THESIS FOR THE DEGREE OF LICENTIATE OF ENGINEERING

Application of the quantum approximate optimization algorithm to combinatorial optimization problems

PONTUS VIKSTÅL

Department of Microtechnology and Nanoscience (MC2)
Applied Quantum Physics Laboratory
Chalmers University of Technology
Göteborg, Sweden, 2020

Application of the quantum approximate optimization algorithm
to combinatorial optimization problems
PONTUS VIKSTÅL

Technical Report MC2-436
ISSN 1652-0769

Applied Quantum Physics Laboratory
Department of Microtechnology and Nanoscience (MC2)
Chalmers University of Technology
SE–412 96 Göteborg
Sweden
Telephone +46 (0)31-772 1000

Cover: Schematic representation of the quantum approximate optimization algorithm

Chalmers Digitaltryck
Göteborg, Sweden, 2020

Application of the quantum approximate optimization algorithm
to combinatorial optimization problems
PONTUS VIKSTÅL
Department of Microtechnology and Nanoscience (MC2)
Applied Quantum Physics Laboratory
Chalmers University of Technology

Abstract

This licentiate thesis is an extended introduction to the accompanying papers, which encompass a study of the quantum approximate optimization algorithm (QAOA). It is a hybrid quantum-classical algorithm for solving combinatorial optimization problems and is a promising algorithm to run on near term quantum devices. In this thesis, we will introduce the workings of the QAOA, together with some applications of it on combinatorial optimization problems.

Keywords: Quantum approximate optimization algorithm, quantum computing, combinatorial optimization

Acknowledgments

First and foremost, I would like to acknowledge the support that I have received from my great supervisor Giulia Ferrini. Thanks for always believing in me. I am also grateful for the help and guidance that I have received from Göran Johansson. I am also thankful for the collaboration that I have had with people from Jeppesen: Mattias Grönkvist, Marika Svensson, and Martin Andersson. Furthermore, I would like to thank all the great co-workers at the AQP and QT department for making MC2 a very enjoyable workplace. I would also like to give a tremendous thanks to Ingrid Strandberg for proof reading this thesis. Finally, this would not be possible without the generous support from Helena Liljenberg and my family.

Pontus Vikstål, Göteborg, December 2020

Publications

A **Applying the Quantum Approximate Optimization Algorithm to the Tail-Assignment Problem**
Pontus Vikstål, Mattias Grönkvist, Marika Svensson, Martin Andersson, Göran Johansson, and Giulia Ferrini
Phys. Rev. Applied **14**, 034009 (2020)

B **Improved Success Probability with Greater Circuit Depth for the Quantum Approximate Optimization Algorithm**
Andreas Bengtsson, Pontus Vikstål, Christopher Warren, Marika Svensson, Xiu Gu, Anton Frisk Kockum, Philip Krantz, Christian Križan, Daryoush Shiri, Ida-Maria Svensson, Giovanna Tancredi, Göran Johansson, Per Delsing, Giulia Ferrini, and Jonas Bylander
Phys. Rev. Applied **14**, 034010 (2020)

Contents

1 Introduction

Over the past half-century there has been an amazing miniaturization in computer size, thanks to technological innovations that has led to the transistor in our computers becoming smaller. But as the transistors are shrinking to the size of a few atoms, quantum effects are beginning to take place and interfere with the functioning of the electronics [1]. To handle this, scientists are trying to use these quantum effects to their advantage by introducing a new computational model, called "quantum computing," based on the recognition that at small scales, quantum mechanics is the most accurate description of reality that is currently known.

1.1 Quantum computing

The quantum computing field started in the early 1980s with the prominent physicists Paul Benioff, Yuri Manin, and Richard Feynman, conceptualizing, independently and simultaneously, the idea of a quantum computer [2–5]. This idea was based on the observation that simulating a quantum system on a classical computer requires resources that scale exponentially with the size of the quantum system. Thus, we better use quantum physics if we want to simulate quantum physics. Later on, David Deutsch formalized the idea of a quantum Turing machine and put forward the quantum circuit model [6, 7]. This was followed by Peter Shor, who found a quantum algorithm that can solve prime factorization exponentially faster than any known classical algorithm [8]. Finding the prime factors to large numbers is believed to be hard for classical computers, and this computational hardness has come to be used in public-key cryptosystems, such as the RSA [9]. However, with a large enough quantum computer, the public-key cryptosystems could easily be hacked.

Today, quantum computers are still in the early stages, and they are much more sensitive to noise than their classical counterpart. This sets a limit on the size of the quantum circuits. Even though quantum error correction is theoretically known to tame errors, it still requires a large overhead of qubits [10, 11]. For example, estimates on the requirements of running Shor's algorithms for factoring cryptographically hard numbers have shown to require millions of qubits with error-correction [12].

Nevertheless, the goal of building a quantum computer is still actively pursued by many institutes and corporations. As small-size, non-fault tolerant quantum computers, commonly referred to as noisy intermediate-scale quantum (NISQ) devices [13], are starting to become available, academics want to explore their usefulness. In fact, it has already been demonstrated that it is possible to run a quantum algorithm on a NISQ device, which is hard to simulate on a classical computer [14]. Although the quantum algorithm of Ref. [14] has little to no application purposes, it still shows the potential power of these NISQ devices. A major task thus remains to find useful quantum algorithms that can solve real-world problems faster than any classical algorithm. One promising candidate for this is the quantum approximate optimization algorithm (QAOA), whose purpose is to solve combinatorial optimization problems. These are types of problems that frequently arise in industry, such as aviation [15, 16].

The QAOA is classified as a heuristic hybrid quantum-classical algorithm. The hybrid character comes from the fact that a quantum computer prepares some n-qubit state that is measured, and the measurement results are then processed by a classical computer that tells the quantum computer how to slightly change how the n-qubit state is prepared. And because no theoretical proof of speed up exists for this algorithm, it is heuristic[1]. Thus one has to simply run the algorithm and see what happens.

In this thesis, we set out to explore the quantum approximate optimization algorithm, its application to combinatorial optimization problems, and its implementation on a quantum computer.

Qubits

Just as a classical computer uses bits that are either 0 or 1 and that can physically be represented as a low and a high voltage in a wire, a quantum computer uses qubits $|0\rangle$ or $|1\rangle$ (often called computational basis states). A qubit can, for example, be physically manifested as two states of an atom (e.g., ground state $|0\rangle$ and excited state $|1\rangle$). The angular-looking bracket that symbolizes the qubit state is called "ket"-notation, and it is a shorthand notation for column vectors

$$|0\rangle \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle \equiv \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{1.1}$$

The reason why qubits are represented as column vectors and not by a single number is because a quantum system, like the atom, can exist in a linear combination or, "superposition", of both states at once:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \equiv \begin{pmatrix} \alpha \\ \beta \end{pmatrix}. \tag{1.2}$$

Here α and β are two complex numbers that satisfy $|\alpha|^2 + |\beta|^2 = 1$. This is to ensure that the qubit is properly normalized. Such that when the qubit is

[1] Although as we will see, for the optimization problem Max-Cut, there exist lower bounds on the performance guarantee, which therefore makes it an approximate algorithm.

measured, there is a $|\alpha|^2$ probability of finding it in the $|0\rangle$ state, and $|\beta|^2$ of finding it in the $|1\rangle$ state. Thus the very act of measuring the qubit affects its state! If the measurement outcome is 0, then the state after the measurement is $|0\rangle$. And if the measurement outcome is 1, then the state after the measurement is $|1\rangle$. Moreover, an intuitive way to visualize a general quantum state of a single qubit as represented by Eq. (1.2) is to picture it as a unit vector inside a unit sphere, called the Bloch sphere, see Fig. 1.1. An arbitrary state of a single qubit state can be written in terms of the polar and the azimuthal angles (φ, θ) on the Bloch sphere as

$$|\psi\rangle = \mathrm{e}^{\mathrm{i}\gamma}\left(\cos\frac{\theta}{2}|0\rangle + \mathrm{e}^{\mathrm{i}\phi}\sin\frac{\theta}{2}|1\rangle\right), \quad 0 \le \theta \le \pi, \quad 0 \le \phi \le 2\pi. \tag{1.3}$$

Here γ is an overall phase factor that is unobservable, i.e., qubit states with different values of γ are indistinguishable and are all represented by the same point on the Bloch sphere.

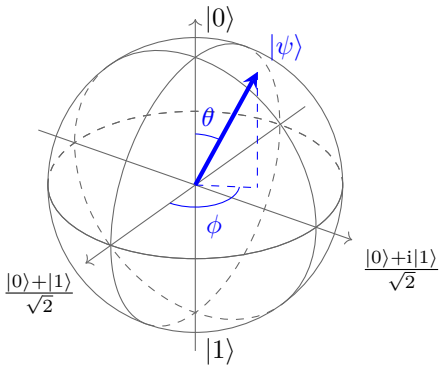


Figure 1.1: Bloch sphere showing the computational basis states $|0\rangle$ and $|1\rangle$, and a general qubit state $|\psi\rangle = \cos\theta/2\,|0\rangle + \mathrm{e}^{\mathrm{i}\phi}\sin\theta/2\,|1\rangle$.

The most general n-qubit state can be written as

$$|\psi\rangle = \sum_{z\in\{0,1\}^n} \alpha_z\,|z\rangle\,, \text{ with } \sum_{z\in\{0,1\}^n} |\alpha_z|^2 = 1, \tag{1.4}$$

where $\{0,1\}^n$ is the tuple of all binary strings $z = z_1 \ldots z_n$ of length n, and thus $|\psi\rangle$ is described by a 2^n column vector.

