Quantum Divide and Conquer for Combinatorial Optimization and Distributed Computing

Zain H. Saleem,[1,*] Teague Tomesh,[2,3,†] Michael A. Perlin,[4,‡] Pranav Gokhale,[3,§] and Martin Suchara[1,¶]

[1] *Argonne National Laboratory, 9700 S. Cass Ave., Lemont, IL 60439, USA.*
[2] *Department of Computer Science, Princeton University, Princeton, NJ 08540, USA.*
[3] *Super.tech, Chicago, IL, USA.*
[4] *JILA, NIST and University of Colorado, 440 UCB, Boulder, Colorado 80309, USA.*

(Dated: July 19, 2021)

We introduce a quantum divide and conquer algorithm that enables the use of distributed computing for constrained combinatorial optimization problems. The algorithm consists of three major components: classical partitioning of a target graph into multiple subgraphs, variational optimization over these subgraphs, and a quantum circuit cutting procedure that allows the optimization to take place independently on separate quantum processors. We simulate the execution of the quantum divide and conquer algorithm to find approximate solutions to instances of the Maximum Independent Set problem which have nearly twice as many nodes than the number of qubits available on a single quantum processor.

I. INTRODUCTION

Building quantum chips with a sufficiently large number of qubits suitable for solving practically sized problems has remained an elusive goal. Distributed quantum computing [1, 2] that uses multiple smaller systems connected by quantum communication channels promises to offer a path to scalability. Suitable quantum connections can be offered by quantum networks such as the quantum internet [3–5], with proof-of-concept experimental realizations under construction by several teams [6–9]. However, distributed quantum computing will require reliable, low-latency quantum connectivity between the quantum systems, and the amount of quantum communication therefore needs to be minimized. This work addresses this challenge by designing a quantum optimization algorithm that minimizes communication between subproblems. In the absence of a working physical quantum communication network, we use the Quantum Divide and Conquer (QDC) approach that allows connecting quantum systems by using only classical communication.

The first component of the QDC algorithm takes an input graph and classically partitions it into multiple subgraphs. Graph partitioning approaches are commonly used for tackling combinatorial optimization problems in the classical algorithms literature [10]. The idea is to break down a large problem into more manageable subproblems. In this work, we use the Kernighan-Lin algorithm [11] to partition graphs into approximately balanced subgraphs.

The second component of QDC consists of a quantum variational optimization on the partitioned subgraphs. We use the dynamic quantum variational ansatz (DQVA) to perform this task [12]. The DQVA algorithm was designed for tackling constrained combinatorial optimization problems, such as Maximum Independent Set (MIS). It is based on an ansatz which dynamically changes its structure to make more efficient use of a fixed allocation of quantum resources. However, DQVA is not the only approach to solving constrained optimization problems. We could have also used quantum local search which was recently demonstrated to efficiently find approximate solutions to the MIS problem on large graph instances [13].

Finally, the third component of QDC is the circuit cutting technique which divides large quantum circuits into smaller fragments [14–17]. These smaller fragments can then be independently executed on separate quantum devices and their outputs can be recombined using classical post-processing to obtain the output of the larger, uncut circuit. While circuit cutting is computationally expensive, with a cost that grows exponentially with the number of cuts required to partition a circuit, in our approach we carefully structure the ansatz in such a way that the number of cuts required is tunable to the available post-processing resources.

In this work, we combine these three components: (1) graph partitioning, (2) quantum variational optimization, and (3) circuit cutting to approximate the Maximum Independent Set (MIS) problem for large graphs that would otherwise require more qubits than are currently available on an individual noisy intermediate-scale quantum (NISQ) device [18]. Our key contributions are as follows. First, we provide specifications for building the variational ansatz, choosing the cut locations, and the dynamic ansatz update that enables the execution of QDC with manageable quantum and classical costs. Second, we provide the first demonstration of quantum circuit cutting for a useful, practical application. Using this technique we are able to find approximate solutions to MIS on graphs containing up to 26 nodes while only requiring the simulation of up to 15-qubit quantum cir-

* zsaleem@anl.gov;
† ttomesh@princeton.edu
‡ mika.perlin@gmail.com
§ pranav@super.tech
¶ msuchara@anl.gov

cuits. Finally, we study how differing amounts of entanglement allowed across the graph partition impact the optimality of the solution.

We begin with the background related to constrained quantum variational optimization. Section II A introduces the quantum approximate optimization algorithm and the quantum alternating operator ansatz is covered in Section II B. We describe the DQVA in Section II C and briefly explain the circuit cutting algorithm in Section II D. In Section III we discuss classical approaches to graph partitioning for the MIS problem. We introduce a classical variant of the divide and conquer algorithm and compare its performance to the well known Boppana-Halldórsson algorithm [19] on both 3-regular and Erdös-Rényi graphs with 100 nodes. In Section IV we cover the QDC algorithm in depth, and compare its performance on random 3-regular graphs to both Boppana-Halldórsson and the classical divide and conquer algorithm. Finally, Section V contains conclusions and future directions.

II. BACKGROUND

A. Quantum Approximate Optimization Algorithm

The Quantum Approximate Optimization Algorithm (QAOA) [20] is a hybrid quantum-classical algorithm for solving combinatorial optimization problems. We start by defining the graph-dependent classical objective function $C(\mathbf{b})$ that we want to optimize over n-bit strings $\mathbf{b} = \{b_1, b_2, b_3 \dots b_n\} \in \{0,1\}^n$,

$$C_{obj} \left|b\right\rangle = C(\mathbf{b}) \left|b\right\rangle . \tag{1}$$

The quantum and classical processors then work together to optimize the expectation value of this objective function. The role of the classical computer is to select and update a set of parameters that define a variational ansatz (i.e., a parametrized quantum circuit) based on the current expectation value. The quantum computer's job is to then execute the ansatz and measure the new expectation value. This variational loop continues until a maximum, minimum, or some other threshold condition is reached.

