Scaling Up the Quantum Divide and Conquer Algorithm for Combinatorial Optimization

Cameron Ibrahim
Computer & Information Sciences
University of Delaware
Newark, DE
cibrahim@udel.edu

Teague Tomesh
Infleqtion
Chicago, IL
teague.tomesh@infleqtion.com

Zain Saleem
Mathematics and Computer Science Division
Argonne National Laboratory
Lemont, IL
zsaleem@anl.gov

Ilya Safro
Computer & Information Sciences
University of Delaware
Newark, DE
isafro@udel.edu

Abstract—Quantum optimization as a field has largely been restricted by the constraints of current quantum computing hardware, as limitations on size, performance, and fidelity mean most non-trivial problem instances won't fit on quantum devices. Even proposed solutions such as distributed quantum computing systems may struggle to achieve scale due to the high cost of inter-device communication. To address these concerns, we propose Deferred Constraint Quantum Divide and Conquer Algorithm (DC-QDCA), a method for constructing quantum circuits which greatly reduces inter-device communication costs for some quantum graph optimization algorithms. This is achieved by identifying a set of vertices whose removal partitions the input graph, known as a separator; by manipulating the placement of constraints associated with the vertices in the separator, we can greatly simplify the topology of the optimization circuit, reducing the number of required inter-device operations. Furthermore, we introduce an iterative algorithm which builds on these techniques to find solutions for problems with potentially thousands of variables. Our experimental results using quantum simulators have shown that we can construct tractable circuits nearly three times the size of previous QDCA methods while retaining a similar or greater level of quality.

Index Terms—Quantum Computing, Hybrid Algorithm, Graph Optimization

I. INTRODUCTION

Quantum advantage describes the potential speedup achievable using quantum computers on tasks traditionally performed on classical hardware, such as cryptography [1], machine learning [2], and optimization [3]. However, many relevant problems in these areas are fairly tractable for small inputs using classical methods. In order to demonstrate an advantage, we must push the boundary on the size of inputs that we can solve using quantum computers, which are currently constrained in both size and error rate [4].

Distributed quantum computing systems may provide the solution to the issue of scalability [5]. Since the error rate of a quantum device tends to inflate as its size increases, advocates of distributed quantum computing systems propose to utilize a variety of smaller quantum devices connected by means of quantum teleportation protocols. In the short term, implementation of these systems would be constrained by the cost of inter-device operations, which can add significant overhead and decrease fidelity depending on what method is used [6]–[8]. In order to successfully study the scalability of these distributed quantum systems, we must then find ways to work around this overhead.

Classical distributed systems similarly incur a significant overhead on inter-device operations, which may lead to a bottleneck in system performance [9]. Systems engineers generally try to limit this bottleneck by mapping processes to individual devices to minimize the necessity of inter-device operations (e.g., this can be done by modeling communication complexity using graph partitioning [10]). However, this is only necessary insofar as the inter-device operations present a bottleneck; once below a certain threshold, it becomes useful to instead try to optimize for device utilization, ensuring that the task at hand is making efficient use of the resources in the distributed system.

These classical techniques inspired methods such as the Quantum Divide and Conquer Algorithm (QDCA), which attempts to decompose a single combinatorial optimization problem into subproblems which can be solved on individual quantum devices [11]. This decomposition models the input as a graph, and attempts to minimize the amount of information which must be passed between devices by applying graph vertex partition algorithms to the underlying graph topology. In the QDCA approach, the problem is solved by applying a series of quantum gates known as mixers to each variable of the problem. If the number of inter-device operations is still prohibitive, the problem is sparsified by deactivating a subset of these mixers in order to further reduce the number of inter-device operations below a tractable threshold.

The benefits of QDCA have previously been shown while working with the Maximum Independent Set problem. This is a classical NP-Hard graph optimization problem with a wide variety of applications. This is a well studied problem in quantum optimization [12], and will be the primary area of focus for this paper.

Our contribution In order to address the limitations of QDCA with regards to scalability and device utilization, we introduce the Deferred Constraint Quantum Divide and Conquer Algorithm (DC-QDCA). Our approach utilizes edge partitioning in order to compute a k-way separator for our input graph. By manipulating the position of constraints associated with this separator, we can construct a circuit whose simplified topology allows it to decomposed into $k + 1$ subcircuits. The first k of these subcircuits are completely disconnected from one another and can be run entirely in parallel with no required inter-device operations. The final circuit corresponds to our separator; for each vertex adjacent to the separator, we must communicate its result to the separator circuit. We can adjust the required number of inter-device operations solely by deactivating mixers in the separator circuit, leaving each of the other k subcircuits completely active.