Quantum gates

Similar to a classical computer that uses elementary gates, like AND or NOT, to perform operations on the bits, a quantum computer uses quantum gates to change the probability amplitudes of one or multiple qubits. How the probability amplitudes change with time in a quantum system is ultimately governed by the

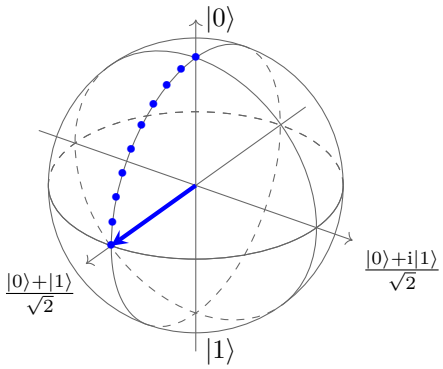


Figure 1.2: Bloch sphere showing the action of the quantum logic operation $\mathrm{R}_y(\pi/2)$ applied to the $|0\rangle$ state.

Schrödinger equation. It is a linear differential equation that describes the time evolution of a closed quantum system:

$$-\mathrm{i}\hbar\frac{\mathrm{d}\,|\psi(t)\rangle}{\mathrm{d}t} = \hat{H}\,|\psi(t)\rangle\,. \tag{1.5}$$

Here $\hbar$ is the reduced Planck's constant. We can set it to 1 by choosing appropriate units, and hence we will ignore it throughout this thesis. $\hat{H}$ is a hermitian operator, called the Hamiltonian, and it corresponds to the quantum system's total energy. Once the Hamiltonian is known, the Schrödinger equation can be solved. For a time-independent Hamiltonian, the solution is

$$\hat{U}(t) = \mathrm{e}^{-\mathrm{i}\hat{H}t}. \tag{1.6}$$

Thus, if the initial state of the system $|\psi(0)\rangle$ is known, all subsequent states can be calculated by acting with the time evolution operator $\hat{U}(t)$ on it, $|\psi(t)\rangle = \hat{U}(t)\,|\psi(0)\rangle$. Important types of Hamiltonians are the Pauli-matrices:

$$\hat{\sigma}_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \hat{\sigma}_y = \begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}, \quad \hat{\sigma}_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{1.7}$$

The dynamics of a quantum state evolving under one of the Pauli matrices, i.e. $\hat{H} = \hat{\sigma}_k$, $k \in \{x, y, z\}$, with $t = \frac{\theta}{2}$ corresponds the rotational quantum gate

$$\mathrm{R}_k(\theta) \equiv \mathrm{e}^{-\mathrm{i}\frac{\theta}{2}\hat{\sigma}_k}. \tag{1.8}$$

These are called rotational quantum gates because they describe rotations around the Bloch-sphere about one of its three principal-axes. For example, acting with the quantum gate $\mathrm{R}_y(\pi/2)$ on the qubit state $|0\rangle$ corresponds to a $\pi/2$ anti-clockwise rotation around the y-axis, see Fig. 1.2.

Another important single-qubit gate is the Hadamard gate, denoted by H (not to be confused with the Hamiltonian $\hat{H}$). Given the input state of the qubit

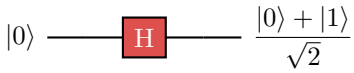


Figure 1.3: Hadamard gate

is $|0\rangle$ it will output the state $(|0\rangle+|1\rangle)/\sqrt{2}$, and given the input $|1\rangle$ it will output $(|0\rangle-|1\rangle)/\sqrt{2}$. The Hadamard gate can be written in matrix representation as

$$\mathrm{H}\,|\psi\rangle \iff \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix}, \tag{1.9}$$

or drawn as a quantum circuit as in Fig. 1.3. The Hadamard gate is equivalent to (up to a global phase) a $\pi/2$ rotation around the y-axis, followed by π rotation around the x-axis, $\mathrm{R}_x(\pi)\mathrm{R}_y(\pi/2) = -\mathrm{iH}$.

Besides the single-qubit gates, there are also multi-qubit gates, with the simplest one being two-qubit gates. A very important two-qubit gate is the controlled-NOT gate, or CNOT gate. It is a quantum gate that will flip the state of the second qubit if the first qubit is in state $|1\rangle$. Hence, $\mathrm{CNOT}\,|00\rangle = |00\rangle$[2] and $\mathrm{CNOT}\,|10\rangle = |11\rangle$, see Fig. 1.4.

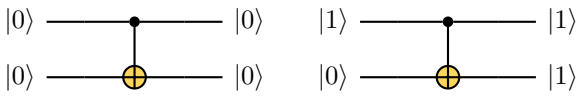


Figure 1.4: Controlled-NOT gate

Running a quantum algorithm

The execution of an algorithm on a classical computer can be broken down into three steps: load, run, and read [17]. The computer needs to load some input data together with a set of instructions on how to operate on the input data. It then needs to read out the result of the calculations. The corresponding steps on a quantum computer become: prepare, evolve, and measure. The quantum computer is prepared in some initial state, typically $|0\rangle^{\otimes n} = |0\rangle\otimes\ldots\otimes|0\rangle$. A set of quantum gates are then applied to the initial state, which then evolves into some final state $\sum_z \alpha_z\,|z\rangle$. This state is then measured, and the probability of observing z is $|\alpha_z|^2$. However, the output of a measurement is random, but as long as the solution string's output probability is high enough, we can repeat the process a couple of times to make the probability of success close to 100% [18].

[2]Here $|00\rangle$ is compact notation for $|0\rangle\otimes|0\rangle$, where $\otimes$ is the tensor product.

1.2 Outline of thesis

This thesis serves as an introduction to the appended papers where we both numerically and experimentally apply the QAOA to a combinatorial optimization problem.

The outline of the thesis is as follows: In Chapter 2 we introduce the notion of complexity theory and combinatorial optimization. This is followed by two examples, Max-Cut, and Exact Cover. We present these problems and their respective mapping to cost Hamiltonians. In Chapter 3 we introduce the QAOA and explain how it can be applied to the Max-Cut and the Exact Cover problems. We then demonstrate how the QAOA can be implemented on a quantum computer in terms of a universal gate set. Then, in Chapter 4 we discuss possibilities of running the QAOA in a continuous variable architecture. Finally, in Chapter 5 we give an overview of the appended papers.

2 Complexity and optimization

A combinatorial optimization problem concerns finding the best solution among a set of feasible candidates. A typical example that stems from traveling is the optimization problem of packing one's belongings such as clothes, hygiene products, etc., in a suitcase while not exceeding the weight limit of your baggage set by the airline company. This optimization problem is more formally known as the knapsack problem. It can be stated as: consider a set of items, where each item has a weight and a value. Pick a set of these items such that the total weight is less than or equal to a given limit such that the value is maximized.

It is possible to construct an algorithm for finding the best solution to an optimization problem, like the knapsack problem. A plethora of algorithms exist for solving it, but with different time complexities. The time complexity of an algorithm depends specifically on how the time to solve the problem scales with the size of the input n. An algorithm is said to be efficient if it has a polynomial running time with the size of the input [19].

As an example, consider the task of finding a number in a sorted list of length n. This can be done in $\mathcal{O}(n)$[1] using linear search, by stepping through the list one element at a time from start to end. In the worst-case, one has to go through the full list. However, using binary search, it is possible to find our number in $\mathcal{O}(\log n)$, by dividing the list in half and looking if our number is smaller or bigger than the element in the middle of the list. One can then discard the other half of the list and repeat the process with the remaining half.

As shown in this example, the running time is dependent on the algorithm. The theory of computational complexity, which is the next section's topic, concerns classifying classes of problems by their hardness. In particular, the hardness of a problem can be characterized by proving upper bounds on the resources, like time and memory, required by the best possible algorithm for solving that problem.

[1] $\mathcal{O}$ is called Big-O notation and a function $f(n)$ is $\mathcal{O}(g(n))$ if there exist a constant $c > 0$ and $n_0 \geq 0$ such that $f(n) \leq cg(n)$ for all $n \geq n_0$.

2.1 Complexity classes

Complexity theory is most easily formulated in terms of decision problems, meaning problems with a *yes* or *no* answer [20]. An example of a decision problem is to ask: is the number 17 prime? In this example, the answer is obvious *yes*. There exist a whole zoo of complexity classes[2], but luckily one does not need to know them all as a quantum computer scientist. Here we will introduce the most important types of complexity classes that appear in the context of optimization problems.

P
Is the complexity class that contains decision problems that can be solved in polynomial time by a deterministic algorithm. Or, alternatively, the decision problems that can be solved in polynomial time by a deterministic Turing machine. An example of a problem in P is determining if a number is prime [21].

BPP
Stands for bounded-error probabilistic polynomial time and is the class of problems that can be solved by a randomized algorithm in polynomial time. Or, alternatively, the complexity class that contains the decision problems that can be solved in polynomial time by a probabilistic Turing machine, with an error-probability less than 1/3. It is known that BPP can simulate P since a deterministic algorithm is a special case of a probabilistic one.

NP
Stands for non-deterministic polynomial time and contains the decision problems such that, when given a *yes* instance, it can be easily checked on a classical computer. Or more formally, the decision problems such that given a *yes* instance can be efficiently verified by a deterministic Turing machine. An example of an NP problem is prime factorization, which is to find two prime factors to an integer n. As mentioned in the introduction, this is a really hard problem to solve on a classical computer, which suggests that the problem is not in P. However, once given a number p, it can be quickly verified if it is a divisor of n by simply dividing n by p. It is conjectured and believed by most researchers that $\mathsf{P} \neq \mathsf{NP}$.

NP-complete
Is the class of problems in NP for which there exists a polynomial time-reduction algorithm of every problem other problem in NP to that problem. Therefore, this complexity class contains the hardest problems in NP in some sense, as if you would find a polynomial-time algorithm for

[2] At http://complexityzoo.com, you will find over 500 complexity classes!

solving an NP-complete problem, you could use the polynomial-time reduction for all other problems to also solve them in polynomial time. An example of an NP-complete problem is the Exact Cover.

NP-hard
Is the class of problems that are at least as hard as the NP-complete ones. They furthermore do not need to be decision problem. In detail, a problem is said to be NP-hard if an algorithm for solving it can be translated into an algorithm for solving any NP-problem. An example of an NP-hard problem is the Max-Cut.

