




Article

Globally Optimizing QAOA Circuit Depth for Constrained Optimization Problems

Rebekah Herrman ^{1,*}, Lorna Treffert ¹, James Ostrowski ¹, Phillip C. Lotshaw ^{2,†}, Travis S. Humble ^{2,†} and George Siopsis ³

¹ Department of Industrial and Systems Engineering, University of Tennessee at Knoxville, Knoxville, TN 37996, USA; ltreffer@vols.utk.edu (L.T.); jostrows@utk.edu (J.O.)

² Quantum Computing Institute, Oak Ridge National Laboratory, Oak Ridge, TN 37830, USA; lotshawpc@ornl.gov (P.C.L.); humblets@ornl.gov (T.S.H.)

³ Department of Physics and Astronomy, University of Tennessee at Knoxville, Knoxville, TN 37996-1200, USA; siopsis@tennessee.edu

* Correspondence: rherrma2@utk.edu

† 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>, accessed on 14 September 2021).

Abstract: We develop a global variable substitution method that reduces n -variable monomials in combinatorial optimization problems to equivalent instances with monomials in fewer variables. We apply this technique to 3-SAT and analyze the optimal quantum unitary circuit depth needed to solve the reduced problem using the quantum approximate optimization algorithm. For benchmark 3-SAT problems, we find that the upper bound of the unitary circuit depth is smaller when the problem is formulated as a product and uses the substitution method to decompose gates than when the problem is written in the linear formulation, which requires no decomposition.

Keywords: quantum approximate optimization algorithm; circuit depth; 3-SAT



Citation: Herrman, R.; Treffert, L.; Ostrowski, J.; Lotshaw, P.C.; Humble, T.S.; Siopsis, G. Globally Optimizing QAOA Circuit Depth for Constrained Optimization Problems. *Algorithms* **2021**, *14*, 294. <https://doi.org/10.3390/a14100294>

Academic Editor: Theodore B. Trafalis

Received: 14 September 2021

Accepted: 5 October 2021

Published: 11 October 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The quantum approximate optimization algorithm (QAOA) was introduced to approximately solve combinatorial optimization problems [1,2]. QAOA research has mostly focused on a small subset of combinatorial optimization (CO) problems such as MaxCut, MaxIndSet, and Max k-cover [3–9]. These problems can be easily written as quadratic unconstrained binary optimization (QUBO) problems by identifying each variable with a qubit. QUBOs are implementable on current hardware [10,11]. Recent work has examined how QAOA can be used on CO problems that can be written as polynomial unconstrained binary optimization (PUBO) problems [12,13]. When solving a CO problem using QAOA, each monomial in k vertices in the problem formulation corresponds to a k qubit gate. The unitary circuit depth for one layer of QAOA for combinatorial optimization problems was shown in [14] to be the edge chromatic number of the graph, or hypergraph, derived from these problems. Here the unitary circuit depth is the number of layers of one- and two-qubit unitary operators that are performed in series to implement the circuit. When the combinatorial optimization problem can be written as a QUBO, the derived graph is not a hypergraph, so the edge chromatic number is either the maximum degree, or the maximum degree plus one [15]. When the derived graph is a hypergraph, the edge chromatic number may be more difficult to compute.

Some classes of combinatorial optimization problems, however, can be written in more than one way, where each formulation may have monomials of different sizes. For example,

Boolean satisfiability problems (SAT) can be written as a QUBO or a more general PUBO. SAT problems have been studied extensively in a classical setting [16,17] but there have been few studies into solving them in a quantum setting [18]. They form the backbone of complexity theory and can be written in a linear form, which requires two-qubit gates to implement, or a product form, which requires larger multi-qubit gates. Since these larger multi-qubit gates are not easily implementable on current hardware, polynomial formulations must be decomposed into sums of two variable monomials. We show in the context of 3-SAT that decomposing the PUBO can lead to a shallower circuit needing fewer qubits than using the natural QUBO formulation [14]. This has implications for more general combinatorial problems, as modeling approaches can have a significant impact on the design of the resulting circuit.

In this paper, we first review QAOA, the classical linear and product formulations of SAT problems, and how they translate to QUBOs in Section 2. Then, in Section 2.5, we introduce a substitution method, called the global variable substitution (GVS) method, to decompose monomials consisting of $k \geq 3$ variables into ones that can be implemented on current hardware. Next, we discuss how to optimize GVS for 3-SAT problems in Section 2.6 and then apply this work to instances from the SATLIB Benchmark Problems suite [19] in Section 3. Finally, we summarize the results and discuss future work in Section 4.

2. Materials and Methods

In this section, we review QAOA, dualization, and the 3-SAT problem.

2.1. QAOA

In order to use QAOA to solve a CO problem, we apply two operators, $U(C, \gamma) = e^{-iC\gamma}$ and $U(B, \beta) = e^{-iB\beta}$, in succession on an initial state. The initial state is the uniform superposition, $|s\rangle = \frac{1}{\sqrt{2^n}} \sum_z |z\rangle$, where the sum is over the computation basis $|z\rangle$. The outcome of one iteration of QAOA is

$$|\gamma, \beta\rangle = U(B, \beta)U(C, \gamma)|s\rangle.$$

Here C encodes the problem to be solved and B is a mixing operator. Often, C is the sum over a collection of clauses, $C = \sum_a C_a$, and B is typically $B = \sum_{v \in V(G)} B_v$, where $B_v = \sigma_v^x$ is the Pauli-X operator acting on the v^{th} qubit. For more detail about QAOA, see [1,2,20].

There is a direct correlation between the circuit depth and level of noise in a quantum circuit [21–23], so it is important to consider the circuit depth when developing a circuit that implements QAOA. Throughout this paper, we use unitary circuit depth to mean the depth of the unitary operators needed in a circuit. The actual circuit depth would be the unitary circuit depth times the number of gates needed to implement each operator. In the next subsection, we review dualization and the related previous work that determines the unitary circuit depth of one iteration of QAOA.

2.2. Dualization and Unitary Circuit Depth

Previously [14], we considered the method of dualizing constraints to solve CO problems of the form

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

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

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

where p_i is contained in the collection of polynomial constraints P . Both p_i and c are polynomial functions in $\mathbb{R}^n[x_1, x_2, \dots, x_n]$ and $b_i \in \mathbb{R}$. We eliminate constraints via dualization by subtracting b_i from both sides, adding in slack variables, squaring both sides, and adding the new expression to the objective function [14]. The *derived (hyper)graph* from dualization

is one in which there is a vertex for each variable and (hyper)edges between variables that appear in a monomial together. For example, if x_1x_2 appears in the dualization, there is an edge between vertices x_1 and x_2 .

In graph theory, a proper edge coloring of a graph is a function $f : E(G) \rightarrow [k]$ such that if two edges share a vertex, they receive different values, e.g., uv and xv , $f(uv) \neq f(xv)$. Given the set $[k] = \{1, \dots, k\}$, each element of this set can be thought of as a color, hence the name edge coloring. The smallest number m such that a proper edge coloring of a graph G is possible with m colors is denoted $\chi'(G)$ and called either the chromatic index or the edge chromatic number. It has been shown that chromatic index of the derived (hyper)graph is directly related to the unitary circuit depth of one iteration of QAOA, if the size of each monomial in the objective function is at most the gate size the hardware can support [14]. In particular, the following theorem holds:

Theorem 1. *Let H be the hypergraph derived from a combinatorial optimization problem instance. Every proper edge coloring of H corresponds to a valid circuit for a PUBO, where the unitary depth of the shallowest circuit is $\chi'(H) + 1$.*

Unitary Circuit Depth Example: MaxCut

We will consider MaxCut as an example for how to construct a quantum circuit with minimal unitary depth. This process applies when determining the unitary circuit depth for QAOA to solve a general combinatorial optimization problem. Consider the triangle graph, which consists of three vertices that are pairwise adjacent, as seen in Figure 1. 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

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

In order to solve MaxCut on the triangle graph, we want to minimize

$$2(x_1x_2 + x_1x_3 + x_2x_3) - 2(x_1 + x_2 + x_3).$$

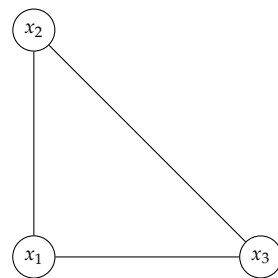


Figure 1. A triangle graph.

There are three vertices in the derived graph for this function, namely x_1 , x_2 and x_3 . There are also three edges, which are x_1x_2 , x_1x_3 , and x_2x_3 . Thus, the derived graph for this combinatorial optimization problem is also a triangle. To calculate the unitary circuit depth, we will properly edge color this graph in Figure 2. This graph requires three colors to be properly edge colored. Since each edge color class must be implemented in series, the unitary circuit depth is four, which is seen in Figure 3.

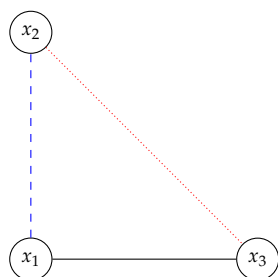


Figure 2. The edge colored graph derived when solving MaxCut on a triangle. The solid black is one color class, and red densely dotted and blue dashed are the other two. There are three colors needed to properly edge color this graph, so the unitary circuit depth is four.