Utilizing this approach, we are able to significantly increase the number of vertices participating in the optimization of a single circuit for a given budget of inter-device operations, increasing the quality of the result. Moreover, for very large graphs where it is infeasible to produce a connected circuit with this approach, we introduce a method which iterates over individual vertices in the separator and attempts to solve the subgraph of partitions connected to that node. Using this approach, we can acquire quality results for graphs with several thousand vertices. Our approach directly benefits the distributed quantum computing algorithms by scaling them up.

II. BACKGROUND & RELATED WORK

A. The Maximum Independent Set Problem

An undirected graph $G = (V, E)$ with no loops and multi-edges is a set V of vertices together with a set $E \subseteq \binom{V}{2}$ of edges. An edge e and a vertex v are incident with one another if $v \in e$. Two vertices u, v are adjacent if there exists an edge e such that $u, v \in e$. The neighborhood $N(v)$ of a vertex v is the set of vertices adjacent to v.

An independent set is a subset $S \subseteq V$ where no two vertices $u, v \in S$ are adjacent. An independent set S is maximal if for all $v \in V$, $S \cup \{v\}$ is not an independent set. The maximum independent set for a graph G is the maximal independent set with largest cardinality.

The Maximum Independent Set Problem admits a formulation as the following binary linear program:

$$\max \quad \sum_{v \in V} x_v,$$

$$\text{such that} \quad x_u + x_v = 1 \qquad \forall uv \in E,$$
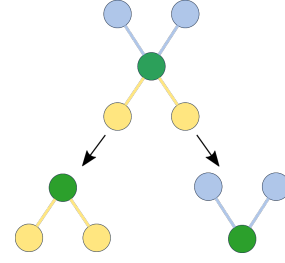$$x_v \in \{0, 1\} \qquad \forall v \in V.$$

Fig. 1: An edge partition assigns a label to each edge of a graph G. The set of vertices with edges belonging to multiple partitions (green) form a separator for the rest of the graph.

B. Vertex & Edge Partitioning

Graph partitioning is generally synonymous with graph vertex partitioning, where the vertex set V of the graph is split into k disjoint sets $V_1 \sqcup \cdots \sqcup V_k$. For $k = 2$, this is commonly known as Graph Bisection. A vertex partitioning solver will generally try to minimize the number of edges crossing between individual partitions. In balanced vertex partitioning, a constraint is added guaranteeing that no one partition takes too great a portion of the graph. The full formulation of balanced vertex partitioning is given by

$$\min_{V = V_1 \sqcup \cdots \sqcup V_k} \quad |\{uv \in E \mid \exists i \neq j : v \in V_i \wedge u \in V_j\}|$$

$$\text{st} \quad \forall i, |V_i| \leq (1 + \epsilon) \left\lceil \frac{|V|}{k} \right\rceil.$$

By comparison, graph edge partitioning attempts to split the edge set E into disjoint sets $E_1 \sqcup \cdots \sqcup E_k$. Edge partitioning has seen success in process assignment in classical distributed systems for systems which a few processes with a large number of required interactions with other processes [13]. Dual to the notion of an edge partition is the idea of a vertex separator, the set of vertices S which are incident to at least two edges assigned to different partitions.

A vertex separator induces a vertex partition $V = V_1 \sqcup \cdots \sqcup V_k \sqcup S$, where every edge in the graph either has one end in the separator or both ends in a single partition. An edge partitioning solver will generally try to find a solution which minimizes the sum of the number of edge partitions each vertex in the set is incident to. An analogous balance constraint can also be defined, giving the final formulation

$$\min_{E = E_1 \sqcup \cdots \sqcup E_k} \quad \sum_{v \in V} |\{i \mid \exists u \in N(v) : uv \in E_i\}| - 1$$

$$\text{such that} \quad \forall i, |E_i| \leq (1 + \epsilon) \left\lceil \frac{|E|}{k} \right\rceil.$$

C. Circuit Cutting and Knitting

A distributed quantum computing (DQC) system consists of a collection of smaller quantum computing devices which are networked together via inter-device operations [14]. In this way, they can have a large number of effective qubits, without the negative effects of increasing the number of qubits within a single device, such as increased error rates [15].