BQP
Stands for bounded-error quantum polynomial time, and contains the class of decision problems that can be solved on a quantum computer in polynomial-time, with an error-probability less than 1/3. It is in some sense the quantum version of BPP.

Although it is not believed by the majority of scientists that quantum algorithms will be able to solve NP-complete problems in polynomial time, it is known that quantum algorithms can solve some problems in NP efficiently, most notably prime factoring using Shor's algorithm.

Moreover, quantum algorithms could also prove useful if they have a *lower* time complexity than the best classical algorithm. An example of this is the unconstructed search problem, where the best classical algorithm has to use linear search with time complexity $\mathcal{O}(n)$, while using Grover's quantum search algorithm can solve the problem in $\mathcal{O}(\sqrt{n})$ [22].

Furthermore, when a problem is computationally hard, i.e. when the only way to solve it is by making use of an algorithm that runs in exponential time, it may be unfeasible to try and compute the exact solution, because it might require months or even years of computer time. In such cases, one may want to resolve to use an *approximate algorithm* instead, which can efficiently compute a solution, though not being an optimal one, it has some provable performance guarantee on the optimality of the returned solution [23]. With this motivation, researchers have started looking for approximate quantum algorithms, such as the QAOA, which is introduced in chapter 3.

2.2 Combinatorial optimization and the Ising model

To solve a combinatorial optimization problem using a quantum algorithm, the quantum algorithm must be able to encode the specific problem that we wish to solve. This can be done by encoding the optimization problem onto a quantum system. In this section we will see how a combinatorial optimization can be framed as a cost Hamiltonian in the Ising form.

Let $C : \{0,1\}^n \to \mathbb{R}$ be a cost function that encodes a combinatorial optimization problem. There are in total 2^n possible strings and the goal is to find the bit-string $z = z_1 \dots z_n$ that maximizes the cost function $C(z)$. Note that a maximization problem can be transformed into and minimization problem by a minus sign $C(z) \to -C(z)$.

One of the most widely used models in physics that is used to represent optimization problems is the Ising model [24]. It is was developed in the 1920s by Ernst Ising and Wilhelm Lenz as a way to understand phase transitions in magnetic materials [25]. The Ising model can be thought of as n Ising spins that sit on a lattice and that can take the values $s_i \pm 1$, where s_i refers to the i:th Ising spin. These spins are coupled together through long-range magnetic interactions that encourage the spins be aligned or anti-aligned. Moreover, an external magnetic field can be applied at each individual spin site which will give a different energy to the two possible spin directions. The total energy of the Ising model consisting of n Ising spins is given by [26]

$$E(s_1, \dots, s_n) = \sum_{1 \le i < j \le n} J_{ij} s_i s_j + \sum_{i=1}^{n} h_i s_i, \quad (s_i = \pm 1), \tag{2.1}$$

where J_{ij} is the coupling strength between the i:th and j:th spin, and h_i is the magnetic field acting on the i:th spin. A quantum version of this model is obtained by simply replacing the spin-variables s_i with Pauli-z operators

$$\hat{H}_C \equiv \hat{H}(\hat{\sigma}_1^z, \dots, \hat{\sigma}_n^z) \sum_{1 \le i < j \le n} J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z + \sum_{i=1}^{n} h_i \hat{\sigma}_i^z, \tag{2.2}$$

were $\hat{\sigma}_i^z$ refers to the Pauli-z matrix acting on the i:th qubit. Many optimization problems, including all of Karp's 21 NP-complete problems, can be written in the form of Eq. (2.2), by choosing appropriate values for J_{ij} and h_i [27]. Furthermore, the spectral decomposition of the this Hamiltonian encodes the different solutions in the computational basis

$$\hat{H}_C = \sum_{z \in \{0,1\}^n} C(z) \, |z\rangle\langle z| \, , \tag{2.3}$$

where $C(z)$ is the cost function. The Hamiltonian of Eq. (2.2) is formally known as an Ising-Hamiltonian, but we will refer to this Hamiltonian as a *cost Hamiltonian*, because its eigenvalues in the computational basis correspond to the possible values of the cost function.

In the two next sections, we will introduce some notable examples of combinatorial optimization problems, and derive their respective cost Hamiltonians in the form of Eq. (2.2)

2.3 Max-Cut

The Max-Cut problem has application in circuit design [28] and is one of the most extensively studied problems in the context of the QAOA [29–32]. The

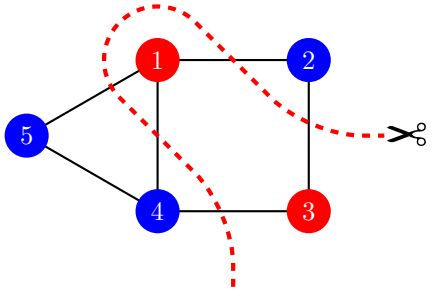


Figure 2.1: A maximum cut of a graph with 5 vertices. The dashed red line corresponds to the cut edges. An edge is cut if two vertices connected by an edge are assigned different colors.

objective of Max-Cut is to partition the set of vertices of a graph into two subsets, such that the sum of the edge weights going from one partition to the other is maximum. Max-Cut is NP-hard because it is not a problem with a *yes* or *no* answer. However the decision version of Max-Cut which asks whether there is a cut of at least size k in a graph is NP-complete [33, 34].

Given an undirected graph $G = (V, E)$, where V is the set of vertices, E is the set of edges with nonnegative edge weights $w_{ij} = w_{ji} : (i, j) \in E$, the formulation of Max-Cut is given by:

$$\text{maximize} \quad \frac{1}{2} \sum_{1 \leq i < j \leq n} w_{ij}(1 - s_i s_j), \tag{2.4}$$

$$\text{subject to:} \; s_i \in \{-1, 1\} \quad i \in V. \tag{2.5}$$

An example of a graph as well as its maximum cut is shown in Fig 2.1.

To map this problem onto a cost Hamiltonian all we have to do is to replace the classical variables s_i with Pauli-z matrices. The corresponding Max-Cut Hamiltonian then reads

$$\hat{H}_C = \frac{1}{2} \sum_{1 \leq i < j \leq n} w_{ij}(1 - \hat{\sigma}_i^z \hat{\sigma}_j^z). \tag{2.6}$$

The eigenstate to this Hamiltonian with the highest eigenvalue corresponds to the maximum cut.

2.4 Exact Cover

The Exact Cover problem is an NP-complete problem [33, 34]. Moreover, it is this problem to which we apply the QAOA in paper A and B. The task in the Exact Cover is given a collection of subsets V of a set U, find a sub-collection of V called R, such that the each element of U is contained in exactly one of the

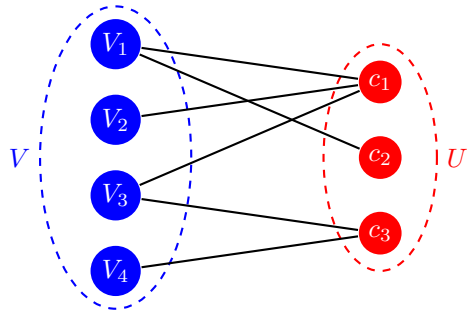


Figure 2.2: An example of the Exact Cover visualized as a bipartite graph. There is an edge from V to U if subset V_i contain the element c_j.

subsets of R. A notable example of an Exact Cover problem is Sudoku, but it also appears in aviation, such as a simplified case of the tail assignment problem [35].

In detail: given a set a set $U = \{c_1, c_2, \ldots, c_n\}$, and a set of subsets $V = \{V_1, \ldots, V_m\}$ with $V_i \subset U$ such that $U = \bigcup_{i=1}^{m} V_i$, the Exact Cover is

$$\sum_{j \in V} K_{ij} z_j = 1, \quad \forall i \in U, \tag{2.7}$$

$$\text{subject to: } z_j \in \{0, 1\}, \tag{2.8}$$

where K_{ij} is the ij:th element of an *incidence matrix* that is 1 if $c_i \in V_j$, and 0 otherwise, and the binary variables z_j represents if the subset V_j should be included in the Exact Cover set R or not. The Exact Cover can moreover be visualized as a bipartite graph as shown in Fig 2.2.

To map the Exact Cover problem onto a cost Hamiltonian, the firsts step is to transform Eq. (2.7) into a cost function. This can be done by subtracting 1 from the r.h.s of Eq. (2.7) and squaring the expression:

$$C(z) = \sum_{i=1}^{u} \left(\sum_{j=1}^{v} K_{ij} z_j - 1 \right)^2 . \tag{2.9}$$

Here $u \equiv |U|$ denotes the cardinality of U and $v \equiv |V|$ denotes the cardinality of V. There exists a solution if and only if there exists a string z such that $C(z)$ is zero (corresponding to a *yes* answer). Note that all the other strings corresponding to a *no* answer will have a positive cost. Therefore we want to minimize the cost for the Exact Cover.

Next, we want to write this cost function in the Ising form Eq. (2.1). This is done by substituting the binary variables $z_j \in \{0, 1\}$ with spin variables $s_j \in \{1, -1\}$,

$$z_j = \frac{1 - s_j}{2}.$$

By using this substitution and expanding the square of Eq. (2.9) we obtain the *Ising energy function* for the Exact Cover problem

$$E(s_1, \ldots, s_v) = \sum_{1 \leq i < j \leq v} J_{ij} s_i s_j + \sum_{i=1}^{v} h_i s_i, \tag{2.10}$$

with J_{ij} and h_i defined by[3]

$$J_{ij} \equiv \frac{1}{2} \sum_{k=1}^{u} K_{ki} K_{kj}, \quad \text{and} \quad h_i \equiv \frac{1}{2} \sum_{j=1}^{u} K_{ji} \left(\sum_{k=1}^{v} K_{jk} - 2 \right). \tag{2.11}$$

Finally, we quantize Eq. (2.10) by promoting the spin variables s_i to Pauli-z matrices as $s_i \to \hat{\sigma}_i^z$. The final cost Hamiltonian for the Exact Cover problem is then

$$\hat{H}_C = \sum_{1 \leq i < j \leq v} J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z - \sum_{i=1}^{v} h_i \hat{\sigma}_i^z. \tag{2.12}$$

The ground state of this Hamiltonian corresponds to the bit-string that minimizes the cost function C.