There are three main components to the variational ansatz. First is the initial state $|s\rangle$ which is the state on which we act with carefully chosen unitary operators to create our ansatz. The other two components are the phase separation unitary $e^{i\gamma C}$ and the mixing unitary $e^{i\alpha M}$. C is a diagonal operator, in the computational basis, related to the problem at hand and is usually the same as the objective operator. This unitary plays the role of a phase separator between successive applications of the mixing operator. The angles γ and α are variational parameters with values between $[0, 2\pi]$. M is non-diagonal in the computational basis, and its job is to mix the states amongst each other during the optimization.

The variational ansatz, $|\psi_p(\boldsymbol{\gamma}, \boldsymbol{\alpha})\rangle$, is constructed by combining these three components,

$$|\psi_p(\boldsymbol{\gamma}, \boldsymbol{\alpha})\rangle = e^{-i\alpha_p M} e^{-i\gamma_p C} \dots e^{-i\alpha_1 M} e^{-i\gamma_1 C} \left|s\right\rangle , \tag{2}$$

where p controls the number of times the unitary operators are applied. The expectation value of C_{obj} in this variational state,

$$E_p(\boldsymbol{\gamma}, \boldsymbol{\alpha}) = \langle \psi_p(\boldsymbol{\gamma}, \boldsymbol{\alpha}) | C_{obj} | \psi_p(\boldsymbol{\gamma}, \boldsymbol{\alpha}) \rangle , \tag{3}$$

is found on a quantum computer and is passed to a classical optimizer to find the optimal parameters $\arg\max_{\boldsymbol{\gamma},\boldsymbol{\alpha}} E_p(\boldsymbol{\gamma}, \boldsymbol{\alpha})$. Since the eigenstates of C_{obj} are computational basis states, this maximization is achieved for the states corresponding to the solutions of the original classical problem.

In the penalty term approach to MIS, the mixing operator is $M = \sum_j X_j$, where X_j is the Pauli-X operator for qubit j, while the objective function (which is used as a phase separator) is [12]

$$C_{obj} = H - \lambda P = \sum_{j \in V} b_j - \lambda \sum_{(j,k) \in E} b_k b_j, \tag{4}$$

where λ is a positive real number,

$$b_j = \frac{1 - Z_j}{2} = |1\rangle\langle 1|_j \tag{5}$$

is a projector onto state $|1\rangle$ of qubit j, and Z_j is the Pauli-Z operator on qubit j. Here H gives us the Hamming weight of the state it acts on, and P is a penalty term for enforcing independent sets by "discouraging" pairs of nodes connected by an edge from simultaneously occupying the $|1\rangle$ state. The inclusion of a penalty term effectively reduces our constrained problem to an unconstrained one, in the sense that now the optimization will be performed over the space of all bit strings $\{0,1\}^n$ during the variational optimization. In this work, however, we will take a different approach. Rather than searching over the space of all bitstrings, we will use a quantum alternating operator ansatz (QAO-Ansatz) [21, 22] that ensures we never leave the space of feasible states (i.e. bitstrings representing truly independent sets) during the optimization.

B. A Quantum Alternating Operator Ansatz for MIS

Much like vanilla QAOA, the QAO-Ansatz [21–23] consists of three parts: an initial state, a phase separation unitary, and a mixing unitary. For the initial state, one can simply pick the all-zero state, or use a "warm start" found by a classical algorithm [24]. The phase separation unitary for our QAO-Ansatz is, as before, generated by the objective function. Since the QAO-Ansatz is designed to keep the computation within the space of feasible states there is now no need to include penalty terms,

so the objective function to maximize for MIS is simply the Hamming weight operator:

$$C_{obj} = H = \sum_{j \in V} b_j. \tag{6}$$

The mixing unitary of the ansatz is again non-diagonal in the computational basis, but it must additionally ensure that the state evolution maps feasible states to other feasible states. One way to ensure that we stay within the space of feasible states is to make sure that a qubit can only rotate from $|0\rangle$ to $|1\rangle$ if none of its neighbors are in the state $|1\rangle$ (equivalently, if all of its neighbors are in $|0\rangle$). Conditioning the individual X_j terms of the mixer on the neighbors of j in this manner results in a mixing unitary of the form $U_M(\alpha) = \prod_j V_j(\alpha)$, where the partial mixers $V_j(\alpha) = e^{i\alpha M_j}$ are generated by $M_j = X_j \bar{B}_j$, and

$$\bar{B}_j = \prod_{k \in \mathcal{N}(j)} \bar{b}_k, \qquad \bar{b}_k = \frac{1 + Z_k}{2} = |0\rangle\langle 0|_k . \tag{7}$$

Here $\mathcal{N}(j)$ denotes the neighbors of node j. Using the fact that the projector $\bar{B}_j^2 = \bar{B}_j$, the partial mixers can equivalently be written as

$$V_j(\alpha) = \bar{B}_j \times e^{i\alpha X_j} + (1 - \bar{B}_j) \times I, \tag{8}$$

where I is the identity operator, which highlights the interpretation of these partial mixers multi-controlled qubit rotations. As the partial mixers V_j generally do not commute with each other ($[V_j, V_k] \neq 0$), it is more natural to define a mixing unitary equipped with some permutation σ of $\{1, 2, \cdots, n\}$:

$$U_M^{\sigma}(\alpha) = V_{\sigma(n)}(\alpha) \cdots V_{\sigma(2)}(\alpha) V_{\sigma(1)}(\alpha). \tag{9}$$

This choice of mixing unitary ensures that we are always in the space of feasible states, but it comes with its own cost. The mixing unitary defined by Eq. (9) is expensive in terms of the number of quantum gates required for its implementation. For each partial mixer, $V_j(\alpha)$, the $\bar{B}_j$ operator that checks whether all of node j's neighbors are in the $|0\rangle$ state can be implemented via a multi-controlled Toffoli gate. Most quantum instruction set architectures express multi-qubit gates in terms of single- and two-qubit gates [25, 26]. The decomposition of such multi-qubit gates can result in an overhead of additional qubits and gates that quickly outstrips the available resources of current quantum computers [27, 28]. However, some quantum computing architectures, such as neutral atoms, may natively support these types of operations and would be a promising candidate for implementing circuits containing multi-controlled gates. With these resource constraints in mind, we utilize the dynamic quantum variational ansatz to minimize the number of multi-controlled Toffoli gates needed by dynamically turning some partial mixers off.