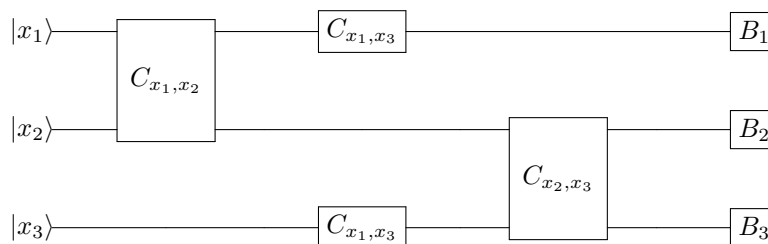


Figure 3. The unitary circuit required to solve MaxCut using QAOA. Note that no two qubit unitary operators can be implemented simultaneously since any two act on one common qubit. Here, C_{x_1,x_3} is a two qubit unitary, but is drawn as two single qubit unitary operators to avoid overlapping with $|x_2\rangle$.

2.3. SAT

A Boolean satisfiability problem (SAT) is one type of CO problem that has the form of Equations (1)–(3). It is defined by a collection of clauses, C , consisting of N literals. The goal is to determine if the values TRUE or FALSE can be assigned to each literal in a clause such that every clause is TRUE. This problem is classically NP-complete [24], even when each clause contains only three literals. When each clause contains precisely three literals, the problem is known as a 3-SAT problem. We will use 3-SAT as the key example throughout the paper. Each clause has the form $y_i \vee y_j \vee y_k$ where $y_i \in \{x_i, \sim x_i\}$. Now, we look at two formulations of 3-SAT problems.

2.3.1. SAT Linear Formulation

Let $\{z_c\}_{c \in C}$ be the collection of variables indicating if clause c is satisfied, and x_i be the indicator variable denoting if literal i is satisfied. Let $TRUE_c$ be the set of literals that must be true to satisfy c , and $FALSE_c$ the rest. Then, 3-SAT can be written as

$$\max \sum_{c \in C} z_c \tag{4}$$

$$\text{s.t.} \quad \sum_{x_i=TRUE_c} x_i + \sum_{x_i=FALSE_c} (1 - x_i) \geq z_c \quad \forall c \in C \tag{5}$$

$$x_i, z_c \in \{0, 1\}. \tag{6}$$

Equation (5) has this form because 3-SAT is defined to have only OR conditions, so at least one x_i must be satisfied when $z_c = 1$. Standard QAOA does not have a way to incorporate constraints; we therefore adopt the aforementioned strategy of dualizing the constraints to yield an unconstrained optimization problem suitable for QAOA as follows.

Before we can dualize Equation (5), we must change the inequality so it matches that of Equation (2). We therefore take the contrapositive of each constraint so Equation (5) has the form $\sum_{x_i=TRUE_c} (1 - x_i) + \sum_{x_i=FALSE_c} x_i \leq 2 + z_c$. We can express this constraint as an equality,

$$\sum_{x_i=TRUE_c} (1 - x_i) + \sum_{x_i=FALSE_c} x_i + \delta_{c,1} + \delta_{c,2} = 2 + z_c,$$

by introducing two slack variables $\delta_{c,1}$ and $\delta_{c,2}$ that guarantee equality holds for any choice of x_i satisfying the constraint.

Let us define

$$f_c(\{x_i, \delta_{c,j}\}, z_c) = \sum_{x_i=TRUE} (1 - x_i) + \sum_{x_i=FALSE} x_i + \delta_{c,1} + \delta_{c,2} - (2 + z_c).$$

The quantity $f_c(\{x_i, \delta_{c,j}\}, z_c)$ is equal to zero when the constraints are satisfied. 3-SAT can now be expressed as an unconstrained optimization problem by introducing $-\lambda f_c^2$ as a penalty term

$$\max \sum_{c \in C} (z_c - \lambda f_c(\{x_i, \delta_{c,i}\}, z_c)^2)$$

$$x_i, \delta_{c,i}, z_c \in \{0, 1\}.$$

where λ is a large number [25]. The variables $x_i, \delta_{c,j}$, and z_c can take values of either 0 or 1, however, choices that violate $f_c = 0$ give penalties $\lambda f_c(\{x_i, \delta_{c,i}\}, z_c)^2 < 0$. This ensures that the optimal solution is identical to the original constrained problem. Since we introduce two δ variables and one z per clause in the dualization, this formulation requires $3|C|$ ancillary qubits. The term $-\lambda \sum_{c \in C} f_c(\{x_i, \delta_{c,j}\}, z_c)^2$ must be expanded in order to determine all of the edges of the derived graph.

The unitary circuit depth for one layer of QAOA is the chromatic index plus one, and the chromatic index is either the maximum degree of the derived graph, or the maximum degree of the derived graph plus one. Thus, the unitary circuit depth is either the maximum degree of the derived graph plus one or the maximum degree of the derived graph plus two.

To compute the maximum degree, let $C_{x_i} \subset C$ be the set of clauses containing x_i . Then the degree of the x_i in the derived graph is

$$\text{deg}(x_i) = 5|C_{x_i}| - \sum_{j, j \neq i} (|C_{x_i, x_j}| - 1)$$

where $C_{x_i, x_j} = C_{x_i} \cap C_{x_j}$ is the set of clauses containing both x_i and x_j . This value lies between $3|C_{x_i}|$ and $5|C_{x_i}|$. The lower bound is because x_i is adjacent to both δ variables and one z per clause and the upper bound is because x_i can also be adjacent to distinct x_j and x_k in each clause. Notice that $\text{deg}(\delta_{i,j}) = 5$ and $\text{deg}(z_c) = 5$. Since $\text{deg}(x_i) \geq 5$ for some i , the maximum degree of the graph is $\max_i \{\text{deg}(x_i)\}$, so the unitary circuit depth for one layer of QAOA is either $\max_i \{\text{deg}(x_i)\} + 1$ or $\max_i \{\text{deg}(x_i)\} + 2$.

2.3.2. SAT Product Formulation

Alternatively, SAT can be written as a product of monomials. To see this, note that $x_i \vee x_j \vee x_k$ is satisfied if $(x_i - 1)(x_j - 1)(x_k - 1) = 0$. Thus, we can write SAT as the polynomial unconstrained binary optimization (PUBO) problem

$$\sum_C \prod_{x_i=TRUE_c} (1 - x_i) \prod_{x_i=FALSE_c} x_i = 0,$$

where x_i indicates if literal x_i is satisfied. There are no ancillary qubits needed to write this PUBO, however expanding it does give monomials in three variables, as seen in Table 1. A straightforward implementation of an n variable monomial requires an n -qubit gate, however, current hardware is often limited to two-qubit gates. We therefore need a method to decompose these large monomials into products of at most two variables.

Table 1. The product reformulation and expansion of 3-SAT clauses. In this chart, we let $x_b x_c = u_{b,c}$. The last three columns count the contribution to the degree of each variable from the monomials in two vertices that are not substituted. This contribution is only added the first time the two variable monomial appears. For example, if $x_a x_b$ appears in more than one clause, we only add one to the degrees of x_a and x_b . Since $x_b x_c = u_{b,c}$, we do not add one to the degrees of x_b and x_c whenever $x_b x_c$ appears in a clause. The number in each degree column is added to the degree from Theorem 2 to calculate the degree of vertex x_i in the derived graph.

Clause	Expansion	deg(x_a)	deg(x_b)	deg(x_c)
$x_a \vee x_b \vee x_c$	$1 - x_a - x_b - x_c + x_a x_c + x_b x_c + x_a x_b - x_a x_b x_c$	2	1	1
$\neg x_a \vee x_b \vee x_c$	$x_a - x_a x_b - x_a x_c + x_a x_b x_c$	2	1	1
$x_a \vee \neg x_b \vee x_c$	$x_b - x_a x_b - x_b x_c + x_a x_b x_c$	1	1	0
$\neg x_a \vee \neg x_b \vee x_c$	$x_a x_b - x_a x_b x_c$	1	1	0
$x_a \vee \neg x_b \vee \neg x_c$	$x_b x_c - x_a x_b x_c$	0	0	0
$\neg x_a \vee \neg x_b \vee \neg x_c$	$x_a x_b x_c$	0	0	0

2.4. Example: Linear and Product Formulations

Consider the 3-SAT problem

Example 1.

$$\begin{aligned}
 &x_1 \vee x_2 \vee \neg x_3 \\
 &x_1 \vee x_3 \vee x_4 \\
 &\neg x_2 \vee x_4 \vee x_5 \\
 &x_1 \vee \neg x_2 \vee x_5.
 \end{aligned}$$

Labeling the indicator variable for the first clause z_1 , the second z_2 , and so on, we can write the objective function for this problem as

$$\max \sum_{c=1}^4 z_c$$

subject to the constraints

$$\begin{aligned}
 &x_1 + x_2 + (1 - x_3) \geq z_1, \\
 &x_1 + x_3 + x_4 \geq z_2, \\
 &(1 - x_2) + x_4 + x_5 \geq z_3, \\
 &x_1 + (1 - x_2) + x_5 \geq z_4, \\
 &x_i, z_c \in \{0, 1\}.
 \end{aligned}$$

We consider the unitary circuit depth for one layer of QAOA to solve this problem. First, we look at the linear characterization found in Section 2.3.1. Then, we look at the product formulation and compare the degrees of the resulting derived graphs to determine the unitary circuit depth for one layer of QAOA from each method, assuming three qubit gates are possible. We next devise a variable-substitution method to use the product formulation in Section 2.5, and analyze the unitary circuit depth for a two-qubit gate implementation of the product formulation in Section 2.5.3.

