on OPT-REM problem. When a color of V_0 is fixed to s_0, we can just uniformly and independently color each vertex in V_1. Now we prove that this gives $\frac{1}{2}$-approximation ratio on OPT-REM problem.

Proof. For each edge $(x_0, y_1) \in E_{01}$, it connects two vertices from two partitions $x_0 \in V_0, y_1 \in V_1$. The x_0 is already colored, and there will be $\frac{1}{2}$ probability that y_1 is colored differently from x_0, making (x_0, y_1) becomes a cut with $\frac{1}{2}$ probability. The argument is similar for edge in E_1. Thus the expected cut size given by this method is then $\frac{1}{2}|E - E_0| \geq \frac{1}{2}\text{OPT-REM}(s_0)$.

□

This $\frac{1}{2}$ approximation is low, thus having the overall algorithm do better than α_C, as mentioned in Lemma 2, is not a decent goal. So we revisit the

approximation ratio (25) given by the overall algorithm and take $\alpha_{\mathsf{C}} = \frac{1}{2}$,

$$\alpha_{\mathsf{CQ}} \geq \alpha_{\mathsf{Q}} \cdot \left(1 + (\alpha_{\mathsf{C}} - 1)\frac{\max_s\{b_s\}}{\max_s\{a_s + b_s\}}\right) \tag{37}$$

$$\geq \alpha_{\mathsf{Q}} \cdot \left(1 - \frac{\max_s\{b_s\}}{2 \cdot \max_s\{a_s + b_s\}}\right) \tag{38}$$

$$\geq \alpha_{\mathsf{Q}} \cdot \left(1 - \frac{|E - E_0|}{|E|}\right) = \alpha_{\mathsf{Q}} \cdot \frac{|E_0|}{|E|} \tag{39}$$

$$\geq \alpha_{\mathsf{Q}} \cdot \frac{n_0(n_0 - 1)}{n(n-1)}, \tag{40}$$

giving us a relationship between α_{CQ}, α_{Q} and the ratio $\big(n_0(n_0-1)\big)/\big(n(n-1)\big)$. We see that if we want an overall α_{CQ} approximation, and given the approximation ratio for QAOA being α_{Q}, it is required to have

$$\frac{n_0(n_0 - 1)}{n(n-1)} \geq \frac{\alpha_{\mathsf{CQ}}}{\alpha_{\mathsf{Q}}}. \tag{41}$$

This is in fact a strict constraint. To see this, we set α_{Q} to 1 which means we assume QAOA obtains an accurate solution, then we need

$$\big(n_0(n_0-1)\big)/\big(n(n-1)\big) \geq \alpha_{\mathsf{CQ}}. \tag{42}$$

Even if the goal is just to achieve 0.9-approximation on Max-Cut (which is not high considering classical computation achieves 0.878), only a few qubits can be saved (see Table 4, col iii). However, it is worth mentioning that the larger the device is, the more effective this method will be. So although this trivial $\alpha_{\mathsf{C}} = \frac{1}{2}$ case is not practically meaningful at present, it may become useful in the future as the quantum computers scale up.

For illustration, we list a table that shows the corresponding largest problem instance that can be solved on current quantum computers, see Table 4. In this table, the n_0 column stands for the actual number of available qubits, $\max n$ stands for the maximum problem size that can be solved with this device using our framework and Δ indicates the difference between n and n_0. setting I means we want to ensure overall approximation ratio larger than α_{C} assuming QAOA and the classical method have α_{Q} and α_{C} approximation ratio, and setting II means we want to ensure overall approximation ratio larger than α_{CQ} assuming QAOA and the classical

Approx		$\alpha_Q = 0.97$ i. $\alpha_C = 0.91$ (setting I)		$\alpha_Q = 0.95$ ii. $\alpha_C = 0.89$ (setting I)		$\alpha_{CQ} = 0.9$ iii. $\alpha_Q = 1$ (setting II)	
Device	n_0	max n	Δn	max n	Δn	max n	Δn
Maryland	40	49	9	47	7	42	2
Sycamore	53	65	12	62	9	55	2
Zuchongzhi2	66	81	15	78	12	69	3
IonQ	79	97	18	93	14	83	4
Eagle	127	156	29	150	23	133	6
Condor	1121	1379	258	1330	209	1181	60

Table 4: The largest problem instance that can be handled.

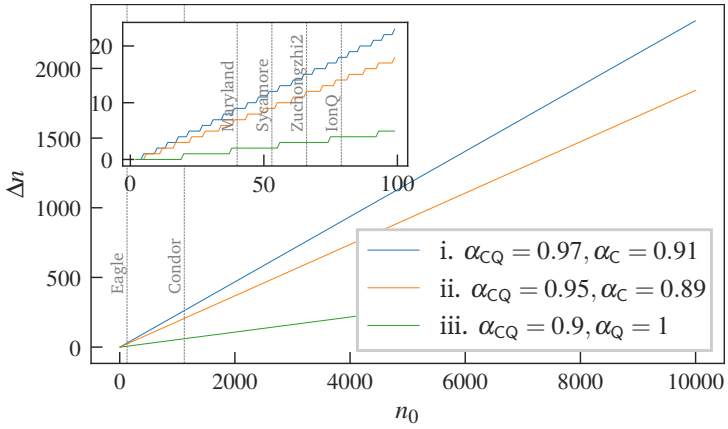


Figure 5: The scaling of the number of saved qubits with respect to the number of available qubits under different approximation ratio settings.

method have α_Q and 0.5 approximation ratio. Moreover, a plot showing the scaling of the number of saved qubits with respect to the number of available qubits under different approximation ratio settings is given in Figure 5. From the figure it is clear that the number of qubits saved grows linearly with the number of available qubits, and the larger the device is, the more effective this method will be.

4. Numerical results

This section is composed of two parts. As the previous section assumes access to an oracle (11) based on a classical algorithm, we now discuss how such an oracle might be built. Building such an oracle proves to be

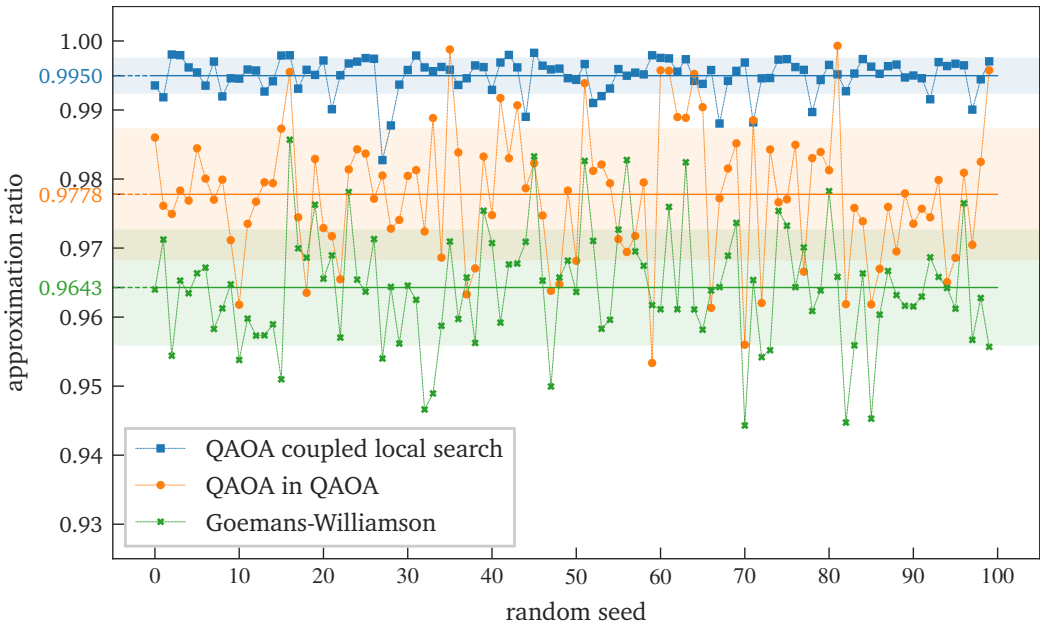


Figure 6: We generate 24-vertex Erdős-Rényi graphs with 0.8 connectivity and use 18-qubit QAOA to solve Max-Cut on them. The blue line (`QAOA coupled local search`) shows the approximation ratio given by our method, the orange line (`QAOA in QAOA`) by the method in [22] and the green line (`Goemans-Williamson`) by Goemans-Williamson algorithm. We can see from the plot that on most instances our method gives better results. In fact, our method outperforms the other two in 96 out of the total 100 instances. The three horizontal lines show the average approximation ratio given by the three methods, and our result is clearly better. The shaded areas around the lines indicate the standard deviations, which are 2.627×10^{-3} (blue), 9.575×10^{-3} (orange) and 8.413×10^{-3} (green), respectively.

hard as we only allow $o(n)$ space overhead (which is discussed in detail in the following text), but we point out a possible direction that is supported by numerical results. For the second part, we propose a heuristic approach for the classical algorithm and perform some numerical experiments on 24-vertex Erdős-Rényi graph using 18 qubits. Our method is compared with Goemans-Williamson algorithm and previous work [22], and the results are shown in Figure 6. We can see that our method with 18 qubits is still able to outperform Goemans-Williamson algorithm by a lot, and outperforms [22] most of the time.