C. The Dynamic Quantum Variational Ansatz

The details of the dynamic quantum variational ansatz are described in [12]. The algorithm has three main steps: warm start, mixer initialization, and the dynamic ansatz update. To warm start the algorithm we start with some initial state that is obtained via a classical optimization algorithm $|\mathbf{c}\rangle = |c^1 c^2 \cdots c^n\rangle$. A similar approach was recently proposed for Max-Cut and QUBOs in [24].

In the next step we prepare the state

$$U_C(\gamma_p) U_M^{\sigma_p}(\boldsymbol{\alpha}_p) \cdots U_C(\gamma_1) U_M^{\sigma_1}(\boldsymbol{\alpha}_1) |\mathbf{c}\rangle \tag{10}$$

where, similarly to Eq. (9), each of the individual mixers given by

$$U_M^{\sigma_k}(\boldsymbol{\alpha}_k) = V_{\sigma_k(n)}\left(\alpha_k^{\sigma_k(n)}\right) \cdots V_{\sigma_k(1)}\left(\alpha_k^{\sigma_k(1)}\right) \tag{11}$$

for $k = 1, 2, \cdots, p$. For simplicity, we fix all permutations $\sigma_1 = \sigma_2 = \cdots = \sigma_p$ in practice, and henceforth drop the permutation index on mixers for brevity.

Wherever the i-th bit of the initial state $c^i = 1$, we set the parameter in the corresponding partial mixer to be $\alpha_k^i = 0$, essentially "turning off" this partial mixer and setting it to the identity operator I. We then calculate the expectation value of the objective function for this variational ansatz. This is where the dynamic ansatz step begins. If the optimization improves the Hamming weight and we get a new state $|\mathbf{q}\rangle$ with a Hamming weight larger than $|\mathbf{c}\rangle$, then we update the initial state $|\mathbf{c}\rangle$ with $|\mathbf{q}\rangle$. Again, we update the mixing unitaries such that if $q_1^i = 1$ then the i-th parameter of each partial mixer is set to zero: $\alpha_k^i = 0$.

D. Circuit Cutting

An n-qubit quantum circuit can be represented by a unitary operator U acting on an initial state $|\psi_0\rangle = |0\rangle^{\otimes n}$ as $U|\psi_0\rangle$. Measuring such a quantum circuit produces bitstings $b_1 \ldots b_n \in \{0, 1\}^n$ that are observed with probability $p(b_1 \ldots b_n) = |\langle b_1 \ldots b_n | U | \psi_0 \rangle|^2$. The unitary operator U is composed of a sequence of gates that can be represented as vertices of a graph. In Ref. [14] it was shown how one can fragment such a graph into two or more subgraphs or subcircuits. These subgraphs can be "stitched together" by performing a complete set of measurements on the output of one subcircuit at a cut, and then preparing a corresponding set of states on the input of the other subcircuit at that cut (see Fig. 1). In practice, one can independently execute and measure the outputs of these subcircuits and classically postprocess these results to obtain the output of the original uncut quantum circuit [15, 16]. Specifically, the probability of measuring bitstring $b_1 \ldots b_n$ at the end of a circuit that

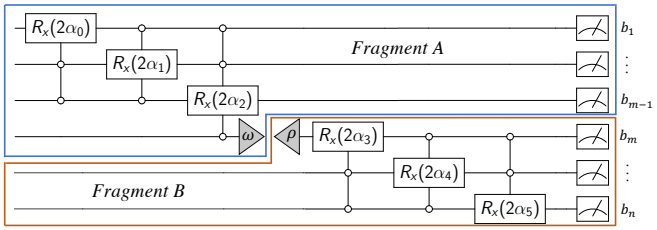

FIG. 1: A circuit can be cut into fragments A, B, which can be executed independently and recombined via classical post-processing of their outputs [14–16]. The inter-fragment entanglement severed by the cut is simulated by correlating the measurements, ω, performed at the cut of fragment A and the states, ρ, prepared at the cut of fragment B.

has been split into two fragments A, B is

$$p(b_1 \dots b_n) = \frac{1}{2} \sum_{\substack{\alpha \in \{X,Y,Z\} \\ b,b' \in \{0,1\}}} \gamma_{bb'} p_A^\alpha(b_1 \dots b_m; b) p_B^\alpha(b'; b_{m+1} \dots b_n), \tag{12}$$

where $\gamma_{bb'} = 2\delta_{bb'} - 2/3$, $p_A^\alpha(b_1 \dots b_m; b')$ is the probability of measuring bitstring $b_1 \dots b_m b'$ on the output of fragment A when measuring its cut qubit in the diagonal basis of the Pauli operator α, and $p_B^\alpha(b'; b_{m+1} \dots b_n)$ is the probability of measuring $b_{m+1} \dots b_n$ on the output of fragment B when its cut qubit is initially prepared in state b of the diagonal basis of α. This procedure can be repeated recursively to break the circuit into ever smaller fragments. Note that the decomposition in Eq. (12) is provided for illustrative purposes, and that more efficient recombination procedures should be used in practice [15, 16].

III. GRAPH PARTITIONING FOR MIS

A graph G is an ordered pair of disjoint sets (Q, E), where $E \subseteq Q \times Q$. Here Q is the set of nodes in the graph G, while E is the set of (unordered) edges in G. If we have n nodes in the graph then $Q = \{q_1, q_2 \cdots q_n\}$. Given a graph G, two nodes $q_i, q_j \in Q$ are said to be neighbours if $\{q_i, q_j\} \in E$. The set of neighbors of a node is defined as $\mathcal{N}(q_i) = \{(q_j) : \{q_i, q_j\} \in E\}$ and its degree $\mathcal{D}(q_i) = |\mathcal{N}(q_i)|$ is equal to the number of neighbors.

