Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

The Quantum Approximate Optimization Algorithm and other applications of alternating Hamiltonians

(Master's Thesis)

Bc. Denisa Lampášová

Supervisor: Mgr. Daniel Nagaj, Ph.D.

Bratislava, 2020

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

The Quantum Approximate Optimization Algorithm and other applications of alternating Hamiltonians

Master's Thesis

Study programme:	Theoretical Physics
Field of study:	4.1.1 Physics (1160)
Department:	Department of Theoretical Physics
Supervisor:	Mgr. Daniel Nagaj, Ph.D.

Bratislava, 2020
Bc. Denisa Lampášová

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Denisa Lampášová
Study programme:	Theoretical Physics (Single degree study, master II. deg., full time form)
Field of Study:	Physics
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: The Quantum Approximate Optimization Algorithm and other applications of alternating Hamiltonians

Annotation: The goal of this thesis is to investigate the possibilities of state preparation for quantum spin systems with Schroedinger evolution with alternating local interaction Hamiltonians (for example X fields and ZZ interactions). First, we want to find out what states are efficiently reachable by this procedure for different interaction geometries. Second, we want to analyze how good the Quantum Approximate Optimization Algorithm (QAOA) can be for local constraint optimization problems and why. In particular we will investigate the algorithm with several rounds as well as with local fields.

Aim: analysis of the QAOA algorithm and the possibilities of state preparation by alternating local Hamiltonians

Keywords: local constraint optimization, approximate solutions, quantum algorithms, local Hamiltonians, spin systems

Supervisor:	Mgr. Daniel Nagaj, PhD.
Department:	FMFI.KTF - Department of Theoretical Physics
Head of department:	doc. RNDr. Tomáš Blažek, PhD.

Assigned: 13.12.2017

Approved: 13.12.2017 prof. Ing. Roman Martoňák, DrSc.
Guarantor of Study Programme

.....
Student

.....
Supervisor

I hereby declare that this thesis is my original authorial work, which I have worked out on my own, under the guidance of my thesis supervisor, and only with the help of referenced literature.

.....

Denisa Lampášová

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my advisor Daniel Nagaj. For introducing me into this exciting research area, as well as for his advice, numerous corrections, and encouragement.

A big heartful thanks also goes to Scott Aaronson for writing excellent blog and articles. The excitement one gets from reading your work provides strong motivation for trying to do a similarly fantastic job.

Very special gratitude goes to my family and friends. Thank you for being there for me. It blows my mind how you (mysteriously) always know what to do to improve my mood.

Last but not least, I want to thank Adam for high-quality coffee supplies, and for introducing me to the magic of this beverage. And I cannot forget all the music composers, writers and YouTube fitness instructors for making their work accessible. You are creating an excellent source of energy and inspiration.

Thank you all for making this time of my life so enjoyable.

Abstract

The goal of this thesis was to investigate the potential and limitations of a promising quantum algorithm for approximate optimization – the Quantum approximate optimization algorithm (QAOA). We have decided to do this by exploring the range of states one can reach using QAOA's lowest-complexity versions, with emphasis on the reachability of computational basis states. That is because the results of the computational basis measurement performed on these states represent candidate solutions/near-optimal solutions to the combinatorial optimization problems we wish to solve. Such analysis led us to three interesting results.

The first one applies to the search for optimal QAOA's parameters – a part that is essential for the success of QAOA but also challenging. In this work, we present (and prove) the existence of several symmetries that reduce the number of parameters needed to examine to $\frac{1}{2 \cdot 8^p}$ of the initially required amount. Furthermore, we have shown that the lowest-complexity ($p = 1$) QAOA can be used for the efficient preparation of GHZ states with an odd number of qubits. Based on numerical evidence, we also assume that $p = 2$ QAOA will extend this possibility to an even number of qubits. The last result concerns the limitations of this algorithm. Even for simple MaxCut instances that can be solved by classical computers in linear time, a low-round QAOA cannot reach computational basis states with sufficient probability (including the MaxCut solutions). More precisely, using numerical tools, we have shown that the probability of reaching the optimal MaxCut solutions for rings (2-regular, connected graphs) decreases exponentially with the number of qubits/vertices. This result leads us to assume that QAOA cannot be very successful when limited only to a few rounds.

