An Efficient QAOA via a Polynomial QPU-Needless Approach

Francisco Chicano
chicano@uma.es
ITIS Software, University of Malaga
Malaga, Spain

Zakaria Abdelmoiz Dahi
zakaria.dahi@uma.es
ITIS Software, University of Malaga
Malaga, Spain

Gabriel Luque
gluque@uma.es
ITIS Software, University of Malaga
Malaga, Spain

ABSTRACT

The Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum algorithm described as ansatzes that represent both the problem and the mixer Hamiltonians. Both are parameterizable unitary transformations executed on a quantum machine/simulator and whose parameters are iteratively optimized using a classical device to optimize the problem's expectation value. To do so, in each QAOA iteration, most of the literature uses a quantum machine/simulator to measure the QAOA outcomes. However, this poses a severe bottleneck considering that quantum machines are hardly constrained (e.g. long queuing, limited qubits, etc.), likewise, quantum simulation also induces exponentially-increasing memory usage when dealing with large problems requiring more qubits. These limitations make today's QAOA implementation impractical since it is hard to obtain good solutions with a reasonably-acceptable time/resources. Considering these facts, this work presents a new approach with two main contributions, including `(I)` removing the need for accessing quantum devices or large-sized classical machines during the QAOA optimization phase, and `(II)` ensuring that when dealing with some k-bounded pseudo-Boolean problems, optimizing the exact problem's expectation value can be done in polynomial time using a classical computer.

CCS CONCEPTS

• **Mathematics of computing** → **Combinatorial optimization**;
• **Computer systems organization** → **Quantum computing**.

KEYWORDS

Quantum Approximate Optimization Algorithm, Combinatorial Optimization, Quantum Computing

ACM Reference Format:
Francisco Chicano, Zakaria Abdelmoiz Dahi, and Gabriel Luque. 2023. An Efficient QAOA via a Polynomial QPU-Needless Approach. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3583133.3596409

1 INTRODUCTION

Quantum Computing (QC) is based on including peculiar principles from quantum mechanics (e.g. superposition, entanglement, etc.) in classical computation [14]. Several QC paradigms exist, but the two main ones are the discrete variable gate-based model and the adiabatic one. The first describes computation as a series of unitary transformations called quantum gates acting on a set of quantum bits (so-called qubits), while the second QC paradigm is based on the adiabatic theorem. The gate-based paradigm is the closest one to universality described by Deutsch [3], while the second is mainly designed for finding an optimum in discrete optimization problems.

These counter-intuitive fundamentals of QC allow it to achieve a computational speed-up that goes beyond the one provided by classical computation (e.g. quasi-exponential in [16, 17]). This turns out to have a large plethora of applications, especially in optimization problem-solving. In this same line of thought, several promising algorithms have been devised in both gate-based and adiabatic QC paradigms. Considering the latter, quantum annealers are specially designed to find the state of minimum energy of an Ising model. Now, when considering the gate-based paradigm, several algorithms also exist, but one of the most general and widely investigated optimizers in that paradigm is the Quantum Approximate Optimization Algorithm (QAOA) [5]. The QAOA is a hybrid quantum algorithm described as variational circuits (i.e. ansatzes) representing the problem and the mixer Hamiltonian. The ansatzes are composed of parameterizable quantum gates whose parameters are iteratively optimized to minimise the expectation value of the problem's Hamiltonian. Considering these facts, the hybrid nature of the QAOA rises from the fact that its hyperparameter tuning is done on a classical machine, while the execution of its variational circuit is performed using a quantum machine/simulator.

The ansatzes hyperparameter-tuning has a big impact on the QAOA efficiency and is correlated with the expectation value of the problem Hamiltonian. Indeed, to assess the quality of a given QAOA hyperparameter value, the expectation value of the problem Hamiltonian need to be computed. To do so, most of today's QAOA literature (e.g. [18, 19]) involves, by default, the use of a Quantum Processing Unit (QPU) or simulator to measure the outcome of the QAOA using the given hyperparameter values. However, such an approach turns out to have a very hard-to-overcome limitation whether using quantum computers or simulators. Actually, most of today's quantum machines are accessible for a fee, and even those publically accessible have long tasks queuing as well as a limited capacity (e.g. the number of qubits). Similarly, the use of quantum simulators suffers from the fact that as the problem variables increase, the number of required qubits increases as well, inducing exponential memory consumption.

Considering the above-mentioned facts, current QAOA implementations are unsuitable since they require substantial time and

resources. This turns out to be even more problematic when considering that today's quantum machines are in their *noisy intermediate scale* era where most devices have very limited capacities (e.g. few and noisy qubits). To solve this issue, this paper proposes a new approach that allows (I) to omit the need to access a quantum computer/simulator for executing the QAOA during its iterative training, and (II) when dealing with a subset of k-bounded pseudo-Boolean functions, it ensures to find the optimum expectation value of the problem's Hamiltonian in *polynomial* time.

The rest of the paper is structured as follows. First, Section 2 presents the fundamental concepts of quantum computation in general and the QAOA in particular. Later, Section 3 highlights the main limitation of today's QPU-dependent QAOA implementation and the challenging problematics it induces, as well as the mathematical demonstration of the proposed approach to cope with it. Afterwards, Section 4 discusses the direct consequences and uses of the main theoretical results derived in this paper. Finally, Section 5 concludes the paper.

2 BACKGROUND

This section presents the basic concepts of gate-based QC and QAOA that are needed to grasp the working principle of the proposed approach.

