Quantum Information Processing (2021) 20:59
https://doi.org/10.1007/s11128-021-03001-7

Lower bounds on circuit depth of the quantum approximate optimization algorithm

Rebekah Herrman[1] · James Ostrowski[1] · Travis S. Humble[2] · George Siopsis[3]

Received: 10 August 2020 / Accepted: 14 January 2021 / Published online: 9 February 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC part of Springer Nature 2021

Abstract

The quantum approximate optimization algorithm (QAOA) is a method of approximately solving combinatorial optimization problems. While QAOA is developed to solve a broad class of combinatorial optimization problems, it is not clear which classes of problems are best suited for it. One factor in demonstrating quantum advantage is the relationship between a problem instance and the circuit depth required to implement the QAOA method. As errors in noisy intermediate-scale quantum (NISQ) devices increase exponentially with circuit depth, identifying lower bounds on circuit depth can provide insights into when quantum advantage could be feasible. Here, we identify how the structure of problem instances can be used to identify lower bounds for circuit depth for each iteration of QAOA and examine the relationship between problem structure and the circuit depth for a variety of combinatorial optimization problems including MaxCut and MaxIndSet. Specifically, we show how to derive a graph, G, that describes a general combinatorial optimization problem and show that the depth of circuit is at least the chromatic index of G. By looking at the scaling of circuit depth, we argue that MaxCut, MaxIndSet, and some instances of vertex covering and Boolean satisfiability problems are suitable for QAOA approaches while knapsack and traveling salesperson problems are not.

✉ James Ostrowski
jostrows@utk.edu

Rebekah Herrman
rherrma2@utk.edu

[1] Department of Industrial and Systems Engineering, University of Tennessee at Knoxville, Knoxville, Tennessee 37996-2315, USA

[2] Quantum Computing Institute, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37830, USA

[3] Department of Physics and Astronomy, University of Tennessee at Knoxville, Knoxville, Tennessee 37996-1200, USA

Keywords Quantum approximate optimization algorithm · Circuit depth · Combinatorial optimization · Chromatic index

1 Introduction

In 2014, Farhi et al. [1] introduced the quantum approximate optimization algorithm (QAOA) to approximately solve combinatorial optimization problems. In classical combinatorial optimization, problems are defined by n bits and m clauses. To solve optimization problems using QAOA, the clauses are converted to Hamiltonians, and the state of the graph is initially $|s\rangle = \frac{1}{\sqrt{2^n}}\Sigma_z|z\rangle$, where $\{|z\rangle\}$ is the computational basis. For $p \in \mathbb{N}$, the p-level QAOA requires $2p$ angles, $\vec{\boldsymbol{\gamma}} = (\gamma_1, \ldots, \gamma_p)$ and $\vec{\boldsymbol{\beta}} = (\beta_1, \ldots, \beta_p)$ and alternates between the mixing Hamiltonian, B, and the problem Hamiltonian, C, to generate the state

$$\left|\psi(\vec{\boldsymbol{\gamma}}, \vec{\boldsymbol{\beta}})\right\rangle = U(B, \beta_p)U(C, \gamma_p)\ldots U(B, \beta_1)U(C, \gamma_1)|s\rangle$$

where $U(A, \phi) = e^{-iA\phi}$. B and C depend on the problem of interest and the angles that maximize them can be found using classical pre-processing [2–4].

Previously, QAOA has been used to solve bounded constraint problems [5] and has been studied on near-term devices [6]. Additionally, it has been used to look at lattice protein folding [7] and the Max-k vertex cover problem [8] and has inspired an approach for solving linear systems using quantum computing [9]. MaxCut and maximum independent set are examples of two problems that have been well studied with QAOA [10–13]. Both can be represented as quadratic unconstrained problems, otherwise known as a QUBO. It has also been shown to exhibit a form of computational advantage in the sense that the output of low depth circuits cannot be efficiently classically simulated [14], and general strategies have been studied for implementing it on hardware graphs [15].

In this paper, we investigate the potential of using quantum computing to solve combinatorial optimization problems of the form

$$\min\ c(x) \tag{1}$$

$$\text{s.t. } p_i(x) \leq b_i \qquad \forall i \in P \tag{2}$$

$$x \in \{0, 1\}^n \tag{3}$$

where both p_i, contained in the collection of polynomial constraints P, and c are polynomial functions in $\mathbb{R}^n[x_1, x_2, ..., x_n]$ and $b_i \in \mathbb{R}$.

We identify the relationship between combinatorial optimization problems and the corresponding depth of circuit for QAOA approaches to solving this problem. It has been shown that the cost function for QAOA decreases with the number of gates and level of noise in NISQ devices [16–18], so in this paper, we specifically focus on circuit depth, although an equally important component of the fidelity of a solution is the number of iterations needed. We only look at a single iteration because we consider all iterations have the same depth.

In Sect. 2, we define graph theory terms that will be used throughout the paper. Next, in Sect. 3, we discuss how to map arbitrary combinatorial optimization problems to polynomial unconstrained binary optimization problems (PUBOs) by dualizing constraints and apply the method to MaxCut, Maximum Independent Set, and a general combinatorial optimization problem. Additionally, we discuss how to use the PUBOs to derive a hypergraph that represents a specific optimization problem and show that one plus the chromatic index of the hypergraph is equal to the depth of QAOA circuit needed to run a combinatorial optimization problem. Using this result, we analyze the depth of circuit for the MaxCut, Maximum Independent Set, and general combinatorial optimization problems. The depth of circuit argument is made assuming that arbitrary n-qubit gates can be performed on the hardware; however, in Sect. 4, we show how the depth of circuit scales if the largest operation that can be made is on two qubits. We then consider vertex covering, knapsack, traveling salesperson, and Boolean satisfiability problems, determine the depth of circuit required to use QAOA to solve them, and discuss the feasibility of performing them on NISQ devices in Sect. 5. Finally, in Sect. 6, we discuss avenues for future work.