In order to map a circuit designed for a single device onto a DQC, we utilize a technique known as circuit cutting. This method decomposes a single circuit into a number of subcircuits which will be mapped to individual devices in the DQC; the results of one subcircuit will be correlated with the initial state of another, communicated using inter-device operations [16]–[18]. These input-output pairings can be represented as cuts on the wires or gates of the quantum circuit. Dual to the notion of circuit cutting is circuit knitting, where individual subcircuits are recombined to acquire an overall solution [17].

There is a considerable cost associated with mapping single device circuits to a DQC. In particular, without quantum methods of communicating between devices such as teleportation, the cost of classical methods for reconstructing the result of a single circuit evaluated on a DQC is exponential in the number of cuts required to decompose the circuit [17]. This places considerable constraints on what circuits can currently be evaluated using these methods.

D. QAOA and Quantum Divide and Conquer

Maximum Independent Set can similarly be formulated in terms of quantum operations. This problem has previously been studied in the context of hybrid quantum-classical optimization, where classical optimization methods such as gradient descent are used to optimize the parameters of a quantum circuit which outputs a valid solution to the given problem [19], [20]. These types of parameterized circuits are generally called *ansatz*.

In this setting, the Maximum Independent Set objective is known as the Hamming weight of the solution bitstring $x \in \{0,1\}^{|V|}$. The Hamming Weight function can be represented in terms of quantum operations as

$$C = \sum_{v \in V} \frac{1 - Z_v}{2},$$

where Z_v is the Pauli-Z operator applied to the qubit corresponding to vertex v [11]. For an ansatz ψ with parameters θ, the loss function which is minimized in order to find an optimal independent set is given by the expectation

$$\mathcal{L}(\theta) = -\mathbb{E}\Big[\langle \psi(\theta)|C|\psi(\theta) \rangle \Big].$$

The Quantum Approximate Optimization Algorithm (QAOA) is a class of algorithms for combinatorial optimization which follow this ansatz construction approach [19]. The ansatz in QAOA is generally constructed using an initial state $|s\rangle$ and p alternating layers of parametrized phase unitaries $U_C(\theta)$ and mixing unitaries $U_M(\theta)$

$$|\psi(\theta)\rangle = U_C^p(\theta)U_M^p(\theta) \cdots U_C^1(\theta)U_M^1(\theta)|s\rangle,$$

where

$$U_C^k(\theta) = \exp\{-i\theta_{C,k}C\}, \quad U_M^k(\theta) = \exp\{-i\theta_{M,k}M\}$$

for some problem dependent operator M. Accelerating QAOA and making it more practical is of great importance [21], [22].

In this paper, we build upon a specific variant of QAOA known as QDCA. In QDCA, the mixing unitaries U_M are defined such that they enforce the constraints of the relevant combinatorial optimization problem [20]. The mixing unitary $U_M(\theta)$ is defined such that $U_M^k(\theta) = \prod_{v \in V} P_v(\theta_{M,k,v})$, where each $P_v(\alpha) = \exp\{-i\alpha M_v\}$ is a partial mixer on the wire corresponding to the vertex v. For Maximum Independent set, M_v is defined as

$$M_v := X_v \prod_{u \in N(v)} \frac{1 + Z_u}{2},$$

where X_v is the Pauli-X operator on the qubit corresponding to the vertex v. In words, M_v will flip x_v if and only if none of its neighbors are in the $|1\rangle$ state.

Furthermore, QDCA incorporates both vertex partitioning and circuit knitting to scale to larger graphs. The QDCA algorithm utilizes graph bisection to acquire a partition of the vertex set $V = V_1 \sqcup V_2$. Individual ansatz are constructed for each of the induced subgraphs corresponding to these partitions. Vertices which are incident to an edge which crosses between these two vertex sets are referred to as cut nodes, and are given by the set $Q \subseteq V$. A certain number of "hot" nodes are then chosen from Q; these hot nodes are used to knit the constructed ansatz together to allow gradient information flow between the two circuits. The remaining "cold" nodes have their mixers deactivated, removing them from consideration for the solution for this circuit so as to remove the possibility of an invalid solution. In this way, decreasing the number of "hot" nodes decreases the number of required inter-device operations at the cost of decreased solution quality.

III. DEFERRED CONSTRAINT DIVIDE AND CONQUER

In this section, we attempt to alleviate the issue of mixer deactivations in order to improve the quality-scalability trade-off in QDCA. To do so, we introduce the Deferred Constraint Quantum Divide and Conquer Algorithm (DC-QDCA). We do so in order to simplify the structure of the circuit, which in turn allows for a greater number of active mixers.