[3]For the full mathematical derivation we refer the reader to the appended paper A.

3 The quantum approximate optimization algorithm

The quantum approximate optimization algorithm (QAOA) is a promising quantum algorithm for solving combinatorial optimization problems. It was invented by Farhi *et al.* in 2014 [36] and has since then spurred a lot of interest in the scientific community for its simplicity and its possibility to run on near term NISQ devices.

3.1 From the quantum adiabatic algorithm to the QAOA

The QAOA is inspired by the quantum adiabatic algorithm (QAA), which was also invented by Farhi *et al.* [37]. The main idea of QAA is to cleverly take advantage of adiabatic evolution, to go from the highest energy eigenstate of an initial Hamiltonian that is easy to prepare, to the highest energy state of a cost Hamiltonian[1]. The QAA Hamiltonian is written as a sum of two non-commuting Hamiltonians,

$$\hat{H}(t) = (1 - s(t))\hat{H}_B + s(t)\hat{H}_C. \tag{3.1}$$

Here $s(0) = 0$ and $s(T) = 1$, T is the total time of the algorithm, $\hat{H}_B$ is the initial Hamiltonian, whose highest energy eigenstate is easy to prepare, and $\hat{H}_C$ is the cost Hamiltonian whose highest energy eigenstate encodes the solution to an optimization problem. A linear time-dependence is commonly assumed where the time-dependent function take the form $s(t) = t/T$. In order to have a high probability of ending up in the optimal state of the cost Hamiltonian, the total time T should be $\mathcal{O}(1/\Delta E_{\text{min}}^2)$, where ΔE_{min} is the minimum energy gap between the two highest energy eigenstates during the evolution [38]. Unfortunately, there are results indicating that QAA requires computation time that is exponential in the number of variables n to reach the optimal state of the cost Hamiltonian [39–41].

[1] Usually QAA is formalized as seeking the lowest energy state; but to make notation uniform with those in the QAOA, we consider the excited state variant.

The QAOA is based on the observation that the easiest way to simulate the QAA is to Trotterize the evolution [42]

$$\hat{U}(T) \equiv \mathcal{T} \exp\left[-\mathrm{i}\int_0^T \hat{H}(t)\,\mathrm{d}t\right] \approx \prod_{k=1}^{p} \exp\Big[-\mathrm{i}\hat{H}(k\Delta t)\Delta t\Big]. \tag{3.2}$$

Here $\hat{U}(T)$ is the evolution operator from 0 to T, $\mathcal{T}$ is the time-ordering operator, and p is a large integer so that $\Delta t = T/p$ is a small time segment. Next, for two non-commuting operators A and B and sufficiently small Δt, one can use the Trotter formula:

$$\mathrm{e}^{\mathrm{i}(A+B)\Delta t} = \mathrm{e}^{\mathrm{i}A\Delta t}\mathrm{e}^{\mathrm{i}B\Delta t} + \mathcal{O}(\Delta t^2), \tag{3.3}$$

and apply it to the discretized time evolution operator (3.2)

$$\hat{U}(T) \approx \prod_{k=1}^{p} \exp\Big[-\mathrm{i}(1-s(k\Delta t))\hat{H}_B\Delta t\Big] \exp\Big[-\mathrm{i}s(k\Delta t)\hat{H}_C\Delta t\Big]. \tag{3.4}$$

Thus, it is possible to approximate the QAA by applying $\hat{H}_C$ and $\hat{H}_B$ in an alternating sequence.

The brilliant yet very simple idea that came from Farhi, Goldstone, and Gutman was to truncate this product to an arbitrary positive integer and redefine the time dependence in each exponent $(1-s(k\Delta t))\Delta t \to \beta_k$ and $s(k\Delta t)\Delta t \to \gamma_k$, such that the fixed time segments become angles to be optimized:

$$\hat{U} = \prod_{k=1}^{p} \mathrm{e}^{-\mathrm{i}\beta_k\hat{H}_B}\mathrm{e}^{-\mathrm{i}\gamma_k\hat{H}_C}, \quad p \in \mathbb{Z}^+. \tag{3.5}$$

Then by choosing $\hat{H}_B$ to be

$$\hat{H}_B = \sum_i \hat{\sigma}_i^x, \tag{3.6}$$

where $\hat{\sigma}_i^x$ refers to the Pauli-x matrix on the i:th qubit, and letting $\hat{U}$ in Eq. (3.5) act on the superposition of all possible states in the computational basis with equal probability

$$|+\rangle^{\otimes n} \equiv H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{z\in\{0,1\}^n} |z\rangle, \tag{3.7}$$

the final variational "QAOA" state is obtained

$$|\psi_p(\vec{\gamma},\vec{\beta})\rangle \equiv \prod_{k=1}^{p} \left(\mathrm{e}^{-\mathrm{i}\beta_k\hat{H}_B}\mathrm{e}^{-\mathrm{i}\gamma_k\hat{H}_C}\right)|+\rangle^{\otimes n}. \tag{3.8}$$

Here $\vec{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_p)$ and $\vec{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)$. Each β_k lies in the interval between 0 and π. This can be seen by inserting $\beta_k \to \beta_k \pm \pi$ into Eq. (3.8)

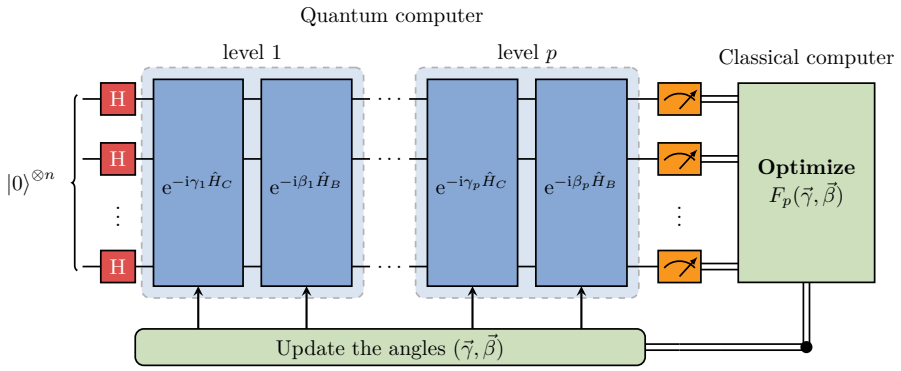


Figure 3.1: Schematic representation of the QAOA. The quantum processor prepares the variational state, depending on the angles. The angles $(\vec{\gamma}, \vec{\beta})$ are optimized in a closed loop using a classical optimizer.

and noting that $e^{\pm i\pi \hat{H}_B} = \prod_i e^{\pm i\pi \hat{\sigma}_i^x} = \prod_i (-I_i)$, is minus the identity on all the qubits, which is mere a global phase. Likewise, if the eigenvalues of the cost Hamiltonian are all integers, it can be shown that γ_k lies between 0 and 2π, for similar reasons. Still, the task of choosing the angles $(\vec{\gamma}, \vec{\beta})$, remains. Let $F_p(\vec{\gamma}, \vec{\beta})$ be the expectation value of $\hat{H}_C$ in this state

$$F_p(\vec{\gamma}, \vec{\beta}) \equiv \langle \psi_p(\vec{\gamma}, \vec{\beta}) | \hat{H}_C | \psi_p(\vec{\gamma}, \vec{\beta}) \rangle \,. \tag{3.9}$$

By finding good angles $\vec{\gamma}$ and $\vec{\beta}$ that maximize the expectation value above, the probability of finding the qubits in a high energy configuration when measuring is increased. Therefore the angles are chosen such that the expectation value is maximized:

$$F_{\max} \equiv \max_{\vec{\gamma}, \vec{\beta}} F_p(\vec{\gamma}, \vec{\beta}). \tag{3.10}$$

In general, this requires the quantum computer to query a classical optimizer, to tell the quantum computer how it should update the variational state by slightly changing the angles to maximize the expectation value, see Fig. 3.1.

Considerable research has been conducted on the classical optimization part of the QAOA [43]. Several numerical investigations have examined different classical optimizers [44, 45]. Other studies have found heuristic methods that enhance the classical optimization procedure [32, 46]. However, in practice, noise and finite sampling error put unique challenges on the optimizers. In the handful of experiments that have run the QAOA [47–49], Bayesian optimization [50], Nelder-mead [51], and Model gradient descent [49] are a few of the optimizers that have been implemented.

In short, the QAOA can be summarized as follows:

1. Pick a positive integer p and start with an initial set of $2p$ angles $(\vec{\gamma}, \vec{\beta})$.

2. Construct the state $|\psi_p(\vec{\gamma},\vec{\beta})\rangle$ using a quantum computer and measure this state in the computational basis. The output is a string z with a probability given by the distribution of states $|z\rangle$.

3. Calculate $C(z)$ using a classical computer. This step is classically efficient.

4. Repeat step 2-3, m times. Record the best observed string z_{best}, and the sample mean $1/m\sum_{i=1}^{m} C(z_i)$, where z_i is the i:th measurement outcome. Note that when $m \to \infty$ the sample mean approaches the expectation value Eq. (3.9) by the law of large numbers.

5. If the optimal or a "good enough" solution is found, output $C(z_{\text{best}})$ together with the string z_{best}. Else, query a classical optimizer that updates the angles $(\vec{\gamma},\vec{\beta})$ based on the minimization of the expectation value and repeat from step 2.