2.4.1. Linear Formulation

When we use the linear formulation of the above problem, the contrapositive of the simplified dualized constraint for the first clause is

$$-x_1 - x_2 + x_3 - z_1 \leq 0.$$

In order to dualize the constraint, we add in two slack variables, $\delta_{1,1}$ and $\delta_{1,2}$ and change the inequality to equality to obtain

$$-x_1 - x_2 + x_3 - z_1 + \delta_{1,1} + \delta_{1,2} = 0.$$

Upon squaring both sides, we get all terms of the form $x_i x_j$, $x_i z_1$, $x_i \delta_{1,k}$, $z_1 \delta_{1,k}$, and $\delta_{1,1} \delta_{1,2}$ where $i \neq j$, $i, j \in \{1, 2, 3\}$ and $k \in \{1, 2\}$. The other clauses are handled similarly and the derived graph is found in Figure 4. The degree of the vertex that is maximal in the derived graph is thirteen, so the unitary circuit depth is either fourteen or fifteen for one layer of QAOA.

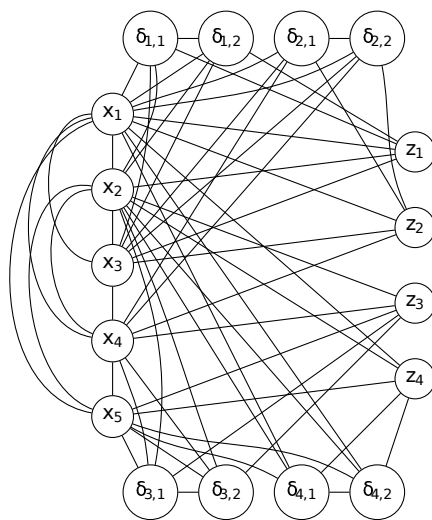


Figure 4. The derived graph for the dualization of the linear formulation of Example 1. The maximum degree of the graph is 13, the chromatic index of this graph is either 13 or 14, and the unitary circuit depth for one layer of QAOA is either 14 or 15.

2.4.2. Product Formulation

Instead of solving the linear formulation, each clause can be rewritten as a product of combinations of x_i and $(1 - x_j)$. For Example 1, the first statement holds if and only if $(1 - x_1)(1 - x_2)x_3$ holds. Upon expanding, we get the expression $x_3 - x_1x_3 - x_2x_3 + x_1x_2x_3$. The other clauses are handled similarly. If there are three-qubit gates, this method requires no ancillary qubits.

Expanding the product formulation of a single 3-SAT clause results in a monomial that contains three variables and possibly one or more monomials in two variables. The variables that occur in degree two monomials are all contained in the three variable monomial. Since each monomial represents a gate that acts on the qubits contained in the monomial, we need only implement the three-qubit gates from each clause. Thus, we can eliminate all edges from the derived graph and are left with a hypergraph, Figure 5. The unitary circuit depth is equal to the chromatic index of the hypergraph plus one.

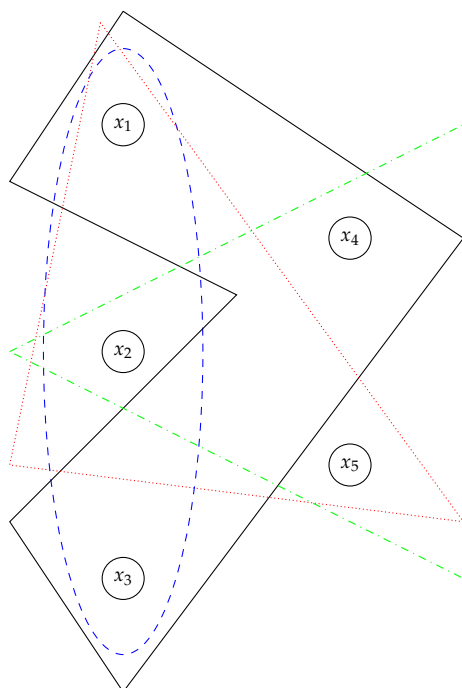


Figure 5. The derived graph for the product formulation of Example 1. There are four hyperedges: $x_1x_2x_3$ in dashed blue, $x_1x_3x_4$ in black, $x_2x_4x_5$ in dashdotted green, and $x_1x_2x_4x_5$ in densely dotted red. The chromatic index of the graph is four and the unitary circuit depth is five.

2.5. Global Variable Substitution

The previous sections assume that three-qubit gates are possible on the hardware, however, that may not always be the case. Thus, we examine how to decompose monomials in three or more variables via a process called global variable substitution (GVS). In order to substitute variables, we require constraints on the problem to ensure the substitution is valid. We then eliminate the constraints via dualization and can determine the unitary circuit depth of one layer of QAOA.

Let us define a substitution of size s as one in which s variables are combined into one, e.g., $x_1x_2\dots x_s = u_{1,2,\dots,s}$. Throughout the remainder of this paper we use the boldface variable \mathbf{j} to refer to lists of indices, $\mathbf{j} = (j_1, j_2, \dots, j_m)$, where $j_i \in \mathbb{N} \forall i \leq m$. A substitution is denoted $u_{\mathbf{j}} = x_{j_1}x_{j_2}\dots x_{j_m}$. When we wish to refer to specific substitutions, we will specify indices of the variables, e.g., $u_{i,j} = x_i x_j$.

The goal of this work is to write an n variable monomial as a product of two variables. One variable replaces s of the variables of the original monomial, and another variable is substituted for the remaining $n - s$. Since the order of the variables in each monomial is not important, we assume without loss of generality that the first s variables are substituted for one variable and the last $n - s$ for the other. In this formulation, we allow no overlap in the substitutions, i.e., if x_i is a variable in the substitution of size s , then it is not a variable in the substitution of size $n - s$.

In order to substitute a new variable $u_{1,\dots,s}$ for $x_1\dots x_s$, we add in the constraints

$$u_{\mathbf{j}} = \sum_{i \in [|\mathbf{u}_{\mathbf{j}}|]} x_i - \sum_{i=1}^{m_{\mathbf{u}_{\mathbf{j}}}} \delta_{c,k} - (|\mathbf{u}_{\mathbf{j}}| - 1) \tag{7}$$

$$u_{\mathbf{j}} = x_i - \delta_{c,m_{\mathbf{u}_{\mathbf{j}}}+k} \quad \forall i \in [|\mathbf{u}_{\mathbf{j}}|] \tag{8}$$

where $|\mathbf{u}_{\mathbf{j}}|$ denotes the number of variables $u_{\mathbf{j}}$ replaces and $m_{\mathbf{u}_{\mathbf{j}}} \in \{1, \dots, s\}$, depending on if x_i takes the value of 0 or 1 for each x_i . The first constraint ensures that if $x_i = 1 \forall i$, then $u_{1,2,\dots,s} = 1$, while the second guarantees that if $x_i = 0$ for some i , then $u_{1,2,\dots,s} = 0$. Note that $|\mathbf{u}_{\mathbf{j}}| = s_j$, so the two can be used interchangeably. After dualizing these constraints,

we are able to determine the degree of each vertex in the derived graph. To facilitate the discussion, in Theorem 1 we focus only on the vertices and edges associated with the dualized substitution terms, which form a sub-graph of the total derived graph. We then show how these are related to the total derived graph and describe how to compute the unitary circuit depth for one layer of QAOA in Section 2.5.2.

2.5.1. Degree of Substituted Variables

We now give a formula for the degrees of vertices $\delta_{c,k}$ and u_j , as well as determine the contribution of each substitution to the degree of x_i . These degrees, in conjunction with the $x_i x_j$ terms, can be used to compute the unitary circuit depth, as shown explicitly in the example of Section 2.5.3.

Theorem 2. *Let U denote the set of all substitutions made. Let us denote the set of substitutions containing x_i as U_i . For $u_j \in U$ we let $|u_j|$ denote the number of variables substituted in substitution u_j . The number of edges incident to each vertex in the derived graph due to the substitution is denoted $\text{deg}_s(v)$ and is*

$$\begin{aligned} \text{deg}_s(\delta_{c,k}) &\in \{2, |u_j| + m_{u_j}\} \\ \text{deg}_s(x_i) &= \sum_{u_j \in U_i} [|u_j| + m_{u_j} + 1] - \sum_{p \in [n] \setminus i} z_{i,p} [|U_i \cap U_p| - 1] + |C'_{x_i}| \\ \text{deg}_s(u_j) &= 2|u_j| + m_{u_j} + |C_{u_j}| \end{aligned}$$

where m_{u_j} denotes the number of $\delta_{c,k}$ variables in Equation (7) for the substitution j , $z_{i,p}$ is an indicator variable denoting whether or not $U_i \cap U_p = \{\emptyset\}$, $|C'_{x_i}|$ refers to the number of monomials containing x_i which do not have any u_s containing x_i substituted, and $|C_{u_j}|$ refers to the number of monomials with more than two variables using the substitution u_j .