4.1. On implementing the classical algorithm solving OPT-REM

Review the definition of OPT-REM (given in (9))

$$\text{OPT-REM}(s_0) = \max_{s_1 \in \{0,1\}^{n_1}} \text{REM}(s_0, s_1).$$

The problem above is a variant of Max-Cut. If we choose $V_0 = \emptyset$, then $\text{CUT}_{01}(s_0, s_1) = 0$ and OPT-REM will degenerate to Max-Cut problem on G_1. Thus, OPT-REM problem is as hard as Max-Cut problem, and in Appendix C, we demonstrate a polynomial space approximation algorithm that solves OPT-REM problem with .878 approximation ratio. The algorithm described in Appendix C naturally gives rise to a quantum circuit with $O(n)$ auxiliary qubits – which is not tolerable as our goal is to save qubits. In fact, due to [36], we know no classical algorithm can break the trivial $\frac{1}{2}$ approximation ratio for Max-Cut with $o(n)$ space. Thus, there is no suitable classical algorithm solving OPT-REM that can be straightly implemented on a quantum circuit to achieve (11). However, we only care about the number of qubits needed in our framework, which means we can use arbitrary (polynomial) classical space to prepare for the quantum circuit, which circumvents the constraint from [36].

When solving OPT-REM, we can do brute force, i.e., enumerating all possible s_1 and take the maximum as defined in (9). The brute force method is inefficient, and we want to do pruning to reduce the search space. Intuitively, some s_1 is unnecessary to be tried. For example, if $\text{CUT}_1(s_1)$ is too small, then this s_1 is unlikely to be selected as the optimal choice for any s_0. We then give the following definition:

Definition 1 (α-guarantee set)**.** *For arbitrary fixed approximation ratio requirement α, there is a set $T \subset \{0,1\}^{n_1}$ such that*

$$\forall s_0 \in \{0,1\}^{n_0}, \frac{\max_{s_1 \in T} \text{REM}(s_0, s_1)}{\text{OPT-REM}(s_0)} \geq \alpha. \tag{43}$$

We call such set T an α-guarantee set.

We ask, what is the minimum number of s_1 elements that a α-guarantee set should contain? This problem is important because a small α-guarantee set straightly leads to an efficient optimization workflow (described in Section 4.1). We manage to show numerically that for Erdős-Rényi graph considered in this work, an α-guarantee set is considerably small. Our workflow is as follows: for a fixed graph, instead of finding T satisfying (43)

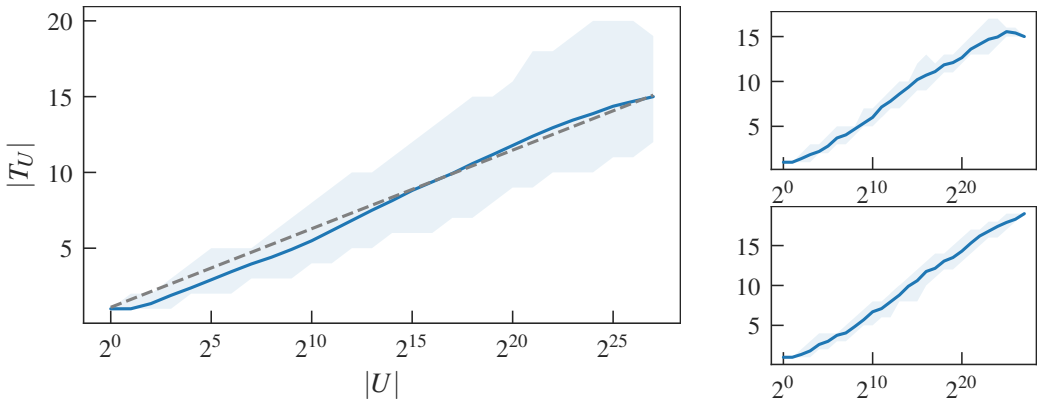


Figure 7: (a) The increment of $|T_U|$ with $n = 35, n = 28, \alpha = 0.9$, averaged over 20 random Erdős-Rényi graphs and 20 random permutations. A straight line (gray) is added for reference. (b) The concentration of $|T_U|$ for fixed graphs and $|U|$, respectively generated using graphs with random seeds 0 and 1. The solid line indicates the average size of $|T_U|$, while the shaded area shows the minimum and the maximum of $|T_U|$ over all samples and graphs.

directly, we select a subset U of $\{0,1\}^{n_1}$ and use an inefficient brute-force algorithm to search for T_U such that

$$\forall s_0 \in U, \frac{\max_{s_1 \in T_U} \text{REM}(s_0, s_1)}{\text{OPT-REM}(s_0)} \geq \alpha. \tag{44}$$

By gradually grow U to $\{0,1\}^{n_1}$, we see that the $|T_U|$ we obtain grows almost linearly with $\log|U|$, see Figure 7(a). This slow speed of scaling indicates that an α-guarantee set should a have polynomial size with respect to n_1.

Moreover, we also observe, that for a fixed graph, when we compute T_U for different U of the same size, $|T_U|$s display signs of concentration, as seen in Figure 7(b). This concentration suggests that when we add elements into U one by one, $|T_U|$ grows following an almost identical speed to a final result, regardless of the order of elements added. This phenomena also supports that $|T_U|$ grows linearly with $\log|U|$. Details on the methodologies of computing T_U and more numerical results are given in Appendix A.

Now we will further support our assertion that α-guarantee sets are small by providing some analysis for large Erdős-Rényi graphs , as shown in the following lemma:

Lemma 3 (α-guarantee set size for large Erdős-Rényi graphs)**.** *For large random Erdős-Rényi graphs and n_0, α satisfying $\frac{n_0}{n} > \frac{\alpha}{2-\alpha}$, then with high probability we have an α-guarantee set of size 2.*

See Appendix D for a detailed proof. This result states for Erdős-Rényi graphs, if n_0/n is sufficiently large, the α-guarantee set size is a constant regardless of the graph size. This provides strong evidence that the α-guarantee set will be small for graphs with much symmetry. But our current result only applies to Erdős-Rényi graphs and large n_0/n, and it is of great interest to generalize the results to other graph ensembles which, along with how to efficiently find a small α-guarantee set for arbitrary graphs, remains for future work to tackle.

QAOA circuit based on α-guarantee set

With an α-guarantee set T, the approximate solution to OPT-REM will hence be

$$\text{OPT-REM'}(s_0) = \max_{s_1 \in T} \text{REM}(s_0, s_1). \tag{45}$$

Assuming QAOA achieves approximation ratio α_{Q}, then we are promised to obtain a cut solution with a size of at least

$$\alpha_{\text{Q}} \cdot \left(\max_{s_0 \in \{0,1\}^{n_0}} \left\{ \text{CUT}_0(s_0) + \text{OPT-REM'}(s_0) \right\} \right) \tag{46}$$

$$= \alpha_{\text{Q}} \cdot \left(\max_{s_0 \in \{0,1\}^{n_0}} \left\{ \text{CUT}_0(s_0) + \max_{s_1 \in T} \text{REM}(s_0, s_1) \right\} \right) \tag{47}$$

$$= \max_{s_1 \in T} \left\{ \alpha_{\text{Q}} \cdot \max_{s_0 \in \{0,1\}^{n_0}} \left\{ \text{CUT}_0(s_0) + \text{REM}(s_0, s_1) \right\} \right\}. \tag{48}$$

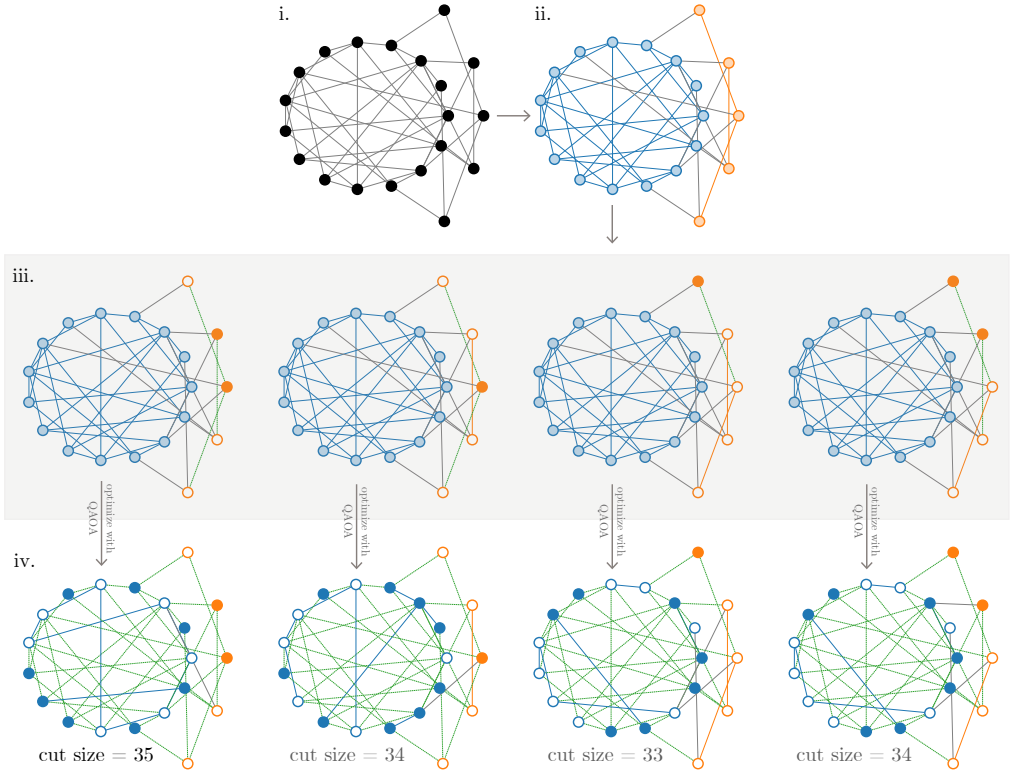
The equation above suggests we can fix s_1 and solve

$$\max_{s_0 \in \{0,1\}^{n_0}} \left\{ \text{CUT}_0(s_0) + \text{REM}(s_0, s_1) \right\} \tag{49}$$