2.1 Gate-Based Quantum Computation

Computation in the gate-based QC paradigm is described as quantum circuits, composed of a set of *quantum gates* represented by *unitary* U and non-unitary (e.g. measurement) transformations, where U verifies the following equation: $U^\dagger U = UU^\dagger = I$. Here $U^\dagger$ is the *conjugate transpose* (also Hermitian transpose) of U and I is the identity matrix. We say that an operator A is *Hermitian* if $A^\dagger = A$. Quantum gates act on a set of qubits in a given quantum state. *States* in a quantum circuit are represented by vectors in a complex vector space. We will use the Dirac notation for vectors, that is, $|s\rangle$, where s is a label of the state. In particular, we will use $|0\rangle$ and $|1\rangle$ to denote the two states of a qubit that represent 0 and 1 with certainty in the computational basis. A general state of a qubit can be written as $|s\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ are complex numbers that fulfil the normalization condition $|\alpha|^2 + |\beta|^2 = 1$. In matrix (column) form, this qubit's state can be represented as:

$$|s\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}. \tag{1}$$

The symbol $\otimes$ denotes the tensor product of two vectors. Thus, $|0\rangle \otimes |0\rangle$ in a 2-qubit quantum circuit represents a state where the two qubits are in their $|0\rangle$ state. We will prefer the shorter notation $|00\rangle = |0\rangle \otimes |0\rangle$ for multi-qubit states. The tensor product of two general states can be computed in matrix (column) form as the Kronecker product of the matrix form of the two states, that is:

$$|s\rangle \otimes |t\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix} \otimes \begin{pmatrix}\gamma\\ \delta\end{pmatrix} = \begin{pmatrix}\alpha\begin{pmatrix}\gamma\\ \delta\end{pmatrix}\\ \beta\begin{pmatrix}\gamma\\ \delta\end{pmatrix}\end{pmatrix} = \begin{pmatrix}\alpha\gamma\\ \alpha\delta\\ \beta\gamma\\ \beta\delta\end{pmatrix}. \tag{2}$$

Single-qubit operators are linear maps that act on the single-qubit states. We will define here, for the sake of completeness, an important family of unitary (and Hermitian) single-qubit operators given by Equations (3)-(6), where I, X, Y and Z define the identity and Pauli X, Y and Z gates.

$$I = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \tag{3}$$

$$X = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} \tag{4}$$

$$Y = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix} \tag{5}$$

$$Z = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix} \tag{6}$$

Any single-qubit operator can be written as a linear combination of the previous four single-qubit operators. We can combine single-qubit operators acting on different qubits using the tensor product to form a multi-qubit operator. Using the Dirac notation, the tensor product of two single-qubit operators A and B acts on the tensor product of two vectors $|s\rangle$ and $|t\rangle$ as follows:

$$(A \otimes B)\,(|s\rangle \otimes |t\rangle) = (A\,|s\rangle) \otimes (B\,|t\rangle)\,. \tag{7}$$

In other words, each single-qubit operator acts on its corresponding vector.

The *scalar product* (or dot product) of two vectors $|s\rangle$ and $|t\rangle$ gives a complex number and is represented in Dirac notation by $\langle s|t\rangle$. Based on the scalar product we can define *co-vectors*, which are linear functions that take a vector and produce a complex number. In Dirac notation they are represented as $\langle s|$, where s is a label. The co-vectors $\langle 0|$ and $\langle 1|$ (dual basis) are defined in such a way that $\langle 0|0\rangle = \langle 1|1\rangle = 1$ and $\langle 0|1\rangle = \langle 1|0\rangle = 0$. The scalar product of $\langle s| \otimes \langle t|$ and $|s'\rangle \otimes |t'\rangle$ is $\langle s|s'\rangle \cdot \langle t|t'\rangle$.

Tensor products of operators can be expressed in matrix form using the Kronecker product. For example, Equation (8) computes the tensor product of both Pauli Y and X gates.

$$\begin{aligned} Y \otimes X &= \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix} \otimes \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} = \begin{pmatrix}0\cdot\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} & -i\cdot\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\\ i\cdot\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} & 0\cdot\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\end{pmatrix}\\ &= \begin{pmatrix}0 & 0 & 0 & -i\\ 0 & 0 & -i & 0\\ 0 & i & 0 & 0\\ i & 0 & 0 & 0\end{pmatrix}. \end{aligned} \tag{8}$$

When we want to make explicit that one operator acts only on a qubit j, we will use j as a subindex of the operator. For example, X_j is operator X (logical not) acting on qubit j and leaving the other qubits intact. This is equivalent to the following tensor product of operators:

$$X_j \equiv \left(\bigotimes_{l=1}^{j-1} I\right) \otimes X \otimes \left(\bigotimes_{k=j+1}^{n} I\right). \tag{9}$$

Using linear combinations of the tensor product of the basic four single-qubit operators, one can build any multi-qubit operator. However, not any multi-qubit operator can be implemented in a quantum computer, only unitary operators do. There is a simple way to build unitary operators from Hermitian operators. If A is

a Hermitian operator, $e^{i\theta A}$ is a unitary operator, where the exponentiation operation is defined by Equation (10). When $A^2 = I$, the exponentiation has a very plain expression in terms of A and I (see Equation (11)). This is the case of the four single-qubit operators I, X, Y, and Z.

$$e^{i\theta A} = \sum_{j=0}^{\infty} \frac{(i\theta A)^j}{j!}. \tag{10}$$

$$e^{i\theta A} = \cos\theta I + i \sin\theta A. \tag{11}$$

2.2 QAOA for Pseudo-Boolean Optimisation

QAOA has been first devised by Farhi et al. [5]. Technically, it is a variational quantum algorithm that is described as a parameterized quantum circuit that models a probability distribution over the set of potential solutions to a given problem. An important feature of QAOA is its hybrid nature, where its parameterizable circuit is executed on a quantum machine/simulator, while its ansatzes hyper-parameters are iteratively optimized on a classical one. Eventually, the goal is to min/maximize the expected value of the objective function of the solutions generated by sampling the variational circuit defined by:

$$U_{(\gamma,\beta)} = \prod_{j=1}^{p} \left(e^{-i\beta_j H_M} e^{-i\gamma_j H_P} \right) H^{\otimes n}, \tag{12}$$

where H_P is the *problem Hamiltonian*, $H_M = \sum_{i=1}^{n} X_i$ is the *mixer Hamiltonian*, H represents the Hadamard gate, n is the number of variables of the problem (qubits of the circuit), and the vectors γ and β are the hyper-parameters that the classical optimizer has to find.