Proof. First, we will consider the degree of each $\delta_{c,k}$ by counting the number of terms that contain $\delta_{c,k}$ in each dualized constraint. Note that in the first constraint, there are $1 + |u_j| + m$ variables total, so $|u_j| + m$ of these are multiplied times a single $\delta_{c,k}$ term upon dualization. Thus, the degree of $\delta_{c,k}$ in this constraint is $|u_j| + m$. In each of the following constraints, note that there are three variables: u_j , x_i and $\delta_{c,k}$. Thus, the degree of $\delta_{c,k}$ from these constraints is 2. Since the $\delta_{c,k}$ terms in each constraint are different, they do not add, so $\text{deg}_s \delta_{c,k} \in \{2, |u_j| + m_{u_j}\}$.

Next, we consider the degree of each x_i in the graph derived from the dualization. In the first constraint, there are $|u_j| + m_{u_j}$ terms containing each x_i and there is exactly one constraint aside from the first that contains x_i . This constraint only has two terms containing x_i , but the contribution to the degree from this is one since the edge $u_j x_i$ already exists in the graph from the first constraint. Now, we must consider how many u_j contain x_i as a variable. If there is more than one substitution containing x_i , there are multiple constraints that have the form of the first one. Each of those constraints contain new u_j variables and new $\delta_{c,k}$ variables since the substitutions are different. Thus, the only double counting that can happen is in products of $x_i x_j$, since these are the only other edges possible in the graph derived from the dualization. We must subtract the number of times each of these terms occurs in each constraint except for one. Additionally, we must count the clauses that contain x_i but in which x_i is not contained in a substitution, since it will be incident to the substitution in the derived graph. We denote the number of these clauses as $|C'_{x_i}|$. Then, $\text{deg}_s(x_i) = [\sum_{u_j \in U_i} |u_j| + m_{u_j} + 1] - \sum_{p \in [n] \setminus i} z_{i,p} [|U_i \cap U_p| - 1] + |C'_{x_i}|$.

Finally, we need to consider the degree of each substituted variable, u_j . The first constraint contributes $|u_j| + m$ to the degree and the other $|u_j|$ constraints contribute 1 to the degree since the first constraint accounts for the $u_j x_i$ terms. Finally, each u_j is substituted into a clause, so is multiplied by the variable not contained in the substitution. We denote the number of clauses into which u_j is substituted as $|C_{u_j}|$. Thus, $\text{deg}_s u_j = 2|u_j| + m_{u_j} + |C_{u_j}|$. \square

The derived graph for the problem consists of variables and edges induced by the substitution, as well as variables and edges $x_i x_j$ that exist in the original problem. The variables specific to the substitutions, $\delta_{c,k}$ and u_j , have degrees determined solely by Theorem 2. The chromatic index of the derived graph is the unitary circuit depth for one layer of QAOA, and the maximum of the degrees from Theorem 2 plus one provides a lower bound for this quantity. We show how to compute the exact unitary circuit depth in Section 2.5.2, including the extra edge terms $x_i x_j$.

2.5.2. 3-SAT

Notice that Theorem 2 reduces to

$$\begin{aligned} \deg(\delta_{c,k}) &\in \{2, 3\} \\ \deg(x_i) &= \left[\sum_{u_j \in U_i} 4 \right] + |C'_{x_i}| \\ \deg(u_j) &= 5 + |C_{s_j}| \end{aligned}$$

for 3-SAT since $|u_j| = 2$, $z_{i,p} |U_i \cap U_p| \in \{0, 1\}$, and $m = 1$. This is the degree for each variable due to the substitution. In order to determine the total degree, we need to look at two variable terms in the expansions and add one to the degree for each unique term.

To compute the unitary circuit depth for one layer of QAOA, we must compute the maximum vertex degree in the derived graph. We focus here on the example of 3-SAT. A similar approach can be used to decompose other classes of combinatorial optimization problems. In order to calculate the total degree of a vertex in the derived graph from 3-SAT, we need to add the degrees of a vertex due to a substitution u_j , from Theorem 2, to the degree from two vertex monomials that were not substituted. Since $\delta_{c,k}$ and $u_{i,j}$ are introduced in order to make the substitutions, their degrees are exactly the quantity in Theorem 2.

In order to count the number of edges induced by pairs that are not substituted, we define P as the set of all two variable monomials that result from the expansion of the product formulation of each clause in a 3-SAT instance. These two variable monomials will be denoted $s_{i,j}$. For example, if we have the clauses $(1 - x_1)x_2x_3 = x_2x_3 - x_1x_2x_3$ and $(1 - x_2)x_4(1 - x_5) = x_4 - x_2x_4 - x_4x_5 - 5 + x_2x_4x_5$ and the substitutions $u_{1,3}$ and $u_{4,5}$ are made, in this case, $P = \{x_2x_3, x_2x_4\}$. The subset of P containing a vertex a is denoted P_a . Here, $P_2 = P$, since x_2 is in each monomial, $P_3 = \{x_2x_3\}$, and $P_4 = \{x_2x_4\}$. Let $||P|| = (|P_1|, |P_2|, \dots, |P_n|)$.

These sets can be used to determine the number of edges incident to a vertex x_i that are from the two variable terms in the expansions of each clause, which we denote $\deg_e(x_i)$. Let us fix variable x_i . Since $|P_i|$ counts the number of two variable terms containing x_i , it is added to Theorem 2. If one of the two variable terms from the expansion of the clauses is substituted, i.e., if $x_i x_j \in P_i$ and $u_{i,j}$ is a substitution, then adding $|P_i|$ to the degree double counts the edge $x_i x_j$. Thus, the number of substitutions that appear in P_i need to be subtracted. If S_i is the set of substitutions containing variable x_i and y_s is an indicator variable that determines if the substitution $s_{i,j}$ was used, the total degree of x_i is

$$\deg(x_i) = 4 \sum_{s_{i,j} \in S_i} y_{s_{i,j}} - \sum_{s_{i,j} \in P_i} y_{s_{i,j}} + |P_i| + |C'_{x_i}| = \deg_s(x_i) + \deg_e(x_i),$$

where $\deg_e(x_i) = - \sum_{s_{i,j} \in P_i} y_{s_{i,j}} + |P_i|$.

The QAOA unitary circuit depth is then given as $\max\{\deg(\delta_{c,k}), \deg(x_i), \deg(u_{i,j})\}$, where

$$\deg(\delta_{c,k}) \in \{2, 3\} \tag{9}$$

$$\deg(x_i) = 4 \sum_{s_{i,j} \in S_i} y_{s_{i,j}} - \sum_{s_{i,j} \in P_i} y_{s_{i,j}} + |P_i| + |C'_{x_i}| \tag{10}$$

$$\deg(u_{i,j}) = 5 + |C_{u_{i,j}}|, \tag{11}$$

for 3-SAT. Note that $\delta_{c,k} \in \{2, 3\}$ since the size of each substitution $|u_{i,j}| = 2$ and $m = n - |u_{i,j}| = 3 - 2 = 1$, which also simplifies $\deg(u_{i,j})$. Furthermore, note that $U_i \cap U_p \in \{0, 1\} \forall i \neq p, i, p \in [n]$, which simplifies $\deg(x_i)$.

2.5.3. Example: Global Variable Substitution

To give an example of the GVS method, we will apply it to the product formulation of Example 1. We want to decompose the three-variable terms into two-variable terms by substituting a new variable to represent the product of the variables. In order to do so, we first choose substitutions and then create the derived graph. For Example 1, we substitute $x_1x_3 = u_{1,3}$ and $x_2x_5 = u_{2,5}$. Thus, the first clause, which can be written as $x_3 - x_2x_3 - x_1x_3 + x_1x_2x_3$ is equal to $x_3 - x_2x_3 - u_{1,3} + u_{1,3}x_2$. With the substitution $x_1x_3 = u_{1,3}$, the edge x_1x_3 is already accounted for in the first constraint. We then need only add the edge between x_2 and x_3 to the derived graph since the monomial x_2x_3 exists in the expansion of the first clause. Note that each clause can be decomposed similarly and will end up with a sum of terms, at least one of which contains three variables, with the others containing one or two depending on the number of $(1 - x_i)$ expressions. The other edges that need to be added due to the two variable terms in the expansions are x_1x_4, x_3x_4, x_1x_2 and x_2x_4 . See Table 1 for the contribution of unsubstituted monomials from a particular clause to $\deg_e(x_i)$ for a generic substitution $x_bx_c = u_{b,c}$. The increase in degree ranges from 0 to 2, with the maximum addition being $k - 1$ for general k -SAT.