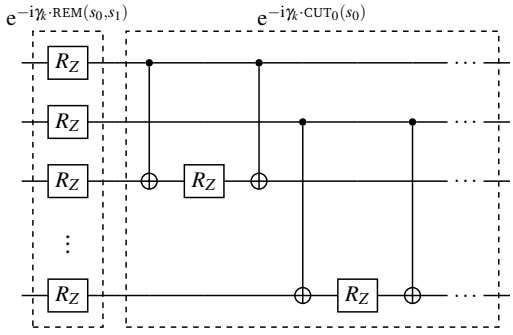
$$= \text{CUT}_1(s_1) + \max_{s_0 \in \{0,1\}^{n_0}} \left\{ \text{CUT}_0(s_0) + \text{CUT}_{01}(s_0, s_1) \right\}, \tag{50}$$

and take the maximum output overall s_1. And the optimization

$$\max_{s_0 \in \{0,1\}^{n_0}} \left\{ \text{CUT}_0(s_0) + \text{CUT}_{01}(s_0, s_1) \right\} \tag{51}$$



(a)



(b)

Figure 8: (a) Overall workflow. From the original 20-vertex graph shown in *i*, we select a 15-vertex dense graph, which is colored blue in *ii*. There exists a .878-guarantee set with 4 elements and is represented with unfilled and filled orange vertices in *iii*. Based on these 4 elements, we can generate 4 QAOA circuits that look like (b), and running optimization separately gives us 4 possible solutions in *iv*, from which we take the first solution, being the optimum, as our final solution. (b) The circuit for $|s_0\rangle \mapsto e^{-i\gamma_k \cdot \left(\mathrm{CUT}_0(s_0)+\mathrm{REM}(s_0,s_1)\right)}|s_0\rangle$.

can also be tackled with QAOA, which gets at least α_{Q} of the optimum. Thus, we can use QAOA to approximately solve (51) for all $s_1 \in T$, and take the maximum answer. The procedure described above achieves solutions no worse than (48), meaning that this multiple QAOA procedure shares the same performance guarantee with the previous procedure described in Section 3 using one single QAOA. The overall procedure is demonstrated using an example in Figure 8(a).

Now it remains to show how to solve (51) with QAOA. That is, we will need to implement

$$|s_0\rangle \mapsto \mathrm{e}^{-\mathrm{i}\gamma_k \cdot (\mathrm{CUT}_0(s_0) + \mathrm{CUT}_{01}(s_0, s_1))} |s_0\rangle. \tag{52}$$

The first part is easy, and it can be done with Figure 1(b) as we have discussed before. For the second part, we use $s_0(u)$ to denote the color of vertex u in s_0, and $s_1(v)$ for the color of vertex v in s_1, then

$$\begin{aligned} \mathrm{CUT}_{01}(s_0, s_1) &= \sum_{(u,v)\in E_{01}} [s_0(u) \neq s_1(v)] \\ &= \sum_{u \in V_0} \sum_{\substack{v \in V_1 \wedge \\ (u,v)\in E_{01}}} [s_1(v) = \neg s_0(u)]. \end{aligned} \tag{53}$$

For convenience, let $A_{u,0/1}(u \in V_0)$ be the contribution from $(u,v) \in E_{01}(v \in V_1)$ to cut when u is colored $0/1$, i.e.,

$$A_{u,0/1} = \sum_{\substack{v \in V_1 \wedge \\ (u,v)\in E_{01}}} [s_1(v) = 1/0], \tag{54}$$

then

$$\mathrm{CUT}_{01}(s_0, s_1) = \sum_{u \in V_0} A_{u, s_0(u)}. \tag{55}$$

So, for the u-th qubit with state $0/1$, we need to add a phase factor $\mathrm{e}^{-\mathrm{i}\gamma_k \cdot A_{u,0/1}}$. This can be done with a single R_Z gate. Therefore, the overall circuit for (52) as a part of the QAOA for solving (51) would look like Figure 8(b). This circuit consists of only CNOT and R_Z gates and, therefore remains NISQ-friendly. We can see from the analysis above, analogous to the original Max-Cut problem (7), problem (51) has a similar Hamiltonian representation

$$\sum_{u \in V_0} (A_{u,1} - A_{u,0}) Z_u + \sum_{(u,v)\in E_{01}} Z_u Z_v, \tag{56}$$

which consists only of one-local and two-local terms.

For a α-guarantee set T, the method described in this section requires $|T|$ runs of QAOA optimization. Hopefully, for graphs with certain properties, we can theoretically derive that there exists T such that $|T| \in \text{poly}(n)$, but even if no theoretical bound can be estimated, a practically acceptable size will still be enough. So it will be of significant interest to design a practical algorithm that constructs a moderate-size T, with or without a polynomial bound for its size.

4.2. A heuristic approach based on local search

Algorithm 3: Coupling QAOA with local search

```
Data: A graph G = (V, E).
Function QAOASolve(s_1):
    H ← the Hamiltonian defined by (56);
    return max_{β,γ} ⟨β, γ|H|β, γ⟩;
Find a dense subgraph G_0 = (V_0, E_0) for G;
s_1 ← an initial coloring for V_1;
c ←QAOASolve(s_1);
while true do
    for v ∈ V_1 do
        s'_1 ← s_1 with the color of v fliped;
        c' ← QAOASolve(s_1);
        r_v ← (c', s'_1);
    end
    (c', s'_1) ← max r;
    if c' > c then
        (c, s_1) ← (c', s'_1);
    else
        break;
    end
end
return c;
```

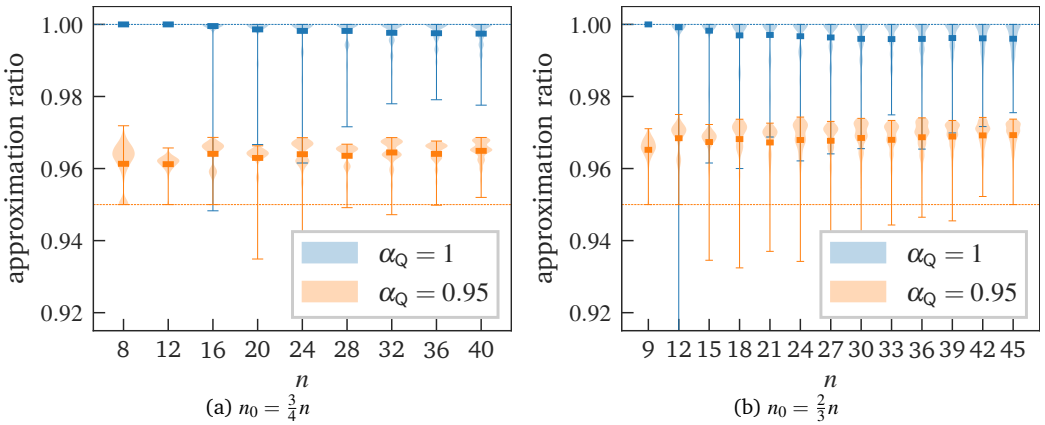
In order to demonstrate the viability of our coupling framework for QAOA circuit design, we set up experiments to show its performance when it is applied to real Max-Cut instances. As pointed out in previous sec-

tions, currently we are not able to efficiently construct α-guarantee set as required, so a heuristic method needs to be devised.

Here we give a heuristic method that converges in polynomial time and demonstrate its numerical results compared with previous methods. Our method is to first fix a coloring s_1 of V_1, and update the chosen s_1 following a local search scheme to gradually reach a good result. The overall structure of our heuristic method will look like Algorithm 3.

We also tested the algorithm proposed by [22] for comparison. We strengthen that [22] is able to split the graph arbitrarily so it is able to solve larger Max-Cut instances using small-scale quantum computers, but here we only consider such occasions that the number of qubits is slightly smaller than the graph size, and we demonstrate that our method has the potential to make better use of such amount of available qubits. The numerical results are shown in Figure 6.