A graph partition is the division of the nodes Q of a graph G into disjoint subsets $Q_s \subset Q$ for which $\bigcup_s Q_s = Q$. In this work, we will consider bi-partitions Q_A, Q_B of roughly equal size. This partitioning can be performed using the Kernighan-Lin algorithm [11], which tries to minimize the number of edges $m = |\{\{q_i, q_j\} \in E : q_i \in Q_A \wedge q_j \in Q_B\}|$ that go between subgraphs $G_A = (Q_A, E_A)$ and $G_B = (Q_B, E_B)$, where $E_s = \{\{q_i, q_j\} \in E : q_i, q_j \in Q_s\}$ is the restriction of E to edges between nodes within Q_s. Within a subgraph G_s we can define two disjoint subsets Q_s^{uc} and Q_s^c of $Q_s = Q_s^{uc} \cup Q_s^c$ where Q_s^{uc} is the set of "uncut" nodes whose neighbors are entirely in Q_s, while Q_s^c is the set of "cut" nodes that have at least one neighbor in $Q_{\bar{s}}$ (with $\bar{s} \neq s$).

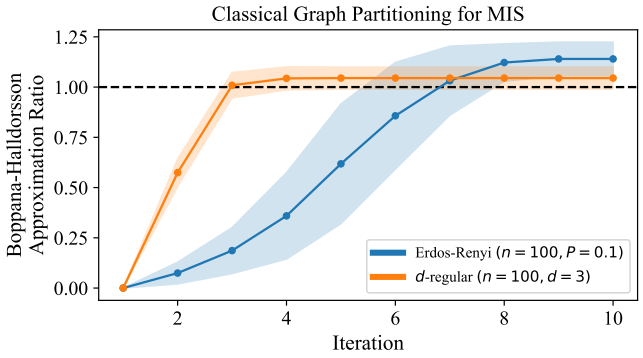

FIG. 2: Finding independent sets on 100-node graphs using graph partitioning and classical MIS algorithms. For fifty random 3-regular (orange lines) and fifty random Erdös-Rényi (blue lines) graphs, we executed an iterative, classical divide and conquer algorithm which first partitions the graph, then applies the classical Boppana-Halldórsson algorithm on the subgraphs. The R_{BH} (see Eq. 13) for each type of graph are shown over the iterations of the algorithm.

We demonstrate the efficacy of graph partitioning for solving large MIS instances in Fig. 2. We consider both random Erdös-Rényi graphs, where each pair of nodes has an edge with probability p, and d-regular graphs, where each node has d random neighbors. We compare the performance of a classical divide and conquer algorithm against Boppana-Halldórsson [19], a polynomial time heuristic algorithm, applied to the full graph. Since a 100-node graph is too large to perform a brute-force-search for the optimal MIS, we measure the relative sizes of the independent sets

$$R_{BH} = \frac{H(\boldsymbol{q}_{DC})}{H(\boldsymbol{q}_{BH})} \tag{13}$$

where H is the Hamming weight operator applied to the solution bitstrings produced by the divide and conquer ($\boldsymbol{q}_{DC}$) and Boppana-Halldórsson ($\boldsymbol{q}_{BH}$) algorithms. The classical divide and conquer algorithm in Fig. 2 uses the Kernighan-Lin algorithm to first partition the full graph, then applies the Boppana-Halldórsson algorithm to the subgraphs. The fact that the graph partitioning leads to larger independent sets demonstrates the usefulness of the divide and conquer approach. This allows the MIS algorithms to focus on finding better solutions to smaller subproblems instead of immediately attempting to find a good global solution.

The algorithm introduced in this work shares many similarities to the divide and conquer algorithm in Fig. 2.

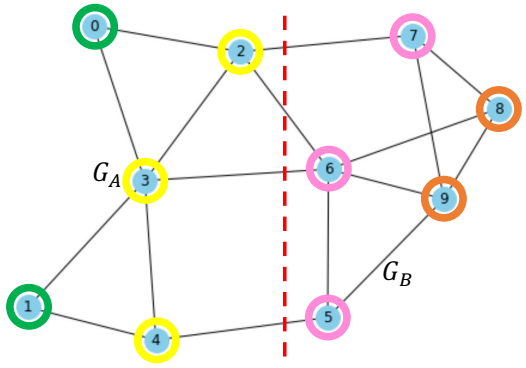

FIG. 3: An example 10-vertex graph partitioned into 5-vertex subgraphs G_A and G_B. The cut and uncut sets of nodes are color coded: $Q_A^{uc} = \{0, 1\}$ (green), $Q_A^c = \{2, 3, 4\}$ (yellow), $Q_B^{uc} = \{8, 9\}$ (orange), $Q_B^c = \{5, 6, 7\}$ (pink).

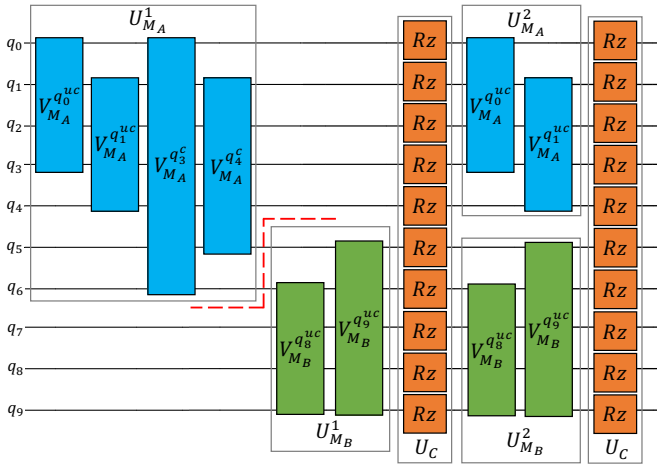

FIG. 4: Example $p = 2$ ansatz for the graph shown in Figure 3. In this example $t = 2$. The dotted red line shows the two cuts needed to separate the circuit into two fragments.

However, instead of using a classical algorithm to solve the subproblems produced by the graph partitioning, we apply DQVA, a hybrid quantum-classical algorithm, together with quantum circuit cutting techniques. Instead of solving each subproblem independently, this strategy allows information to flow between the subgraphs.