Next, we list each constraint of the form Equations (7) and (8) and dualize them. For the first clause $c = 1$ in the example,

$$u_{1,3} = x_1 + x_3 + \delta_{1,1}$$

$$u_{1,3} = x_1 + \delta_{1,2}$$

$$u_{1,3} = x_3 + \delta_{1,3}.$$

When dualizing these constraints, the terms

$$(u_{1,3} - x_1 - x_3 - \delta_{1,1})^2 = u_{1,3} + x_1 + x_3 + \delta_{1,1} - u_{1,3}x_1 - u_{1,3}x_3 - u_{1,3}\delta_{1,1} + x_1\delta_{1,1} + x_3\delta_{1,1} + x_1x_3 \tag{12}$$

$$(u_{1,3} - x_1 - \delta_{1,2})^2 = u_{1,3} + x_1 + \delta_{1,2} - u_{1,3}x_1 - u_{1,3}\delta_{1,2} + x_1\delta_{1,2} \tag{13}$$

$$(u_{1,3} - x_3 - \delta_{1,3})^2 = u_{1,3} + x_3 + \delta_{1,3} - u_{1,3}x_3 - u_{1,3}\delta_{1,3} + x_3\delta_{1,3} \tag{14}$$

are added to the objective function, up to constant λ , and similar terms are added for the $u_{2,5}$ substitution. Since all variables, v , in the equations above have the value zero or one, $v^2 = v$. The entire derived graph can be seen in Figure 6. Note that the degrees match the theorem plus the number of edges induced by pairs that are not substituted. The largest degree vertex is x_2 , which has degree eight, so the unitary circuit depth for one layer of QAOA is either nine or ten. This is less than the unitary circuit depth for the linear 3-SAT formulation and requires fewer ancillary qubits.

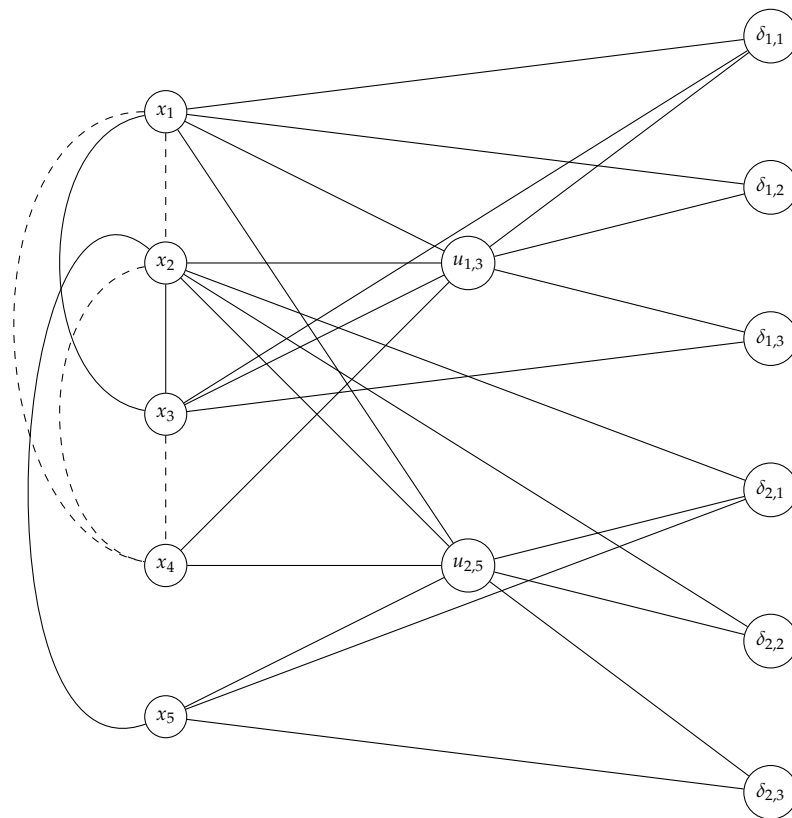


Figure 6. The derived graph for solving Example 1 using GVS with substitutions $x_1x_3 = u_{1,3}$ and $x_2x_5 = u_{2,5}$. The vertices of this graph are the variables from the problem, x_i for $i \in [5]$, along with the substitution variables, $u_{1,3}$ and $u_{2,5}$, and the three $\delta_{c,k}$ variables needed per substitution. The solid edges correspond to the edges induced by the substitution, which correspond to products of two monomials in the Equations (12)–(14). The dashed edges represent the two variable terms from the expansion of the 3-SAT clauses that are not involved in substitutions. The maximum degree of this graph is 8, thus the unitary circuit depth for one layer of QAOA is 9 or 10.

2.6. Optimizing Global Variable Substitutions for 3-SAT

As seen above, the number of substitutions impacts the degrees of the vertices. We want to minimize both the number of ancillary qubits and the maximum degree of the derived graph. This problem is difficult for large gates, but in this section, we explore how to solve this problem with gates with three or fewer variables such as in 3-SAT.

2.6.1. Global Variable Substitution Integer Program

Each substitution introduces new variables that correspond to ancillary qubits and impacts the degree of each vertex in the derived graph. In order to determine which of the possible combinations of substitutions are feasible, we create a graph $G = (V(G), E(G))$, which we call the covering graph. There are two sets of vertices in this graph. One set is the set of all 2-sets representing all possible two-variable substitutions $u_{i,j} = x_i x_j$. We call this set of vertices S and denote the pairs $x_i x_j$ in S as $s_{i,j}$. Each $s_{i,j}$ is a possible substitution, but not each $s_{i,j}$ will result in a substitution $u_{i,j}$. The other set is the one containing all three variable terms from the expansion, which we will call the expansion 3-set, ES_3 . Edges are placed between vertices g and h if the variables in the label of g are a proper subset of the variables in the label of h . There are no edges between any sets of the same size, making the graph bipartite. See Figure 7 for an example, where S is the left set of vertices and ES_3 is the right set for Example 1.

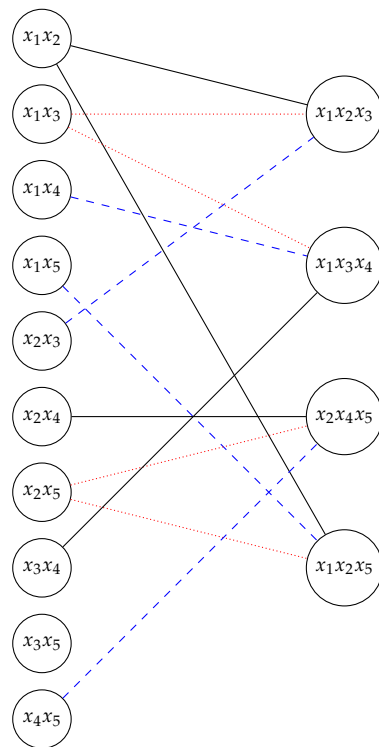


Figure 7. The covering graph for Example 1. We have emphasized all three different coverings by coloring the edges differently for each covering. They are the red dotted, blue dashed, and black solid edges.

The nature of this problem naturally lends itself to a set covering or bipartite matching style formulation as each substitution $u_{i,j}$ can cover, or be matched with, any clause containing the literals in that substitution. We have chosen to use a set covering problem formulation, i.e., we want to find a subset of S such that each vertex in ES_3 is incident to a vertex in the subset. Since covering a vertex in ES_3 more than once leads to unnecessary ancillary qubits and increases the degree, we need to add constraints to ensure that each clause $c \in C$ is covered by exactly one substitution $u_{i,j}$. Notice that more than one covering may be possible as seen in Figure 7.

The objective of 3-SAT is to minimize the maximum degrees found in Equations (9) and (11),

$$\min \max\{\deg(\delta_{c,k}), \deg(x_i), \deg(u_{i,j})\}$$

which are derived from the GVS method. The covering graph developed in the previous paragraphs is used to aid in the minimization process. The integer program formulation of this problem is subject to the set covering constraints. Each clause $x_i x_j x_k = c$ is covered by a pair $s_{i,j} = x_i x_j$ and has a variable x_k that is not in a substitution. The indicator variable $z(c, s_{i,j}, x_k)$ is defined as

$$z(c, s_{i,j}, x_k) = \begin{cases} 1 & \text{if covering } (s_{i,j}, x_k) \text{ is selected to cover clause } c \\ 0 & \text{else} \end{cases}$$

The full integer program formulation to minimize the maximum degree vertex in the derived graph, obj , is as follows:

$$\min obj + \frac{1}{10|C|} * \sum_{s \in S} y_s \tag{15}$$

$$s.t. \ 4 \sum_{s_{i,j} \in S_a} y_s - \sum_{s_{i,j} \in P_a} y_{s_{i,j}} + |P_a| + \sum_{c \in C} \sum_{\{s_{i,j} \in S \mid a \notin s_{i,j}\}} z(c, s_{i,j}, x_a) \leq obj \quad \forall a \in V \tag{16}$$

$$5 + \sum_{c \in C} z(c, s_{i,j}, x_k) \leq obj \quad \forall s \in S \tag{17}$$

$$\sum_{s \in S} z(c, s_{i,j}, x_k) = 1 \quad \forall c \in C \tag{18}$$

$$\sum_{c \in C} z(c, s_{i,j}, x_k) - y_{s_{i,j}} \leq 0 \quad \forall s \in S \tag{19}$$

$$obj \in \mathbb{Z}^+ \tag{20}$$

$$y_{s_{i,j}}, z(c, s_{i,j}, x_k) \in \{0, 1\}. \tag{21}$$

Equation (15) is our objective function. We have added the penalty $\frac{1}{10|C|} * \sum_{s_{i,j} \in S} y_{s_{i,j}}$ in Equation (15) which minimizes the total number of unique substitutions to help minimize the number of ancillary qubits. Equation (16) indicates that the maximum degree of each vertex must be less than or equal to the objective value. The last term of this constraint counts the number of clauses that contain variable x_a but in which x_a is not substituted. This is equivalent to $|C'_{x_a}|$ in Theorem 2 and the example. Similarly, Equation (17) indicates that the maximum degree of any substituted variable must be less than or equal to the objective value. These two constraints effectively allow us to minimize the maximum degree of the graph. Note, the degree of the slack variables are not included in this formulation as they are either 2 or 3 and will never yield the maximum degree in a problem of sufficient size. Equation (18) is our set covering constraint. This asserts that each $c \in C$ is required to be covered by exactly one covering $s_{i,j}, x_k$. Equation (19) asserts that if at least one $c \in C$ substitutes the pair $s_{i,j}, y_{s_{i,j}}$ is set to 1.