Keywords: local constraint optimization, approximate solutions, quantum algorithms, local Hamiltonians, spin systems

Abstrakt

Cieľom tejto práce bolo skúmať potenciál a limity sľubného kvantového algoritmu pre aproximačnú optimalizáciu – Kvantového aproximačného optimalizačného algoritmu (QAOA). Za týmto účelom sme sa rozhodli preskúmať rozsah stavov, ktoré vieme dosiahnuť použitím jeho verzií s najnižšou komplexitou. A to s dôrazom na preskúmavanie efektivity dosiahnuteľnosti počítačovo-bázových stavov, keďže meraním do počítačovej bázy získavame kandidátov pre optimálny/takmer optimálny výsledok riešeného problému. Pri takejto analýze sme dospeli k trom zaujímavým výsledkom.

Prvý sa týka voľby vhodných (optimálnych) parametrov pre QAOA – časti, ktorá je pre úspech QAOA kľúčová, ale zároveň náročná. V práci prezentujeme (a dokazujeme) existenciu niekoľkých symetrií, ktoré skresajú množstvo parametrov potrebných na preskúmanie na $\frac{1}{2\cdot 8^p}$ pôvodne potrebného množstva. Ďalej sme dokázali, že verzia QAOA s najnižšou komplexitou ($p = 1$) umožňuje efektívnu prípravu GHZ stavov s nepárnym počtom qubitov. Z numerických výsledkov predpokladáme, že $p = 2$ QAOA už rozšíri túto možnosť aj na párne množstvo qubitov. Tretí výsledok sa týka limitácií tohoto algoritmu. Už pri veľmi jednoduchých inštanciách MaxCutu, ktoré klasické počítače vyriešia v lineárnom čase, sú mnohé stavy pomocou malého množstva kôl QAOA nepripraviteľné dosť dobre. Konkrétne sme numericky ukázali, že pravdepodobnosť dosiahnutia optimálnych stavov pre kruhy (2-regulárne, súvislé grafy) klesá exponenciálne s počtom qubitov/vrcholov. Predpokladáme teda, že QAOA nemôže byť pre málo kôl príliš úspešné.

Kľúčové slová: optimalizácia s lokálnymi obmedzeniami, približné riešenia, kvantové algoritmy, lokálne Hamiltoniány, spinové systémy

Contents

Notational Conventions

- We use the convention, where $\hbar = 1$.
- The Pauli matrices/operators:

$$\mathbb{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \text{NOT},$$

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = iXZ.$$

- The computational basis (the Z (eigen)basis):

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \& \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

- The X basis:

$$|+\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix},$$
$$|-\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

- In the case of 2 or more qubits, we often leave the tensor product symbol $\otimes$ out and abbreviate, e.g. $|0\rangle \otimes |1\rangle \otimes |0\rangle \otimes |1\rangle = |0\rangle\,|1\rangle\,|0\rangle\,|1\rangle = |0101\rangle$.
- When writing a tensor product of single-qubit operators, we usually erase all identities and tensor product symbols. For the remaining single-qubit operators, we specify their location in the tensor product (= which qubit the single-qubit operator acts on) by a subscript. For example:

$$X \otimes \mathbb{I} = X_1,$$
$$\mathbb{I} \otimes Z \otimes \mathbb{I} \otimes \mathbb{I} \otimes Z = Z_2 Z_5.$$

- For the identity matrix $\mathbb{I}_d$, we usually do not specify the dimension d as it is clear from the context. Hence, even though we defined $\mathbb{I}$ as a 2-dimensional identity, we write $\mathbb{I}$ for any 2^n-dimensional identity $\underbrace{\mathbb{I} \otimes \mathbb{I} \otimes \cdots \otimes \mathbb{I}}_{n} = \mathbb{I}$.