Scaling performance



(a) $n_0 = \frac{3}{4}n$ (b) $n_0 = \frac{2}{3}n$

Figure 9: Scaling performance of our method. For increasing n, we use $\frac{3}{4}n$ ($\frac{2}{3}n$) qubits to solve Max-Cut problem on 1000 Erdős-Rényi graph instances, assuming QAOA achieves an approximation ratio α_Q. The thick dash shows the average approximation ratio, and the shaded area shows the density distribution of the approximation ratio. We can see from the plot that the overall approximation ratio is promising and stable even at large-scale problems, indicating that our framework may well extend quantum advantage to problems whose sizes are larger than number of available qubits.

In order to show the scaling performance of our coupling method, we provide some more numerical experiments on various graph sizes. Note that as the cost of QAOA simulation and parameter optimization is really

high, in this section, we just assume that QAOA achieves some certain approximation ratio α_Q in our numerical experiments. In other words, we do not perform QAOA simulation and parameter optimization, but just return α_Q times the optimum solution when given a problem Hamiltonian. This will allow us to perform larger-scale experiments faster, giving us more sufficient statistical data. The results are shown in Figure 9.

5. Summary and discussion

In this work, we propose a coupling framework for QAOA circuit design and demonstrate it using Max-Cut as an example. Our method replaces a certain portion of quantum computation with classical computation to save quantum resources at the expense of losing a bit of approximation performance.

Our method relies on an efficient quantum-implementable classical algorithm that approximately solves the OPT-REM problem defined in (9). As the problem is as hard as Max-Cut and the classical algorithm can only consume $o(n)$ qubits when implemented as a quantum oracle, currently we are not able to design a classical algorithm as required, but we present some numerical observation that will lead our way towards a possible solution. However, despite designing a required algorithm with a theoretical guarantee being hard, we give a heuristic method inspired by the local search algorithm. Numerical experiments show good performance which suggests our coupling framework might be able to help give more satisfactory solutions to larger Max-Cut problem instances using currently limited number of qubits. Future work remains to design a classical algorithm that solves (9) with rigorous analysis and performance guarantee.

Moreover, our framework applies not only to Max-Cut problem. In [37, 38], the Ising formulations of many hard problems are given, which enable us to solve them using quantum adiabatic algorithms and QAOA. It is an interesting direction for future work to explore the applications of our framework to other combinatorial optimization problems. For problems with similar structures to Max-Cut, such as number partitioning, we expect a straightforward generalization. For other problems, the generalization would be more non-trivial but still possible.

6. Data and code availability

The datasets and the source code that generates them are available at [39].

References

[1] E. Farhi, J. Goldstone, S. Gutmann, A Quantum Approximate Optimization Algorithm, arXiv:1411.4028 [quant-ph] (2014). doi:10.48550/arXiv.1411.4028.
URL http://arxiv.org/abs/1411.4028

[2] J. Preskill, Quantum Computing in the NISQ era and beyond, Quantum 2 (2018) 79. doi:10.22331/q-2018-08-06-79.
URL https://quantum-journal.org/papers/q-2018-08-06-79/

[3] W. Li, D.-L. Deng, Recent advances for quantum classifiers, Science China Physics, Mechanics, and Astronomy 65 (2) (2022) 220301. doi:10.1007/s11433-021-1793-6.

[4] H.-L. Huang, X.-Y. Xu, C. Guo, G. Tian, S.-J. Wei, X. Sun, W.-S. Bao, G.-L. Long, Near-term quantum computing techniques: Variational quantum algorithms, error mitigation, circuit compilation, benchmarking and classical simulation, Science China Physics, Mechanics, and Astronomy 66 (5) (2023) 250302. doi:10.1007/s11433-022-2057-y.
URL https://doi.org/10.1007/s11433-022-2057-y

[5] D. Headley, T. Müller, A. Martin, E. Solano, M. Sanz, F. K. Wilhelm, Approximating the quantum approximate optimization algorithm with digital-analog interactions, Phys. Rev. A 106 (4) (2022) 042446, publisher: American Physical Society. doi:10.1103/PhysRevA.106.042446.
URL https://link.aps.org/doi/10.1103/PhysRevA.106.042446

[6] P. W. Shor, Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, SIAM Journal on Computing 26 (5) (1997) 1484–1509. doi:10.1137/S0097539795293172.
URL https://epubs.siam.org/doi/10.1137/S0097539795293172

[7] L. K. Grover, A fast quantum mechanical algorithm for database search, in: Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing, STOC '96, Association for Computing Machinery, New York, NY, USA, 1996, pp. 212–219. doi:10.1145/237814.237866.
URL https://doi.org/10.1145/237814.237866

[8] E. Farhi, J. Goldstone, S. Gutmann, L. Zhou, The Quantum Approximate Optimization Algorithm and the Sherrington-Kirkpatrick Model at Infinite Size, Quantum 6 (2022) 759. doi:10.22331/q-2022-07-07-759.
URL https://doi.org/10.22331/q-2022-07-07-759

[9] J. Basso, E. Farhi, K. Marwaha, B. Villalonga, L. Zhou, The Quantum Approximate Optimization Algorithm at High Depth for MaxCut on Large-Girth Regular Graphs and the Sherrington-Kirkpatrick Model, in: F. Le Gall, T. Morimae (Eds.), 17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022), Vol. 232 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2022, pp. 7:1–7:21. doi:10.4230/LIPIcs.TQC.2022.7.
URL https://drops.dagstuhl.de/opus/volltexte/2022/16514

[10] M. P. Harrigan, K. J. Sung, M. Neeley, K. J. Satzinger, F. Arute, K. Arya, J. Atalaya, J. C. Bardin, R. Barends, S. Boixo, M. Broughton, B. B. Buckley, D. A. Buell, B. Burkett, N. Bushnell, Y. Chen, Z. Chen, Ben Chiaro, R. Collins, W. Courtney, S. Demura, A. Dunsworth, D. Eppens, A. Fowler, B. Foxen, C. Gidney, M. Giustina, R. Graff, S. Habegger, A. Ho, S. Hong, T. Huang, L. B. Ioffe, S. V. Isakov, E. Jeffrey, Z. Jiang, C. Jones, D. Kafri, K. Kechedzhi, J. Kelly, S. Kim, P. V. Klimov, A. N. Korotkov, F. Kostritsa, D. Landhuis, P. Laptev, M. Lindmark, M. Leib, O. Martin, J. M. Martinis, J. R. McClean, M. McEwen, A. Megrant, X. Mi, M. Mohseni, W. Mruczkiewicz, J. Mutus, O. Naaman, C. Neill, F. Neukart, M. Y. Niu, T. E. O'Brien, B. O'Gorman, E. Ostby, A. Petukhov, H. Putterman, C. Quintana, P. Roushan, N. C. Rubin, D. Sank, A. Skolik, V. Smelyanskiy, D. Strain, M. Streif, M. Szalay, A. Vainsencher, T. White, Z. J. Yao, P. Yeh, A. Zalcman, L. Zhou,

H. Neven, D. Bacon, E. Lucero, E. Farhi, R. Babbush, Quantum approximate optimization of non-planar graph problems on a planar superconducting processor, Nature Physics 17 (3) (2021) 332–336. doi:10.1038/s41567-020-01105-y.
URL https://doi.org/10.1038/s41567-020-01105-y

[11] J. Wurtz, P. Love, MaxCut quantum approximate optimization algorithm performance guarantees for $p > 1$, Physical Review A 103 (4) (2021) 042612. doi:10.1103/PhysRevA.103.042612.
URL https://link.aps.org/doi/10.1103/PhysRevA.103.042612

[12] S. H. Sack, M. Serbyn, Quantum annealing initialization of the quantum approximate optimization algorithm, Quantum 5 (2021) 491. doi:10.22331/q-2021-07-01-491.
URL https://doi.org/10.22331/q-2021-07-01-491

[13] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, M. D. Lukin, Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices, Phys. Rev. X 10 (2) (2020) 021067. doi:10.1103/PhysRevX.10.021067.
URL https://link.aps.org/doi/10.1103/PhysRevX.10.021067

[14] I. Waldspurger, A. d'Aspremont, S. Mallat, Phase recovery, MaxCut and complex semidefinite programming, Mathematical Programming 149 (1) (2015) 47–81. doi:10.1007/s10107-013-0738-9.
URL https://doi.org/10.1007/s10107-013-0738-9

[15] M. Frackiewicz, How much does 1 qubit cost? (Oct. 2023).

[16] J. Koetsier, Million-Qubit Quantum Computing? How SEEQC Plans To Scale Quantum Computers.
URL https://www.forbes.com/sites/johnkoetsier/2022/01/11/million-qubit-quantum-computing-how-seeqc-plans-to-scale-quantum-computers/

[17] R. Van Meter, S. J. Devitt, The Path to Scalable Distributed Quantum Computing, Computer 49 (9) (2016) 31–42. doi:10.1109/MC.2016.291.

[18] D. P. Franke, J. S. Clarke, L. M. K. Vandersypen, M. Veldhorst, Rent's rule and extensibility in quantum computing, Microprocessors and Microsystems 67 (2019) 1–7. doi:10.1016/j.micpro.2019.02.006.

[19] M. Kjaergaard, M. E. Schwartz, J. Braumüller, P. Krantz, J. I.-J. Wang, S. Gustavsson, W. D. Oliver, Superconducting Qubits: Current State of Play, Annual Review of Condensed Matter Physics 11 (1) (2020) 369–395. doi:10.1146/annurev-conmatphys-031119-050605.