2.6.2. IP Formulation and Solution for Example 1

Let us examine Example 1,

$$\begin{aligned} x_1 \vee x_2 \vee \sim x_3 \\ x_1 \vee x_3 \vee x_4 \\ \sim x_2 \vee x_4 \vee x_5 \\ x_1 \vee \sim x_2 \vee x_5. \end{aligned}$$

The reformulation and expansion of each of these four 3-SAT clauses yields a three variable monomial c , two variable monomials $s_{i,j} \in P$, and potential coverings $(s_{i,j}, x_k)$. The first statement holds if and only if $(1 - x_1)(1 - x_2)x_3 = x_3 - x_1x_3 - x_2x_3 + x_1x_2x_3$ holds. The other clauses, when expanded, give expressions $1 + x_1x_4 + x_3x_4 - x_1x_3x_4$, $x_2 - x_2x_4 - x_2x_5 + x_2x_4x_5$, and $x_2 - x_1x_2 - x_2x_5 + x_1x_2x_5$. The two variable terms are $s_{i,j} \in P$. Each $c \in C$ has $\binom{3}{2}$ pairs that can cover it. In this example, we have five variables and four clauses. A summary of the clauses, their corresponding P and potential covering are found in Table 2.

Table 2. This table shows the result of the reformulation of each 3-SAT clause including the three variable monomials to be covered c , the two variable monomial pairs from the expansion of each clause $s \in P$, and each of the three potential coverings of c , $(s_{i,j}, x_k)$.

	$c(x_i x_j x_k)$	$s_{i,j} \in P$	Covering 1	Covering 2	Covering 3
c0	$x_1 x_2 x_3$	$s_{1,3}, s_{2,3}$	$(s_{1,2}, x_3)$	$(s_{1,3}, x_2)$	$(s_{2,3}, x_1)$
c1	$x_1 x_3 x_4$	$s_{1,3}, s_{1,4}, s_{3,4}$	$(s_{1,3}, x_4)$	$(s_{1,4}, x_3)$	$(s_{3,4}, x_1)$
c2	$x_2 x_4 x_5$	$s_{2,4}, s_{2,5}$	$(s_{2,4}, x_5)$	$(s_{2,5}, x_4)$	$(s_{4,5}, x_2)$
c3	$x_1 x_2 x_5$	$s_{1,2}, s_{2,5}$	$(s_{1,2}, x_5)$	$(s_{1,5}, x_2)$	$(s_{2,5}, x_1)$

$$S = \{s_{1,2}, s_{1,3}, s_{1,4}, s_{1,5}, s_{2,3}, s_{2,4}, s_{2,5}, s_{3,4}, s_{4,5}\}$$

$$P = \{s_{1,2}, s_{1,3}, s_{1,4}, s_{2,3}, s_{2,4}, s_{2,5}, s_{3,4}\}$$

$$||P|| = (3, 4, 3, 3, 1).$$

After determining these sets, we are able to formulate our constraints. First we add a constraint for each literal vertex $v_a \in V$ in the form of Equation (16). For $a = 1$:

$$\begin{aligned} \deg(x_1) = & 3y_{s_{1,2}} + 3y_{s_{1,3}} + 3y_{s_{1,4}} + 4y_{s_{1,5}} + z(c_0, s_{2,3}, x_1) + z(c_1, s_{3,4}, x_1) + z(c_3, s_{2,5}, x_1) \\ & + 3 \leq obj. \end{aligned}$$

Next, we add a constraint for each unique substitution u_s corresponding to an $s \in S$ in the form of Equation (17). For $s = (x_1, x_2)$:

$$\deg(u_{1,2}) = 5 + z(c_0, s_{1,2}, x_3) + z(c_3, s_{1,2}, x_5) \leq obj.$$

For each $c \in C$ we add a constraint in the form of Equation (18) to assert that a clause c must be covered by exactly one of its three potential coverings found in Table 2. For clause c_0 :

$$z(c_0, s_{1,2}, x_3) + z(c_0, s_{1,3}, x_2) + z(c_0, s_{2,3}, x_1) = 1.$$

Finally, for each covering containing a pair $s_{i,j}$, we add a constraint in the form of Equation (19). This asserts that if any covering containing $s_{i,j}$ is selected for a substitution, $x_{s_{i,j}}$ is forced to 1. For $s = (x_1, x_2)$ we add the constraints

$$z(c_0, s_{1,2}, x_3) - y_{s_{1,2}} \leq 0$$

$$z(c_3, s_{1,2}, x_5) - y_{s_{1,2}} \leq 0.$$

The solution to this IP is

$$y_{1,3}, y_{2,5} = 1$$

$$z(c_0, s_{1,3}, x_2), z(c_1, s_{1,3}, x_4), z(c_2, s_{2,5}, x_4), z(c_3, s_{2,5}, x_1) = 1,$$

which indicates that the optimal solution is to substitute pairs $s_{1,3}$ and $s_{2,5}$. Using these substitutions, $\Delta_G = 8$ and the degree of each vertex in the derived graph is

$$\deg(x_1) = 7 \quad \deg(x_2) = 8 \quad \deg(x_3) = 6 \quad \deg(x_4) = 5$$

$$\deg(x_5) = 4 \quad \deg(u_{1,3}) = 7 \quad \deg(u_{2,5}) = 7$$

so the unitary circuit depth is at most nine.

2.6.3. Heuristic Approximation

The previous integer programming approach will provide an optimal solution. However, depending on the 3-SAT instance, it might be a very difficult problem to solve. As an alternative, we have developed a heuristic to approximate it. Greedy algorithms are a common way to approximate large scale set covering problems [26]. Generally, this involves a greedy selection of elements that cover the greatest amount of sets until all sets are covered. We have chosen to use a similar approach as the degree of the graph derived using GVS is directly impacted by the number of substitutions made. Therefore, we expect by selecting the coverings with the highest degree in the covering graph, we will be able to minimize the total number of substitutions made and obtain a locally minimal solution. Let U be the set of uncovered clauses. Let K be the set of covered clauses. At the beginning of the iteration, K is empty and $U = C$, where C is the set of all clauses in the 3-SAT instance. The greedy algorithm is described in Algorithm 1. Often in step 2, multiple $s_{i,j}$ pairs may cover the same number of clauses. To break these ties, we randomly select a pair $s_{i,j}$ of the current largest degree in the covering graph.

Algorithm 1 Greedy by Covering Heuristic

```

while  $|U| > 0$  do
  select  $s_{i,j} \in S$  which covers the the greatest number of  $u \in U$ 
  add all covered  $u$  to  $K$ 
  remove all covered  $u$  from  $U$ 
  remove  $s_{i,j}$  from  $S$ 

```

3. Results

In this section, we evaluate the performance of the product 3-SAT model and covering heuristic formulated to apply the global variable substitution method to large 3-SAT instances. Since the unitary circuit depth for one iteration of QAOA is the maximum degree plus one or the maximum degree plus two, in all cases, we take the unitary circuit depth to be the maximum degree plus two since is the upper bound of the unitary circuit depth for one iteration. Thus, we compare the upper bound of the unitary circuit depth for one iteration achieved using the integer programming model and covering heuristic, which we denote Δ'_{IP} and Δ'_C , respectively, to the upper bound of the unitary circuit depth, denoted Δ'_L , calculated using the linear model described in Section 2.3.1. We apply each formulation and method to 3-SAT problem instances from the SATLIB Benchmark Suite developed by Holger Hoos and Thomas Stütze [19]. This suite is composed of thousands of SAT instances of varying families and sizes. We choose the first instance of each problem set to evaluate.