- Often, instead of $X_1X_2\ldots X_n$, we write simply NOT.
- A tensor product of an $m \times n$ matrix A with an $p \times q$ matrix B is an $mp \times nq$ matrix

$$A \otimes B = \left(\begin{array}{ccc|ccc} \overbrace{\begin{matrix} A_{11}B_{11} & \cdots & A_{11}B_{1q} \\ \vdots & \ddots & \vdots \\ A_{11}B_{p1} & \cdots & A_{11}B_{pq} \end{matrix}}^{=A_{11}[B]} & & & A_{12}\,[B] & \cdots & A_{1n}\,[B] \\ \hline A_{21}\,[B] & & & A_{22}\,[B] & \cdots & A_{2n}\,[B] \\ \vdots & & & \vdots & \ddots & \vdots \\ A_{m1}\,[B] & & & A_{m2}\,[B] & \cdots & A_{mn}\,[B] \end{array}\right).$$

- $\#_a(z)$ – the number of occurrences of a symbol a in a string z.

Introduction

"Someone has to do it. It's all very well calling for eye of newt, but do you mean Common, Spotted or Great Crested? Which eye, anyway? Will tapioca do just as well? If we substitute egg white will the spell a) work b) fail or c) melt the bottom out of the cauldron?"

— Terry Pratchett, *Wyrd Sisters*

"I don't know anything, but I do know that everything is interesting if you go into it deeply enough."

— Richard Feynman, *The Pleasure of Finding Things Out*

The original motivation for exploring quantum computing, conceived in 1980s by Richard Feynmann, was to considerably accelerate the simulation of quantum physics and chemistry. Since then, researchers have discovered many other promising applications of quantum computers. For instance, using the famous Shor's algorithm [31], one could factor large numbers in polynomial time (quickly), which would render public-key cryptosystems commonly used today on the Internet, such as RSA, insecure. Grover's search algorithm [18] would allow us to find an item in an (unstructured) database in $\mathcal{O}(\sqrt{n})$ steps as opposed to classical algorithms that cannot guarantee better performance than $\mathcal{O}(n)$. Another interesting algorithm is HHL [21] (named after its discoverers: Harrow, Hassidim, Lloyd), which would enable us to solve systems of linear equations in logarithmic time – a rare exponential speedup, albeit one for which we are still seeking applications.

Unfortunately, for almost all other problems, we have no idea whether quantum computation could provide much advantage, compared to classical approaches. Moreover, implementing quantum computing-based solutions to most of these problems in practice would require a vast number of qubits, reliable quantum memories, and scalable error correction, which we still have a long way to go to achieve.

In the current area of classical computing, there are often no efficient ("fast enough") algorithms that would solve computationally challenging optimization problems that arise in the practical world.[1] Since we cannot find the optimal solution "fast enough",

[1] Even innocent-looking problems such as: "I have a set of containers. Each has a different weight

we often settle for a near-optimal solution brought by an approximation algorithm. This naturally begs the question: *Can we do better? Could we somehow leverage the quantum phenomena (superposition, entanglement, or tunneling) to gain some advantage in the area of approximate optimization?*

In 2014, for the first time, a quantum algorithm for approximate optimization, creatively named as the Quantum approximate optimization algorithm (QAOA), was introduced [11]. And as it turns out, QAOA seems to be a promising application for the near-term quantum devices [13, 19, 37]. The purpose of this thesis is to analyze this algorithm. We do it by investigating the range of states one can reach with low levels of complexity of QAOA, with emphasis on the reachability of computational basis states as these represent possible solutions to the combinatorial optimization problems we solve. Concretely, we will take a look at some simple instances of the (NP-hard) MaxCut problem – these will constitute our playground for this investigation as well as for exploring various other QAOA's properties. We discuss our observations throughout the thesis, prove some of them, and based on numerical evidence, lay out several conjectures.