In order to understand both the potential and the cost of combining DQVA with circuit cutting, it is necessary to examine the number of circuit edges that must be cut. In the case of the MIS problem, there are two levels of partitioning (cutting) which need to take place. First, the input graph must be *partitioned* into two separate subgraphs. Then, the DQVA circuit that is constructed must be *cut* into two subcircuits corresponding to each of the subgraphs. Here, we try to use the words "partition" when referring to the input graph, and "cut" when referring to the quantum circuit. Note that the number of split edges in the graph partition may not equal the number of cut circuit edges. The number of circuit edges which need to be cut corresponds to the number of graph edges crossing the partition that are included within the DQVA's set of *active* partial mixer unitaries. Depending on the amount of classical and quantum resources at our disposal we can choose to include more or less of the partitioned graph edges within the DQVA circuit.

A property of particular interest is the *bisection width* of a graph, which is the minimum number of edges that must be cut in order to partition a graph into two sets of (roughly) equal size [29]. For large ($n \to \infty$) Erdös-Rényi graphs the bisection width is at least $\frac{n^2 p}{4} - O\left(n^{3/2}\sqrt{p(1-p)}\right)$ and at most $\frac{n^2 p}{4} - O\left(n\sqrt{p(1-p)}\right)$ [30]. For large d-regular graphs the bisection width is at most $\frac{d-2}{4}n + O(d\sqrt{n}\log n)$ [31]. To the best of our knowledge, explicit bisection widths for general *random* d-regular graphs are not known. However, for the special case of random 4-regular graphs, it was proven that the bisection width is at most $(\frac{1}{3} + \epsilon)n$ [32].

In practice, we see these differences in bisection width manifest in Fig. 2 as distinct rates of convergence. For the 100-node graphs considered in Fig. 2, the Erdös-Rényi and d-regular graphs required splitting an average of 180 and 35 edges, respectively, for a successful bisection of the full graph. The d-regular graphs can be partitioned with fewer edges so more nodes within the subgraphs can be added to the current independent set in each iteration.

From the standpoint of DQVA with circuit cutting, these results suggest that d-regular random graphs with $d \in O(1)$ – which are sparser than Erdös-Rényi graphs with $p \in O(1)$ – are a favorable target, simply because these graphs have lower maximum bisection width. Note that finding the bisection width is NP-Hard in general [33]. Therefore, we do not expect to always achieve optimal cuts with heuristics such as Kernighan-Lin partitioning. Nonetheless, the bounds imposed on bisection width capture the best case results we could expect using our methods.

IV. QUANTUM DIVIDE AND CONQUER

A. Algorithm Components

Here we combine the dynamic quantum variational ansatz and circuit cutting to find the MIS of a particular problem graph. The basic idea will be to partition a graph and perform local optimization that tries to solve MIS within each subgraph. Circuit cutting will be used to mediate correlations between the local optimizations within subgraphs, which should enable finding better solutions to MIS than an uncorrelated local optimization. We outline the algorithm below, with pseudocode given in Alg. 1, and an implementation is available via Github [34]:

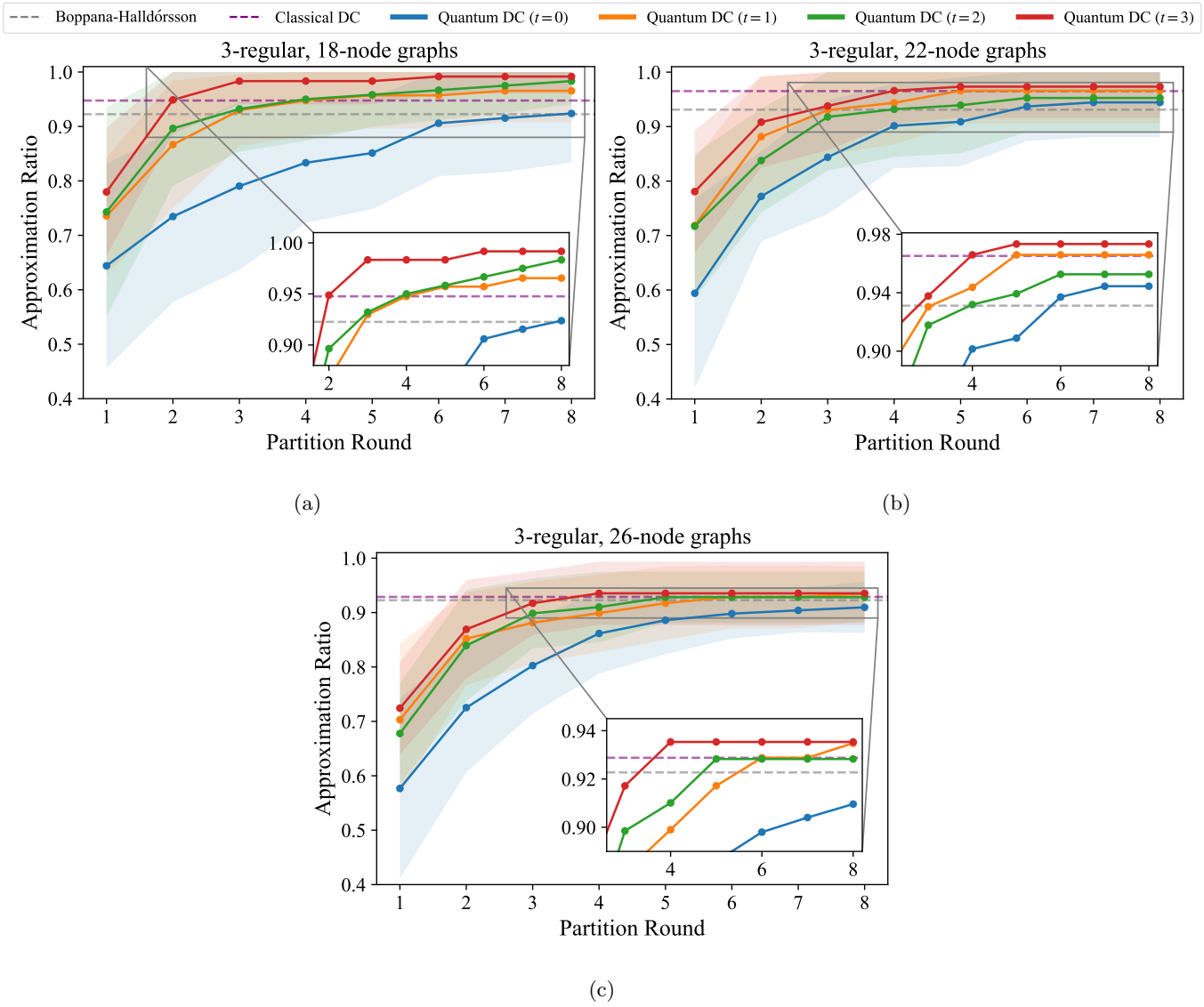

FIG. 5: Average performance of the quantum divide and conquer algorithm for increasing values of t (the number of edges allowed to cross the cut) collected over 15 random 3-regular graphs with 18, 22, and 26 nodes. For each point, we retain the best of five repetitions and average over the 15 graphs. As t increases, and partial mixers with more edges crossing the partition are allowed, the algorithm is (1) able to find larger independent sets and (2) converges to the solution faster.

1. Graph Partitioning: Given a graph $G = (Q, E)$, partition the graph into two subgraphs G_A and G_B of roughly equal sizes using the Kernighan-Lin algorithm [11] to minimize the number of crossing edges.

In subsequent iterations of this algorithm, the graph partitioning may be biased to favor partitioning edges containing qubits that have been assigned to $|1\rangle$ in the initial state (i.e. included in the current independent set). Such a partition is preferable because these vertices do not require any partition-crossing partial mixers (discussed in the steps below). This biasing may be accomplished by reducing the weight of edges involving vertices who have been assigned to $|1\rangle$ in the initial state. Thereafter, the Kernighan-Lin algorithm should minimize the weight (rather than the number) of partition-crossing edges.

2. Circuit Cutting and "Hot Node" Selection: We now need to choose the set Q^{hot} of "hot nodes", or the subset $Q^{\text{hot}} \subset Q^c$ of cut nodes $Q^c \equiv Q^c_A \cup Q^c_B$ to which we will apply nontrivial partial mixers in the first layer of the QAO-Ansatz. In contrast, the "cold nodes" $Q^{\text{cold}} = Q^c \setminus Q^{\text{hot}} = \{q \in Q_c : q \notin Q^{\text{hot}}\}$ (i.e. the set of cut nodes that are not hot) will *always* have trivial partial mixers, with $\alpha^i_k = 0$ for all nodes $q \in Q^{\text{cold}}$. As the number of cut nodes $|Q^c|$ is typically small, we use a brute-force search to find the subsets $Q^{\text{hot}}, Q^{\text{cold}}$ that

Algorithm 1: Quantum Divide and Conquer

Input : $G = (Q, E)$, m = # of partition rounds, t = max # of cuts, p = circuit depth
Output: Approximate MIS of G

$s \leftarrow$ initial state selection;
$s_{best} \leftarrow$ "00...0";
for $r \in [m]$ **do**
  $h_{new} \leftarrow H(\mathrm{s})$;
  $h_{old} \leftarrow -1$;
  /* Graph Partitioning */
  $G_A, G_B, Q^c \leftarrow$ kernighan_lin_bisection(G);
  while $h_{new} > h_{old}$ **do**
    /* Hot Node Selection */
    $Q^{hot} \leftarrow \min(C_{cut}(Q^{cold}),$ s.t. $\left|\mathcal{I}(Q^{\mathrm{hot}})\right| \leq t)$;
    /* Ansatz Construction */
    /* For this example we assume $p = 1$ */
    $\vec{\alpha} \leftarrow [0_1, 0_2, \ldots, 0_n]$;
    $\gamma \leftarrow \mathrm{random}(0, 2\pi)$;
    for $q \in \{Q_A^{uc} \bigcup Q_B^{uc} \bigcup Q^{hot}\}$ **do**
      if $s_q = 0$ **then**
        $\alpha_q \leftarrow \mathrm{random}(0, 2\pi)$
      end
    end
    $U_{ansatz}(\vec{\alpha}, \gamma) \leftarrow U_M(\vec{\alpha}) U_C(\gamma)$;
    /* Inner Variational Loop */
    while *not converged* **do**
      /* Circuit Cutting */
      subcircs, cuts $\leftarrow \mathrm{cut}(U_{ansatz}(\vec{\alpha}, \gamma)\,|s\rangle\,, t)$;
      subcirc_counts $\leftarrow$ evaluate(subcircs);
      counts $\leftarrow$ reconstruct(subcirc_counts, cuts);
      $E \leftarrow$ expectation_value(H, counts);
    end
    $h_{old} \leftarrow H(s)$;
    $h_{new} \leftarrow \max_b([H(b)$ for b in counts$])$;
    $s \leftarrow \arg\max_b([H(b)$ for b in counts$])$;
    if $h_{new} > H(s_{best})$ **then**
      $s_{best} \leftarrow s$;
    end
  end
end
return s_{best}

minimize the "cut cost"

$$C_{\mathrm{cut}}(Q^{\mathrm{cold}}) = \sum_{i \in Q^{\mathrm{cold}}} \sum_{\substack{j \in \mathcal{N}(i) \\ g(j) \neq g(i)}} \mathcal{D}(j), \tag{14}$$

where $g(q) \in \{G_A, G_B\}$ is the subgraph containing node q, and $\mathcal{D}(q)$ is the degree of node q (i.e., its number of neighbors). The minimization of $C_{\mathrm{cut}}(Q^{\mathrm{cold}})$ is performed with the constraint that the number of inter-graph neighbors of hot nodes $q \in Q^{\mathrm{hot}}$, i.e. the size of the set

$$\mathcal{I}(Q^{\mathrm{hot}}) = \bigcup_{i \in Q^{\mathrm{hot}}} \{j \in \mathcal{N}(i) : g(j) \neq g(i)\}, \tag{15}$$

does not exceed some number t of maximal cuts that will be tolerated for circuit cutting. In words, partial mixers for the cold nodes in Q^{cold} are essentially "thrown out" in the sense that they are always trivial: $V_i(\alpha_k^i) = V_j(0) = I$ (the identity operator) for all $j \in Q^{\mathrm{cold}}$. The cut cost C_{cut} then penalizes throwing out the partial mixers for nodes i with high-degree neighbors j in the complement subgraph $g(j) \neq g(i)$. This penalty heuristically tries to maximize the amount of inter-subgraph correlation that is preserved by circuit cutting.