3.2 QAOA for Max-Cut

In the first publication of the QAOA, the authors Farhi *et al.* applied it to the Max-Cut problem. In particular, they showed that the QAOA for $p = 1$ will always yield a cut that is at least .6924 times the maximum cut for all 3-regular graphs. These are graphs where every vertex's degree is 3, i.e., every vertex is connected by an edge to three other vertices. Following the spirit of Ref. [36], we will demonstrate exactly how this result can be obtained.

To begin, we define the approximation ratio of a graph to be

$$\frac{F_p(\vec{\gamma},\vec{\beta})}{C_{\max}}, \tag{3.11}$$

where $F_p(\vec{\gamma},\vec{\beta})$ is given by Eq. (3.9) and corresponds to the average cut produce by the QAOA for Max-Cut, and $C_{\max}$ is the value of the maximum cut for the graph. The approximation ratio is a value between 0 and 1, which is a measure of how close the variational state $|\psi_p(\vec{\gamma},\vec{\beta})\rangle$ is to the optimal state. A value of 1 means that the variational state is equal to the optimal state. It is this approximation ratio that we will show is guaranteed to not be less than .6924 on 3-regular graphs.

Recall the Max-Cut Hamiltonian from Eq. (2.6). The expectation value for $p = 1$ of this cost Hamiltonian with unit edge weights ($w_{ij} = 1$), can be expressed as a sum over the individual edges' expectation values

$$F_1(\gamma,\beta) = \sum_{(i,j)\in E} f_{\langle ij\rangle}(\gamma,\beta), \tag{3.12}$$

$$\text{with} \quad f_{\langle ij\rangle}(\gamma,\beta) = \frac{1}{2}\langle\psi_p(\gamma,\beta)|1-\hat{\sigma}_i^z\hat{\sigma}_j^z|\psi_p(\gamma,\beta)\rangle\,. \tag{3.13}$$

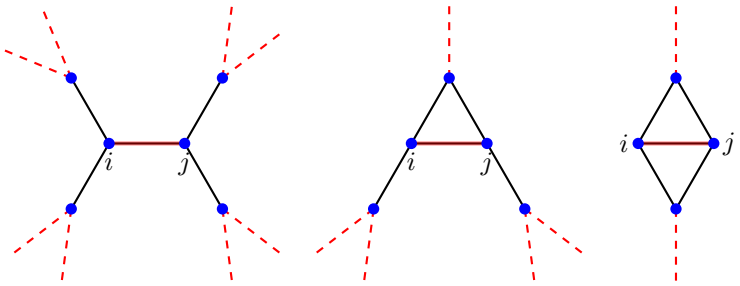


Figure 3.2: The three possible subgraphs for $p = 1$. The red dashed lines represent edges connecting to the rest of the graph, while the red solid line represent the $\langle ij \rangle$ edge.

Focusing on one pair of edges $\langle ij \rangle$, the expectation value for that edge is

$$\frac{1}{2} - \frac{1}{2} \langle + |^{\otimes n} \, \mathrm{e}^{\mathrm{i}\gamma \hat{H}_C} \mathrm{e}^{\mathrm{i}\beta \hat{H}_B} \hat{\sigma}_i^z \hat{\sigma}_j^z \mathrm{e}^{-\mathrm{i}\beta \hat{H}_B} \mathrm{e}^{-\mathrm{i}\gamma \hat{H}_C} \, | + \rangle^{\otimes n} . \tag{3.14}$$

Since $\mathrm{e}^{-\mathrm{i}\beta \hat{H}_B} = \mathrm{e}^{-\mathrm{i}\beta \sum_{k=1}^n \hat{\sigma}_k^x}$ and $\hat{\sigma}_i^z \hat{\sigma}_j^z$ only involves qubit i and j, all $\hat{\sigma}_k^x$ terms in $\mathrm{e}^{-\mathrm{i}\beta \hat{H}_B}$ which do not involve qubit i or j commutes through $\hat{\sigma}_i^z \hat{\sigma}_j^z$ and cancel, leaving us with

$$\frac{1}{2} - \frac{1}{2} \langle + |^{\otimes n} \, \mathrm{e}^{\mathrm{i}\gamma \hat{H}_C} \mathrm{e}^{\mathrm{i}\beta (\hat{\sigma}_i^x + \hat{\sigma}_j^x)} \hat{\sigma}_i^z \hat{\sigma}_j^z \mathrm{e}^{-\mathrm{i}\beta (\hat{\sigma}_i^x + \hat{\sigma}_j^x)} \mathrm{e}^{-\mathrm{i}\gamma \hat{H}_C} \, | + \rangle^{\otimes n} . \tag{3.15}$$

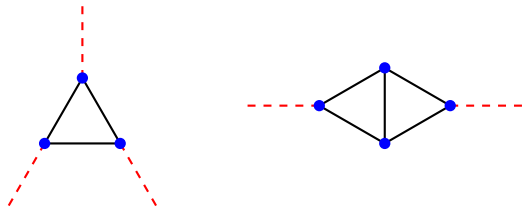
Likewise for $\mathrm{e}^{-\mathrm{i}\gamma \hat{H}_C}$, all the terms which do not involve qubit i or j commutes through $(\hat{\sigma}_i^x + \hat{\sigma}_j^x)$ and cancels out. Hence the expectation value for the $\langle ij \rangle$ edge involves vertex i and j, and those vertices that are adjacent to i and/or j.

Therefore, vertex i and j together with their adjacent vertices form a subgraph. For 3-regular graphs, there exist exactly three such types of subgraphs that they can form, see Fig. 3.2. Other pairs of qubits in Eq. (3.12) will also form one of these three subgraphs types. Isomorphic subgraphs will correspond to the same expectation value for the same value of (γ, β). Hence we can express the expectation value as the occurrences of each subgraph type

$$F_1(\gamma, \beta) = \sum_\lambda N_\lambda f_\lambda(\gamma, \beta) = N_{\diagup\!\!\diagdown} f_{\diagup\!\!\diagdown}(\gamma, \beta) + N_{\triangle} f_{\triangle}(\gamma, \beta) + N_{\diamond} f_{\diamond}(\gamma, \beta), \tag{3.16}$$

where N_λ is the number of subgraphs that look like λ and $f_\lambda(\gamma, \beta)$ is the expectation value of that subgraph.

The number of subgraph occurrences are not independent of one another. Suppose that a graph has n vertices with accordingly $3n/2$ edges, and assume that it contains $N_\triangle$ isolated triangles and $N_\diamond$ diamonds, see Fig. 3.3. An isolated triangle means that the outgoing edges of the triangle land on distinct vertices.



(a) (b)

Figure 3.3: (a) Triangle graph. (b) Diamond graph. The red dashed lines represent edges connecting to the rest of the graph.

In general $3N_{\triangle}+4N_{\diamondsuit} \leq n$, because no isolated triangle and diamond can share a vertex. Every isolated triangle gives rise to 3 subgraphs that looks like Fig. 3.2b. Similarly, there are 4 edges in a diamond that give rise to a subgraph that also looks like Fig. 3.2b. Hence, $N_{\bigwedge} = 4N_{\diamondsuit} + 3N_{\triangle}$, and the remaining edges must therefore form tree subgraphs $N_{\rightthreetimes} = 3n/2 - 5N_{\diamondsuit} - 3N_{\triangle}$.

A lower bound on the approximation ratio can be obtained by the fraction of the two terms $F_{\max}/C_{\max}$. The value of $F_{\max}$ is bounded from below by

$$F_{\max} = \max_{\gamma,\beta} F_1(\gamma,\beta) \geq$$
$$\geq N_{\diamondsuit} f_{\diamondsuit}(\gamma,\beta) + (4N_{\diamondsuit} + 3N_{\triangle}) f_{\bigwedge}(\gamma,\beta) + \left(\frac{3n}{2} - 5N_{\diamondsuit} - 3N_{\triangle}\right) f_{\rightthreetimes}(\gamma,\beta). \quad (3.17)$$

An exact value of $C_{\max}$ is hard to find. Nevertheless, an upper bound is enough to lower bound the approximation ratio. There are in total $3n/2$ edges. But for every isolated triangle and diamond, there must be at least one uncut edge. This sets the upper limit of $C_{\max}$ to $\leq 3n/2 - N_{\diamondsuit} - N_{\triangle}$. The approximation ratio is therefore lower bounded by

$$\frac{F_{\max}}{C_{\max}} \geq$$
$$\geq \frac{N_{\diamondsuit} f_{\diamondsuit}(\gamma,\beta) + (4N_{\diamondsuit} + 3N_{\triangle}) f_{\bigwedge}(\gamma,\beta) + (\frac{3n}{2} - 5N_{\diamondsuit} - 3N_{\triangle}) f_{\rightthreetimes}(\gamma,\beta)}{\frac{3n}{2} - N_{\diamondsuit} - N_{\triangle}}. \quad (3.18)$$

The right hand side is a function of $N_{\diamondsuit}$, $N_{\triangle}$, γ, and β. It is favorable to multiply the numerator and denominator by $1/n$, and redefine $N_{\diamondsuit}/n \equiv n_{\diamondsuit}$ and $N_{\triangle}/n \equiv n_{\triangle}$, so that we can write

$$\frac{n_{\diamondsuit} f_{\diamondsuit}(\gamma,\beta) + (4n_{\diamondsuit} + 3n_{\triangle}) f_{\bigwedge}(\gamma,\beta) + (\frac{3}{2} - 5n_{\diamondsuit} - 3n_{\triangle}) f_{\rightthreetimes}(\gamma,\beta)}{\frac{3}{2} - n_{\diamondsuit} - n_{\triangle}}, \quad (3.19)$$

where $n_{\diamondsuit}, n_{\triangle} \geq 0$ and $4n_{\diamondsuit}+3n_{\triangle} \leq 1$. By numerically minimizing over $n_{\diamondsuit}$ and $n_{\triangle}$ while maximizing over the angles (γ,β), we find that the minimum is obtained

when $n_{\diamondsuit} = n_{\triangle} = 0$ with $(\gamma, \beta) = (35.3°, 22.5°)$, and that the corresponding approximation ratio is lower bounded by .6924. Thus the worst graph is the one that is only made up of tree subgraphs, Fig. 3.2a.