In the first chapter, we introduce combinatorial optimization problems (COPs) together with the practical motivation behind this area of research. We continue by introducing quantum algorithms for COPs. First, we explain the idea behind the Quantum adiabatic algorithm (QAA) – the predecessor of QAOA, which happens to be also a universal tool for quantum computation [2]. Next, we move to QAOA. We attempt to provide a detailed explanation of this algorithm. After introducing it, we offer a thorough analysis of QAOA on the simplest instance (2-vertex line) of MaxCut. For the rest of the chapter, we present the current research results.

The second chapter provides an analysis of QAOA on slightly harder instances of MaxCut – on rings, lines, and complete graphs. We present here our observations – symmetries on the parameter space, an exponential decrease of success, and possible efficient preparation of $|GHZ\rangle$ states.

If any unclarities in notation should arise while reading the thesis, we recommend taking a look at the Notational Conventions section.

---

and a different price. What containers should I load to my ship with limited load capacity to maximize my profit?"

1 QAOA: The Quantum Approximate Optimization Algorithm

The goal of this chapter is to introduce the Quantum approximate optimization algorithm (QAOA). We would like to properly explain how the algorithm works, what interesting properties it has, what are its potential applications and already known limitations, what circumstances inspired its origin and in general, where its research led us so far. We have decided to start by introducing a class of algorithms QAOA could be used to solve. But before proceeding to the following section, we would like to point the reader towards http://www.dr-qubit.org/Boulder.html for a brief introduction to the theory of computation and complexity theory and towards Appendix A where we explain the big $\mathcal{O}$ notation.

1.1 Background and Motivation

Combinatorial optimization problems (COPs), problems where we search for an optimal solution from a finite set of possible solutions, are of great practical and theoretical importance today. To illustrate this, let us take a quick tour through some of the most famous examples.

- *The traveling salesman problem.* The knowledge of the order in which you should visit desired destinations so that the cost of travel is minimal would be invaluable for travelers, delivery companies or even astronomers.[1]

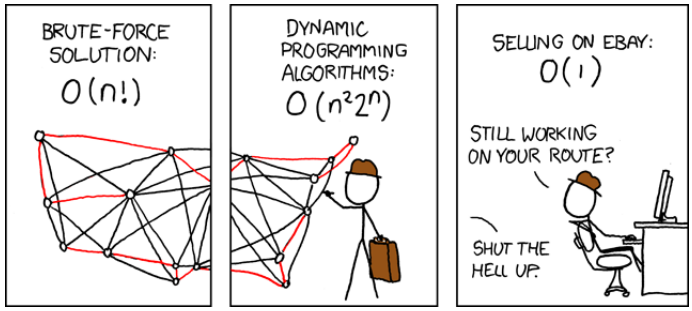


Figure 1.1: Illustration of the travelling salesman problem from xkcd.

[1] They would like to minimize the time spent on moving the telescope between the observed sources.

- *The knapsack problem.* Imagine you have just one night to prepare for an exam. Which chapters should you learn in order to maximize the points obtained? Or another situation, you are going somewhere for a few days and you end up wanting to pack too many things. How to choose the most valuable subset that your backpack will handle?

- *The Maximum-Cut problem (MaxCut).* Given a graph

$$G = (V - \text{set of vertices}, E - \text{set of edges}),$$

how to divide the vertices into two sets (how to color/label them with black and white/0 and 1) in order to maximize the cut, i.e., the number of edges connecting these two sets?[2]