The power of the deferred constraint approach lies in the simplicity with which we can compute its overhead. The number of wire cuts required to split a deferred constraint circuit is exactly equal to the size of the neighborhood of the separator, $\bigcup_{v \in S} N(v) \setminus S$. By trying to minimize this quantity, we minimize the number of cuts required to separate the circuit. Once a separator is chosen, we have the option to sparsify it, removing vertices until the number of required cuts is within our budget. This process is equivalent to deactivating the mixers corresponding to those vertices in the separator.

A. Construction of the QC-QDCA Circuit

A key insight enabling DC-QDCA is that if a vertex v has a neighbor u, where u was not in the independent set in the initial state and u has not yet had a partial mixer applied to it, then a partial mixer on v does not have to check the current state of u. Given an ordering $v_1, \ldots, v_{|V|} \in V$ of the vertices of the input graph which matches the ordering of partial mixers
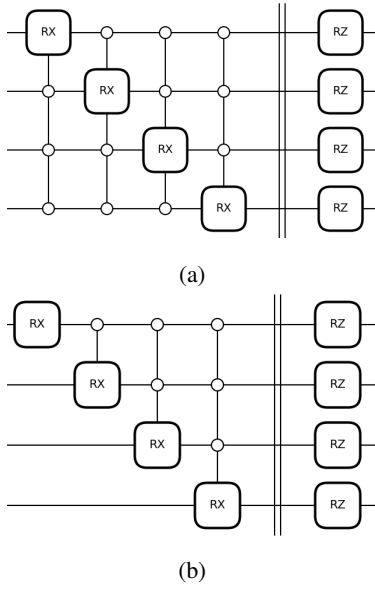
(a)

(b)

Fig. 2: Example of circuit simplification. Circuit (a) is simplified into (b) by removing unnecessary connections.

(a) 5 node path graph
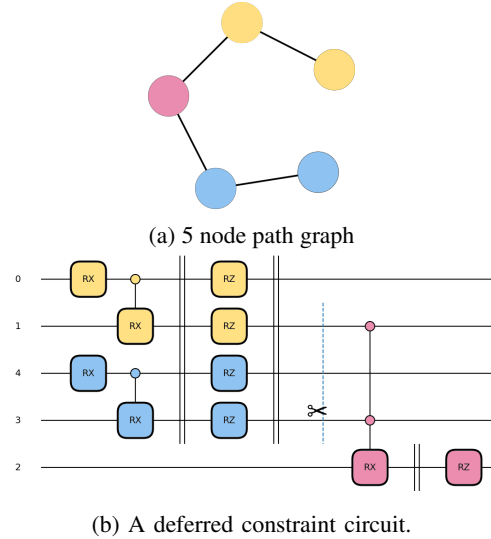

(b) A deferred constraint circuit.

Fig. 3: Example using a vertex separator (Red) with two partitions (Blue, Yellow). The DC-DQCA circuit is separated into 3 subcircuits using cuts on wires 1 and 3, corresponding to nodes adjacent to the separator.

in the circuit, this can be expressed in terms of gate operations as

$$M_v := X_v \prod_{v_j \in \mathcal{N}} \frac{1 + Z_{v_j}}{2}, \quad \mathcal{N} := \{v_j \in N(v_i) \mid j < i\}.$$

This is reflected in Fig. 2, where for an initial state corresponding to the empty set, the partial mixer for each vertex only needs to check the wires corresponding to neighbors coming before it in the order.

Without circuit cutting, this benefit may seem negligible. However, this reduction in the number of connections between wires can significantly reduce the number of required inter-device operations. Consider a vertex separator $V = V_1 \sqcup \cdots \sqcup V_k \sqcup S$ of the input graph. Each edge of the graph is either contained completely in one partition, has at least one end in S. If the mixers corresponding to the vertices in S are placed at the end of the circuit, then no gate earlier in the circuit will need to connect two wires from different partitions. As a result, if we identify partition subcircuits ψ_i corresponding to each partition V_i, as well as a separator subcircuit ψ_S corresponding to S, no inter-device operations will need to be used between any two ψ_i, ψ_j. This is construction is demonstrated in Fig. 3, where deferring the separator produces a circuit with three subcircuits. Notably, this construction also means that each of the partition subcircuits can be executed completely in parallel, even on a classical simulator.