2 Background

In this section, we define graph theory terms that will be used in upcoming sections. An *edge coloring* of a simple graph $G = (V, E)$ is a labeling $f : E \longrightarrow [k]$, where each number represents a color. An edge coloring is *proper* if for all edges uv and xv, $f(uv) \neq f(xv)$. The smallest number of colors needed for a proper coloring of G is the edge chromatic number, sometimes referred to as the *chromatic index*, denoted $\chi'(G)$, and we say all edges with the same label belong to the same color class. A well-known result by Vizing states that $\chi'(G) \in \{\Delta, \Delta+1\}$, where Δ is the maximum degree of G [19].

A *hypergraph* $H = (V_H, E_H)$ is a generalization of a graph in which an edge may join more than two vertices. If there are n vertices in H, then $E \subset P \setminus \{\emptyset\}$. H is *linear* if two edges share at most one vertex, and it is *k-uniform* if all edges contain exactly k vertices. A *hypergraph clique* is a collection of edges, $H_c \subset E_H$, such that every element of H_c is pairwise intersecting. A *proper hypergraph edge coloring* is analogous to an edge coloring of a graph in that if a vertex is contained in multiple edges, they all receive distinct colors.

3 Mapping arbitrary combinatorial optimization problems to PUBO

When considering combinatorial optimization problems, we will use the method of dualizing constraints to solve them and analyze circuit depth. Other methods may give different results. Consider a constraint $p_i(x) \leq b_i$, where $x = \{x_1, ..., x_n\}$. We can *dualize* this constraint by penalizing any solution x' with $p_i(x') \geq b_i$ as follows. Let $\underline{p_i} = \min_{x \in \{0,1\}^n} p_i(x)$. The "most feasible" solution with respect to constraint i is going

to be $feas_i = b_i - \underline{p_i}$ away from the constraint. Let $k_i = \lfloor \ln feas_i \rfloor + 1$. We can omit constraint i from the set of constraints and add the term

$$\lambda_i \left(p_i(x) + \sum_{j \in [k_i]} 2^{j-1} \delta_{ij} - b_i \right)^2 \tag{4}$$

where λ_i is any large, positive parameter penalizing violation of constraint i and δ_{ij} are additional binary variables.

In multiplying out the above constraint, we get

$$\lambda_i \left(p_i(x)^2 + 2 \sum_{j = \in [k_i]} 2^{j-1} p_i(x) \delta_{ij} - 2 b_i p_i(x) + \left(\sum_{j \in [k_i]} 2^{j-1} \delta_{ij} \right)^2 - 2 \sum_{j \in [k_i]} 2^{j-1} b_i \delta_{ij} + b_i^2 \right). \tag{5}$$

The cost of this transformation is an increase in the potentially large number of new δ_{ij} variables.

Using the above process, we can write any combinatorial optimization problem of type (1) as

$$\min_{x \in \{0,1\}^n, \delta \in \{0,1\}^{k_i}} c(x) + \sum_{i \in P} \lambda_i \left(p_i(x)^2 + 2 \sum_{j \in [k_i]} 2^{j-1} p_i(x) \delta_{ij} - 2 b_i p_i(x) + \left(\sum_{j \in [k_i]} 2^{j-1} \delta_{ij} \right)^2 - 2 \sum_{j \in [k_i]} 2^{j-1} b_i \delta_{ij} + b_i^2 \right). \tag{6}$$

If all p_i constraints are linear and c is quadratic, the resulting unconstrained problem is a QUBO. Simplifying notation, we can think of a combinatorial optimization problem as the sum of monomials of the polynomial $p_i(x)$,

$$\min_{x \in \{0,1\}, \delta \in \{0,1\}^{k_i}} \sum_{m_i \in M_i} m_i(x, \delta),$$

where M_i is the set of monomials of $p_i(x)$

3.1 Examples

In this section, we give examples of problems whose edge operators act on at most two qubits and how to map them to PUBOs.

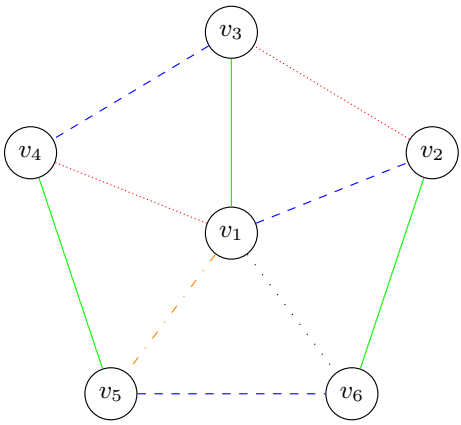

Fig. 1 The wheel graph on six vertices with the edges properly colored. There are five color classes: solid green, dashed blue, densely dotted red, loosely dotted black, and dotted dashed orange (Color figure online)

Example: MaxCut

In the combinatorial optimization problem MaxCut, the vertices of a graph, $G = (V, E)$, are partitioned into two sets such that the number of edges with an end point in each set is maximized. This problem can be formulated as

$$\min_{x\in\{0,1\}^n} \sum_{ij\in E(G)} x_j(x_i - 1) + x_i(x_j - 1) = \min_{x\in\{0,1\}^n} \sum_{ij\in E(G)} 2x_i x_j - x_i - x_j$$

Note that $P = \{\emptyset\}$, so there are no δ_{ij} terms when the problem is dualized. For example, consider the wheel graph on six vertices, W_6, as seen in Fig. 1.

In this example, we want to minimize

$$\begin{aligned} &2(x_1x_2 + x_1x_3 + x_1x_4 + x_1x_5 + x_1x_6 + x_2x_3 + x_2x_6 + x_3x_4 + x_4x_5 + x_5x_6) \\ &\quad - 5x_1 - 3(x_2 + x_3 + x_4 + x_5 + x_6) \end{aligned}$$

where $x_i \in \{0, 1\}$ for all $i \in [6]$.