In principle, the number of cuts required to separate the variational ansatz depends on the choice of partial mixer order. To simplify the brute-force search for hot nodes, we therefore enforce that partial mixers are always ordered by their subgraph (we test both possible orders), and that the set of hot nodes is chosen from the first subgraph in this ordering (below, we call this subgraph G_A without loss of generality). This restriction also ensures that the number of circuit cuts that will be required for our ansatz is precisely $\left|\mathcal{I}(Q^{\mathrm{hot}})\right|$.

3. Ansatz Construction and Optimization: For a chosen integer p, create the variational ansatz $U_{\mathrm{ansatz}} = U_C(\gamma_p)U_M(\boldsymbol{\alpha}_p)\cdots U_C(\gamma_1)U_M(\boldsymbol{\alpha}_1)$ with some (fixed) order of partial mixers. In the $k = 1$ layer, fix $\alpha_1^j = 0$ for all cold nodes $j \in Q^{\mathrm{cold}}$, and subsequently fix $\alpha_k^j = 0$ in all layers $k > 1$ for all cut nodes $j \in Q^c$.

Once U_{ansatz} has been constructed, optimize its (non-fixed) parameters within a quantum-classical variational loop and store the bitstring obtained upon completion $\mathbf{q}_A$. The circuit cutting technique applied within this variational loop (see Figure 4), effectively increases the size of the quantum circuits that can be run on the quantum computer and therefore allows this algorithm to target larger graphs [15, 16].

4. Rerun with a new random partition: Repeat the above steps as many times as desired, each time using a new random bi-partition of the graph, and using the best feasible state (with largest Hamming weight) observed throughout the optimization in Step 31 as a new initial state for the algorithm.

B. Simulation Results

We evaluated the performance of QDC on 18-, 22-, and 26-node 3-regular graphs over the course of $m = 8$ partition rounds with increasing number of inter-partition edges t. In addition, we also ran the Boppana-Halldórsson and classical divide and conquer (introduced in Sec. III) on the same graphs for comparison. The results in Figure 5 are averaged over 15 random 3-regular graphs. On every graph, we executed each algorithm five separate times and selected the best of these five repetitions.

We studied how increasing the value of t, which increases the amount of entanglement allowed across the graph partitions, impacts the optimality of the solution. It can be easily seen in Figure 5 that allowing more inter-partition edges in the quantum circuit increases not only the speed of convergence to the solutions but also im-

proves the quality of the solutions. The QDC algorithm benefits from the increased flow of information between subgraphs provided by the inter-partition edges, and performs better than both the Boppana-Halldórsson and the classical divide and conquer algorithms.

V. CONCLUSION AND FUTURE DIRECTIONS

The divide and conquer paradigm is a powerful technique for tackling difficult combinatorial optimization problems. In the case of MIS, graph partitioning can be used to improve the tractability of solving this problem on large graphs. Rather than directly searching for the MIS of the full graph, we can first partition the graph into multiple subgraphs where these subproblems can be solved independently and then recombine their results to produce a solution over the full graph. In this classical picture, it is advantageous to minimize the number of edges that cross between the partitions because each of the subproblems are independent of each other, and no information flows between the subgraphs.

There are several improvements that can be made to QDC algorithm in the future. In this work we have used the Kernighan-Lin algorithm [11] to partition the graph but in the future we can use more advanced graph partitioning algorithms that find partitions of the graphs with fewer edges crossing between the partitions. We have also only considered the case of two subgraphs but we can apply the QDC algorithm to the case with more than two subgraphs. The quantum-classical variational algorithm presented in this work utilized the dynamic quantum variational ansatz (DQVA) to perform the constrained optimization on each of the subgraphs produced by the partition. Recently, a quantum local search algorithm has been proposed, also targeting the MIS problem, which tackles large graphs using a fixed allocation of quantum resources [13]. The goal of both of these works is scaling up the ability of hybrid variational algorithms to target larger problem instances. It will be interesting to investigate ways of combining the quantum local search with the divide and conquer approach presented here.

Now that we have demonstrated the ability of this algorithm to find independent sets on large graphs via simulation, we would like to run it on quantum hardware. The small qubit counts and noisy execution of current NISQ hardware is enough to make any implementation a challenging endeavor. However, there are a number of ways we can hope to overcome the difficulties of implementing this algorithm on NISQ computers. Quantum computing platforms (mainly neutral-atom architectures) will soon become available that natively support the multi-control gates that compose the partial mixers used by DQVA [35]. The effects of noise in current NISQ systems will be problematic for the constrained evolution between feasible states, however, by modifying the objective function of the constrained optimization some of these concerns may be mitigated [12]. Finally, the small qubit counts of current devices can be alleviated with the help of quantum circuit cutting techniques, and we would like to see what is the largest graph we can tackle quantumly.

ACKNOWLEDGMENTS

T.T. is supported in part by EPiQC, an NSF Expedition in Computing, under grant CCF-1730082. Z.S. and M.S. are supported by the National Science Foundation under Award No. 2037984, and by the US Department of Energy (DOE) Office of Science Advanced Scientific Computing Research (ASCR) Accelerated Research in Quantum Computing (ARQC). P.G. acknowledges funding by the US Department of Energy Office, Advanced Manufacturing Office (CRADA No. 2020-20099.); and by the National Science Foundation under Grant No. 2110860.