The number of cuts required to split the circuit in this way is then going to be $\bigcup_{v \in S} N(v) \setminus S$, which is the number of vertices not in the separator whose value must be communicated to the separator circuit. Each individual subcircuit ψ will be constructed as in QDCA, but using the simplified mixers defined in this section. The overall DC-QDCA circuit has the

form

$$\psi(\theta) := \psi_S(\theta) \prod_{i=1}^{k} \psi_i(\theta).$$

In order to further simplify the circuit, we order the partial mixers of the graph in order of neighborhood size, $|N(v)|$. This is a criterion used in some greedy algorithms for Maximum Independent set, and leads to an an order in which there few vertices with neighbors earlier in the order [23]. When combined with the simplified mixers defined in this section, this can in practice reduce the number of wire connections within each subcircuit.

B. Separator Sparsification

While the DC-QDCA algorithm produces a circuit requiring only a small number of cuts, this can still quickly become intractable due to the exponential cost of overhead associated with inter-device-communication. As such, we require some way to simplify the problem so that it may still be tractably evaluated. In this section, we present two approaches to achieving tractability, one through sparsification and one through iteration.

As the number of cuts required to split the QC-QDCA circuit is given $\bigcup_{v \in S} N(v) \setminus S$, one approach to decreasing this cost is to remove vertices, deactivating their associated mixers, in order to decrease the size of the neighborhood adjacent to the separator. In this paper, we choose a subset of the separator by fixing an ordering of the vertices in the separator and iteratively attempting to add them to a candidate set. By maintaining the current neighborhood of the candidate set, rather than evaluating the neighborhoods of each vertex independently, we are able to capture vertices

with overlapping neighborhoods that are not going to increase the number of required cuts. We once again utilize an ordering based on neighborhood size in order to prioritize vertices with few connections to the rest of the graph. This approach is appropriate when only a few vertices need to be removed to produce a tractable circuit, one which can solve the input problem using a single circuit. This approach is summarized as Alg. 1.

Algorithm 1: Deferred Constraint Quantum Divide and Conquer Algorithm (Single Circuit)

Input : A graph $G = (V, E)$, a number of partitions k, an imbalance ϵ, an optimizer opt, a cut budget b, and a number of shots s

Output: A bitstring encoding an independent set of G

$parts, S \leftarrow \text{edgepartition}(G, k, \epsilon)$
$S' \leftarrow \{\}$
$\mathcal{N} \leftarrow \{\}$
sort $s_1, \ldots, s_{|S|} \in S$ by $|N(s_i)|$
for $s_i \in S$ **do**
    $\mathcal{N}' \leftarrow \mathcal{N} \cup (N(v_i) \setminus S)$
    if $|\mathcal{N}'| \leq b$ **then**
        $N \leftarrow N'$
        $S' \leftarrow S' \cup \{v_i\}$
    end
end
$\psi, \theta \leftarrow \text{circuitgen}(G, parts, S')$
$\theta^* \leftarrow opt(\mathcal{L}, \psi, \theta)$
$X \leftarrow \text{sample}(\psi, \theta^*, s)$
return $\text{argmax}_{x \in X} \sum_{v \in V} x_v$

If the size of the separator is significant, this sort of sparsification likely isn't tractable. In this case, it likely isn't feasible to find a solution for the entire graph with only a single circuit. As an alternative to the sparsification approach, one could try an iterative strategy, where small subproblems are solved and recombined for find an overall solution. In this case, the local subproblems in question are defined by the subset of partitions connected by a single vertex s in the separator, $\{V_i \mid i \leq k, |N(s) \cap V_i| > 0\}$.

Together with s, these form an induced subgraph which can be solved to find a local part of the solution for the overall graph. The rest of the current solution remains fixed. Any mixers that may produce a conflict with the fixed solution is deactivated.

By iterating over all vertices in the separator, we generate a set of subproblems which covers the entire graph. Over the course of multiple sweeps, we can find and refine a global solution for the input graph. This approach is highly scalable, and can be used to find solutions for graphs with several thousand vertices. This algorithm is given as Alg. 2.