Example: Maximum Independent Set

Let $G = (V, E)$ be a simple, undirected graph. In the maximum independent set problem, often denoted MaxIndSet, the goal is to find the largest set of independent vertices, or vertices that are not pairwise adjacent. This problem can be written as

$$\max \sum_{i\in V} x_i \tag{7}$$

$$\text{s.t. } x_i x_j = 0 \qquad \forall (i, j) \in E \tag{8}$$

$$x \in \{0, 1\}^n \tag{9}$$

$P \neq \{\emptyset\}$, as $|P| = |E|$, but for all p_i, $\underline{p_i} = 0$, so $\texttt{feas}_i = 0$ for all i, and the new formulation is

$$\max_{x\in\{0,1\}^n} \sum_i x_i + \sum_{(i,j,)\in E} \lambda x_i x_j.$$

For the graph given in Fig. 1, the resulting optimization function is:

$$\begin{aligned} & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \\ & \quad + \lambda(x_1x_2 + x_1x_3 + x_1x_4 + x_1x_5 + x_1x_6 + x_2x_3 \\ & \quad\quad + x_3x_4 + x_4x_5 + x_5x_6 + x_2x_6) \end{aligned}$$

Example: General Problem As a general example, consider

$$\max \sum_{i\in[3]} x_i \tag{10}$$

$$\text{s.t. } x_1x_2 + x_2x_3 + 2x_1x_3 \leq 3 \tag{11}$$

$$x_i \in \{0, 1\} \tag{12}$$

Now, $\underline{p_i} = 0$, so $\texttt{feas} = 3$. Dualizing the first constraint, we get

$$\begin{aligned} & \max(x, \lambda) \sum_{i\in[3]} x_i + \lambda(x_1x_2 + x_2x_3 + 2x_1x_3 - \delta_{11} - 2\delta_{12} - 3)^2 \\ & = \sum_{i\in[3]} x_i + \lambda(-5x_1x_2 + 10x_1x_2x_3 - 5x_2x_3 - 8x_1x_3 - 2\delta_{11}x_1x_2 \\ & \quad - 2\delta_{11}x_2x_3 - 4\delta_{11}x_1x_3 - 4\delta_{12}x_1x_2 - 4\delta_{12}x_2x_3 \\ & \quad - 7\delta_{11} + 16\delta_{12} + 4\delta_{11}\delta_{12} + 9). \end{aligned}$$

3.2 A QAOA approach

This section assumes we are optimizing a problem of the form $\min_{x\in\{0,1\}^n} \sum_{m_i\in M} m_i(x)$. The natural extension of QAOA on general PUBOs is to define the unitary operator $U(C, \gamma) = e^{-i\gamma C} = \prod_{m_i\in M} e^{-i\gamma m_i} = \prod_{m_i\in M} U(m_i, \gamma)$, while the mixing operator remains $U(B, \beta) = e^{-i\beta B}$ where $B = \Sigma_{v\in V(G)} B_v$, $B_v = \sigma_v^x$ for $v \in V(G)$ and σ_v^x is the Pauli X operator acting on qubit v. $U(C, \gamma)$ can be compiled on a circuit by decomposing it into a sequence of gates performing all $U(m_i, \gamma)$ operators. The number of qubits each $U(m_i, \gamma)$ acts on is the number of variables in monomial m_i, which depends on the size of the support of p_i, denoted $\text{supp}(p_i)$. We seek to explore the relationship between the structure of the monomial and the minimum depth required for a quantum circuit to optimize such a function.

First, we assume that all m_i have been combined optimally to fit the hardware, meaning each polynomial has size at most the maximal gate size the hardware supports, and any monomials that can be combined and fit on one gate have been combined. Although current hardware currently supports gate width of two, we look at larger gate width for completeness. Operators $U(m_i, \gamma)$ and $U(m_j, \gamma)$ cannot be performed in parallel unless they act on disjoint sets of qubits. With that in mind, we construct a proper hypergraph edge coloring that minimizes the total depth of circuit, where edges of the same color represent sets of operators that can be performed in parallel. We let

$H = (V_H, E_H)$ be such a hypergraph where $V_H = \{1, \ldots, n\}$ and E_H consists of edges $e_i = \text{supp}(m_i)$ for all $m_i \in M$.

Theorem 1 *Every proper edge coloring of H corresponds to a valid circuit for PUBO, where the depth of the shallowest circuit is* $\chi'(H) + 1$.

Proof Let $v_a v_b ... v_d$ be the support of monomial m_i and $v_e v_f ... v_h$ be the support of monomial m_j such that $\{v_a, v_b, ..., v_d\} \cap \{v_e, v_f, ..., v_h\} = \{\emptyset\}$. Then, $U(C_{m_i}, \gamma)$ and $U(C_{m_j}, \gamma)$ can be implemented simultaneously in a circuit. Since the intersection is empty, the edges may receive the same color in a proper coloring, but do not necessarily, as there may be several proper colorings of one graph. Thus, a proper edge coloring gives a feasible implementation of a circuit that can be used to solve a combinatorial optimization problem. There exists a coloring of H that uses exactly $\chi'(H)$ colors, and by definition, any coloring that uses fewer colors is not proper. If the coloring is not proper, two edges that share a vertex have the same color and their corresponding gates cannot be implemented simultaneously. Hence, the depth of the shallowest circuit is $\chi'(H) + 1$, as one must be added to account for $U(B, \beta)$. □

Determining the chromatic index of hypergraphs in general is a difficult problem. In 1972, Erdös, Faber, and Lovász conjectured that the chromatic index of any linear hypergraph on n vertices is at most n [20]. Since then, the conjecture has been proven if H satisfies $\Delta(H) \leq \sqrt{n + \sqrt{n} + 1}$ [21]. Additionally, Chang and Lawler showed that the chromatic index of a hypergraph H on n vertices is at most $\lceil 1.5n - 2 \rceil$ with no restriction on the degrees of the vertices. In 1992, Kahn showed that $\chi'(H) \leq n + o(n)$ for linear H [22]. Note that since any two edges in a linear hypergraph intersect in at most one vertex, that is equivalent to saying any two monomials in an optimization problem share at most one common variable. As there are bounds on the chromatic index of linear hypergraphs, in a general combinatorial optimization problem, one could attempt to relax the problem such that for any two monomials a and b, $|\text{supp}(a) \cap \text{supp}(b)| \leq 1$ in order to have a rough bound on the depth of the circuit.