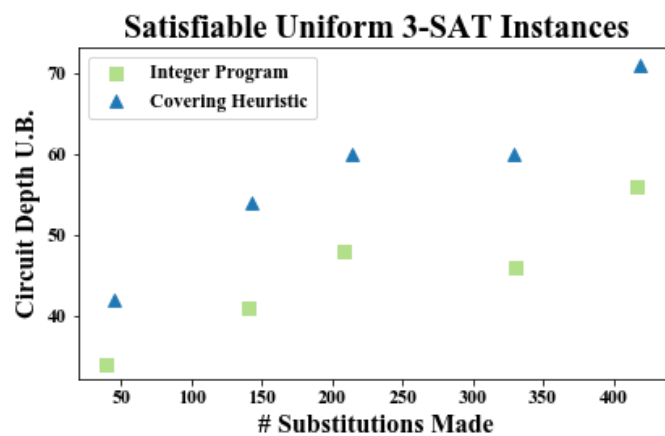
The results in Table 3 indicate that the graphs derived using the product formulation combined with the GVS method result in significantly lower unitary circuit depth for one iteration of QAOA than the graphs derived from the linear formulation. For every problem instance, $\Delta'_{IP} < 1/2\Delta_L$. The heuristic does not yield as large of a reduction, but most instances see a significant reduction in degree compared to the linear method. It is interesting to note that as the size of each 3-SAT instance increases, the Δ of the derived graph does not increase significantly. We can attribute this to the uniformity in the ratio of the literals to clauses and the uniformity of the distribution of those literals amongst all problem instances.

In nearly every 3-SAT instance, the GVS integer program makes more than or approximately equal to the amount of substitutions made by the covering heuristic method. However, the maximum degree of the IP derived graphs Δ_{IP} are significantly lower than Δ'_C . This trend can be seen in Figure 8a,b which plot the maximum degree of the derived graph against the number of substitutions made for the uniform 3-SAT instances. This seems to indicate that simply minimizing the number of substitutions does not necessarily minimize the maximum degree of the GVS derived graph. In particular the uf50-218 instance from [19], which is a satisfiable instance with 50 variables and 218 clauses, achieves

$\Delta'_{IP} = 41$, which is accomplished by making 141 substitutions. The covering heuristic makes three more substitutions than the LP, but produces $\Delta'_C = 54$. The unsatisfiable instance of the same size, uuf50-218, achieves $\Delta'_{IP} = 41$ by making 138 substitutions. The covering heuristic in this instance made only six more substitutions, but resulted in a $\Delta'_C = 50$. To investigate this trend, we plot the distribution of the degrees of each vertex in the derived graph for each method.

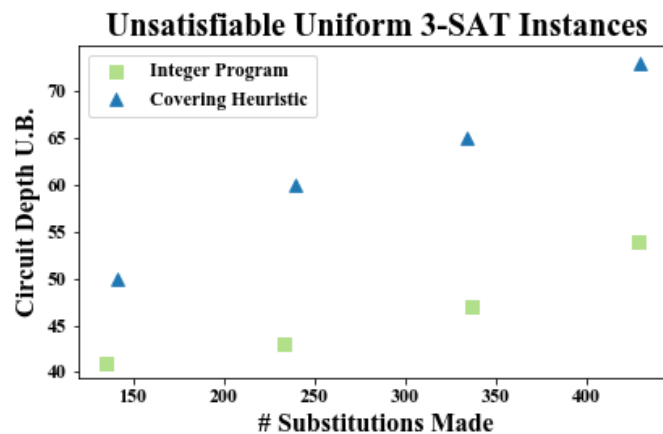
Table 3. The unitary circuit depth upper bound for each SATLIB 3-SAT instance using the linear formulation and product formulation with GVS methods. The IP and Covering Heuristic columns also displays the number of substitutions made to cover every $c \in C$.

Problem Name	Linear Formulation	IP Solution	Covering Heuristic
	Δ'_L	$(\Delta'_{IP}, \#Subs)$	$(\Delta'_C, \#Subs)$
uf20-91	84	(34, 39)	(42, 45)
uf50-218	100	(41, 141)	(54, 143)
uf75-325	115	(48, 208)	(60, 214)
uf100-430	103	(46, 330)	(60, 329)
uf125-538	133	(56, 417)	(71, 419)
uuf50-218	95	(41, 135)	(50, 141)
uuf75-325	106	(43, 233)	(60, 240)
uuf100-430	105	(47, 337)	(65, 334)
uuf125-538	126	(54, 429)	(73, 430)
RTI_k3_n100_m429	132	(61, 332)	(75, 334)
BMS_k3_n100_m289	110	(53, 240)	(63, 236)
CBS_k3_n100_m403_b10	94	(43, 315)	(56, 309)
CBS_k3_n100_m403_b30	94	(45, 318)	(60, 309)
CBS_k3_n100_m403_b50	96	(43, 317)	(61, 319)



(a)

Figure 8. Cont.

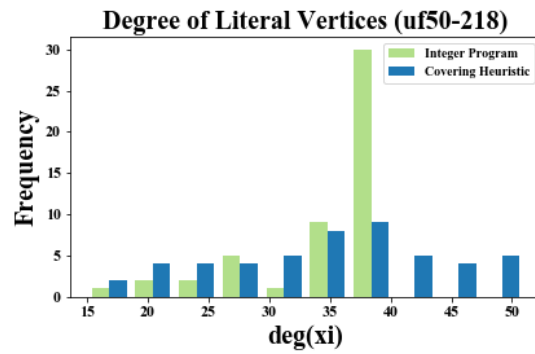


(b)

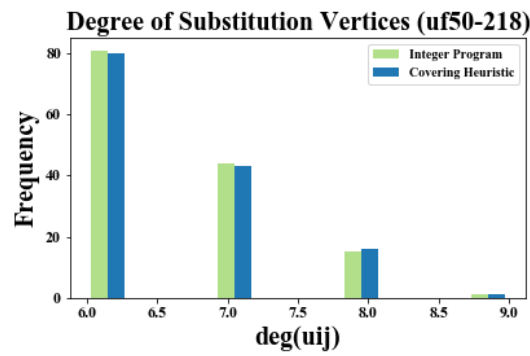
Figure 8. Plots of the upper bound of the maximum degree of the derived graph versus the number of substitutions made for the uniform 3-SAT instances found in Table 3. The satisfiable instances plotted are denoted ‘ $ufn-|C|$ ’. The unsatisfiable instances are denoted ‘ $uufn-|C|$ ’. These instance names indicate the number of literals n and clauses C in each problem instance. We choose the upper bound of the maximum degree to be the y-axis since the largest unitary circuit depth for one iteration is the maximum degree plus two. Calculating the exact unitary depth requires finding the edge chromatic number of each graph, which is NP-hard, and results in a number that differs from the upper bound by at most one. (a) Plot of the maximum degree of the derived graph versus the number of substitutions made for the satisfiable uniform 3-SAT instances in Table 3. (b) Plot of the maximum degree of the derived graph versus the number of substitutions made for the unsatisfiable uniform 3-SAT instances found in Table 3.

As shown in Figure 9a–d, the degrees of the literal vertices v_i are larger than those of the substitution vertices $u_{i,j}$ for each 3-SAT instance we evaluated. We can attribute this to the GVS method of determining the degree for each vertex in the derived graph and the sparseness of the literals in each instance. Each unique pair $s_{i,j}$ which is substituted will add three or four edges to the degree of the literal vertices v_i and v_j . However, making this substitution only adds one edge to the corresponding substitution vertex $u_{i,j}$. In both instances displayed here, a literal x_i is included in approximately thirteen clauses. Clearly, this severely limits the amount of clauses each substitution is able to cover. Consequently, this significantly increases the degree of each literal vertex as more unique substitutions are required to cover all clauses and simultaneously limits the degree of each substitution vertex $u_{i,j}$ since each $s_{i,j}$ is used very few times. We can see this represented in Figure 9b,d as nearly 40% of the substitutions made in both instances only cover one clause. The most clauses covered by a any substitution is five.

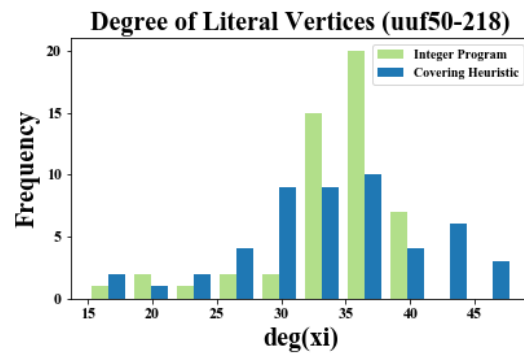
The distribution of literal vertices of the integer program differs significantly from the distribution of the heuristics in both problem instances. A majority of the literal vertices in the graph derived from the IP take on the value of Δ'_{IP} . It is clear that the integer program is not simply minimizing the number of substitutions made, but rather appears to limit the amount of substitutions per literal x_i . For these sparse and uniform 3-SAT instances, the best method of minimizing the max degree of any v_i is to attempt to distribute the number of substitutions made evenly amongst all literal vertices v_i .



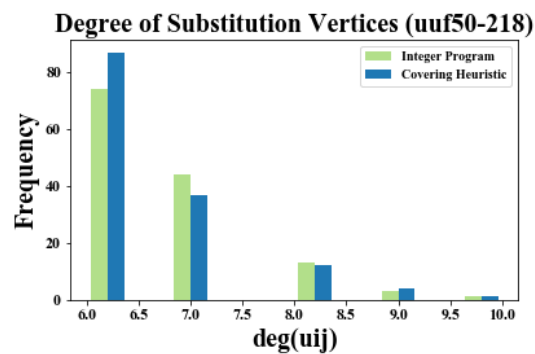
(a)



(b)



(c)



(d)

Figure 9. Histograms of the degrees of each literal vertex or substitution vertex for specific uniform 3-SAT problems. We plot the degrees of each vertex since the unitary circuit depth for one iteration is either the maximum degree plus one or the maximum degree plus two.