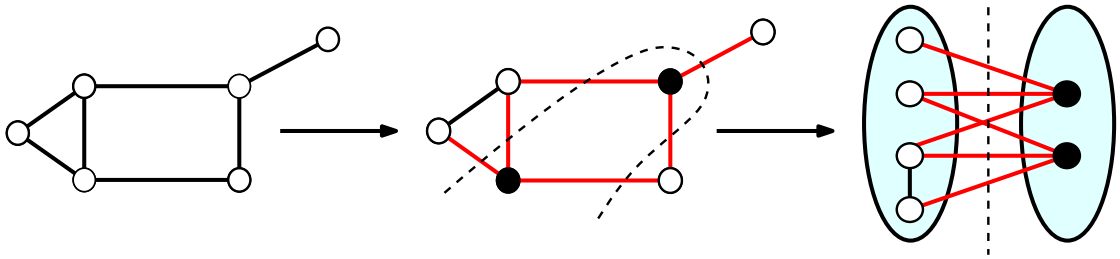


Figure 1.2: An example of a MaxCut problem. The value of the maximum cut is 6.

You can find this problem in physics when looking for a ground state of an antiferromagnetic Ising model. The frustrated magnets have minimal energy when the spins of the nearest neighbors point in different directions.

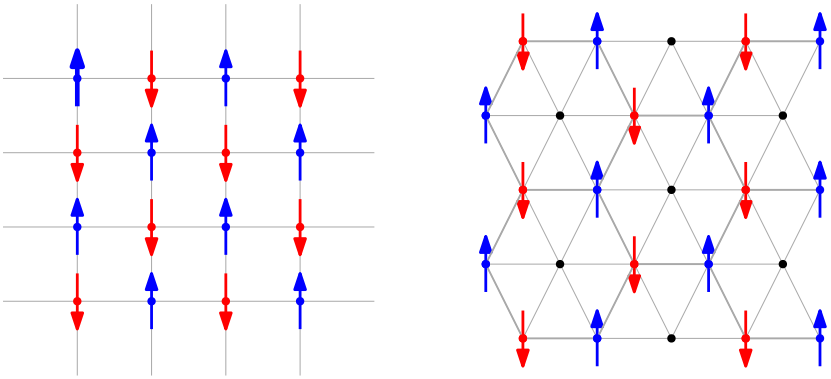


Figure 1.3: A ground state of a square-lattice antiferromagnetic Ising model on the left and a ground state of a triangular-lattice antiferromagnetic Ising model on the right. In the triangular lattice case, the black spins' values do not affect the MaxCut solution.

[2] A more general formulation gives each edge a weight and the cut is defined as the sum of the weights of the edges joining the two sets. In this thesis, we will consider only the case when all edges have the same weight (1).

In practice, combinatorial optimization problems are often described by a set of m constraints/clauses on an n-bit string (solution). You can see a constraint as a boolean formula – if the string satisfies the formula, the clause is equal to 1; otherwise it is 0. For example, a constraint $C_{12}(z) = (z_1 \vee z_2) \wedge (\neg z_1 \vee \neg z_2)$ asks for the first two bits (z_1 and z_2) of a string z to have opposite values. The goal is to find n-bit string $z = z_1 z_2 \dots z_n$ which meets as many constraints as possible.

To decribe how good some string z is, we define an *objective function*

$$C(z) = \sum_{\alpha=0}^{m} C_\alpha(z), \tag{1.1}$$

which tells us how many clauses does the string z satisfy. There are several problems we can look at.

- *Satisfiability (SAT)* asks whether there exists a string z which satisfies the whole list of m constraints, i.e., $C(z) = m$.
- *Maximum satisfiability (MAX-SAT)* asks for a string z that minimizes the number of broken clauses, i.e., $C(z)$ is the maximum of C.
- *Approximate Optimization* gives us z such that $C(z)$ is close to maximum.