This lower bound on the approximation ratio is better than random guessing. However it is worse than the current best classical approximation algorithm for Max-Cut, which gives an approximation ratio of .87856 for all graphs [52], and .9326 for 3-regular graphs [53]. Nevertheless, the possibility that quantum advantage might exist for higher level p is not ruled out.

3.3 QAOA for Exact Cover

Although the QAOA was initially introduced as an approximation algorithm for Max-Cut, nothing prohibits us from using it to try and solve decision problems. In fact, because the QAOA can be thought of as a Trotterization of the QAA, in the limit of large p, there is exist a set of angles $\vec{\gamma}$ and $\vec{\beta}$ that make the overlap between the optimal state close to unity

$$\lim_{p\to\infty}\left(\max_{\vec{\gamma},\vec{\beta}} F_p(\vec{\gamma},\vec{\beta})\right) = \max_z C(z). \tag{3.20}$$

For a decision problem it is possible to define the success probability of the QAOA as the sum of the probability amplitudes over all feasible solutions f (all the "yes" answers)

$$\sum_{z\in f} \left|\langle z|\psi_p(\vec{\gamma},\vec{\beta})\rangle\right|^2. \tag{3.21}$$

As a demonstration, we will show how the QAOA solves a simple exact-cover problem. Consider the problem where $U = \{c_1, c_2\}$ and $V = \{V_1, V_2\}$, with $V_1 = \{c_1, c_2\}$ and $V_2 = \{0, c_2\}$. It has the solution $|10\rangle$. The cost Hamiltonian for this problem is $\hat{H}_C = J\hat{\sigma}_1^z\hat{\sigma}_2^z - h_1\hat{\sigma}_1^z - h_2\hat{\sigma}_2^z$ with $J = 1/2$, $h_1 = -1/2$, and

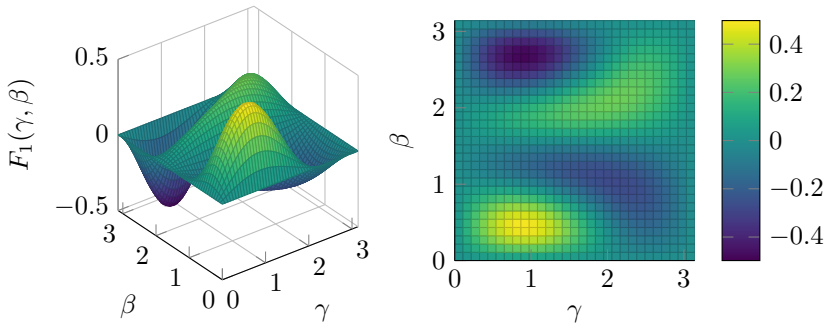


Figure 3.4: The expectation value Eq. (3.22) as a surface plot (left) and contour plot (right).

$h_2 = 0$, that we get from Eq. (2.11). For $p = 1$, it is possible to write down an analytical expression of the expectation value Eq. (3.9) by performing the matrix vector multiplication:

$$F_1(\gamma, \beta) = \frac{1}{2}\big(\cos(2\beta) + 2\cos^2(\beta)\cos(\gamma)\big)\sin 2\beta \sin(\gamma). \tag{3.22}$$

This expectation value can be visualized on a grid, see Fig. 3.4. From the grid the optimal values $(\gamma_{\text{opt}}, \beta_{\text{opt}})$ can be read off. Note that since we seek the lowest energy eigenstate of the Exact Cover cost Hamiltonian, it is favorable to minimize the expectation value $F_1(\gamma, \beta)$ instead of maximizing it. In this example, there is only one feasible solution, and we find that the success probability is 50% for $p = 1$ using the angles that minimize Eq. (3.22). By numerically minimizing $F_2(\vec{\gamma}, \vec{\beta})$ we find a 100% success probability for $p = 2$. This example demonstrates that the QAOA can be used as a means for solving decision problems. However, since there exists no known lower bound on the success probability, one has to simply run the algorithm and see how it performs.

3.4 Implementation of the QAOA

The QAOA has already been implemented on several different hardware systems, including photonic [54], trapped ion [55], and superconducting quantum processors [47–49, 56]. The unitary matrices that create the variational state $|\psi_p(\vec{\gamma}, \vec{\beta})\rangle$ needs to be decomposed into 1-qubit and 2-qubit gates in order to run on an actual quantum computer. In general, the decomposition will depend on the primitive quantum gates for the specific hardware. Here we will give an example of how the variational state $|\psi_p(\vec{\gamma}, \vec{\beta})\rangle$ can be constructed in terms of single and two-qubit gates starting from the $|0\rangle^{\otimes n}$ state and using the following universal gate set: $\{\mathrm{R}_x(\theta), \mathrm{R}_z(\theta), \mathrm{CZ}\}$. Here CZ is the control-Z gate that applies a (-1) phase to the $|11\rangle$ state. This is, moreover, the gate-set that we used in paper B [48].

The starting state $|+\rangle^{\otimes n}$ of the QAOA can be constructed by applying the Hadamard gate to each individual qubit in the $|0\rangle^{\otimes n}$ state Eq. (3.7). However, since the Hadamard gate is not in the available gate set it has to be compiled using a product of gates. This can be achieved by a $\pi/2$ rotation around the x-axis of the Bloch-sphere, followed by a $\pi/2$ rotation around the z-axis, and yet another $\pi/2$ rotation around the x-axis: $\mathrm{H} = \mathrm{i}\mathrm{R}_x\big(\frac{\pi}{2}\big)\mathrm{R}_z\big(\frac{\pi}{2}\big)\mathrm{R}_x\big(\frac{\pi}{2}\big)$.

Next, the unitary that involves the sum of Pauli-x matrices can be written as a product because all terms in the Hamiltonian commute

$$\mathrm{e}^{-\mathrm{i}\beta\hat{H}_B} = \mathrm{e}^{-\mathrm{i}\beta\sum_{i=1}^{n}\hat{\sigma}_i^x} = \prod_{i=1}^{n}\mathrm{e}^{-\mathrm{i}\beta\hat{\sigma}_i^x}. \tag{3.23}$$

This unitary can be implemented as n parallel single-qubit rotations around the

x-axis of the Bloch sphere with the same angle. The rotation on qubit i is

$$e^{-i\beta\hat{\sigma}_i^x} \equiv \quad \boxed{R_x(2\beta)} \tag{3.24}$$

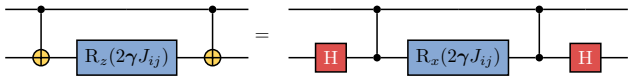
The cost Hamiltonian Eq. (2.3), consists of two parts: a two-body Hamiltonian $\hat{\sigma}_i^z\hat{\sigma}_j^z$ and a single-body Hamiltonian $\hat{\sigma}_i^z$:

$$e^{-i\gamma\hat{H}_C} = \prod_{1\leq i<j\leq n} e^{-i\gamma J_{ij}\hat{\sigma}_i^z\hat{\sigma}_j^z} \prod_{i=1}^{n} e^{-i\gamma h_i\hat{\sigma}_i^z}. \tag{3.25}$$

All terms in this product commute so the order with which they are applied does not matter. Starting with the single-qubit term, it can be implemented as n rotations around the z-axis of the Bloch-sphere, where the rotation for the i:th qubit is

$$e^{-i\gamma h_i\hat{\sigma}_i^z} \equiv \quad \boxed{R_z(2\gamma h_i)} \tag{3.26}$$

The two-qubit interaction $\hat{\sigma}_i^z\hat{\sigma}_j^z$ in the cost Hamiltonian can be implemented using a local single-qubit gate between two CNOT gates [29]

$e^{-i\gamma J_{ij}\hat{\sigma}_i^z\hat{\sigma}_j^z} \equiv$ 

(3.27)

Since CNOT is not in our available gate set we have used the following identity to express the CNOT gates into CZ gates:

(3.28)

where X and Z are the Pauli-x and z matrices. A problem where every element of J_{ij} is non-zero would require a total of $n(n-1)$ CZ gates for each level p. This is of course under the assumption that it is possible to apply the two-qubit gates directly between any two qubits. In reality, quantum processors have limited hardware connectivity. Still, SWAP gates can be used to move distant qubits, see Fig. 3.5. The number of SWAP gates needed to make all the qubits interact

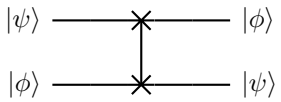


Figure 3.5: The SWAP gate swaps the state of two qubits.

with each other depends on the connectivity of the hardware. For example, in a linear array of qubits a swap network can be implemented using $\mathcal{O}(n)$ number of SWAP gates [29].

3.5 History and further reading

We will end this chapter with a list of results in connection to the QAOA, taking a historical perspective. As mentioned, the QAOA was invented by Farhi, Goldstone, and Gutmann. After their initial publication, where they had applied this algorithm to the Max-Cut, they applied it to the Max E3LIN2 problem. For this problem, they were able to show that the QAOA can achieve a better approximation ratio than the best classical approximation algorithm [57]. This was not happily accepted by the computer science community that came together to flex their muscles and soon came up with a better classical algorithm [58].

In 2018, Brandao *et al.* [59] demonstrated that if the problem instances come from a reasonable distribution[2], then the expectation value of the cost function concentrates. Meaning that for fixed angles $(\vec{\gamma}, \vec{\beta})$, the expectation value will be the same on all typical instances. This suggests that it is possible to train a classical optimizer to find good angles on small instances and reuse those angles on larger instances, as long as they come from the same distribution.