[20] S. A. Moses, C. H. Baldwin, M. S. Allman, R. Ancona, L. Ascarrunz, C. Barnes, J. Bartolotta, B. Bjork, P. Blanchard, M. Bohn, J. G. Bohnet, N. C. Brown, N. Q. Burdick, W. C. Burton, S. L. Campbell, J. P. Campora, C. Carron, J. Chambers, J. W. Chan, Y. H. Chen, A. Chernoguzov, E. Chertkov, J. Colina, J. P. Curtis, R. Daniel, M. DeCross, D. Deen, C. Delaney, J. M. Dreiling, C. T. Ertsgaard, J. Esposito, B. Estey, M. Fabrikant, C. Figgatt, C. Foltz, M. Foss-Feig, D. Francois, J. P. Gaebler, T. M. Gatterman, C. N. Gilbreth, J. Giles, E. Glynn, A. Hall, A. M. Hankin, A. Hansen, D. Hayes, B. Higashi, I. M. Hoffman, B. Horning, J. J. Hout, R. Jacobs, J. Johansen, L. Jones, J. Karcz, T. Klein, P. Lauria, P. Lee, D. Liefer, S. T. Lu, D. Lucchetti, C. Lytle, A. Malm, M. Matheny, B. Mathewson, K. Mayer, D. B. Miller, M. Mills, B. Neyenhuis, L. Nugent, S. Olson, J. Parks, G. N. Price, Z. Price, M. Pugh, A. Ransford, A. P. Reed, C. Roman, M. Rowe, C. Ryan-Anderson, S. Sanders, J. Sedlacek, P. Shevchuk, P. Siegfried, T. Skripka, B. Spaun, R. T. Sprenkle, R. P. Stutz, M. Swallows, R. I. Tobey, A. Tran, T. Tran, E. Vogt, C. Volin, J. Walker, A. M. Zolot, J. M. Pino, A Race-Track Trapped-Ion Quantum Processor, Physical Review X 13 (4) (2023) 041052. doi:10.1103/PhysRevX.13.041052.

[21] J. Li, M. Alam, S. Ghosh, Large-scale Quantum Approximate Optimization via Divide-and-Conquer, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 42 (6) (2023) 1852–1860. doi:10.1109/TCAD.2022.3212196.

[22] Z. Zhou, Y. Du, X. Tian, D. Tao, QAOA-in-QAOA: Solving Large-Scale MaxCut Problems on Small Quantum Machines, Phys. Rev. Applied 19 (2) (2023) 024027, publisher: American Physical Society. doi:10.1103/PhysRevApplied.19.024027.
URL https://link.aps.org/doi/10.1103/PhysRevApplied.19.024027

[23] M. X. Goemans, D. P. Williamson, Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefi-

nite Programming, J. ACM 42 (6) (1995) 1115–1145. doi:10.1145/227683.227684.
URL https://doi.org/10.1145/227683.227684

[24] H. Karloff, How good is the Goemans-Williamson MAX CUT algorithm?, in: Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing, 1996, pp. 427–434. doi:10.1145/237814.237990.

[25] P. Festa, P. M. Pardalos, M. G. Resende, C. C. Ribeiro, Randomized heuristics for the MAX-CUT problem, Optimization methods and software 17 (6) (2002) 1033–1058. doi:10.1080/1055678021000090033.

[26] S. Khot, G. Kindler, E. Mossel, R. O'Donnell, Optimal Inapproximability Results for MAX-CUT and Other 2-Variable CSPs?, SIAM J. Comput. 37 (1) (2007) 319–357. doi:10.1137/S0097539705447372.
URL https://doi.org/10.1137/S0097539705447372

[27] C. Mathieu, W. Schudy, Yet Another Algorithm for Dense Max Cut: Go Greedy, in: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '08, Society for Industrial and Applied Mathematics, USA, 2008, pp. 176–182. doi:10.5555/1347082.1347102.

[28] C. Wu, J. Wang, F. Zuo, Stabilizer approximation III: Maximum cut (2023). doi:10.48550/arXiv.2303.17215.
URL http://arxiv.org/abs/2303.17215

[29] A. Misra-Spieldenner, T. Bode, P. K. Schuhmacher, T. Stollenwerk, D. Bagrets, F. K. Wilhelm, Mean-Field Approximate Optimization Algorithm, PRX Quantum 4 (3) (2023) 030335. doi:10.1103/PRXQuantum.4.030335.

[30] J. Håstad, Some Optimal Inapproximability Results, J. ACM 48 (4) (2001) 798–859. doi:10.1145/502090.502098.
URL https://doi.org/10.1145/502090.502098

[31] L. Trevisan, G. Sorkin, M. Sudan, D. Williamson, Gadgets, approximation, and linear programming, in: Proceedings of 37th Con-

ference on Foundations of Computer Science, 1996, pp. 617–626. doi:10.1109/SFCS.1996.548521.

[32] G. E. Crooks, Performance of the Quantum Approximate Optimization Algorithm on the Maximum Cut Problem, arXiv:1811.08419 [quant-ph] (2018). doi:10.48550/arXiv.1811.08419.
URL http://arxiv.org/abs/1811.08419

[33] F. Hadlock, Finding a Maximum Cut of a Planar Graph in Polynomial Time, SIAM Journal on Computing 4 (3) (1975) 221–225, publisher: Society for Industrial and Applied Mathematics. doi:10.1137/0204019.
URL https://epubs.siam.org/doi/10.1137/0204019

[34] F. Della Croce, M. J. Kaminski, V. Th. Paschos, An exact algorithm for MAX-CUT in sparse graphs, Operations Research Letters 35 (3) (2007) 403–408. doi:10.1016/j.orl.2006.04.001.

[35] Y. Kobayashi, Y. Kobayashi, S. Miyazaki, S. Tamaki, An Improved Fixed-Parameter Algorithm for Max-Cut Parameterized by Crossing Number, in: C. J. Colbourn, R. Grossi, N. Pisanti (Eds.), Combinatorial Algorithms, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2019, pp. 327–338. doi:10.1007/978-3-030-25005-8_27.

[36] M. Kapralov, D. Krachun, An optimal space lower bound for approximating MAX-CUT, in: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, ACM, 2019, pp. 277–288. doi:10.1145/3313276.3316364.
URL https://dl.acm.org/doi/10.1145/3313276.3316364

[37] V. Choi, Adiabatic Quantum Algorithms for the NP-Complete Maximum-Weight Independent Set, Exact Cover and 3SAT Problems (Apr. 2010). arXiv:1004.2226.

[38] A. Lucas, Ising formulations of many NP problems, Frontiers in Physics 2 (2014). doi:10.3389/fphy.2014.00005.

[39] Y. Lu, LucidaLu/QAOA-with-fewer-qubits (Jul. 2023).
URL https://github.com/LucidaLu/QAOA-with-fewer-qubits

[40] B. Korte, J. Vygen, Combinatorial Optimization: Theory and Algorithms, 5th Edition, Springer Publishing Company, Incorporated, 2012.

[41] S. S. Ravi, D. J. Rosenkrantz, G. K. Tayi, Heuristic and Special Case Algorithms for Dispersion Problems, Oper. Res. 42 (2) (1994) 299–310. doi:10.1287/opre.42.2.299.
URL https://doi.org/10.1287/opre.42.2.299

[42] R. K. R. Yarlagadda, J. E. Hershey, Hadamard Matrix Analysis and Synthesis: With Applications to Communications and Signal/Image Processing, Kluwer Academic Publishers, USA, 1996.

[43] S. K. Lam, A. Pitrou, S. Seibert, Numba: A LLVM-Based Python JIT Compiler, in: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, LLVM '15, Association for Computing Machinery, New York, NY, USA, 2015. doi:10.1145/2833157.2833162.
URL https://doi.org/10.1145/2833157.2833162

[44] C. G. BROYDEN, The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations, IMA Journal of Applied Mathematics 6 (1) (1970) 76–90. doi:10.1093/imamat/6.1.76.
URL https://doi.org/10.1093/imamat/6.1.76

[45] R. Fletcher, A new approach to variable metric algorithms, The Computer Journal 13 (3) (1970) 317–322. doi:10.1093/comjnl/13.3.317.
URL https://doi.org/10.1093/comjnl/13.3.317

[46] D. Goldfarb, A Family of Variable-Metric Methods Derived by Variational Means, Mathematics of Computation 24 (109) (1970) 23–26. doi:10.1090/S0025-5718-1970-0258249-6.
URL http://www.jstor.org/stable/2004873

[47] D. F. Shanno, Conditioning of Quasi-Newton Methods for Function Minimization, Mathematics of Computation 24 (111) (1970) 647–656. doi:10.2307/2004840.
URL http://www.jstor.org/stable/2004840

[48] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, Nature Methods 17 (2020) 261–272. doi:10.1038/s41592-019-0686-2.

[49] M. ApS, Introducing the MOSEK Optimization Suite 10.0.39 (2023). URL https://docs.mosek.com/latest/intro/index.html

[50] S. Diamond, S. Boyd, CVXPY: A Python-embedded modeling language for convex optimization, Journal of Machine Learning Research 17 (83) (2016) 1–5. doi:10.5555/2946645.3007036.