In addition to linear hypergraphs, there has been work on bounding the chromatic index of k-uniform hypergraphs. Pippenger and Spencer proved that if a k-uniform hypergraph has minimum degree asymptotic to the maximum degree and asymptotic codegree negligible compared to the maximum degree, then for some $\delta > 0$, $\chi'(H) \leq (1 + \delta)\Delta(H)$ [23]. Later, Alon and Kim showed that if H is k-uniform and if any two edges have at most t vertices in common and maximum degree sufficiently large as a function of k, then $|E_h| \leq (t - 1 + \frac{1}{t})\Delta(H)$, which bounds the chromatic index of H from above [24]. As each edge in a k-uniform hypergraph contains k vertices, it is equivalent to the original combinatorial optimization problem containing monomials that consist of precisely k variables. Thus, the circuit depth of problems that can be written such that each monomial has the same size support can be bounded.

We can potentially combine $U(C_a, \gamma)$ and $U(C_b, \gamma)$ into $U(C_{a,b}, \gamma)$, which could reduce the number of colors needed for the corresponding graph. Doing so, however, requires solving a potentially difficult optimization problem. Consider the problem:

$$\min \sum_{c \in C} z_c \tag{13}$$

$$\text{s.t.} \sum_{s \in S} x_s^c \leq |S| - 1 \qquad \forall\, c \in C,\ S \text{ s.t. } |\cup_{s \in S} s| > L \tag{14}$$

$$x_e^c \leq z_c \qquad \forall\, e \in E,\ c \in C \tag{15}$$

$$x,\ z \in \{0, 1\} \tag{16}$$

where L is the number of qubits in the largest gate the hardware can perform, c is a color in the collection of colors C, s is an edge of $S \subset E(\mathcal{H})$, and x_s^c indicates that s receives color c in a particular proper coloring. One obvious example of how to combine gates is, if for monomials a and b, $\text{supp}(a) \subset \text{supp}(b)$ then the size of the gate required for $U(C_{a,b}, \gamma)$ will be identical to $U(C_b, \gamma)$. Thus, $U(C_a, \gamma)$ and $U(C_{a,b}, \gamma)$ can be combined since $\text{supp}(a) \cup \text{supp}(b) = \text{supp}(b)$.

Throughout the rest of this paper, we define the *derived graph* as the graph corresponding to a combinatorial optimization problem whose vertex set consists of the variables in the problem and whose edges connect vertices that are found in a common monomial. The *derived hypergraph* is similarly defined.

3.2.1 Examples continued

In this section, we analyze the structure of the resulting hypergraphs built from the examples in Sect. 3.1 and discuss how this impacts the difficulty of performing each problem on NISQ devices.

Example: MaxCut, continued

The support of the cost function is six, but each gate acts on two qubits in the circuit since each monomial has at most two unique x_i terms. We define gates $U(C_{i,j}, \gamma)$ for monomials that have two variables, x_i and x_j.

As i, j, m, and n must be unique in order to run $C_{i,j}$ and $C_{m,n}$ at the same time, we can color the edges of a graph G and perform operators associated to the edges of the same color class at once. Thus, the depth of the circuit for MaxCut is either $\chi'(G) + 1$, as one must be added to account for the B gates, and the depth scales linearly with the number of iterations of the algorithm. Figure 2 is a circuit diagram for implementing MaxCut on W_6 using the PUBO mapping and QAOA approach.

Example: Maximum Independent Set, continued

The support of the optimization function has size six, and each monomial is comprised of at most two variables. The circuit diagram for this example is the same as in **Example: MaxCut, continued**, as it contains the same monomials, up to constants and signs.

Example: General Problem, continued

Since several monomials in the function to optimize are contained in the support of others, the gates needed in the QAOA circuit are those acting on $x_1x_2x_3$, $x_1x_2\delta_{11}$, $x_1x_2\delta_{12}$, $x_1x_3\delta_{11}$, $x_1x_3\delta_{12}$, $x_2x_3\delta_{11}$, $x_2x_3\delta_{12}$ and $\delta_{11}\delta_{12}$, and the associated hypergraph and coloring for it is Fig. 3. The circuit diagram for this example is seen in Fig. 4.

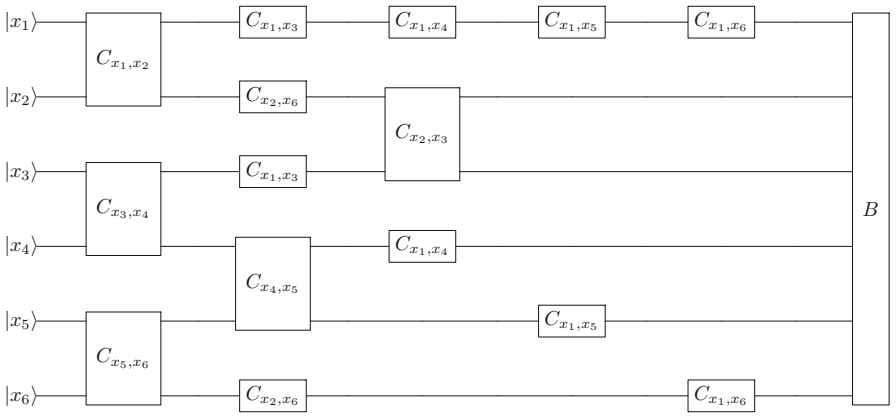

Fig. 2 The circuit diagram for a 1-level QAOA on the wheel graph on six vertices. We use the notation $C_{i,j}$ to represent $U(C_{i,j}, \gamma)$ to make the image clearer to read. If two qubits are used in one gate but not next to each other in number order, the gate in the diagram appears split, but the pieces are in the same column

4 Decomposing gates

The above arguments assume that the given PUBO fits onto the given hardware. In cases where the required gate sized are larger than what is available, we would need to decompose the problem. This comes with an additional cost.

4.1 Classically converting a PUBO to a QUBO

Classically, we can convert a PUBO of degree k to a QUBO in the following manner.