Having a pseudo-Boolean problem to be solved, QAOA evolves the Hamiltonian representing the problem to be solved, being the goal of finding the eigenstate of the corresponding problem Hamiltonian H_P that optimizes the expected value of H_P in the solutions sampled from the circuit $U_{(\gamma,\beta)}$ applied to the initial state $|00\ldots0\rangle$. This state will be represented by:

$$\left|\psi_{(\gamma,\beta)}\right\rangle = U_{(\gamma,\beta)} \left|00\ldots0\right\rangle. \tag{13}$$

The expected value of H_P in state $\left|\psi_{(\gamma,\beta)}\right\rangle$ is given by:

$$F_{(\gamma,\beta)} = \left\langle\psi_{(\gamma,\beta)}\right| H_P \left|\psi_{(\gamma,\beta)}\right\rangle. \tag{14}$$

The classical optimization algorithm of QAOA min/maximizes the expectation value $F_{(\gamma,\beta)}$ of the problem Hamiltonian H_P when the quantum machine is sampled in state $\left|\psi_{(\gamma,\beta)}\right\rangle$. In other words, QAOA optimizes the probability distribution over the search space provided by the state $\left|\psi_{(\gamma,\beta)}\right\rangle$ in order to min/maximize the expected value for the objective function of the problem. Figure 1 depicts the general workflow of the standard QAOA.

It is worth noting that in the literature, it is common to show how an optimization problem should be expressed as a Quadratic Unconstrained Binary Optimisation problem (QUBO) as a previous step to solve the problem with a quantum annealer or QAOA. In fact, the original QAOA paper illustrates its use to solve MAX-CUT, a combinatorial optimisation problem that can be easily expressed as a QUBO. But, QAOA is not restricted to QUBOs since it can be used to solve higher-order pseudo-Boolean functions, as shown in [8]. However, this fact is not spread in the QC community.

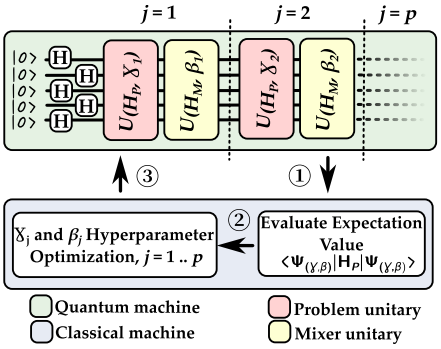

Figure 1: Workflow of the vanilla QAOA

Although the QAOA was originally devised to solve polynomials of a given order [5], it has also been repurposed in other works to prove quantum supremacy [7], or describe universal computation [10, 12].

2.3 Pseudo-Boolean functions

A pseudo-Boolean function f takes a binary string as input and provides a real value as output, that is, $f : \{0,1\}^n \rightarrow \mathbb{R}$, where n is the length of the binary string (or number of Boolean variables). Without loss of generality, we can always write a pseudo-Boolean function f as a polynomial:

$$f(x) = \sum_{S \subseteq [n]} C_S \prod_{j \in S} x_j, \tag{15}$$

where $[n]$ denotes the set $\{1, 2, \ldots, n\}$, S is a subset of variables from $[n]$, C_S is the coefficient of the term involving the product of variables in S and $x \in \{0,1\}^n$ is a vector of n Boolean variables $x = \{x_1, \ldots, x_n\}$ (binary string).

We say that a pseudo-Boolean function is k-bounded for k a constant, when it can be written as a sum of sub-functions, each one depending on at most k Boolean variables:

$$f(x) = \sum_{i=1}^{m} f_i(x, M_i), \tag{16}$$

where f_i depends at most on k variables from x, M_i is a mask that defines the variables involved in the i^{th} sub-function and m is the total number of sub-functions in the sum.

3 THE PROPOSED QPU-NEEDLESS QAOA

This section explains the main problematics induced by the vanilla (i.e. standard) QAOA implementation, as well as the mathematical proof of the proposed approach to cope with it.

3.1 Analysis of QAOA QPU-Dependency

QAOA is increasingly being investigated in the literature. Actually, for several reasons, growing interest is given from both academics and industry. This work does not aim at sketching the QAOA literature, but roughly speaking, the former can be brought to analyzing

the QAOA performances [11, 23], proposing more efficient variants of it [4, 13, 15], and eventually applying it for solving different real-world problems (e.g. in routing [1], scheduling [9], etc.). When going over the QAOA literature, it can be seen that most of the vanilla implementations involve the use of a quantum machine/simulator in each iteration of the QAOA. Such an approach is also widely-spread in introductory materials given by major quantum technology manufacturers such as IBM[1] and Google[2].

An important fact to keep in mind is that, from a mathematical point of view, it is possible to express QAOA as a discretized version of a quantum Annealer process [22]. Thus, both approaches can be explained using the same principles. Also, as explained in Section 2.2, a key feature of the QAOA is the cyclic quantum vs classical machine interactions where a classical optimizer iteratively optimizes the QAOA ansatz hyperparameters in order to min/maximize the expected value $F_{(\gamma,\beta)}$. However, to assess the quality of the newly-optimized hyperparameters, the QAOA variational circuit needs to be executed on a quantum machine/simulator and the produced quantum state should be measured. Said differently, in the original QAOA, the expected value is approximated by sampling solutions in state $\left|\psi_{(\gamma,\beta)}\right\rangle$ and evaluating H_P on the sampled solutions to compute the average. This implies preparing the circuit, running it on the quantum computer and sampling a solution several (around a thousand) times. Technically speaking, the variational quantum circuit represents a probability distribution over the search space. The goal of the classical optimizer is to maximize the average quality of the solutions sampled with that probability distribution.

However, considering today's state of quantum technology, performing executions on real quantum machines is subject to several constraints. For instance, few quantum machines are freely accessible and, those which do, have long queuing systems, and very limited capacities (e.g. few and noisy qubits). Likewise, quantum simulators have also their limitations. For instance, as the number of variables of the problem to be solved increases, the number of required qubits increases as well. The former causes the memory requirements during simulation to exponentially scale. As a matter of fact, it has been proven that, when reaching a given circuit complexity, the QAOA cannot be efficiently simulated on a classical machine [7]. Therefore, the main *problematic* is that the iterative use of a quantum device/simulator in today's QAOA implementation poses a very serious limitation towards the practical applicability of the QAOA on real-life problems requiring real-time solving, where good solutions need to be obtained in a reasonable time and with affordable resources.