[51] A. Dembo, A. Montanari, S. Sen, Extremal cuts of sparse random graphs, The Annals of Probability 45 (2) (Mar. 2017). doi:10.1214/15-AOP1084.
URL https://projecteuclid.org/journals/annals-of-probability/volume-45/issue-2/Extremal-cuts-of-sparse-random-graphs/10.1214/15-AOP1084.full

[52] A. Hagberg, P. Swart, D. S Chult, Exploring network structure, dynamics, and function using networkx, Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2008).

[53] D. V. Andrade, M. G. C. Resende, R. F. Werneck, Fast local search for the maximum independent set problem, Journal of Heuristics 18 (4) (2012) 525–547. doi:10.1007/s10732-012-9196-4.

[54] Z. Yang, M. Zolanvari, R. Jain, A Survey of Important Issues in Quantum Computing and Communications, IEEE Communications Surveys & Tutorials 25 (2) (2023) 1059–1094. doi:10.1109/COMST.2023.3254481.

Appendix A. Methodologies to obtain a α-guarantee set

This section describes the methodology to obtain a α-guarantee set in our numerical experiments. Consider the process of constructing a α-guarantee set T. For each $s_1 \in \{0,1\}^{n_1}$, if it is added to a set T, it will be able to ensure that all elements in a certain set $\text{COVER}_\alpha(s_1) \subset \{0,1\}^{n_0}$ meet the approximation requirement, namely,

$$\text{COVER}_\alpha(s_1) := \left\{ s_0 : \frac{\text{REM}(s_0, s_1)}{\text{OPT-REM}(s_0)} \geq \alpha \right\}. \tag{A.1}$$

Thus, our goal of constructing a α-guarantee set can be reformulated as the well-known set-covering problem.

Definition 2 (Candidate set search problem)**.** *Given* 2^{n_1} *subsets*

$$\left\{\text{COVER}_\alpha(s_1)\right\}_{s_1 \in \{0,1\}^{n_1}}$$

of $U \subset \{0,1\}^{n_0}$*, we'd like to construct a minimum set* $T \subset \{0,1\}^{n_1}$ *such that*

$$\bigcup_{s_1 \in T} \text{COVER}_\alpha(s_1) = U. \tag{A.2}$$

Set-covering is known to be NP-hard [40], and we make use of a greedy algorithm. Note that this algorithm outputs a solution larger than the optimal one, so if the solution $|T_U|$ given by this algorithm is small, then the optimal size can only be smaller.

Algorithm 4: $\ln|U|$-approximation set covering

Data: m subsets $\mathscr{C} = \{S \mid S \subset U\}$ that guarantee $\bigcup_{S\in\mathscr{C}} S = U$.
Result: Covering set $\mathscr{T} \subset \mathscr{C}$ that satisfies $\bigcup_{T\in\mathscr{T}} T = U$.
$\mathscr{T} \leftarrow \emptyset$;
while $\bigcup_{T\in\mathscr{T}} T \neq U$ **do**
  $S^* \leftarrow \arg\max_{S\in\mathscr{C}} \#\left\{S - \bigcup_{T\in\mathscr{T}} T\right\}$;
  $\mathscr{T} \leftarrow \mathscr{T} \cup \{S^*\}$; `/* Choose the most profitable subset */`
end
return $\mathscr{T}$;

In our numerical experiments, we generate n-vertex Erdős-Rényi graphs with 0.8 connectivity and select a dense subgraph with n_0 vertices using the heuristic algorithm in [41]. Under such settings, we compute how $|T_U|$ increases as $|U|$ gradually grows to 2^{n_0}. We first generate a random permutation p of $\{0,1,2,\ldots,2^{n_0}-1\}$, and for a random graph, we compute the T_U when $U=\{p_0\},\{p_0,p_1\},\{p_0,p_1,p_2,p_3\},\ldots,\{p_0,p_1,\ldots,p_{2^{n_0}-1}\}$. The procedure above is done for 20 random permutations and 20 random graphs, and the results for $n=35, n_0=27$ are shown in Figure 7(a). It can be observed that $|T_U|$ seems to grow almost linearly with $\log|U|$, indicating the final $|T|$ will not be very large.

We change the size of graph, approximation ratio requirement α, and split ratio n_0/n and compute T_U. Experiments under different setups all give positive results, showing slow (non-exponential) increment of $\log|T_U|$ in accordance with $|U|$, see Figure A.10.

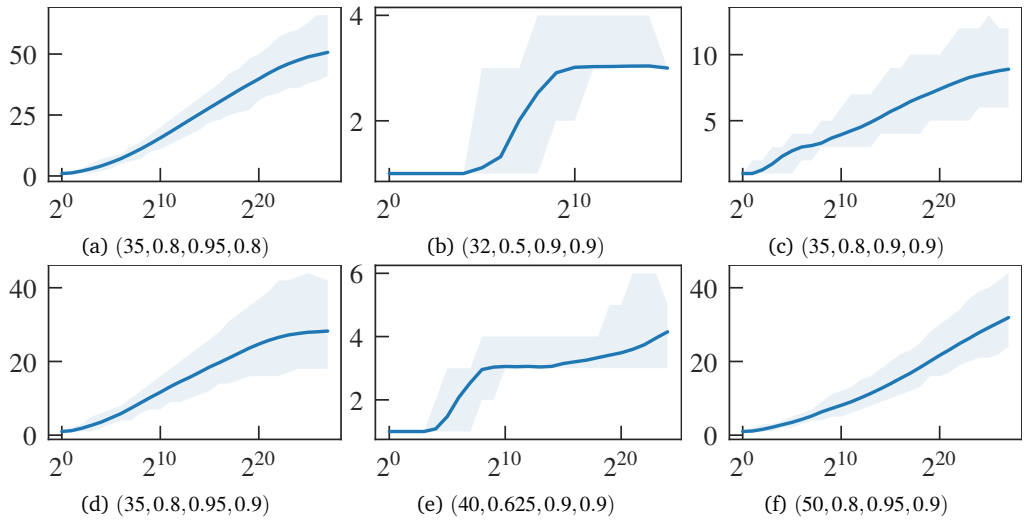


Figure A.10: Supplementary experiments for T_U size. The captions indicate $(n_0, \frac{n_0}{n}, \alpha, p)$.

Appendix B. Experiment settings

QAOA circuit for Max-Cut only involves $\exp(\mathrm{i}\theta X)$ and $\exp(\mathrm{i}\theta ZZ)$ gates, which is simply element-wise vector multiplication instead of matrix-vector multiplication under $\{|+\rangle,|-\rangle\}^{\otimes n}$ and $\{|0\rangle,|1\rangle\}^{\otimes n}$, respectively, while the basis transform between the two bases can be done with fast Walsh-Hadamard transform [42]. So here we implement a statevector simulator dedicated to

QAOA with the help of FWHT, and we use *Numba* [43] library to achieve parallel speedup with GPU.

In order to fully present the power of our circuit design and prevent performance loss due to unsatisfactory optimization results given by QAOA, we want to have a QAOA approximation ratio as close to 1 as possible. Therefore, we add much redundancy to p, the number of QAOA layers – p is set to n when we use QAOA to perform optimization over n-qubit problem instance. More layers means more parameters which leads to better circuit expressibility, thus we will more likely reach a state with a higher expectation.

Like many related works, we use BFGS method [44, 45, 46, 47] to optimize QAOA parameters for its quasi-Newton nature provides quick convergence rate. We choose the implementation of BFGS via *scipy* [48] library, and limit the maximum number of iterations to 100, with other parameters of BFGS left default. And to avoid getting stuck in abysmal local optima, we adopt the parameter initialization method given by [12] and set its hyper-parameter $\delta t = 0.56$.

While implementing our coupling method, we select dense subgraphs using a heuristic algorithm given in [41] and partition the graph into two parts (V_0, V_1), and run the algorithm defined in Algorithm 3 starting with $S_0 = \emptyset$. And when implementing [22], we use the random partition policy described in the paper and respectively use $|V_0|$-qubit QAOA and $|V_1|$-qubit QAOA to solve Max-Cut inside the two subgraphs, and numerically compute the expected output according to the two resultant probability distributions. When implementing [22], the parameters are changed to have better optimization performance: we set $p = 2n$ and iteration limit of BFGS to 500 up to try to decrease the performance loss of [22] due to the inaccuracy of QAOA.

When testing the performance of Goemans-Williamson algorithm, we use the SDP solver provided in *MOSEK* [49], wrapped with *cvxpy* [50] package.

Appendix C. Approximation algorithm solving OPT-REM problem

When polynomial space (instead of only logarithmic) is allowed, OPT-REM problem can be approximately solved using a modified version of Goemans-Williamson algorithm, enjoying the same .878 approximation guarantee.

But note that this algorithm requires $\Omega(n)$ space, thus it cannot be straightly used in constructing (11).