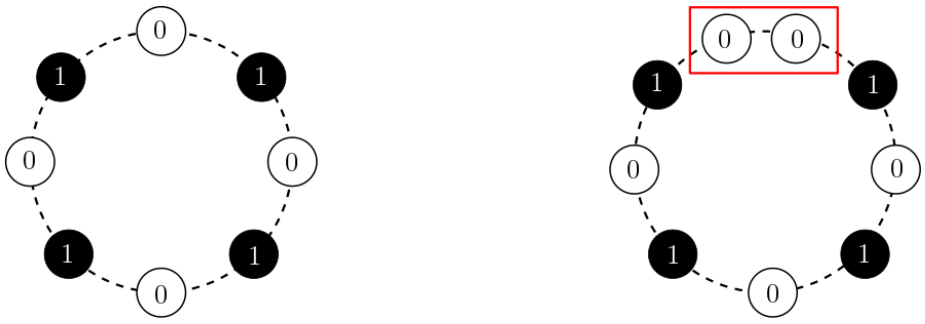
For a better illustration, let us look at a problem we call *The ring of disagrees* – MaxCut on 2-regular, connected rings. To find the solution, we must ask for all the neighboring bits to have opposite values (to disagree). Therefore, we use the following set of constraints.

$$\begin{aligned}
C_1(z) &= (z_1 \vee z_2) \wedge (\neg z_1 \vee \neg z_2)\,, \\
C_2(z) &= (z_2 \vee z_3) \wedge (\neg z_2 \vee \neg z_3)\,, \\
&\vdots \\
C_i(z) &= (z_i \vee z_{i+1}) \wedge (\neg z_i \vee \neg z_{i+1})\,, \\
&\vdots \\
C_n(z) &= (z_n \vee z_1) \wedge (\neg z_n \vee \neg z_1)\,.
\end{aligned}$$

Now, let us put them together into the objective function

$$C(z) = \sum_{\alpha=0}^{m} C_\alpha(z) = \sum_{\langle kl \rangle} (z_k \vee z_l) \wedge (\neg z_k \vee \neg z_l)\,, \tag{1.2}$$

where $\langle kl \rangle$ goes through the set of edges $E = \{\langle 1,2\rangle, \langle 2,3\rangle, \dots, \langle i, i+1\rangle, \dots, \langle n-1, n\rangle, \langle n, 1\rangle\}$. Notice that depending on the parity of n, $\max_z C(z)$ is either equal to m or $m-1$.



(a) An optimal solution for $n = 8$ (even). (b) An optimal solution for $n = 9$ (odd).

If n is even, there are always 2 solutions satisfying all the clauses:

$$z = 0101\ldots01 \quad \text{and} \quad \neg z = 1010\ldots10.$$

On the other hand, if n is odd, there is always at least one unsatisfied clause and $2n$ possible solutions as there are n clauses which can be broken and two possibillities of breaking the clause – 00 and 11.

Now that we have discussed what combinatorial optimization problems are, we ask: *How to solve them? Can we solve them efficiently? Can we solve them at all? If we are unable to find the optimal solution efficiently, can we solve these problems at least approximately?*

Let us start with the SAT problem. A straightforward ("brute-force") algorithm would be the following:

```
# Iterate through all the 2^n possible strings.
for x in {0,1}^n:
    check whether x satisfies all the constraints
    if it does so:
        we found the solution
    otherwise continue
If none satisfies, there is no string that satisfies all of them.
```

We say that such an algorithm takes $\mathcal{O}(2^n)$ time or that its time complexity is $\mathcal{O}(2^n)$ [big $\mathcal{O}$ notation: appendix A].[3] It is not very heartwarming as the number of computational steps (needed in the worst case) grows exponentially with n. But given the solution, it is quite quick to verify its correctness. All we would have to do is to check whether all the clauses are really satisfied. This can be done in polynomial time. Therefore,

[3] We have assumed that each loop takes a constant time. In general, we cannot forget m, as well as the time it takes to evaluate a clause.

SAT belongs to the *nondeterministic polynomial time* complexity class (abbreviated as NP).[4]

Moreover, *the Cook-Levin theorem* states that 3-SAT (or more generally SAT) is NP-complete. It means that in addition to being in NP, any problem in NP can be reduced to 3-SAT in polynomial time. So 3-SAT is at least as hard as the hardest problems in NP. We have already defined SAT. But what is 3-SAT, or more generally k-SAT?