IV. SIMULATION RESULTS

In this section, we evaluate the performance of the DC-QDCA using the Pennylane library, evaluated on University

---

Algorithm 2: Deferred Constraint Quantum Divide and Conquer Algorithm (Iterative)

Input : A graph $G = (V, E)$, a number of partitions k, an imbalance ϵ, an optimizer opt, a cut budget b, a number of shots s, and a number of iteratations $iter$

Output: A bitstring encoding an independent set of G

$x \leftarrow [0] * |V|$
$parts, S \leftarrow \text{edgepartition}(G, k, \epsilon)$
sort $s_1, \ldots, s_{|S|} \in S$ by $|N(s_i)|$
for $1 \leq i \leq iter$ **do**
    for $s_i \in S$ **do**
        if $|\mathcal{N}| \leq b$ **then**
            $local \leftarrow \{p \in parts \mid \exists u \in p : s_i \in N(u)\}$
            $L \leftarrow \{s_i\} \cup \{u \mid \forall p \in local, \forall u \in p\}$
            $L \leftarrow \{v \in L \mid \exists u \in N(v) : x[u] = 1\}$
            $local \leftarrow \{p \setminus L \mid p \in local\}$
            $\psi, \theta \leftarrow \text{circuitgen}(G, local, \{s_i\} \setminus L)$
            $\theta^* \leftarrow opt(\mathcal{L}, \psi, \theta)$
            $X \leftarrow \text{sample}(\psi, \theta^*, s)$
            $x_{local} \leftarrow \text{argmax}_{x \in X} \sum_{v \in V} x_v$
        end
    end
end
return x

---

of Delaware Caviness cluster. Our testing uses a collection of real world graphs from the Suite Sparse Matrix Collection [24], as well as a Random 3-Regular graph and a Connected Watts-Strogatz graph. The code for this project, the graphs we tested on, and a summary of the collected data will be available at [25]. Our primary source of comparison will be the original QDCA implementation, found at [26]. Where optimal results were required in order to give an approximation ratio, we used the classical KaMIS library [27], while for the calculation of vertex separators we used an edge partition provided by the KaHIP library [28].

A. Single Circuit Inputs

For these tests, we compare directly with QDCA on a collection of small inputs for which we can construct a circuit covering the majority of the graph. For DC-DQVA, each of these graphs are partitioned into 2-4 parts. We are primarily concerned with number of inactive mixers and how that relates to the quality of the result. A large number of inactive mixers means a large portion of the graph is not being considered in the optimization, and further applications of the circuit may be required to find a solution. Both approaches in this test were given a cut budget of 6, allowing us to compare the difference in coverage given the same amount of overhead. Both random graphs where generated with 16 vertices, while the real world graphs are up to size 25.

In Fig. 4, we see that DC-QDCA tends to produce circuits with a fraction of the inactive mixers in QDCA. In some cases, DC-QDCA is able to capture the entire graph with no inactive
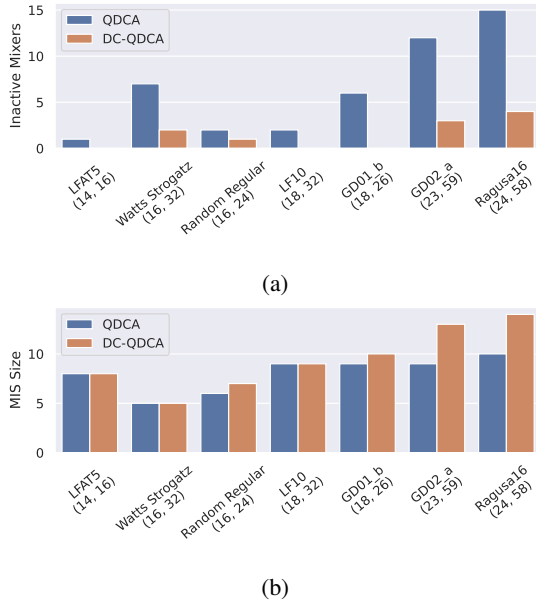
(a)

(b)

Fig. 4: Comparison of DC-QDCA with QDCA with respect to mixer activations on a number of small graphs with size $(|V|, |E|)$. Fig 4a shows the significant decrease in inactive mixers achieved by DC-QDCA while Fig 4b shows the accompanying increase in quality.

mixers whatsoever. Notably, there is not a perfect correlation between inactive mixers and final quality.

However, we can see that where the difference in inactive mixers is greatest, so too is the difference in quality. In fact, on each of these graphs, DC-QDCA was able to achieve an optimal result, whereas QDCA fell short on several, especially those where it had a great number of inactive mixers, such as Ragusa16 or GD02_a.