We first review the workflow of Goemans-Williamson algorithm. Max-Cut has the following standard QUBO (Quadratic unconstrained binary optimization) formalism

$$\max\left\{\sum_{(u,v)\in E}\frac{1-c_u\cdot c_v}{2}:c_u^2=1\right\}. \tag{C.1}$$

Goemans-Williamson algorithm uses unit real vectors $\boldsymbol{x}_u\in\mathbb{R}^n$ to approximate c_u, relaxing (C.1) as

$$\max\left\{\sum_{(u,v)\in E}\frac{1-\boldsymbol{x}_u\cdot\boldsymbol{x}_v}{2}:\|\boldsymbol{x}_u\|^2=1\right\}. \tag{C.2}$$

Consider a matrix M with entries defined as $M_{uv}:=\langle\boldsymbol{x}_u|\boldsymbol{x}_v\rangle$, then it is known that M is positive semidefinite. Moreover, for any positive semidefinite M, there exists $\{\boldsymbol{x}_u\}$ satisfying $M_{uv}:=\langle\boldsymbol{x}_u|\boldsymbol{x}_v\rangle$. Thus, there is a one-to-one correspondence between $\{\boldsymbol{x}_u:\|\boldsymbol{x}_u\|^2=1\}$ and $\{M:M_{uu}=1\}$. Now let L be the Laplacian matrix of the graph with entries defined as

$$L_{uv}:=\begin{cases}\deg(u), & u=v,\\ -1, & (u,v)\in E,\\ 0, & \text{otherwise},\end{cases} \tag{C.3}$$

where $\deg(u)$ means the degree of u, then,

$$\begin{aligned}\frac{\mathbf{tr}(LM)}{4}&=\frac{1}{4}\sum_{u\in V}\sum_{v\in V}L_{uv}(\boldsymbol{x}_u\cdot\boldsymbol{x}_v)\\ &=\frac{1}{4}\left(\sum_{u\in V}\deg(u)-\sum_{(u,v)\in E}2\cdot\boldsymbol{x}_u\cdot\boldsymbol{x}_v\right)\\ &=\sum_{(u,v)\in E}\frac{1-\boldsymbol{x}_u\cdot\boldsymbol{x}_v}{2}.\end{aligned}$$

From the discussion above, (C.1) is relaxed to the following SDP formulation:

$$\max_{M\succeq 0}\left\{\mathbf{tr}(LM)/4:M_{uu}=1\right\}. \tag{C.4}$$

From (C.4), we can get the optimal $\{\boldsymbol{x}_u\}$, after which we select a random hyperplane through the origin, separating $\{\boldsymbol{x}_u\}$ into two sets, which forms a cut.

Lemma 4 ([23, Theorem 3.3])**.** *For an arbitrary set of* $\{\boldsymbol{x}_u\}$*, the expected size of the cut obtained by separating the set using a random hyperplane is at least* .878 *times* $\mathbf{tr}(LM)/4$.

As (C.4) is a relaxation of Max-Cut (C.1), the optimal $\mathbf{tr}(LM)/4$ will be no less than Max-Cut. This fact along with Lemma 4 implies Lemma 5.

Lemma 5. *Goemans-Williamson algorithm has* .878 *approximation ratio on Max-Cut problem.*

To approximately solve the modified version of Max-Cut, OPT-REM, only a slight amendment to the described Goemans-Williamson algorithm is required. In OPT-REM, for some vertices, their colors are antecedently assigned, so we fix their $\{\boldsymbol{x}_u\}$ vectors in advance. For vertices that are assigned different colors, we need to make sure their corresponding vectors appear at different sides of the random hyperplane. To do this, we fix the vectors so that if two vertices have different colors, their corresponding vectors point to opposite directions. One possible choice is

$$\boldsymbol{x}_u := \begin{cases} (+1,0,\ldots,0), & \text{color of } u \text{ is } 0, \\ (-1,0,\ldots,0), & \text{color of } u \text{ is } 1. \end{cases} \tag{C.5}$$

Under such configuration, the entries of M are subject to further constraints. We use z to denote an arbitrary vertex whose vector is not fixed, and let the vertices whose colors are fixed 0 be $x_1, x_2, \ldots, x_a$ and the vertices whose colors are fixed 1 be $y_1, y_2, \ldots, y_b$, then for each z,

$$M_{zx_i} - M_{zx_{i+1}} = 0 \quad (\forall 1 \le i \le a-1), \tag{C.6}$$
$$M_{zy_i} - M_{zy_{i+1}} = 0 \quad (\forall 1 \le i \le b-1), \tag{C.7}$$
$$M_{zx_1} + M_{zy_1} = 0. \tag{C.8}$$

And,

$$M_{x_i y_j} = -1 \quad (\forall 1 \le i \le a, 1 \le j \le b). \tag{C.9}$$

Then we solve (C.4) under the additional constraints above. Similar to the reasoning towards Lemma 5, we have Lemma 6.

Lemma 6. *The modified Goemans-Williamson algorithm described in this section have .878 approximation ratio on* OPT-REM *problem.*

The existence of an efficient classical non-trivial approximation algorithm solving OPT-REM suggests that this problem, non-rigorously speaking, has close difficulty to the original Max-Cut problem.

Appendix D. Analyzing T size for Erdős-Rényi graphs

In this section, we give an elementary proof for Lemma 3. Remember we want to construct a set T satisfying

$$\forall s_0 \in \{0,1\}^{n_0}, \frac{\max_{s_1 \in T} \text{REM}(s_0, s_1)}{\text{OPT-REM}(s_0)} \geq \alpha.$$

We will start by showing a warm-up case where the graph is a complete bipartite graph, where the optimal T is trivial. Then, we will show that this trivial solution applies to random bipartite graphs where each pair of vertices is connected by an edge with probability p, which can somehow be named *Erdős-Rényi bipartite graph*. At last, we will show for (large) Erdős-Rényi graphs, the trivial solution can also achieve good performance if $\frac{n_0}{n}$ is large enough, because that the contribution from CUT_1 is comparatively small.

Appendix D.1. Warm-up: complete bipartite graph

Assume we have a complete bipartite graph $G = (V_0 \cup V_1, V_0 \times V_1)$, and consider if we want to construct T that satisfying (43).

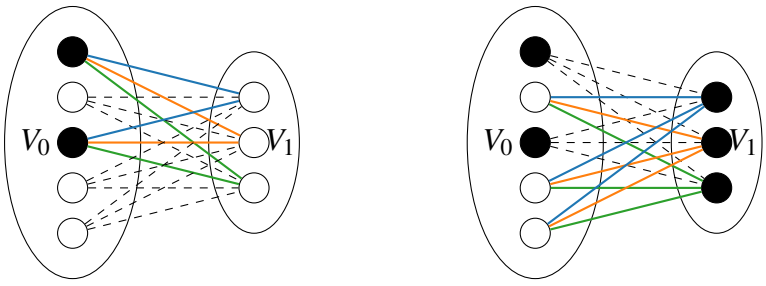


Figure D.11: An example of a complete bipartite graph. We should color all vertices in V_1 black since there are more white vertices in V_0.

First, we look at elements in $\{0,1\}^{n_0}$ independently and see how to compute its corresponding optimal coloring in $\{0,1\}^{n_1}$. For a coloring $s_0 \in \{0,1\}^{n_0}$ of vertices in V_0, assume there are t vertices colored black, and accordingly, there are $n_0 - t$ vertices colored white. Then for each vertex $v \in V_1$, if it is colored black, it will contribute $n_0 - t$ to the cut value, and t otherwise. Therefore, we simply compare t with $n_0 - t$, and color vertices in V_1 all black or all white, which is indeed the optimal solution, i.e.,

$$\text{OPT-REM}(s_0) = \max\{t, n_0 - t\} \cdot n_1. \tag{D.1}$$

Through the argument above, we can see that if we want a T satisfying (43), we can simply set $T = \left\{0^{\otimes n_1}, 1^{\otimes n_1}\right\}$ and this will give us a 1-guarantee set with $|T| = 2$. See Figure D.11 for reference.

Appendix D.2. Partial Erdős-Rényi graph

Consider a graph $G = (V_0 \cup V_1, E_0 \cup E_1 \cup E_{01})$, where $E_0 \subset V_0 \times V_0$, $E_1 \subset V_1 \times V_1$, $E_{01} \subset V_0 \times V_1$. And for arbitrary set S and probability p, we define a subset $T \sim \mathsf{ER}_p(S)$ if each element in S is included in T independently with probability p. Then for Erdős-Rényi graph,

$$\begin{aligned} E_0 &\sim \mathsf{ER}_p\Big(\big\{(u,v) : (u,v) \in V_0 \times V_0, u < v\big\}\Big), \\ E_1 &\sim \mathsf{ER}_p\Big(\big\{(u,v) : (u,v) \in V_1 \times V_1, u < v\big\}\Big), \\ E_{01} &\sim \mathsf{ER}_p(V_0 \times V_1). \end{aligned} \tag{D.2}$$

In this section, we first consider the following graph ensemble, which will later be referred to as *partial Erdős-Rényi graph*:

$$\begin{aligned} E_0 &\text{ is arbitrary}, \\ E_1 &= \emptyset, \\ E_{01} &\sim \mathsf{ER}_p(V_0 \times V_1). \end{aligned}$$

Similarly, we consider a coloring of V_0, and assume there are t vertices colored black, and accordingly, there are $n_0 - t$ vertices colored white. As a consequence of Erdős-Rényi property, for each vertex $v \in V_1$, its neighboring vertices in V_0 have $p \cdot t$ black vertices and $p \cdot (n_0 - t)$ white vertices in expectation. If there are precisely $p \cdot t$ black and $p \cdot (n_0 - t)$ white vertices, the case will be analogous to what we have argued in the previous section: simply compare t with $n_0 - t$ and color vertices in V_1 all black or all

white accordingly. Now we will argue, that this simple strategy still applies despite the graph being random.