Definition 1.1.1 (k-Satisfiability, k-SAT)**.** *Given a set of boolean disjunctive clauses*[5] $\{C_1, C_2, \ldots, C_m\}$ *for an n-bit string $z = z_1 z_2 \ldots z_n$, such that each clause involves exactly k literals*[6]*. Does a string z satisfying a boolean formula $C_1 \wedge C_2 \wedge \ldots \wedge C_m$ exist?*

A straightforward algorithm for MAX-SAT would be very similar to the one for SAT. We would just in addition count the number of satisfied clauses and remember the maximum. This is an $O(2^n)$ algorithm, too. As well as in the SAT case, if we find a string that satisfies all the clauses, we can terminate the iteration as we have found the solution, otherwise, we need to iterate through all the 2^n possibilities – only this time, one unsatisfied clause may not be sufficient for moving to another string. The currently iterated string x can be left after $m-$*(current maximum)* unsatisfied clauses. It does not seem to take much more time. Even though, this algorithm is somehow better/more complex as we can use MAX-SAT as an *oracle* for SAT, i.e., having a black box solving MAX-SAT in a single step, we could easily solve also SAT as it would be sufficient just to check whether the maximal number of satisfied clauses is equal to the number of all clauses. Now, here is some food for thought: *Could we reduce also MAX-SAT to SAT?*

To illustrate the complexity difference between SAT and MAX-SAT better, let us mention that in stark contrast to 2-SAT for which there is a polynomial-time algorithm [3], (the decision version of)[7] MAX-2-SAT was proven to be NP-complete [14].

[4] NP is the set of all decision problems for which the problem instances, where the answer is "yes", have proofs verifiable in polynomial time by a deterministic Turing machine. Or alternatively, the set of decision problems solvable in polynomial time by a non-deterministic Turing machine.

[5] Clauses which are a disjunction of boolean variables and their negations. Comming back to the MaxCut ring of disagrees example, when talking about k-SAT or MAX-k-SAT, each of the 4-literal clauses $((z_k \vee z_l) \wedge (\neg z_k \vee \neg z_l))$ we used there should be broken into two 2-literal disjunctive clauses: $(z_k \vee z_l)$ and $(\neg z_k \vee \neg z_l)$. Thus, MaxCut is easily reducible into MAX-2-SAT.

[6] A literal is either a boolean variable (x) or its negation ($\neg x$).

[7] A decision problem is a problem posted as a "yes-no question" on the input values. Hence, in the decision version of MAX-SAT, we would only ask: *is there a string satisfying at least k clauses?* Let us note that this restriction to the decision problem can still give us a good idea about the original version of the problem because, as we know the maximal number of clauses, we can use binary search, which increases the time complexity only by a logarithmic factor.

Many combinatorial optimization problems are NP-hard and therefore very time-consuming to solve – there is no known algorithm that would solve these problems in polynomial time and a strong belief that such an algorithm even can not exist (the P vs. NP question)[8]. However, the exact solution is often unnecessary. In that case, one may use some algorithm for approximate optimization and obtain a near-optimal solution in quite a reasonable time. The question is then how well can we estimate the optimum efficiently. This is described by *approximation factor* (or *approximation ratio*) $r^\star$ which tells us what fraction of the optimal value are we guaranteed to obtain in the worst case:

$$\frac{C(z)}{C_{\max}} \geq r^\star. \tag{1.3}$$

Let us demonstrate this on the MaxCut problem whose decision version was proven to be NP-complete [16]. A very simple approximation algorithm would be the following:

```
#Partition the vertices arbitrarily: V → (A, B).
repeat (until there is nothing to do):
    for each vertex:
        if less than half of its edges cross the cut:
            move the vertex to the other side of the cut
return the final cut (A,B)
```