On the other hand, an interesting fact is that in the original paper, Farhi et al [5] illustrate the use of QAOA over MAX-CUT and explain how for this particular problem, the evaluation of the QAOA variational circuit can be efficiently done on a classical computer and therefore, the solutions sampling via a QPU can be avoided thanks to the particular mathematical structure of MAX-CUT problem. The authors in [20] have investigated this alternative, starting mainly from the observation of the QAOA parameters' concentration. However, their work only considers the classical 2-order Ising model. Later, the work in [6] proposed a formula, as a function of $2p$ QAOA parameters, for the expected value of the problem's Hamiltonian in the QAOA that can be evaluated on a computer with $O(16^p)$ complexity. The proposal has been investigated on the Sherrington-Kirkpatrick model. Afterwards, the authors in [2] could achieve a more efficient calculation than the one given in [6] with the complexity of $O(p^2 4^p)$ proved for both the Sherrington-Kirkpatrick model and generalized to the Max-q-XORSAT. In [6], the authors went up to p=12, while in [2] they reached p=20, where for p=11, the QAOA was able to outperform a classical algorithm that can find an approximate solution for Sherrington-Kirkpatrick model within (1 - ϵ) times the ground state energy. This being said, experiments using p=20 in [2] took almost 14 hours to complete.

In this same line of thoughts, this present work extends the results given by Farhi et al. [5] and [20] and rigorously demonstrates that for any k-bounded pseudo-Boolean problem or k-bounded generalized Ising model, under very mild assumptions of the problem Hamiltonian H_P, we can exactly and efficiently evaluate the expected value $F_{(\gamma,\beta)}$ and its gradient in a classical computer in *polynomial* time, thus, reducing the time and resources needed to run QAOA. The efficient evaluation of the variational circuit in a classical computer would make QAOA much faster than its vanilla implementation, where a quantum computer is used to evaluate its circuit (see Figure 2). In this paradigm, the quantum computer is used only in the final stage of the algorithm, when the circuit is already designed, to sample solutions for the problem with high average quality. In the following section, more insight into the mathematical proof of the proposed approach is given.

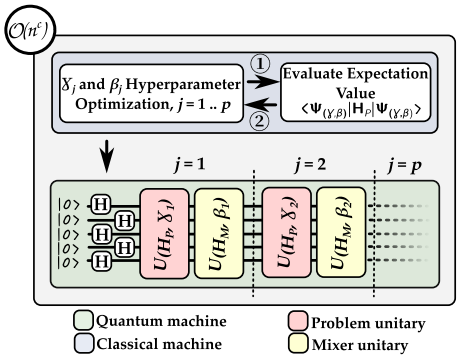

Figure 2: The proposed polynomial QPU-needless QAOA

3.2 Average Hamiltonian

We will assume that the problem Hamiltonian, H_P, is a polynomial in the operators Z_l with a maximum of k operators per term, that is, a k-bounded generalized Ising model. In formal terms, H_P can be written as:

$$H_P = \sum_{\substack{S \subseteq [n] \\ |S| \leq k}} w_S \bigotimes_{j \in S} Z_j. \tag{17}$$

[1]IBM QAOA Materials: https://learn.qiskit.org/course/ch-applications/solving-combinatorial-optimization-problems-using-qaoa
[2]Google QAOA Materials: https://quantumai.google/cirq/experiments/qaoa/example_problems

Let us consider the following example:

$$H_P = 2Z_1 \otimes Z_4 - 5Z_2 \otimes Z_3 \otimes Z_5 + 8Z_1 \otimes Z_2, \tag{18}$$

where in this case $k = 3$. The maximum number of terms of a k-bounded generalized Ising model for k constant is $\binom{n}{k} \in O(n^k)$, which is polynomial in n.

It is well-known that any compressible pseudo-Boolean function can be rewritten in polynomial time as a k-bounded pseudo-Boolean function, and a k-bounded pseudo-Boolean function can be transformed into a k-bounded generalized Ising model by applying the Walsh-Hadamard transform in polynomial time. Thus, any compressible pseudo-Boolean function can be written in the form (17) in polynomial time. In order to write an efficient expression for $F_{(\gamma,\beta)}$, we need to introduce some mathematical results.

We will frequently work with multi-qubit operators that are tensor products of single-qubit operators. With a little abuse of notation, we will represent with capital letters, like A, the multi-qubit operator and will add subindices to this symbol, e.g., A_j, to represent the single-qubit operator acting on a particular qubit j. In formal terms, this can be expressed as follows:

$$A \equiv \bigotimes_{j=1}^{n} A_j. \tag{19}$$

The first mathematical result provides an expression for the commutator of multi-qubit operators that are tensor products of single-qubit operators.

Lemma 3.1. *Let A and B be two n-qubit operators that can be expressed as the tensor product of single-qubit operators. Then, the commutator of A and B can be written as the sum of at most n tensor products and the concrete value is:*

$$[A, B] = \sum_{j=1}^{n} \bigotimes_{l=1}^{j-1} B_l A_l \otimes [A_j, B_j] \otimes \bigotimes_{l=j+1}^{n} A_l B_l \tag{20}$$

Proof. We can prove this by induction over the number of qubits, n. If $n = 1$, we have $[A, B] = [A_1, B_1]$ and the claim trivially holds. Let us assume that the expression is true when we have $n-1$ qubits (induction hypothesis) and let us prove it for n qubits. In that case the expression for $[A, B]$ can be-written as follows:

$$\begin{aligned}
[A, B] &= \left[\bigotimes_{l=1}^{n} A_l, \bigotimes_{l=1}^{n} B_l\right] = \bigotimes_{l=1}^{n} A_l B_l - \bigotimes_{l=1}^{n} B_l A_l \\
&= A_1 B_1 \otimes \bigotimes_{l=2}^{n} A_l B_l - B_1 A_1 \otimes \bigotimes_{l=2}^{n} B_l A_l \\
&= A_1 B_1 \otimes \bigotimes_{l=2}^{n} A_l B_l - B_1 A_1 \otimes \bigotimes_{l=2}^{n} A_l B_l \\
&+ B_1 A_1 \otimes \bigotimes_{l=2}^{n} A_l B_l - B_1 A_1 \otimes \bigotimes_{l=2}^{n} B_l A_l \\
&= [A_1, B_1] \otimes \bigotimes_{l=2}^{n} A_l B_l + B_1 A_1 \otimes \left(\bigotimes_{l=2}^{n} A_l B_l - \bigotimes_{l=2}^{n} B_l A_l\right)
\end{aligned}$$