Suppose we have a monomial of the form $u = x_1 x_2 \dots x_k$. Note that dualizing u will create monomials with degrees larger than $\frac{k}{2}$. Instead of doing this, we enforce this equality by ensuring that u takes the appropriate values by adding the following set of linear constraints.

$$\begin{aligned} u &\leq x_i && \forall\, i \in [k] \\ u &\geq \sum_{i\in[k]} x_i - k + 1 \end{aligned}$$

Adding in the slack variables, $s_{m,n}$, we have

$$\begin{aligned} u - x_i + s_{1,i} &= 0 && \forall\, i \in [k] \\ u - \sum_{i\in[k]} x_i + k - 1 + \sum_{j\in[\lfloor \log(k)\rfloor+1]} s_{2,j} &= 0 \end{aligned}$$

Note that k-many slacks are added to the first constraint and the latter requires at least $\lfloor \log(k)\rfloor$ many. With the additional u variable, this means that $k + \lfloor \log(k)\rfloor + 2$ ancillary qubits are needed for this decomposition.

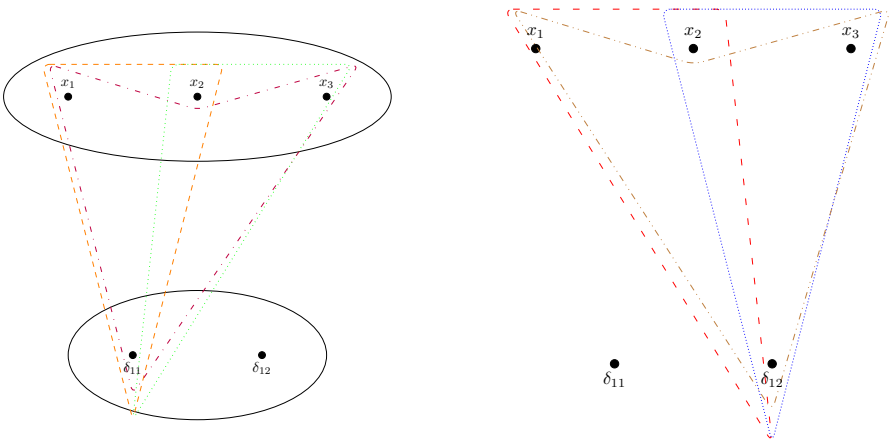

Fig. 3 The coloring for the hypergraph in *Example: General Problem*. It has vertices $x_1, x_2, x_3, \delta_{11}$, and δ_{12}. The edges have been placed into two separate images to show the coloring more clearly, though the entire hypergraph contains the edges found in both figures. No colors are repeated between the left and right sides, and the hypergraph requires seven colors (Color figure online)

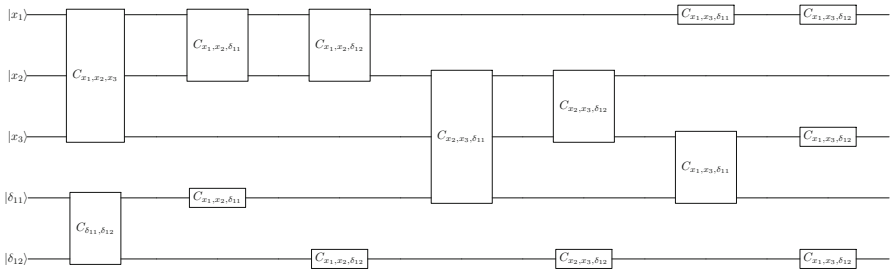

Fig. 4 The circuit diagram for the general combinatorial optimization example. We use $C_{i,j,k}$ to represent $U(C_{i,j,k}, \gamma)$ to make the image clearer to read. If multiple qubits are used in one gate but not next to each other in number order, the gate in the diagram appears split, but the pieces are in the same column

Dualizing the first constraint gives the following binomials:

$$ux_i \quad us_{1,i} \quad x_i s_{1,i} \quad \forall i \in [k],$$

while dualizing the second gives binomials:

$$ux_i \quad x_i x_{i'} \quad us_{2,j} \quad x_i s_{2,j} \quad s_{2,j} s_{2,j'} \quad \forall i, i' \in [k],\ j,\ j' \in [\lfloor \log(k) \rfloor + 1].$$

In the graph representation of the above constraints, the degree of u is $2k + \lfloor \log(k) \rfloor + 1$ so it will take a circuit of at least this depth to enforce the decomposition.

Example: General Problem, continued In this example, we show how to decompose $x_1x_2x_3$ into a sum of monomials in two variables. This method applies in general.

Let $u = x_1x_2x_3$. Then the constraints $u \leq x_1$, $u \leq x_2$, and $u \leq x_3$ must be added to the problem, and we want to penalize the solution $u \geq x_1 + x_2 + x_3$, so we have six new variables, u, $s_{1,i}$ for $i \in [3]$, and $s_{2,j}$ for $j \in [2]$. Dualizing this

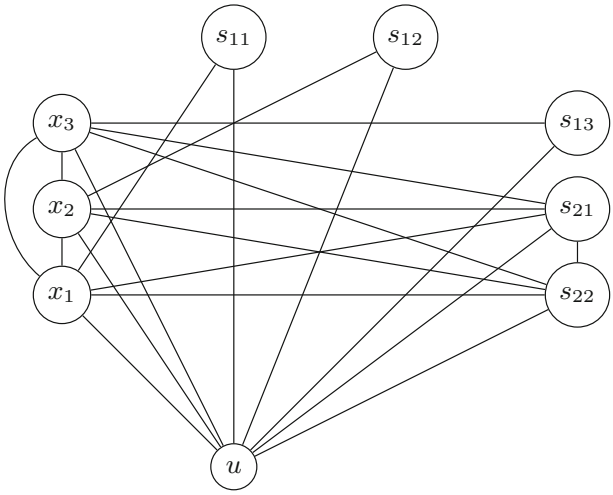

Fig. 5 The derived graph for $x_1x_2x_3$. The chromatic index of this graph is eight

problem, we get all terms of the forms $x_i x_{i'}$, $x_i s_{1,i}$, $x_i s_{2,1}$, ux_i, $us_{1,i}$, $us_{2,j}$ $s_{2,1}s_{2,2}$ and ux_i for $i, i' \in [3]$ and $j \in [2]$. The derived graph is shown in Fig. 5, and eight colors are needed for a proper edge coloring, which is minimal as u has degree eight.