B. Iterative Approach

To demonstrate the application of the iterative version of our algorithm, we look at a collection of 4 large real world graphs from the Suite Spares Matrix Collection. In particular, we look at both citation and reference networks, as a well as a network representation of the US power grid. The largest of these graphs, USpowergrid, requires a full 250 partitions in order to get the size of the largest partition small enough to be reasonably tractable. A full list of parameters used for each individual graph is available at [25].

Our initial results in Fig. 5, are highly promising, achieving an approximation ratio as high as 97%. Moreover, an increase in size does not necessarily correlate with a decrease in the quality of this iterative method, as some of our best results were achieved for the largest graph USpowergrid.

To more closely examine the behavior of our iterative method, we demonstrate the quality of the solution overtime in Fig. 6. Here, we see the increase in quality over several hundred circuit applications over two full sweeps of the iterative method. Here we see a significant amount of improvement
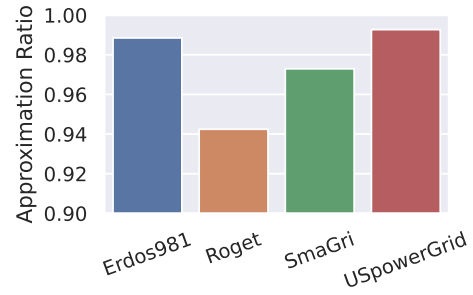

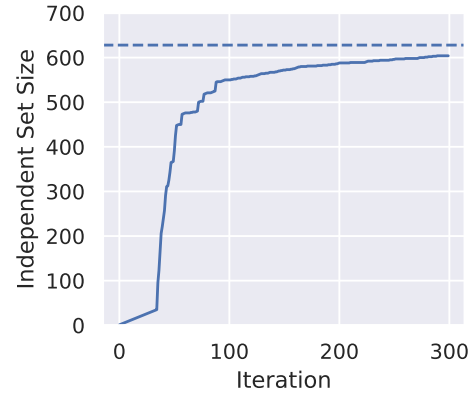Fig. 5: Approximation ratios for a collection of real world graphs using iterative DC-QDCA.

Fig. 6: Shows the progress of iterative DC-QDCA on the SmaGri network over two sweeps of the iterative method.

initially, which slowly tapers of as we approach the optimum, noted by the dotted line. This behavior is highly promising, demonstrating that given access to fast quantum hardware, we could feasibly find high quality solutions for graphs with several thousand vertices, without actually requiring a device or system of devices which support several thousand qubits.

V. CONCLUSION AND FUTURE WORK

Scalability in quantum computing will remain one of the preeminent obstacles to achieving wide scale use in practice for the foreseeable future. In particular, it is a considerable barrier to demonstrating practical applications of quantum optimization. Previous attempts to address this problem include QDCA, which utilized a divide and conquer approach to the Maximimum Independent Set problem. We demonstrate an approach to achieve scalability in a distributed setting, utilizing edge partitioning and separator sparsification to lower communication costs and improve results over QDCA. Moreover, we introduced an iterative variant of our algorithm which allows us to find high quality solutions on real world graphs with several thousand vertices. Ongoing efforts are being made to show the broad applicability of these methods to other optimization problems, as well as expanding on existing results through improved heuristics.

REFERENCES

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*, Ieee, 1994, pp. 124–134.

[2] K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmann, D. Scheiermann, and R. Wolf, "Training deep quantum neural networks," *Nature communications*, vol. 11, no. 1, p. 808, 2020.

[3] D. Herman, C. Googin, X. Liu, Y. Sun, A. Galda, I. Safro, M. Pistoia, and Y. Alexeev, "Quantum computing for finance," *Nature Reviews Physics*, vol. 5, no. 8, pp. 450–465, 2023.

[4] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[5] D. Cuomo, M. Caleffi, and A. S. Cacciapuoti, "Towards a distributed quantum computing ecosystem," *IET Quantum Communication*, vol. 1, no. 1, pp. 3–8, 2020.

[6] X.-M. Hu, Y. Guo, B.-H. Liu, C.-F. Li, and G.-C. Guo, "Progress in quantum teleportation," *Nature Reviews Physics*, vol. 5, no. 6, pp. 339–353, 2023.

[7] T. Ayral, F.-M. L. Régent, Z. Saleem, Y. Alexeev, and M. Suchara, "Quantum divide and compute: Exploring the effect of different noise sources," *SN Computer Science*, vol. 2, no. 3, p. 132, 2021.