using the induction hypothesis

$$\begin{aligned}
&= [A_1, B_1] \otimes \bigotimes_{l=2}^{n} A_l B_l \\
&+ B_1 A_1 \otimes \sum_{j=2}^{n} \bigotimes_{l=2}^{j-1} B_l A_l \otimes [A_j, B_j] \otimes \bigotimes_{l=j+1}^{n} A_l B_l \\
&= [A_1, B_1] \otimes \bigotimes_{l=2}^{n} A_l B_l \\
&+ \sum_{j=2}^{n} \bigotimes_{l=1}^{j-1} B_l A_l \otimes [A_j, B_j] \otimes \bigotimes_{l=j+1}^{n} A_l B_l \\
&= \sum_{j=1}^{n} \bigotimes_{l=1}^{j-1} B_l A_l \otimes [A_j, B_j] \otimes \bigotimes_{l=j+1}^{n} A_l B_l.
\end{aligned}$$

□

Equation (20) allows us to express any commutator of single-qubit operator tensor products as a sum of single-qubit operator tensor products. This will be useful to prove the efficiency of evaluating the ansatz of QAOA in a classical computer. We can observe that the number of terms in the sum of the right-hand side of Equation (20) is the number of nonzero single-qubit commutators $[A_l, B_l]$. This is an important number for which we provide a short notation:

$$\#(A, B) = |\{l \in [n], [A_l, B_l] \neq 0\}| \tag{21}$$

Lemma 3.2. *Let A and B be n-qubit operators that are the tensor product of single-qubit operators. We also assume that B is its own inverse: $B^2 = I$. Then, we have:*

$$e^{i\theta B} A e^{-i\theta B} = A - (iI \cos\theta - B \sin\theta) \sin\theta [A, B], \tag{22}$$

which is a sum of at most $2\#(A, B) + 1$ tensor products of single-qubit operators.

Proof.

$$\begin{aligned}
e^{i\theta B} A e^{-i\theta B} &= (I\cos\theta + iB\sin\theta)\, A\, (I\cos\theta - iB\sin\theta) \\
&= (I\cos\theta + iB\sin\theta)\, (A\cos\theta - iAB\sin\theta) \\
&= A\cos^2\theta - i[A, B]\cos\theta\sin\theta + BAB\sin^2\theta \\
&= A\cos^2\theta - i[A, B]\cos\theta\sin\theta + B([A, B] + BA)\sin^2\theta \\
&= A - i[A, B]\cos\theta\sin\theta + B[A, B]\sin^2\theta \\
&= A - (iI\cos\theta - B\sin\theta)\sin\theta[A, B]
\end{aligned}$$

We know by Lemma 3.1 that $[A, B]$ is a sum of $\#(A, B)$ tensor products of single-qubit operators. When we pre-multiply these tensor products by B we will get another $\#(A, B)$ tensor products of single-qubit operators that, in general, are not linear combinations of the ones in the sum of $[A, B]$. We also have A. In total, this means that the $e^{i\theta B} A e^{-i\theta B}$ can be written as the sum of $2\#(A, B)+1$ tensor products of single-qubit operators. □

Corollary 3.3. *If B is an operator representing one term in the sum of the problem Hamiltonian, H_P, representing a k-bounded generalized Ising model, then $\#(A, B) \leq k$ and the number of $e^{i\theta B} A e^{-i\theta B}$ can be written as a sum of no more than $2k + 1$ tensor products of single-qubit operators.*

Proof. A k-bounded generalized Ising model contains terms with at most k variables. Each term is transformed into a multi-qubit operator which is a tensor product of single-qubits operators, where for each qubit the identity I or the Z operator is used. The identity operator commute with any other operator and the number of Z operators in B is bounded by k. Thus, $\#(A,B) \leq k$ and the results follows as a consequence of Lemma 3.2. □

Corollary 3.4. *If B is an operator acting on a single-qubit with $B^2 = I$, and A is a tensor product of single-qubit operators then $e^{i\theta B}Ae^{-i\theta B}$ is a tensor product of single-qubit operators.*

Proof. Let j be the only index such that $[A_j, B_j] \neq 0$. Then:

$$
\begin{aligned}
e^{i\theta B}Ae^{-i\theta B} &= \bigotimes_{l=1}^{n} A_l \\
&\quad - \sin\theta \left(i\cos\theta \bigotimes_{l=1}^{n} I_l - \sin\theta \bigotimes_{l=1}^{n} B_l \right) \cdot \\
&\quad \bigotimes_{l=1}^{j-1} A_l \otimes [A_j, B_j] \otimes \bigotimes_{l=j+1}^{n} A_l \\
&= \bigotimes_{l=1}^{n} A_l - i\sin\theta\cos\theta \bigotimes_{l=1}^{j-1} A_l \otimes [A_j, B_j] \otimes \bigotimes_{l=j+1}^{n} A_l \\
&\quad + \sin^2\theta \bigotimes_{l=1}^{j-1} A_l \otimes B_j [A_j, B_j] \otimes \bigotimes_{l=j+1}^{n} A_l \\
&= \bigotimes_{l=1}^{j-1} A_l \otimes \left(A_j - \sin\theta \left(iI_j \cos\theta - \sin\theta B_j \right) [A_j, B_j] \right) \\
&\quad \otimes \bigotimes_{l=j+1}^{n} A_l
\end{aligned}
$$

which is a tensor product of operators that differ from A only in the j-th factor. □

Proposition 3.5. *Let A be a tensor product of single-qubit operators A_l for $1 \leq l \leq n$, and H_M the mixing Hamiltonian. Then, we have:*

$$
e^{i\beta H_M}Ae^{-i\beta H_M} = \bigotimes_{j=1}^{n} \left(A_j - \sin\beta \left(iI_j \cos\beta - \sin\beta X_j \right) [A_j, X_j] \right) \tag{23}
$$

which is a tensor product of single-qubit operators.