Thus, it takes a depth of eight to implement $U(C_{x_i x_j x_k}, \gamma)$ in a circuit. When decomposing multiple gates on the same circuit, many of the operations can be either aggregated or done in parallel. For example, in the above example, decomposition of both $U(C_{x_1 x_2 \delta_{11}})$ and $U(C_{x_1 x_2 \delta_{12}})$ contain operations on x_1 and x_2 that can be aggregated while operations regarding the ancilla added during each decomposition can all be done in parallel.

Note that as the above decomposition converts a PUBO into a QUBO, we can identify the depth of circuit for the decomposed problem by finding the maximum degree vertex in the graph (non-hypergraph) encoding of the QUBO. In the above decomposition, a vertex representing x_i will be adjacent to other vertices x_j if and only if they are in at least one monomial term in the PUBO. Moreover, each vertex representing x_i is adjacent to $\lfloor \log(k) \rfloor + 3$ unique ancilla vertices for each monomial of degree k. We can similarly easily identify the maximim degree of the ancilla vertices, as the degree of the u vertex is shown to be $2k + \lfloor \log(k) \rfloor + 1$, which will always be larger than the degree of the s-vertices. With this, we can easily identify the depth of circuit required to enforce this decomposition.

Note that there may be different ways to decompose a given monomial. For example, we can use the above to decompose $x_1x_2x_3x_4$ or we can write it as $u_{12}u_{34}$ where $u_{12} = x_1x_2$ and $u_{34} = x_3x_4$. Enforcing this decomposition may result in more ancilla being used at the benefit of a smaller depth of circuit. We will explore this tradeoff in future work.

4.2 Decomposing a *n* qubit gate into two qubit gates

In general, any unitary acting on n qubits can be decomposed into 4^n controlled- not gates [25], so each $U(C_{x_1,\ldots x_k}, \gamma)$ can be implemented with 4^k controlled-not gates. Let D be the set consisting of the number of qubits on which each unitary operates in the circuit diagram. Since the proof is not constructive in general, it is unknown how many of these controlled-not operations can be performed in parallel, but it does provide an upper bound for the depth of circuit $\sum_{j\in D} 4^j$. As a special case, Bullock and Markov showed that if a unitary operator is diagonal in the computational basis, such as C for MaxCut, it can be decomposed into $2^{n+1} - 3$ controlled not and single qubit z rotation gates [26]. In general, a complex valued matrix is diagonalizable if it commutes with its conjugate transpose or is Hermitian, so all but a subset of matrices of Lesbesgue measure zero are diagonalizable.

Thus, if all $C_{x_1,\ldots,x_k}$ are diagonal, each $U(C_{x_1,\ldots x_k}, \gamma)$ can be implemented in $2^{k+1} - 3$ one or two qubit gates. Then, the depth of circuit for all diagonal unitaries is at most $\sum_{j\in D} 2^{j+1} - 3$. In either case, the depth of each circuit increases exponentially in the number of qubits upon which each operator acts.

The number of one and two qubit gates needed to implement $U(C_{x_1,\ldots x_k}, \gamma)$ can be reduced at the cost of ancillary qubits. Cao et al. [27] introduced an iterative method to construct a logical sequence consisting of 3 qubit systems from a k qubit system by subdividing the space in $\lceil \log(k-2) \rceil$ iterations. The method uses Hamiltonian subdivision gadgets and requires $k - 3$ ancillary qubits to perform the operations. After partitioning the space, the subdivision gadget is used to construct a new Hamiltonian. Then, a penalty Hamiltonian is applied and the space is perturbed. To then reduce to two-qubit interactions, one additional qubit is needed. Thus, an arbitrary k qubit operation in the circuit can be written as a similar logical sequence using $k - 2$ ancillary qubits. The overall cost is linear in the number of operators that need be reduced, but error is accumulated with the number of subdivisions needed. The number of two qubit gates needed to perform the subdivision depends on the structure of each term in the k body Hamiltonian.

5 QAOA circuit depth bounds for some combinatorial optimization problems

In this section, we review some combinatorial optimization problems and discuss the depth of circuit required for one QAOA iteration of each problem instance. NP-complete problems can be reduced to other NP-complete problems; however, the act of reducing may impact the depth of circuit, which may or may not be desirable depending on the hardware. In each case below, the depth of circuit assumes fully connected hardware, so these scenarios are the best possible.

5.1 Vertex covering

A vertex cover of a graph $G = (V, E)$ is a collection $S \subset V$ such that for all $xy \in E$, at least one of x or y is contained in S. Finding a minimum vertex covering is classically NP-complete [28] and is written as

$$\begin{aligned} &\min \sum_{i \in V} x_i \\ &\text{s.t. } (1 - x_i) + (1 - x_j) \leq 1 \qquad \forall (i, j) \in E \\ &x_i \in \{0, 1\} \end{aligned}$$

As each constraint consists of the sum of two unique variables and a constant, dualizing them gives monomials consisting of at most two distinct variables, as each monomial corresponds to an edge in the graph. The derived graph has vertex set $V = x_i \cup \delta_{ij}$ for $i \in [n]$ and $j \neq i$ with $j \in [n]$. Note that δ_{ij} is incident only to vertices x_i and x_j, as it only occurs in constraints including those variables. Thus, the depth of circuit is $2\chi(G) + 1$, so the difficulty of covering problems is directly related to the maximum degree of the problem. Graphs with low degree allow for a shallow circuit in one iteration of QAOA, so they should be suitable for NISQ devices.

5.2 Knapsack

In the knapsack problem, a collection of objects, x_i for $i \in [n]$, are assigned a weight, w_i, and a value, v_i. The goal is to maximize the sum of the value of the objects while the sum of the weights of the objects is restricted to be less than some constant W. This problem is NP-complete classically, as well [28]. As an integer program, it is written

$$\begin{aligned} &\max \sum_{i \in [n]} v_i x_i \\ &\text{s.t. } \sum_{i \in [n]} w_i x_i \leq W \\ &x_i \in \{0, 1\} \end{aligned}$$

Knapsack problems where W is large pose problems for testing on quantum computers because the larger W is, the more δ_{ij} variables are needed when dualizing the constraint. The derived graph contains vertices for each x variable and δ variable and is complete, meaning that its edge coloring has minimum coloring of at least $n + ln(W)$; however, pre-processing can be used to reduce the depth of the embedding. Assuming the weights are ordered such that $w_1 \leq w_2 \leq \cdots \leq w_n$, the total weight of an optimal solution must be at least $W - w_n$, since, if not, there is room in the knapsack for an additional item. With this in mind, the knapsack constraint can be written as $W - w_n \leq \sum_{i \in [n]} w_i x_i \leq W$, which now requires $\ln(w_n)$ many additional variables and leads to a circuit depth of $n + \ln(w_n)$. In order for there to be nontrivial

instances of the knapsack problem with small w_n, there necessarily must be small W. Knapsack problems with small W can be suitable for experimentation; however, classically, problems with bounded W are polynomial and can be easily solved by conventional computing by dynamic programming [29,30]. Thus, they may not be suitable for quantum computing.