[1] A. S. Cacciapuoti, M. Caleffi, F. Tafuri, F. S. Cataliotti, S. Gherardini, and G. Bianchi, Quantum internet: Networking challenges in distributed quantum computing, IEEE Network **34**, 137 (2020).
[2] L. Gyongyosi and S. Imre, Scalable distributed gate-model quantum computers, Scientific Reports **11**, 5172 (2021).
[3] S. Pirandola and S. Braunstein, Unite to build a quantum internet, Nature **532**, 169 (2016).
[4] S. Wehner, D. Elkouss, and R. Hanson, Quantum internet: A vision for the road ahead, Science **362** (2018).
[5] R. Van Meter, *Quantum Networking* (Wiley, 2014).
[6] Argonne and UChicago scientists take important step in developing national quantum internet, https://bit.ly/3wKJS8Z (2020).
[7] J. Chung, G. Kanter, N. Lauk, R. Valivarthi, W. Wu, R. R. Ceballos, C. Peña, N. Sinclair, J. Thomas, S. Xie, R. Kettimuthu, P. Kumar, P. Spentzouris, and M. Spiropulu, Illinois Express Quantum Network (IEQNET): Metropolitan-scale experimental quantum networking over deployed optical fiber, arXiv preprint arXiv:2104.04629 (2021).
[8] D. Du, P. Stankus, O.-P. Saira, M. Flament, S. Sagona-Stophel, M. Namazi, D. Katramatos, and E. Figueroa, An elementary 158 km long quantum network connecting room temperature quantum memories, arXiv preprint arXiv:2101.12742 (2021).
[9] J. F. Dynes, A. Wonfor, W. W. S. Tam, A. W. Sharpe, R. Takahashi, M. Lucamarini, A. Plews, Z. L. Yuan, A. R. Dixon, J. Cho, Y. Tanizawa, J. P. Elbers, H. Greißer, I. H. White, R. V. Penty, and A. J. Shields,

Cambridge quantum network, npj Quantum Information **5**, 101 (2019).
[10] B. Hendrickson and T. G. Kolda, Graph partitioning models for parallel computing, Parallel computing **26**, 1519 (2000).
[11] B. W. Kernighan and S. Lin, An efficient heuristic procedure for partitioning graphs, The Bell system technical journal **49**, 291 (1970).
[12] Z. H. Saleem, T. Tomesh, B. Tariq, and M. Suchara, Approaches to constrained quantum approximate optimization, arXiv preprint arXiv:2010.06660 (2021).
[13] T. Tomesh, Z. H. Saleem, and M. Suchara, Quantum local search with quantum alternating operator ansatz, arXiv preprint arXiv:2107.04109 (2021).
[14] T. Peng, A. W. Harrow, M. Ozols, and X. Wu, Simulating large quantum circuits on a small quantum computer, Physical Review Letters **125**, 150504 (2020).
[15] M. A. Perlin, Z. H. Saleem, M. Suchara, and J. C. Osborn, Quantum circuit cutting with maximum-likelihood tomography, npj Quantum Information **7**, 1 (2021).
[16] W. Tang, T. Tomesh, M. Suchara, J. Larson, and M. Martonosi, Cutqc: using small quantum computers for large quantum circuit evaluations, in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (2021) pp. 473–486.
[17] T. Ayral, F.-M. Le Régent, Z. Saleem, Y. Alexeev, and M. Suchara, Quantum divide and compute: exploring the effect of different noise sources, SN Computer Science **2**, 1 (2021).
[18] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum **2**, 79 (2018).
[19] R. Boppana and M. M. Halldórsson, Approximating maximum independent sets by excluding subgraphs, BIT Numerical Mathematics **32**, 180 (1992).
[20] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, arXiv preprint arXiv:1411.4028 (2014).
[21] S. Hadfield, Quantum algorithms for scientific computing and approximate optimization, arXiv preprint arXiv:1805.03265 (2018).
[22] S. Hadfield, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, From the quantum approximate optimization algorithm to a quantum alternating operator ansatz, Algorithms **12**, 34 (2019).
[23] Z. H. Saleem, Max-independent set and the quantum alternating operator ansatz, International Journal of Quantum Information **18**, 2050011 (2020).
[24] D. J. Egger, J. Mareček, and S. Woerner, Warm-starting quantum optimization, Quantum **5**, 479 (2021).
[25] R. S. Smith, M. J. Curtis, and W. J. Zeng, A practical quantum instruction set architecture, arXiv preprint arXiv:1608.03355 (2016).
[26] A. W. Cross, A. Javadi-Abhari, T. Alexander, N. de Beaudrap, L. S. Bishop, S. Heidel, C. A. Ryan, J. Smolin, J. M. Gambetta, and B. R. Johnson, Openqasm 3: A broader and deeper quantum assembly language, arXiv preprint arXiv:2104.14722 (2021).
[27] V. V. Shende and I. L. Markov, On the cnot-cost of toffoli gates, Quantum Info. Comput. **9**, 461–486 (2009).
[28] Y. He, M.-X. Luo, E. Zhang, H.-K. Wang, and X.-F. Wang, Decompositions of n-qubit toffoli gates with linear circuit complexity, International Journal of Theoretical Physics **56**, 2350 (2017).
[29] J. Díaz, M. J. Serna, and N. C. Wormald, Bounds on the bisection width for random d-regular graphs, Theoretical Computer Science **382**, 120 (2007).
[30] T. N. Bui, *On Bisecting Random Graphs.*, Tech. Rep. (Massachusetts Inst. of Tech., Cambridge Lab For Computer Science, 1983).
[31] A. Kostochka and L. Mel'nikov, On bounds of the bisection width of cubic graphs, in *Annals of Discrete Mathematics*, Vol. 51 (Elsevier, 1992) pp. 151–154.
[32] J. Dıaz, N. Do, M. J. Serna, and N. C. Wormald, Bounds on the max and min bisection of random cubic and random 4-regular graphs, Theoretical computer science **307**, 531 (2003).
[33] D. Johnson and L. Stockmeyer, Some simplified npcomplete graph problems, Theoretical Computer Science **1**, 237 (1976).
[34] Quantum-Software-Tools, dqva-and-circuit-cutting, https://github.com/Quantum-Software-Tools/dqva-and-circuit-cutting (2021).
[35] M. Saffman, Quantum computing with atomic qubits and rydberg interactions: progress and challenges, Journal of Physics B: Atomic, Molecular and Optical Physics **49**, 202001 (2016).