Proof. We iteratively apply Corollary 3.4 to the $e^{i\beta X_j}Ae^{-i\beta X_j}$ factors and the result follows. □

Proposition 3.6. *Let A be a tensor product of single-qubit operators A_l for $1 \leq l \leq n$ where m of them are different from the identity, and let H_P be the problem Hamiltonian of a k-bounded generalized Ising model. Then, $e^{i\gamma H_P}Ae^{-i\gamma H_P}$ can be written as a sum of at most $(2k+1)^{cm}$ tensor products of single-qubit operators, where c is the maximum number of appearances of a variable in the terms of the generalized Ising model. The newly generated tensor products have, at most, $m(c(k-1)+1)$ single-qubit operators different from the identity.*

Proof. First, we notice that $e^{-i\gamma H_P}$ can be written as a product of operators as follows, due to the commutativity of all the terms in the sum of H_P:

$$
e^{-i\gamma H_P} = \prod_{S \subseteq [n]} e^{-i\gamma w_S \bigotimes_{l \in S} Z_l}. \tag{24}
$$

All the factors in the product commute among them. Thus, in the expression $e^{i\gamma H_P}Ae^{-i\gamma H_P}$, we can move the factors that do commute with A in $e^{-i\gamma H_P}$ to the left, commute them with A and cancel them with the inverse factor in $e^{i\gamma H_P}$. Doing this, the only factors that remain in the exponentials are those that do not commute with A. They are exactly those factors $e^{-i\gamma w_S \bigotimes_{l \in S} Z_l}$ where there is an $l \in S$ for which $A_l \neq I_l$. If we denote with c the maximum number of appearances of a Z_l in the terms of H_P, the number of factors remaining is, at most, cm, where m is the number of factors A_l which are not the identity.

According to Corollary 3.3, after applying each of these factors the results can be expressed as a sum of at most $(2k+1)$ single-qubit tensor products. Thus, the expression for $e^{i\gamma H_P}Ae^{-i\gamma H_P}$ can be written as a sum of at most $(2k+1)^{cm}$ single-qubit tensor products.

For each of the non-identity factors, A_j, the terms $e^{-i\gamma w_S \bigotimes_{l \in S} Z_l}$ can add $k-1$ new non-identity terms in the resulting tensor products. Since there are at most c of these factors, the new tensor products of single-qubit operators will have at most $c(k-1)$ non-identity terms for each $A_j \neq I_j$. We have m of such single-qubit operators A_j, and, thus, the new tensor products of single-qubit operators, have at most $mc(k-1) + m = m(c(k-1)+1)$ non-identity factors. □

Combining Propositions 3.5 and 3.6 we conclude that the expression corresponding to one layer of the ansatz,

$$
e^{i\gamma H_P}e^{i\beta H_M}Ae^{-i\beta H_M}e^{-i\gamma H_P},
$$

for a tensor product of single-qubit operators, A, with m non-identity single-qubit operators, can be written using, at most, $m(c(k-1)+1)$ tensor products of single-qubit operators. The reason is that the factor including the mixing Hamiltonian does not change the number of terms in the sum and the factor including the problem Hamiltonian can be written with, at most, $m(c(k-1)+1)$ terms. The next proposition provides an upper bound on the number of linearly independent tensor products of single-qubit operators for $U^{\dagger}_{(\gamma,\beta)}H_P U_{(\gamma,\beta)}$.

Proposition 3.7. *Let H_P be a k-bounded generalized Ising model with t terms where each variable Z_l appears in at most c terms. Then, the expression $U^{\dagger}_{(\gamma,\beta)}H_P U_{(\gamma,\beta)}$ can be written as a sum of at most $t(2k+1)^{\frac{k}{k-1}((c(k-1)+1)^p-1)}$ linearly independent tensor products of single-qubit operators.*

Proof. Let us start the proof by counting the number of non-identity single-qubit operators in the tensor products of the expression for $U^{\dagger}_{(\gamma,\beta)}H_P U_{(\gamma,\beta)}$ depending on the number of layers p. The terms in H_P have at most k non-identity single-qubit operators in their tensor products (by definition of k-bounded). Let us call this $m_0 = k$. By Proposition 3.6, after one layer, the tensor products of single-qubit operators will have $m_1 = m_0(c(k-1)+1) = k(c(k-1)+1)$, which increase to $m_2 = m_1(c(k-1)+1) = k(c(k-1)+1)^2$

after two layers. It is easy to see that the general result is

$$m_p = k(c(k-1)+1)^p. \tag{25}$$

Let us focus now in the upper bound for the number of linearly independent tensor products of single-qubit operators. This number starts in t and Proposition 3.6 proves that this number is multiplied by $(2k+1)^{cm_{p-1}}$ in after applying layer p. Thus the final number will be:

$$\begin{aligned} t\prod_{l=1}^{p}(2k+1)^{cm_{l-1}} &= t(2k+1)^{c\sum_{l=1}^{p} m_{l-1}} \\ &= t(2k+1)^{ck\sum_{l=1}^{p}(c(k-1)+1)^{l-1}} \\ &= t(2k+1)^{ck\frac{(c(k-1)+1)^p-1}{c(k-1)}} \\ &= t(2k+1)^{\frac{k}{k-1}((c(k-1)+1)^p-1)}, \end{aligned}$$

as we claim in the proposition. We can now complete the proof by considering that the application of the Hadamard gates to each single-qubit does not add new linearly independent tensor products of single-qubit operators. □

Lemma 3.8. *Let A be a tensor product of single-qubit operators, where each A_l is written as the sum of the I_l, X_l, Y_l, and Z_l single-qubits operators as follows: $A_l = w_{l,I}I_l + w_{l,X}X_l + w_{l,Y}Y_l + w_{l,Z}Z_l$. Then, we have:*

$$\langle 0\ldots0|\, H^{\otimes n}AH^{\otimes n}\,|0\ldots0\rangle = \prod_{l=1}^{n}\left(w_{l,I}+w_{l,X}\right), \tag{26}$$

which can be computed in $O(n)$ time.