5.3 Traveling salesperson

The traveling salesperson problem (TSP) can be viewed as a problem on a graph $G = (V, E)$ where each edge e has an associated weight, w_e. The goal is to start in a vertex, say v_1, use edges to visit each vertex exactly once, and return to v_1, all while minimizing the sum of the weights of the edges used. This problem is classically NP-hard, but there exist some heuristics for the problem [31,32].

Let x_e represent if the salesperson travels along edge e. One formulation of the problem is

$$\begin{aligned}
&\min \sum_{e\in E} w_e x_e \\
&\text{s.t. } 0 \le x_e \le 1 \ \forall e \in E \\
&\sum_{e\ni i} x_e = 2 \ \forall i \in V \\
&\sum_{e=(i,\,j),\ i\in Q,\ j\in Q} x_e \le |Q| - 1 \qquad\qquad Q \subsetneq [n], |Q| \ge 2.
\end{aligned}$$

The derived graph for TSP has vertex set $x_i \cup \delta_{ij}$, where there are two δ_{ij} variables per constraint. The edges form a complete graph on all x_i vertices and connect δ_{ij} to δ_{ik} for $j \neq k$. As there are two δ variables per constraint, they form a disjoint collection of edges. The rest of the edges connect every x_i variable to every δ_{ij} variable. Thus, the maximum degree of the graph is $n - 1$ plus twice the number of constraints. Denoting the number of constraints as N_c, the depth of circuit is $n - 1 + 2N_c$, where N_c can be large, depending on the problem instance. Similarly to knapsack problems, it can be difficult to implement TSP on NISQ devices because of the subtour constraint, $Q \subsetneq [n]$, and the fact that so many new variables are introduced in dualizing.

5.4 SAT

In Boolean satisfiability problems (SAT), there are a set of clauses, C, containing a set of literals, N. The goal is to determine if the values of TRUE or FALSE can be assigned to each literal in a clause such that it evaluates to TRUE. This problem, again, is classically NP-complete [28], even when each clause contains only three literals. Let $\{z_c\}_{c\in C}$ be a collection of indicator variables for clauses in three variables, where $z_i = 0$ if clause i is satisfied and 1 if not. Let x_i be the indicator variable denoting if literal i is satisfied. Let TRUE_c (FALSE_c) be the set of literals that must be true (false) to satisfy clause c. Then, the problem can be written as

$$\begin{aligned} &\min \sum_{c\in C} z_c \\ &\text{s.t.} \sum_{x_i=\text{TRUE}_c} x_i + \sum_{x_i=\text{FALSE}_c} (1-x_i) \geq 1 + z_c \qquad \forall c \in C \\ &x_i,\ z_c \in \{0,1\} \end{aligned}$$

However, taking the contrapositive, we have $\sum_{x_i=\text{TRUE}}(1-x_i)+\sum_{x_i=\text{FALSE}} x_i \leq 2+z_c$. The derived graph, is again, a graph consisting of all x_i vertices and two dummy variables, δ_c^1 and δ_c^2, per clause. If x_i appears in the set of clauses $C_{x_i} \subset C$, the degree of the x_i, d_{x_i}, is $d_{x_i} = |\cup_{c\in C_{x_i}} c| - 1 + 2|C_{x_i}|$. SAT can be a good problem for NISQ devices if the set of literals is large while the number of literals in each clause and the number of clauses are relatively small, as this guarantees a literal cannot occur in many clauses and each literal does not appear in clauses with several others.

6 Discussion

We have shown how to map arbitrary combinatorial optimization problems to polynomial unconstrained binary optimization problems (PUBOs) by dualizing constraints, and applied the method to a few combinatorial optimization problems. Additionally, we discussed how use the PUBOs to derive a graph that represents problem instances and used this to show that the depth of the QAOA circuit needed to run the problem is $\chi'(G) + 1$. We then considered various combinatorial optimization problems and determined the depth of circuit required to use QAOA to solve them. In particular, since the vertex covering problem has a low depth of circuit, it appears to be suitable for NISQ devices, as do instances of SAT problems that have large sets of literals but few clauses and few literals in each clause. Due to the number of new variables that must be introduced to dualize knapsack and TSP, they do not appear to be good problems to test on NISQ devices.

Clearly, the maximum degree of a vertex affects the circuit depth in combinatorial optimization problems in which each monomial consists of at most two unique variables, such as MaxCut and MaxIndSet. Specifically, the depth of the QAOA circuit is $\chi'(G) + 1$, where $\chi'(G)$ for graphs that are not hypergraphs is Δ or $\Delta + 1$, by a classic result of Vizing [19]. In the case of monomials of at least three variables, a lower bound for the circuit depth is the number of colors needed in a proper edge coloring of the associated hypergraph, H, which is a difficult problem. A trivial lower bound on this number is the maximum degree of H while a trivial upper bound is the number of edges in H.

The depth of circuit is hard to determine in part because when dualizing, squaring $p(x)$ can potentially yield monomials with larger support than any in c. Sparser constraints are preferable because the polynomials have smaller support, which decreases the size of each gate. However, sparser constraints does not imply a shallower circuit depth. For example. consider MaxCut on a star graph on n vertices, that is a connected bipartite graph in which one part contains one vertex and the other contains $n - 1$. The depth of circuit is n, as each edge must have a unique color.