4. Discussion

In this paper, we analyze an approach to minimizing the unitary circuit depth of the quantum approximate optimization algorithm by expressing general combinatorial optimization problems in varying forms. We compare a linear formulation that is the natural choice in conventional optimization algorithms with a product formulation that we posit as natural for QAOA. The product formulation leads to monomials in more than two variables, which cannot be directly implemented on a quantum computer with two qubit gates. Thus, we introduce the global variable substitution method to decompose them into two variable terms which can be implemented on a quantum computer with two qubit gates. For each of these formulations, we analytically compute the unitary circuit depth in terms of the maximum degree of a graph derived from the problem instance and formulation. We demonstrate that the product formulation gives shallower circuits than the linear formulation for benchmark 3-SAT problems.

The global variable substitution requires constraints that must be satisfied in order to obtain the optimal solution, as does the linear formulation. We can derive graphs for the linear and product formulations from the objective function and the appropriate constraints. The unitary circuit depth is directly related to the maximum degree of the derived graph.

We evaluate the unitary circuit depth of the product formulation with global variable substitutions by writing an integer program that computes the minimal unitary circuit depth of the linear formulation and product formulation for a collection of benchmark problems. In all cases, the product formulation gives unitary circuit depth roughly half that of the linear formulation. The linear formulation for 3-SAT requires exactly three ancillary qubits per clause, where the product formulation requires four per substitution, although substitutions can sometimes be reused to reduce the number of ancillary qubits. We find several additional interesting features of the approach.

We find that minimizing the number of substitutions per problem instance does not necessarily minimize the maximum degree. For example, when solving “uf-100-430”, the covering heuristic makes 329 substitutions for a maximum degree of 60, whereas the IP makes 330 substitutions for a maximum degree of 46. We also note that the objective function for the IP can be modified to limit the number of substitutions. While this may drive up the degree of vertices, it also reduces the number of ancillary qubits, as each substitution requires four additional qubits. Thus, the problem formulation can be changed to accommodate different hardware.

The focus of this work has been on using the product formulation of 3-SAT instances to minimize QAOA unitary circuit depth relative to a conventional linear formulation. Extending the analysis of linear and product formulations to more general problems will help determine additional types of problems that benefit from this approach. Additionally, there may be other formulations for specific problems that result in shallower circuits than the linear or product formulations. While the product formulation with GVS gives shallower circuits for 3-SAT, future work should determine if the reformulation gives a comparable outcome to the linear formulation in the same number of QAOA iterations. A final note is that the global variable substitution method can be used to rewrite problems in terms of gates acting on m qubits. If more general gates become available on quantum computers, then a similar analysis could lead to new approaches for minimizing depth.

Author Contributions: Conceptualization, R.H., L.T., J.O., P.C.L., T.S.H. and G.S.; methodology, R.H., L.T. and J.O.; software, R.H., L.T. and J.O.; validation, R.H., L.T., J.O., P.C.L., T.S.H. and G.S.; formal analysis, R.H., L.T., J.O., P.C.L., T.S.H. and G.S.; investigation, R.H., L.T., J.O., P.C.L., T.S.H. and G.S.; resources, R.H., L.T., J.O., P.C.L., T.S.H. and G.S.; data curation, R.H., L.T., J.O., P.C.L., T.S.H. and G.S.; writing—original draft preparation, R.H., L.T. and J.O.; writing—review and editing, R.H., L.T., J.O., P.C.L., T.S.H. and G.S.; visualization, R.H., L.T., J.O., P.C.L., T.S.H. and G.S.; funding acquisition, J.O., T.S.H. and G.S. All authors have read and agreed to the published version of the manuscript.

Funding: 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.

G. Siopsis acknowledges the Army Research Office award W911NF-19-1-0397. J. Ostrowski and G. Siopsis acknowledge the National Science Foundation award OMA-1937008.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Farhi, E.; Goldstone, J.; Gutmann, S. A quantum approximate optimization algorithm. *arXiv* **2014**, arXiv:1411.4028.
2. Farhi, E.; Goldstone, J.; Gutmann, S. A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem. *arXiv* **2014**, arXiv:1412.6062.
3. Lotshaw, P.C.; Humble, T.S.; Herrman, R.; Ostrowski, J.; Siopsis, G. Empirical Performance Bounds for Quantum Approximate Optimization. *arXiv* **2021**, arXiv:2102.06813.
4. Herrman, R.; Treffert, L.; Ostrowski, J.; Lotshaw, P.C.; Humble, T.S.; Siopsis, G. Impact of Graph Structures for QAOA on MaxCut. *Quantum Inf. Process.* **2021**, *20*, 1–12. [[CrossRef](#)]
5. Saleem, Z.H. Max-independent set and the quantum alternating operator ansatz. *Int. J. Quantum Inform.* **2020**, *18*, 2050011. [[CrossRef](#)]
6. Wang, Z.; Hadfield, S.; Jiang, Z.; Rieffel, E.G. Quantum approximate optimization algorithm for MaxCut: A fermionic view. *Phys. Rev. A* **2018**, *97*, 022304. [[CrossRef](#)]
7. Crooks, G.E. Performance of the quantum approximate optimization algorithm on the maximum cut problem. *arXiv* **2018**, arXiv:1811.08419.
8. Guerreschi, G.G.; Matsuura, A.Y. QAOA for Max-Cut requires hundreds of qubits for quantum speed-up. *Sci. Rep.* **2019**, *9*, 6903. [[CrossRef](#)]
9. Cook, J.; Eidenbenz, S.; Bärttschi, A. The quantum alternating operator ansatz on max-k vertex cover. *Bull. Am. Phys. Soc.* **2020**, *165*.
10. Ryan, C.A.; Johnson, B.R.; Ristè, D.; Donovan, B.; Ohki, T.A. Hardware for dynamic quantum computing. *Rev. Sci. Instrum.* **2017**, *88*, 104703. [[CrossRef](#)] [[PubMed](#)]
11. Linke, N.M.; Maslov, D.; Roetteler, M.; Debnath, S.; Figgatt, C.; Landsman, K.A.; Wright, K.; Monroe, C. Experimental comparison of two quantum computing architectures. *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 3305–3310. [[CrossRef](#)] [[PubMed](#)]
12. Liu, X.; Angone, A.; Shaydulin, R.; Safro, I.; Alexeev, Y.; Cincio, L. Layer VQE: A Variational Approach for Combinatorial Optimization on Noisy Quantum Computers. *arXiv* **2021**, arXiv:2102.05566.
13. Guerreschi, G.G. Solving Quadratic Unconstrained Binary Optimization with divide-and-conquer and quantum algorithms. *arXiv* **2021**, arXiv:2101.07813.
14. Herrman, R.; Ostrowski, J.; Humble, T.S.; Siopsis, G. Lower bounds on circuit depth of the quantum approximate optimization algorithm. *Quantum Inf. Process.* **2021**, *20*, 59. [[CrossRef](#)]
15. Vizing, V.G. On an estimate of the chromatic class of a p-graph. *Discret. Anal.* **1964**, *3*, 25–30.
16. Tovey, C.A. A simplified NP-complete satisfiability problem. *Discret. Appl. Math.* **1984**, *8*, 85–89. [[CrossRef](#)]
17. Marques-Silva, J. Practical applications of boolean satisfiability. In Proceedings of the 2008 9th International Workshop on Discrete Event Systems, Gothenburg, Sweden, 28–30 May 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 74–80.
18. Hill, A.D.; Hodson, M.J.; Didier, N.; Reagor, M.J. Realization of arbitrary doubly-controlled quantum phase gates. *arXiv* **2021**, arXiv:2108.01652.
19. Hoos, H.; Stützel, T. SATLIB: An online resource for research on SAT. *Sat* **2000**, *2000*, 283–292.
20. Farhi, E.; Harrow, A.W. Quantum supremacy through the quantum approximate optimization algorithm. *arXiv* **2019**, arXiv:1602.07674.
21. Xue, C.; Chen, Z.Y.; Wu, Y.C.; Guo, G.P. Effects of quantum noise on quantum approximate optimization algorithm. *Chin. Phys. Lett.* **2021**, *38*, 030302. [[CrossRef](#)]
22. Wang, S.; Fontana, E.; Cerezo, M.; Sharma, K.; Sone, A.; Cincio, L.; Coles, P.J. Noise-Induced Barren Plateaus in Variational Quantum Algorithms. *arXiv* **2020**, arXiv:2007.14384.
23. Marshall, J.; Wudarski, F.; Hadfield, S.; Hogg, T. Characterizing local noise in QAOA circuits. *IOP SciNotes* **2020**, *1*, 025208. [[CrossRef](#)]
24. Karp, R.M. Reducibility among combinatorial problems. In *Complexity of Computer Computations*; Springer: Berlin/Heidelberg, Germany, 1972; pp. 85–103.
25. Rockafellar, R.T. *Convex Analysis*; Princeton University Press: Princeton, NJ, USA, 2015.
26. Grossman, T.; Wool, A. Computational experience with approximation algorithms for the set covering problem. *Eur. J. Oper. Res.* **1997**, *101*, 81–92. [[CrossRef](#)]