Proof. By simple algebraic manipulations we can find that $HX_lH = Z_l$, $HY_lH = -Y_l$, $HZ_lH = X_l$ and $HI_lH = I_l$. Thus, we have:

$$\begin{aligned} HA_lH &= w_{l,I}HI_lH + w_{l,X}HX_lH + w_{l,Y}HY_lH + w_{l,Z}HZ_lH \\ &= w_{l,I}I + w_{l,X}Z - w_{l,Y}Y + w_{l,Z}X. \end{aligned}$$

Now using the definitions of I, X, Y and Z in Equations (3) to (6) we can easily check that:

$$\langle 0|\, X\,|0\rangle = \langle 0|\, Y\,|0\rangle = 0, \tag{27}$$

$$\langle 0|\, I\,|0\rangle = \langle 0|\, Z\,|0\rangle = 1. \tag{28}$$

Now we can use the expression

$$\langle 0\ldots0|\bigotimes_{l=1}^{n}A_l\,|0\ldots0\rangle = \prod_{l=1}^{n}\langle 0|\, A_l\,|0\rangle$$

to complete the proof. □

Theorem 3.9. *Let H_P be a k-bounded generalized Ising model with t terms where each variable Z_l appears in at most c terms. Then, the expression $F_{(\gamma,\beta)}$, which QAOA optimizes, and its gradient with respect to variables γ_j and β_j can be evaluated in a classical computer in time $O\left(nt(2k+1)^{\frac{k}{k-1}((c(k-1)+1)^p-1)}\right)$.*

Proof. The expression for $F_{(\gamma,\beta)}$ is:

$$\left\langle\psi_{(\gamma,\beta)}\right|H_P\left|\psi_{(\gamma,\beta)}\right\rangle = \langle 00\ldots0|\, U^{\dagger}_{(\gamma,\beta)}H_PU_{(\gamma,\beta)}\,|00\ldots0\rangle\,. \tag{29}$$

We know by Proposition 3.7 that $U^{\dagger}_{(\gamma,\beta)}H_PU_{(\gamma,\beta)}$ can be written as a sum of at most $t(2k+1)^{\frac{k}{k-1}((c(k-1)+1)^p-1)}$ linearly independent tensor products of single-qubit operators. Lemma 3.8 proves that the evaluation of each tensor product of single-qubit operators can be computed in $O(n)$ time. Thus, the computation of $F_{(\gamma,\beta)}$ can be done in $O\left(nt(2k+1)^{\frac{k}{k-1}((c(k-1)+1)^p-1)}\right)$ time.

The expression $F_{(\gamma,\beta)}$ can be written in closed form depending on the sines and cosines of the parameter vectors β and γ. Thus, its gradient can be computed in a time which is also

$$O\left(nt(2k+1)^{\frac{k}{k-1}((c(k-1)+1)^p-1)}\right).$$

□

4 CONSEQUENCES

Theorem 3.9 have interesting consequences for QAOA. We discuss them in this section.

If k, c and p are constant, then $F_{(\gamma,\beta)}$ can be evaluated in a time that is linear in n. Farhi et al. [5] shows this for MAX-CUT when the degree of the nodes is bounded by a constant. The bound on the degrees of the nodes implies that c is a constant. In this case, it makes little sense to use a quantum computer to evaluate this expression, unless k, c or p are large. Furthermore, we have access to the exact value of the expression and we can evaluate the gradient exactly. This opens the door to the use of gradient-based optimization techniques, which are not available when the circuit in run in a quantum computer. Using the evaluation in the classical computer, we avoid the noise associated with the finite sampling of solutions and quantum machine, improving the quality of the parameters for the variational circuit.

Once the circuit is optimized using the classical computer, we still need to implement and run the circuit in a quantum computer to sample the solutions to the problem. This sampling is not efficient in a classical computer because we need to represent the state $\left|\psi_{(\gamma,\beta)}\right\rangle$ in a classical computer and, as far as we know, this would require $O(2^n)$ space and time. However, we can also imagine a quantum-inspired algorithm based on QAOA where we get the values of some variables in high-quality or even optimal solutions. The idea is to evaluate the expressions $\left\langle\psi_{(\gamma,\beta)}\right|Z_j\left|\psi_{(\gamma,\beta)}\right\rangle$ for $1 \le j \le n$. We can do this efficiently as a consequence of Theorem 3.9. If the values obtained are close to 1 or -1, we can guess that the corresponding variable has that value in a (quasi-)optimal solution. If the value is around 0 we cannot conclude what is the value for Z_j. However, in that case, we could evaluate pairs of variables in the ansatz: $\left\langle\psi_{(\gamma,\beta)}\right|Z_jZ_l\left|\psi_{(\gamma,\beta)}\right\rangle$. This can also be done efficiently and provides information about the correlation among the variables.

The consequences of Theorem 3.9 generalize beyond k-bounded problems. For instance, it is proven that there exist general and specialized transforms that allow reducing multi-linear pseudo-Boolean polynomials to k-bounded pseudo-Boolean functions for all $k \ge 2$ [21], which ensures always having k-bounded pseudo-Boolean functions. In this particular case, these two aforementioned types of functions are defined by Equations (15) and (16), respectively.

5 CONCLUSIONS AND PERSPECTIVES

QAOA is one of the widely-investigated quantum algorithms whose working principles rely on the iterative optimization of its ansatz hyperparameters. This repetitive process, de facto, involves the intensive use of both quantum and classical machines. Regarding the nature of quantum computation and the state of today's quantum technology, the use of a quantum machine/simulator in the case of the QAOA poses several challenges and limitations that substantially limit its real-time applicability and induces the need for substantial resources whether using quantum machines or simulators. To cope with this, this work proposes a new approach that has two main *contributions* (I) eliminates the need for the use of QPU/simulator during the QAOA hyperparameter-training, and (II), when dealing with k-bounded pseudo-Boolean problems, the set of optimal parameters, and therefore the minimum expectation value, can be found in *polynomial* time. The proposed approach is thought to lead the way for wider and more efficient implementation and use of QAOA.

In addition to the theoretical development provided in the current work, the next step stands in the practical demonstration of these theoretical findings on a diverse and wide set of benchmarks representing QAOA for tackling different k-bounded and unbounded pseudo-Boolean problems.