Moreover, it has been shown that the QAOA is universal, meaning that for a problem of size n, and a choice of $\hat{H}_C$ and $\hat{H}_B$, the QAOA can approximate any unitary U of dimension $2^n \times 2^n$ to arbitrary precision [60, 61]. Also, it has been shown that the QAOA is able to realize Grover's search algorithm [62, 63].

Later on, Farhi *et al.* argued that for certain choices of $\hat{H}_C$ and (γ, β), level $p = 1$ is computationally hard to simulate on a classical computer without collapsing the polynomial hierarchy to the third level [64]. However, this would not imply that $\mathsf{P} = \mathsf{NP}$.

In 2019, Hadfield *et al.* created the *quantum operator alternating ansatz*, which generalizes the original QAOA ansatz to allow for more general types of Hamiltonians and initial states [65].

In 2020 Farhi *et al.* showed that for certain types of graphs, the algorithm has limited power when p is less than $\mathcal{O}(\log n)$ [66, 67]. This is because qubits in a graph that are further than $2p$ edges apart can not influence each other, which will make the measurement of these qubits uncorrelated. The same year, Wurtz and Love [68] conjectured that the approximation ratio for Max-Cut on 3-regular graphs for $p = 3$ is .7924, and speculated that for $p < 6$, there is no quantum advantage.

[2] For example, random graphs, where each edge is included with a probability p.

4 Discussion and outlook

In this thesis, we have studied the working principles of the QAOA and shown how performance bounds can be derived for the Max-Cut problem and how the QAOA can be applied to the Exact Cover problem. We have also demonstrated how the QAOA can be compiled onto a quantum circuit in terms of a universal-gate set.

The QAOA has shown to be an extremely versatile algorithm that can be applied to a broad set of combinatorial optimization problems [35, 36, 69, 70]. As long as the problem can be mapped onto a cost Hamiltonian, the QAOA can be used as a means to solve it. Although quantum advantage remains to be seen, the application of the QAOA to combinatorial optimization problems is still a fascinating topic to research and to run on NISQ devices.

For many levels of the QAOA, i.e. for large p, there is a high probability of measuring the optimal solution. However, in reality, there exists a trade-off between the number p and the algorithm's performance [49]. This is because quantum circuits are prone to errors due to decoherence mechanisms. To battle these surreptitious errors, quantum error-correcting (QEC) codes has to be implemented. A good candidate for this is continuous variable encoding, which has shown great success in realizing hardware-efficient QEC [71]. In short, a continuous variable system is associated with an infinite-dimensional Hilbert space, e.g., the quantum harmonic oscillator [72], while a discrete variable system, such as a qubit, is associated with a finite-dimensional Hilbert space, e.g., the ground and excited state of an atom. A numerical study of the QAA found that when implemented in a continuous variable encoding, the success probability was considerably higher than for discrete variable qubits under the same amount of noise [73].

As a future research project, we would like to build upon these results and investigate if a similar performance advantage is obtained for the QAOA when implemented in a continuous variable encoding. This could translate into the possibility of running more levels of the QAOA in experiments without performance loss.

5 Summary of papers

In **paper A**, we explore the performance of the QAOA when applied to the tail assignment (TAS) problem. It is a problem that consists of assigning aircraft to routes such that all routes are covered by an aircraft and that two aircraft do not share the same route. The instances that we investigate have only one feasible solution, making it possible to frame the TAS problem as an Exact Cover problem. In the paper, we perform numerical experiments to investigate the performance of the QAOA for solving the TAS problem. Specifically, we look at the dependence of its success probability as a function of circuit depth p and problem size n. We simulated several instances consisting of 8, 15, and 25 qubits. From the simulation results, we observe patterns in the angles $(\vec{\gamma}, \vec{\beta})$, and use this to employ an interpolation strategy that significantly simplifies the classical optimization part of the QAOA [32].

Furthermore, we find a strong size dependence on the success probability. In detail, we find that the 15 qubit instances are more challenging to solve than the 25 qubit ones. We attribute this to the problems' graph connectivity, where we find that the 15 qubit instances have more than twice the average vertex degree compared to the 25 qubit instances.

In **paper B**, we implement the QAOA on a quantum processor, consisting of two superconducting transmon qubits. This requires compiling the QAOA in terms of the available gate set for the hardware.

In the paper, we solve small instances of the exact-cover problem and find a 96.6% success probability by iterating the algorithm up to $p = 2$. The experiment serves as a technological demonstration that the algorithm works and that there is nothing fundamentally wrong with the theory. In the experiment, we implemented three different classical optimizers, Bayesian optimization with Gaussian processes, Nelder-mead, and covariance matrix adaptation evolution optimization. Of these three, the Bayesian optimizer showed the best performance.

Bibliography

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).

[2] P. Benioff, "The computer as a physical system", J. Stat. Phys. **22**, 563–591 (1980).

[3] Y. Manin, *Vychislimoe i nevychislimoe (computable and noncomputable)* (Soviet Radio, 1980) pp. 13–15, in Russian.

[4] P. Benioff, "Quantum mechanical hamiltonian models of turing machines", J. Stat. Phys. **29**, 515–546 (1982).

[5] R. P. Feynman, "Simulating physics with computers", Int J Theor Phys **21**, 467–488 (1982).

[6] D. Deutsch, "Quantum theory, the church–turing principle and the universal quantum computer", Proc. R. Soc. Lond. A **400**, 97–117 (1985).

[7] D. Deutsch, "Quantum computational networks", Proc. R. Soc. Lond. A **425**, 73–90 (1989).

[8] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", SIAM J. Comput. **26**, 1484–1509 (1997).

[9] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", Commun. ACM **21**, 120–126 (1978).

[10] D. Gottesman, "An introduction to quantum error correction and fault-tolerant quantum computation", arXiv:0904.2557 [quant-ph] (2009).

[11] D. A. Lidar and T. A. Brun, *Quantum Error Correction* (Cambridge University Press, 2013).

[12] C. Gidney and M. Ekerå, "How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits", arXiv:1905.09749 [quant-ph] (2019).

[13] J. Preskill, "Quantum computing in the NISQ era and beyond", Quantum **2**, 79 (2018).

[14] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, "Quantum supremacy using a programmable superconducting processor", Nature **574**, 505–510 (2019).

[15] M. Grönkvist, *The Tail Assignment Problem*, Ph.D. thesis, Chalmers University of Technology and Göteborg University (2005).

[16] A. Parmentier and F. Meunier, "Aircraft routing and crew pairing: Updated algorithms at Air France", Omega **93**, 102073 (2020).

[17] C. P. Williams, *Explorations in quantum computing*, 2nd ed., Texts in computer science (Springer, London, 2011).

[18] S. Dasgupta, C. H. Papadimitriou, and U. Vazirani, *Algorithms* (McGraw-Hill, Inc., USA, 2006).

[19] A. Bengtsson, *Quantum information processing with tunable and low-loss superconducting circuits*, Ph.D. thesis, Chalmers University of Technology (2020).

[20] C. H. Papadimitriou, *Computational complexity* (Addison-Wesley, Reading, Mass, 1994).

[21] M. Agrawal, N. Kayal, and N. Saxena, "PRIMES is in p", Annals of Mathematics **160**, 781–793 (2004).

[22] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack", Phys. Rev. Lett. **79**, 325–328 (1997).

[23] G. Ausiello, A. Marchetti-Spaccamela, P. Crescenzi, G. Gambosi, M. Protasi, and V. Kann, *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*, 1st ed. (Springer-Verlag Berlin Heidelberg, 1999).

[24] Z. Bian, F. Chudak, W. G. Macready, and G. Rose, *The Ising model: teaching an old problem new tricks* (D-Wave systems, 2010).

[25] S. P. Singh, "The ising model: Brief introduction and its application", in *Metastable, Spintronics Materials and Mechanics of Deformable Bodies*, edited by S. Sivasankaran, P. K. Nayak, and E. Günay (IntechOpen, Rijeka, 2020) Chap. 8, p. 19.

[26] P. Vikstål, *Continuous-variable quantum annealing with superconducting circuits*, Master's thesis, Linköping University (2018).

[27] A. Lucas, "Ising formulations of many NP problems", Front. Phys. **2**, 5 (2014).

[28] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt, "An Application of Combinatorial Optimization to Statistical Physics and Circuit Layout Design", Operations Research **36**, 493–513 (1988).

[29] G. E. Crooks, "Performance of the Quantum Approximate Optimization Algorithm on the Maximum Cut Problem", arXiv:1811.08419 [quant-ph] (2018).

[30] Z. Wang, S. Hadfield, Z. Jiang, and E. G. Rieffel, "Quantum approximate optimization algorithm for MaxCut: A fermionic view", Phys. Rev. A **97**, 022304 (2018).

[31] G. G. Guerreschi and A. Y. Matsuura, "QAOA for Max-Cut requires hundreds of qubits for quantum speed-up", Sci Rep **9**, 6903 (2019).

[32] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices", Phys. Rev. X **10**, 021067 (2020).

[33] R. M. Karp, "Reducibility among combinatorial problems", in *Complexity of Computer Computations*, edited by R. E. Miller, J. W. Thatcher, and J. D. Bohlinger (Springer US, Boston, MA, 1972) pp. 85–103.

[34] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness* (W.H. Freeman and Company, 1979).

[35] P. Vikstål, M. Grönkvist, M. Svensson, M. Andersson, G. Johansson, and G. Ferrini, "Applying the Quantum Approximate Optimization Algorithm to the Tail-Assignment Problem", Phys. Rev. Applied **14**, 034009 (2020).

[36] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm", arXiv:1411.4028 [quant-ph] (2014).

[37] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, "Quantum Computation by Adiabatic Evolution", arXiv:quant-ph/0001106 (2000).

[38] T. Albash and D. A. Lidar, "Adiabatic quantum computing", Rev. Mod. Phys. **90**, 015002 (2018).

[39] B. Altshuler, H. Krovi, and J. Roland, "Anderson localization makes adiabatic quantum optimization fail", PNAS **107**, 12446–12450 (2010).