As any combinatorial optimization problem can be mapped to a PUBO via dualizing constraints, we can examine the resulting QAOA circuit and bound the depth of it by the edge coloring of the hypergraph associated to the problem instance. Although current hardware is limited to two qubit gates, larger gates can be decomposed into two qubit gates. It would be interesting to see if there is a way to construct a graph associated to the decomposed gates and if its chromatic number, or some other property of the graph, determines the depth of circuit.

Acknowledgements This work was supported by DARPA ONISQ program under award W911NF-20-2-0051. J. Ostrowski acknowledges the Air Force Office of Scientific Research award, AF-FA9550-19-1-0147. This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan. (http://energy.gov/downloads/doe-public-access-plan).

References

1. Farhi, E., Goldstone, J., Gutmann, S.: A quantum approximate optimization algorithm. arXiv preprint arXiv:1411.4028 (2014)
2. Guerreschi, G.G., Smelyanskiy, M.: Practical optimization for hybrid quantum-classical algorithms. arXiv preprint arXiv:1701.01450 (2017)
3. Streif, M., Leib, M.: Training the quantum approximate optimization algorithm without access to a quantum processing unit. arXiv preprint arXiv:1908.08862 (2019)
4. Shaydulin, R., Safro, I., Larson, J.: Multistart methods for quantum approximate optimization. In: 2019 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–8. IEEE (2019)
5. Farhi, E., Goldstone, J., Gutmann, S.: A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem. arXiv preprint arXiv:1412.6062 (2014)
6. Zhou, L., Wang, S.-T., Choi, S., Pichler, H., Lukin, M.D.: Quantum approximate optimization algorithm: performance, mechanism, and implementation on near-term devices. arXiv preprint arXiv:1812.01041 (2018)
7. Fingerhuth, M., Babej, T., et al.: A quantum alternating operator ansatz with hard and soft constraints for lattice protein folding. arXiv preprint arXiv:1810.13411 (2018)
8. Cook, J., Eidenbenz, S., Bärtschi, A.: The quantum alternating operator ansatz on max-k vertex cover. arXiv preprint arXiv:1910.13483 (2019)
9. Huang, H.-Y., Bharti, K., Rebentrost, P.: Near-term quantum algorithms for linear systems of equations. arXiv preprint arXiv:1909.07344 (2019)
10. Saleem, Z.H.: Maximum independent set and quantum alternating operator ansatz. arXiv preprint arXiv:1905.04809 (2019)
11. Wang, Z., Hadfield, S., Jiang, Z., Rieffel, E.G.: Quantum approximate optimization algorithm for maxcut: a fermionic view. Phys. Rev. A **97**(2), 022304 (2018)
12. Crooks, G.E.: Performance of the quantum approximate optimization algorithm on the maximum cut problem. arXiv preprint arXiv:1811.08419 (2018)
13. Guerreschi, G.G., Matsuura, A.Y.: Qaoa for max-cut requires hundreds of qubits for quantum speed-up. Sci. Rep. **9**, 1–7 (2019)
14. Farhi, E., Harrow, A.W.: Quantum supremacy through the quantum approximate optimization algorithm. arXiv preprint arXiv:1602.07674 (2019)
15. Wang, Z., Rubin, N.C., Dominy, J.M., Rieffel, E.G.: xy-mixers: analytical and numerical results for qaoa. arXiv preprint arXiv:1904.09314 (2019)
16. Xue, C., Chen, Z.-Y., Wu, Y.-C., Guo, G.-P.: Effects of quantum noise on quantum approximate optimization algorithm. arXiv preprint arXiv:1909.02196 (2019)

17. Wang, S., Fontana, E., Cerezo, M., Sharma, K., Sone, A., Cincio, L., Coles, P.J.: Noise-induced barren plateaus in variational quantum algorithms. arXiv preprint arXiv:2007.14384 (2020)
18. Marshall, J., Wudarski, F., Hadfield, S., Hogg, T.: Characterizing local noise in qaoa circuits. arXiv preprint arXiv:2002.11682 (2020)
19. Vizing, V.G.: On an estimate of the chromatic class of a p-graph. Discret Anal. **3**, 25–30 (1964)
20. Erdős, P.: Problems and results in graph theory and combinatorial analysis. In: Proceedings of the 5th British Combinatorial Conference, pp. 169–192 (1975)
21. Paul, V., Germina, K.A.: On edge coloring of hypergraphs and erdös-faber-lovász conjecture. Discrete Math. Algorithms Appl. **4**(01), 1250003 (2012)
22. Kahn, J.: Coloring nearly-disjoint hypergraphs with n+ o (n) colors. J. Comb. Theory, Ser. A **59**(1), 31–39 (1992)
23. Pippenger, Nicholas, Spencer, Joel: Asymptotic behavior of the chromatic index for hypergraphs. J. Comb. Theory, Ser. A **51**(1), 24–42 (1989)
24. Alon, N., Kim, J.H.: On the degree, size, and chromatic index of a uniform hypergraph. J. Comb. Theory, Ser. A **77**(1), 165–170 (1997)
25. Vartiainen, J.J., Möttönen, M., Salomaa, M.M.: Efficient decomposition of quantum gates. Phys. Rev. Lett. **92**(17), 177902 (2004)
26. Bullock, S.S., Markov, I.L.: Asymptotically optimal circuits for arbitrary n-qubit diagonal computations. arXiv preprint quant-ph/0303039 (2008)
27. Cao, Y., Babbush, R., Biamonte, J., Kais, S.: Hamiltonian gadgets with reduced resource requirements. Phys. Rev. A **91**(1), 012315 (2015)
28. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) Complexity of computer computations, pp. 85–103. Springer (1972)
29. Andonov, R., Poirriez, V., Rajopadhye, S.: Unbounded knapsack problem: dynamic programming revisited. Eur. J. Oper. Res. **123**(2), 394–407 (2000)
30. Frieze, A.M.: Shortest path algorithms for knapsack type problems. Math. Program. **11**(1), 150–157 (1976)
31. Ouaarab, A., Ahiod, B., Yang, X.-S.: Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput. Appl. **24**(7–8), 1659–1669 (2014)
32. Masutti, T.A.S., de Castro, L.N.: A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem. Inf. Sci. **179**(10), 1454–1468 (2009)

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