ACKNOWLEDGMENTS

This work is partially funded by *Universidad de Málaga*, *Ministerio de Ciencia, Innovación y Universidades del Gobierno de España* under grants PID 2020-116727RB-I00 (funded by MCIN/AEI/10.13039/501100011033) and PRX21/00669; and TAILOR ICT-48 Network (No 952215) funded by EU Horizon 2020 research and innovation programme.

REFERENCES

[1] Utkarsh Azad, Bikash K. Behera, Emad A. Ahmed, Prasanta K. Panigrahi, and Ahmed Farouk. 2022. Solving Vehicle Routing Problem Using Quantum Approximate Optimization Algorithm. *IEEE Transactions on Intelligent Transportation Systems* (2022), 1–10. https://doi.org/10.1109/TITS.2022.3172241

[2] Joao Basso, Edward Farhi, Kunal Marwaha, Benjamin Villalonga, and Leo Zhou. 2022. The Quantum Approximate Optimization Algorithm at High Depth for MaxCut on Large-Girth Regular Graphs and the Sherrington-Kirkpatrick Model. In *17th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2022, July 11-15, 2022, Urbana Champaign, Illinois, USA (LIPIcs, Vol. 232)*, François Le Gall and Tomoyuki Morimae (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 7:1–7:21. https://doi.org/10.4230/LIPIcs.TQC.2022.7

[3] David Deutsch. 1985. Quantum theory, the Church–Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 400 (1985), 117 – 97.

[4] Daniel J. Egger, Jakub Mareček, and Stefan Woerner. 2021. Warm-starting quantum optimization. *Quantum* 5 (June 2021), 479. https://doi.org/10.22331/q-2021-06-17-479

[5] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028 [quant-ph]

[6] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Leo Zhou. 2022. The Quantum Approximate Optimization Algorithm and the Sherrington-Kirkpatrick Model at Infinite Size. *Quantum* 6 (July 2022), 759. https://doi.org/10.22331/q-2022-07-07-759

[7] Edward Farhi and Aram W Harrow. 2019. Quantum Supremacy through the Quantum Approximate Optimization Algorithm. arXiv:1602.07674 [quant-ph]

[8] Camille Grange, Michael Poss, and Eric Bourreau. 2022. An introduction to variational quantum algorithms on gate-based quantum computing for combinatorial optimization problems. arXiv:2212.11734 [math.OC]

[9] Krzysztof Kurowski, Tomasz Pecyna, Mateusz Slysz, Rafał Różycki, Grzegorz Waligóra, and Jan Weglarz. 2023. Application of quantum approximate optimization algorithm to job shop scheduling problem. *European Journal of Operational Research* (2023). https://doi.org/10.1016/j.ejor.2023.03.013

[10] Seth Lloyd. 2018. Quantum approximate optimization is computationally universal. arXiv:1812.11075 [quant-ph]

[11] Thomas Lubinski, Carleton Coffrin, Catherine McGeoch, Pratik Sathe, Joshua Apanavicius, and David E. Bernal Neira. 2023. Optimization Applications as Quantum Performance Benchmarks. (2 2023). arXiv:2302.02278 [quant-ph]

[12] Mauro E. S. Morales, Jacob D. Biamonte, and Zoltán Zimborás. 2020. On the universality of the quantum approximate optimization algorithm. *Quantum Inf. Process.* 19, 9 (2020), 291. https://doi.org/10.1007/s11128-020-02748-9

[13] Moussa, Charles, Wang, Hao, Bäck, Thomas, and Dunjko, Vedran. 2022. Unsupervised strategies for identifying optimal parameters in Quantum Approximate Optimization Algorithm. *EPJ Quantum Technol.* 9, 1 (2022), 11. https://doi.org/10.1140/epjqt/s40507-022-00131-4

[14] Michael A. Nielsen and Isaac L. Chuang. 2000. *Quantum Computation and Quantum Information*. Cambridge University Press.

[15] Ruslan Shaydulin, Ilya Safro, and Jeffrey Larson. 2019. Multistart Methods for Quantum Approximate optimization. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)*. 1–8. https://doi.org/10.1109/HPEC.2019.8916288

[16] P.W. Shor. 1994. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 124–134. https://doi.org/10.1109/SFCS.1994.365700

[17] Peter W. Shor. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* 26, 5 (oct 1997), 1484–1509. https://doi.org/10.1137/S0097539795293172

[18] Vicente P. Soloviev, Concha Bielza, and Pedro Larrañaga. 2023. Quantum approximate optimization algorithm for Bayesian network structure learning. *Quant. Inf. Proc.* 22, 1 (2023), 19. https://doi.org/10.1007/s11128-022-03769-2 arXiv:2203.02400 [quant-ph]

[19] Vicente P. Soloviev, Concha Bielza, and Pedro Larrañaga. 2023. Quantum approximate optimization algorithm for Bayesian network structure learning. *Quant. Inf. Proc.* 22, 1 (2023), 19. https://doi.org/10.1007/s11128-022-03769-2 arXiv:2203.02400 [quant-ph]

[20] Michael Streif and Martin Leib. 2020. Training the quantum approximate optimization algorithm without access to a quantum processing unit. *Quantum Science and Technology* 5, 3 (may 2020), 034008. https://doi.org/10.1088/2058-9565/ab8c2b

[21] Darrell Whitley, Hernan Aguirre, and Andrew Sutton. 2020. Understanding Transforms of Pseudo-Boolean Functions. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (Cancún, Mexico) *(GECCO '20)*. Association for Computing Machinery, New York, NY, USA, 760–768. https://doi.org/10.1145/3377930.3390144

[22] Sheir Yarkoni, Elena Raponi, Thomas Bäck, and Sebastian Schmitt. 2022. Quantum annealing for industry applications: introduction and review. *Rept. Prog. Phys.* 85, 10 (2022), 104001. https://doi.org/10.1088/1361-6633/ac8c54 arXiv:2112.07491 [quant-ph]

[23] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. 2020. Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices. *Phys. Rev. X* 10 (Jun 2020), 021067. Issue 2. https://doi.org/10.1103/PhysRevX.10.021067