[40] I. Hen and A. P. Young, "Exponential Complexity of the Quantum Adiabatic Algorithm for certain Satisfiability Problems", Phys. Rev. E **84**, 061152 (2011).

[41] E. Farhi, D. Gosset, I. Hen, A. W. Sandvik, P. Shor, A. P. Young, and F. Zamponi, "The performance of the quantum adiabatic algorithm on random instances of two optimization problems on regular hypergraphs", Phys. Rev. A **86**, 052334 (2012).

[42] Y. Sun, J.-Y. Zhang, M. S. Byrd, and L.-A. Wu, "Adiabatic Quantum Simulation Using Trotterization", arXiv:1805.11568 [quant-ph] (2018).

[43] D. Wecker, M. B. Hastings, and M. Troyer, "Training a quantum optimizer", Phys. Rev. A **94**, 022309 (2016).

[44] G. G. Guerreschi and M. Smelyanskiy, "Practical optimization for hybrid quantum-classical algorithms", arXiv:1701.01450 [quant-ph] (2017).

[45] W. Lavrijsen, A. Tudor, J. Müller, C. Iancu, and W. de Jong, "Classical Optimizers for Noisy Intermediate-Scale Quantum Devices", arXiv:2004.03004 [quant-ph] (2020).

[46] K. M. Nakanishi, K. Fujii, and S. Todo, "Sequential minimal optimization for quantum-classical hybrid algorithms", Phys. Rev. Research **2**, 043158 (2020).

[47] J. S. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. S. Fried, S. Hong, P. Karalekas, C. B. Osborn, A. Papageorge, E. C. Peterson, G. Prawiroatmodjo, N. Rubin, C. A. Ryan, D. Scarabelli, M. Scheer, E. A. Sete, P. Sivarajah, R. S. Smith, A. Staley, N. Tezak, W. J. Zeng, A. Hudson, B. R. Johnson, M. Reagor, M. P. da Silva, and C. Rigetti, "Unsupervised Machine Learning on a Hybrid Quantum Computer", arXiv:1712.05771 [quant-ph] (2017).

[48] A. Bengtsson, P. Vikstål, C. Warren, M. Svensson, X. Gu, A. F. Kockum, P. Krantz, C. Križan, D. Shiri, I.-M. Svensson, G. Tancredi, G. Johansson, P. Delsing, G. Ferrini, and J. Bylander, "Improved Success Probability with Greater Circuit Depth for the Quantum Approximate Optimization Algorithm", Phys. Rev. Applied **14**, 034010 (2020).

[49] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, S. Boixo, M. Broughton, B. B. Buckley, D. A. Buell, B. Burkett, N. Bushnell, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, S. Demura, A. Dunsworth, E. Farhi, A. Fowler, B. Foxen, C. Gidney, M. Giustina,

R. Graff, S. Habegger, M. P. Harrigan, A. Ho, S. Hong, T. Huang, L. B. Ioffe, S. V. Isakov, E. Jeffrey, Z. Jiang, C. Jones, D. Kafri, K. Kechedzhi, J. Kelly, S. Kim, P. V. Klimov, A. N. Korotkov, F. Kostritsa, D. Landhuis, P. Laptev, M. Lindmark, M. Leib, E. Lucero, O. Martin, J. M. Martinis, J. R. McClean, M. McEwen, A. Megrant, X. Mi, M. Mohseni, W. Mruczkiewicz, J. Mutus, O. Naaman, M. Neeley, C. Neill, F. Neukart, H. Neven, M. Y. Niu, T. E. O'Brien, B. O'Gorman, E. Ostby, A. Petukhov, H. Putterman, C. Quintana, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, A. Skolik, V. Smelyanskiy, D. Strain, M. Streif, K. J. Sung, M. Szalay, A. Vainsencher, T. White, Z. J. Yao, P. Yeh, A. Zalcman, and L. Zhou, "Quantum Approximate Optimization of Non-Planar Graph Problems on a Planar Superconducting Processor", arXiv:2004.04197 [quant-ph] (2020).

[50] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization", Proceedings of the IEEE **104**, 148–175 (2016).

[51] J. A. Nelder and R. Mead, "A simplex method for function minimization", The Computer Journal **7**, 308–313 (1965).

[52] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming", Journal of the ACM **42**, 1115–1145 (1995).

[53] E. Halperin, D. Livnat, and U. Zwick, "MAX CUT in cubic graphs", Journal of Algorithms **53**, 169–185 (2004).

[54] X. Qiang, X. Zhou, J. Wang, C. M. Wilkes, T. Loke, S. O'Gara, L. Kling, G. D. Marshall, R. Santagati, T. C. Ralph, J. B. Wang, J. L. O'Brien, M. G. Thompson, and J. C. F. Matthews, "Large-scale silicon quantum photonics implementing arbitrary two-qubit processing", Nature Photonics **12**, 534–539 (2018).

[55] G. Pagano, A. Bapat, P. Becker, K. S. Collins, A. De, P. W. Hess, H. B. Kaplan, A. Kyprianidis, W. L. Tan, C. Baldwin, L. T. Brady, A. Deshpande, F. Liu, S. Jordan, A. V. Gorshkov, and C. Monroe, "Quantum approximate optimization of the long-range ising model with a trapped-ion quantum simulator", PNAS **117**, 25396–25401 (2020).

[56] N. Lacroix, C. Hellings, C. K. Andersen, A. Di Paolo, A. Remm, S. Lazar, S. Krinner, G. J. Norris, M. Gabureac, J. Heinsoo, A. Blais, C. Eichler, and A. Wallraff, "Improving the performance of deep quantum optimization algorithms with continuous gate sets", PRX Quantum **1**, 110304 (2020).

[57] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem", arXiv:1412.6062 [quant-ph] (2015).

[58] B. Barak, A. Moitra, R. O'Donnell, P. Raghavendra, O. Regev, D. Steurer, L. Trevisan, A. Vijayaraghavan, D. Witmer, and J. Wright, "Beating the random assignment on constraint satisfaction problems of bounded degree", arXiv:1505.03424 [cs.CC] (2015).

[59] F. G. S. L. Brandao, M. Broughton, E. Farhi, S. Gutmann, and H. Neven, "For Fixed Control Parameters the Quantum Approximate Optimization Algorithm's Objective Function Value Concentrates for Typical Instances", arXiv:1812.04170 [quant-ph] (2018).

[60] S. Lloyd, "Quantum approximate optimization is computationally universal", arXiv:1703.06199 [quant-ph] (2018).

[61] M. E. S. Morales, J. Biamonte, and Z. Zimborás, "On the universality of the quantum approximate optimization algorithm", arXiv:1909.03123 [quant-ph] (2019).

[62] Z. Jiang, E. G. Rieffel, and Z. Wang, "Near-optimal quantum circuit for Grover's unstructured search using a transverse field", Phys. Rev. A **95**, 062317 (2017).

[63] M. Y. Niu, S. Lu, and I. L. Chuang, "Optimizing QAOA: Success Probability and Runtime Dependence on Circuit Depth", arXiv:1905.12134 [quant-ph] (2019).

[64] E. Farhi and A. W. Harrow, "Quantum supremacy through the quantum approximate optimization algorithm", arXiv:1602.07674 [quant-ph] (2019).

[65] S. Hadfield, Z. Wang, B. O'Gorman, E. Rieffel, D. Venturelli, and R. Biswas, "From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz", Algorithms **12**, 34 (2019).

[66] E. Farhi, D. Gamarnik, and S. Gutmann, "The Quantum Approximate Optimization Algorithm Needs to See the Whole Graph: A Typical Case", arXiv:2004.09002 [quant-ph] (2020).

[67] E. Farhi, D. Gamarnik, and S. Gutmann, "The Quantum Approximate Optimization Algorithm Needs to See the Whole Graph: Worst Case Examples", arXiv:2005.08747 [quant-ph] (2020).

[68] J. Wurtz and P. J. Love, "Bounds on MAXCUT QAOA performance for p>1", arXiv:2010.11209 [quant-ph] (2020).

[69] M. Hodson, B. Ruck, H. Ong, D. Garvin, and S. Dulman, "Portfolio rebalancing experiments using the Quantum Alternating Operator Ansatz", arXiv:1911.05296 [quant-ph] (2019).

[70] Utkarsh, B. K. Behera, and P. K. Panigrahi, "Solving Vehicle Routing Problem Using Quantum Approximate Optimization Algorithm", arXiv:2002.01351 [quant-ph] (2020).

[71] A. Grimm, N. E. Frattini, S. Puri, S. O. Mundhada, S. Touzard, M. Mirrahimi, S. M. Girvin, S. Shankar, and M. H. Devoret, "Stabilization and operation of a Kerr-cat qubit", Nature **584**, 205–209 (2020).

[72] S. L. Braunstein and P. van Loock, "Quantum information with continuous variables", Rev. Mod. Phys. **77**, 513–577 (2005).

[73] S. Puri, C. K. Andersen, A. L. Grimsmo, and A. Blais, "Quantum annealing with all-to-all connected nonlinear oscillators", Nature Communications **8**, 15785 (2017).

Paper A

Applying the Quantum Approximate Optimization Algorithm to the Tail-Assignment Problem

Pontus Vikstål, Mattias Grönkvist, Marika Svensson, Martin Andersson, Göran Johansson, and Giulia Ferrini

Phys. Rev. Applied **14**, 034009 (2020)

Paper B

Improved Success Probability with Greater Circuit Depth for the Quantum Approximate Optimization Algorithm

Andreas Bengtsson, Pontus Vikstål, Christopher Warren, Marika Svensson, Xiu Gu, Anton Frisk Kockum, Philip Krantz, Christian Križan, Daryoush Shiri, Ida-Maria Svensson, Giovanna Tancredi, Göran Johansson, Per Delsing, Giulia Ferrini, and Jonas Bylander

Phys. Rev. Applied **14**, 034010 (2020)