[8] T. Ayral, F.-M. Le Régent, Z. Saleem, Y. Alexeev, and M. Suchara, "Quantum divide and compute: Hardware demonstrations and noisy simulations," in *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, IEEE, 2020, pp. 138–140.

[9] D. P. Woodruff and Q. Zhang, "When distributed computation is communication expensive," *Distributed Computing*, vol. 30, no. 5, pp. 309–323, 2017.

[10] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz, "Recent advances in graph partitioning," in *Algorithm Engineering: Selected Results and Surveys*, L. Kliemann and P. Sanders, Eds. Springer International Publishing, 2016.

[11] T. Tomesh, Z. H. Saleem, M. A. Perlin, P. Gokhale, M. Suchara, and M. Martonosi, "Divide and conquer for combinatorial optimization and distributed quantum computation," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01, 2023, pp. 1–12.

[12] T. Tomesh, Z. H. Saleem, and M. Suchara, "Quantum local search with the quantum alternating operator ansatz," *Quantum*, vol. 6, p. 781, 2022.

[13] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, "{Powergraph}: Distributed {graph-parallel} computation on natural graphs," in *10th USENIX symposium on operating systems design and implementation (OSDI 12)*, 2012, pp. 17–30.

[14] M. Caleffi, M. Amoretti, D. Ferrari, D. Cuomo, J. Illiano, A. Manzalini, and A. S. Cacciapuoti, "Dis-

tributed quantum computing: A survey," *arXiv preprint arXiv:2212.10609*, 2022.

[15] T. Tomesh, P. Gokhale, V. Omole, *et al.*, "Supermarq: A scalable quantum benchmark suite," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, IEEE, 2022, pp. 587–603.

[16] T. Peng, A. W. Harrow, M. Ozols, and X. Wu, "Simulating large quantum circuits on a small quantum computer," *Physical review letters*, vol. 125, no. 15, p. 150 504, 2020.

[17] C. Piveteau and D. Sutter, "Circuit knitting with classical communication," *IEEE Transactions on Information Theory*, 2023.

[18] K. Mitarai and K. Fujii, "Overhead for simulating a non-local channel with local channels by quasiprobability sampling," *Quantum*, vol. 5, p. 388, 2021.

[19] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," *Physical Review X*, vol. 10, no. 2, p. 021 067, 2020.

[20] Z. H. Saleem, T. Tomesh, B. Tariq, and M. Suchara, "Approaches to constrained quantum approximate optimization," *SN Computer Science*, vol. 4, no. 2, p. 183, 2023.

[21] H. Ushijima-Mwesigwa, R. Shaydulin, C. F. Negre, S. M. Mniszewski, Y. Alexeev, and I. Safro, "Multilevel combinatorial optimization across quantum architectures," *ACM Transactions on Quantum Computing*, vol. 2, no. 1, pp. 1–29, 2021.

[22] A. Galda, E. Gupta, J. Falla, X. Liu, D. Lykov, Y. Alexeev, and I. Safro, "Similarity-based parameter transferability in the quantum approximate optimization algorithm," *Frontiers in Quantum Science and Technology*, vol. 2, p. 1 200 975,

[23] A. Kako, T. Ono, T. Hirata, and M. M. Halldórsson, "Approximation algorithms for the weighted independent set problem in sparse graphs," *Discrete Applied Mathematics*, vol. 157, no. 4, pp. 617–626, 2009.

[24] T. A. Davis and Y. Hu, "The university of florida sparse matrix collection," *ACM Transactions on Mathematical Software (TOMS)*, vol. 38, no. 1, pp. 1–25, 2011.

[25] C. Ibrahim, *Mis qce2024*. [Online]. Available: https://github.com/cameton/MIS_QCE2024.

[26] T. Tomesh, *Dqva-and-circuit-cutting*. [Online]. Available: https://github.com/teaguetomesh/dqva-and-circuit-cutting.

[27] D. Hespe, C. Schulz, and D. Strash, "Scalable kernelization for maximum independent sets," *ACM Journal of Experimental Algorithmics*, vol. 24, no. 1, 1.16:1–1.16:22, 2019.

[28] S. Schlag, C. Schulz, D. Seemaier, and D. Strash, "Scalable Edge Partitioning," in *Proceedings of the 21th Workshop on Algorithm Engineering and Experimentation (ALENEX)*, SIAM, 2019, pp. 211–225.