Lemma 7 ($1-\varepsilon$-guarantee set for partial Erdős-Rényi graph)**.** *For partial Erdős-Rényi graph and* $\forall \varepsilon > 0$*,*

$$w.h.p \quad T = \{0^{\otimes n_0}, 1^{\otimes n_0}\} \text{ is a } 1-\varepsilon\text{-guarantee set.}$$

Proof. Still, assume there are t black vertices and $n_0 - t$ white ones in V_0. Then $\forall i \in V_1$, we let B_i, W_i be the number of black and white vertices connected to i, respectively. By definition,

$$B_i \sim \mathsf{Bionomial}(t, p), \tag{D.3}$$

$$W_i \sim \mathsf{Bionomial}(n_0 - t, p). \tag{D.4}$$

Then, the overall approximation ratio will be

$$\frac{\sum_i B_i}{\sum_i \max\{B_i, W_i\}}. \tag{D.5}$$

For arbitrary approximation ratio requirement α, we want to upper bound the probability where the approximation ratio is smaller than α. First note that

$$\forall a, b, \lambda > 0, \quad \max\{a, b\} \geq \frac{\lambda a + b}{\lambda + 1}. \tag{D.6}$$

Then we have

$$\frac{\sum_i B_i}{\sum_i \max\{B_i, W_i\}} \leq \frac{\sum_i B_i}{\sum_i \frac{\lambda B_i + W_i}{\lambda + 1}}, \tag{D.7}$$

and it suffices to upper bound

$$\mathbb{P}\left(\frac{\sum_i B_i}{\sum_i \frac{\lambda B_i + W_i}{\lambda + 1}} < \alpha \right) \tag{D.8}$$

$$= \mathbb{P}\left(\big(\lambda(1-\alpha) + 1\big) \sum_i B_i - \alpha \sum_i W_i < 0 \right). \tag{D.9}$$

From the properties of binomial distribution, we have

$$\boldsymbol{B} := \sum B_i \sim \mathsf{Bionomial}(n_1 \cdot t, p), \tag{D.10}$$

$$\boldsymbol{W} := \sum W_i \sim \mathsf{Bionomial}\big(n_1 \cdot (n_0 - t), p\big), \tag{D.11}$$

then $\big(\lambda(1-\alpha)+1\big)\boldsymbol{B}-\alpha\boldsymbol{W}$ can be made arbitrarily large once $\alpha<1$ and $\boldsymbol{B}\neq 0$. Specifically, let $\lambda_0=\frac{\alpha\cdot n_1\cdot(n_0-t)-1}{1-\alpha}$, then

$$\mathbb{P}\Big(\big(\lambda_0(1-\alpha)+1\big)\boldsymbol{B}-\alpha\boldsymbol{W}<0\Big) \tag{D.12}$$

$$\leq \mathbb{P}(\boldsymbol{B}=0)=\binom{n_1 t}{0}(1-p)^{n_1 t} \tag{D.13}$$

$$=(1-p)^{n_1 t}. \tag{D.14}$$

That is, if there are t black vertices and n_0-t white vertices in V_0, the probability that the approximation ratio is smaller than α is upper bounded by $(1-p)^{n_1 t}$.

Finally, we enumerate the number of black vertices in V_0 and employ union bound, and obtain that for all colorings of V_0, the probability where the approximation ratio is smaller than α is upper bounded by

$$2\sum_{t=\left\lceil\frac{n_0}{2}\right\rceil}^{n_0}\binom{n_0}{t}(1-p)^{n_1 t} \tag{D.15}$$

$$\leq 2\sum_{t=\left\lceil\frac{n_0}{2}\right\rceil}^{n_0}\left(\frac{en_0}{t}\right)^{t}(1-p)^{n_1 t} \tag{D.16}$$

$$=2\sum_{t=\left\lceil\frac{n_0}{2}\right\rceil}^{n_0}\exp\Big(t\big(1+\log n_0+n_1\log(1-p)-\log t\big)\Big) \tag{D.17}$$

$$\leq 2\sum_{t=\left\lceil\frac{n_0}{2}\right\rceil}^{n_0}\exp\Big(t\big(1+\log n_0+n_1\log(1-p)\big)\Big). \tag{D.18}$$

$$\tag{D.19}$$

Now we look at

$$1+\log n_0+n_1\log(1-p), \tag{D.20}$$

which is clearly dominated by the negative polynomial term $n_1\log(1-p)$. Let $\kappa:=n_0/n$, then we have

$$\begin{aligned}&\mathbb{P}\left(\frac{\sum_i B_i}{\sum_i \max\{B_i,W_i\}}<\alpha\right)\\ \leq& n\exp\left(\frac{\kappa(1-\kappa)}{2}\cdot n^2\log(1-p)+o(n^2)\right),\end{aligned} \tag{D.21}$$

meaning that the probability where the approximation ratio is smaller than α is exponentially small in n. □

Appendix D.3. (Large) Erdős-Rényi graph

Now we consider the general case where E_1 is not empty and defined as (D.2). We still analyze the approximation ratio given by $T = \{0^{\otimes n_1}, 1^{\otimes n_1}\}$. From Lemma 7, we can see if we neglect the edges in E_1, the optimal cut given by edges in E_{01} can be written as $\boldsymbol{B}/(1-\varepsilon)$ for an arbitrarily small $\varepsilon > 0$. And if we neglect the edges in E_{01}, the optimal cut given by edges in E_1 is Max-Cut of an Erdős-Rényi graph $G_1 = (V_1, E_1)$. So the joint optimal cut in $E_1 \cup E_{01}$ is upper bounded by $\boldsymbol{B}/(1-\varepsilon) + \textsc{Max-Cut}(G_1)$. Therefore, the approximation ratio is lower bounded by

$$\frac{\boldsymbol{B}}{\boldsymbol{B}/(1-\varepsilon) + \textsc{Max-Cut}(G_1)}. \tag{D.22}$$

Now we will use the fact that $\boldsymbol{B}$ and $\textsc{Max-Cut}(G_1)$ are both in fact concentrated.

Lemma 8 (Concentration of binomial distributions)**.** *For $X \sim \mathsf{Bionomial}(n,p)$, we have*

$$\mathbb{P}\Big(|X - pn| \le n^{0.6}\Big) = o(1). \tag{D.23}$$

Proof. Using the Hoeffding's inequality,

$$\mathbb{P}\Big(X \le pn - n^{0.6}\Big) \le \exp\left(-2 \cdot \frac{n^{1.2}}{n}\right) = o(1), \tag{D.24}$$

and the full inequality can be derived from the symmetry of binomial distribution. □

For $\boldsymbol{B}$, as a consequence of Lemma 8, we have with high probability,

$$\boldsymbol{B} > pn_1 t - (n_1 t)^{0.6} \tag{D.25}$$

$$\boldsymbol{B} < pn_1 t + (n_1 t)^{0.6}. \tag{D.26}$$

For $\textsc{Max-Cut}(G_1)$ [51], we have w.h.p,

$$\textsc{Max-Cut}(G_1) = n_1\left(\frac{pn_1}{4} + o(n_1)\right), \tag{D.27}$$

meaning that $\exists \lambda_1, \lambda_2$ s.t.

$$\textsc{Max-Cut}(G_1) < n_1 \left(\frac{pn_1}{4} + \lambda_1 n_1^{1-\lambda_2} \right) \tag{D.28}$$

Combining two concentration results, we have w.h.p

$$\frac{\boldsymbol{B}}{\boldsymbol{B}/(1-\varepsilon) + \textsc{Max-Cut}(G_1)} \tag{D.29}$$

$$> \frac{pn_1 t - (n_1 t)^{0.6}}{(pn_1 t + (n_1 t)^{0.6})/(1-\varepsilon) + n_1 \left(\frac{pn_1}{4} + \lambda_1 n_1^{1-\lambda_2} \right)} \tag{D.30}$$

$$= \frac{(1-\varepsilon) + o(1)}{\left(1 + (1-\varepsilon)\frac{n_1}{4t}\right) + o(1)}. \tag{D.31}$$

To have the above ratio larger than α in the limit of large n, it suffices to require

$$\frac{n_1}{4t} < \frac{1}{\alpha} - 1, \tag{D.32}$$

then as $t \geq \frac{n_0}{2}$, finally we need

$$\frac{n_0}{n} > \frac{\alpha}{2-\alpha}. \tag{D.33}$$

In other words, this result says if we have determined $n_0/n := \kappa$, then trivially set $T = \{0^{\otimes n_1}, 1^{\otimes n_1}\}$ gives us a α-guarantee set with $\alpha = 2\kappa/(\kappa+1)$. This result suggests that Erdős-Rényi graphs could have a small candidate set T that gives a good approximation ratio.