The approximation ratio of this simple algorithm is guaranteed to be $1/2$ as the final cut is crossed by at least half of the edges ($|E|/2$) and the maximal cut can not be greater than the number of all edges $|E|$. What about its time complexity? The number of iterations is at most $|E|$ because each iteration improves the cut by at least 1. Each iteration takes $O(|E| \cdot |V|)$ time as it consists of a for-loop through all the vertices, where for each vertex one must look at no more than $|E|$ edges. Thus, the overall time complexity is $O(|E|^2 \cdot |V|)$ and therefore polynomial. In fact, the best known polynomial-time approximation algorithm for MaxCut [17] has approximation ratio $r^\star \approx 0.878$.

Besides a great many classical methods for finding approximate solutions to combinatorial optimization problems, there are also quantum algorithms. These are based on steering the dynamics of quantum many-body systems in such a way that their final states encode the optimal (or near-optimal) solutions. For example, these are the Quantum adiabatic algorithm (QAA)[12] and the Quantum approximate optimization algorithm (QAOA) [11] which we are going to discuss in greater detail in the next sections.

[8] However, P=NP would not imply that there is a polynomial-time algorithm for all NP-hard problems – only for the NP-complete subset.

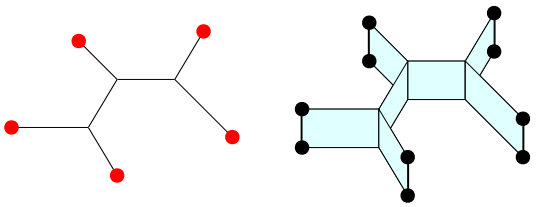
1.2 Quantum algorithms for COPs

Natural physical systems tend to settle into their ground states. One could, therefore, ask: *If finding the ground state was NP-hard, would it mean that we can use physical systems to solve NP-complete problems in polynomial time "naturally"?*[9]

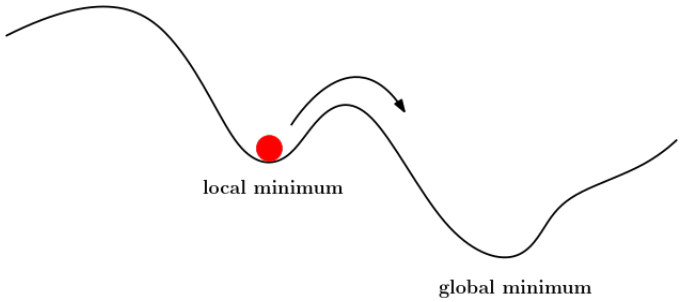
Let us take a look at one such attempt, which led to an arXiv post entitled "P=NP" [7]. For that, we need to define the *Steiner tree* problem, which was proven to be NP-hard [15].

Definition 1.2.1 (The Euclidean Steiner tree problem)**.** *Given n points in a plane, how to interconnect them by line segments allowing the addition of extra vertices (Steiner points) so that the total line length is minimal?*

There is a beautiful physics-based method for solving this problem. Put two parallel transparent planes connected by pins at desired locations into soapy water and when you pull it out, as the soap film is trying to minimize its surface, you will see the solution.



It sounds too beautiful to be true, doesn't it? *So where is the catch?* It is not always true that physical systems quickly settle into their ground states. For instance, will the ball in the picture below get to the global minimum? Probably not anytime soon and some algorithm (shaking) is needed.



In the discussed case of solving the Steiner tree, the system may as well end up in a local minimum. Therefore, we need to repeat the process many times. The crucial question is: *how many?* One can imagine that for large systems, the film will be much more likely to break before settling.

[9] *Note*: If interested in physics-based methods for solving NP-complete problems, you might find [1] worthwhile.

Unfortunately, the problem landscape of many real-world optimization problems is very bumpy with plenty of local optima to get trapped in. The hope that quantum computing could help us wade through these local optima was the starting point for the upcoming topic.

When going from classical to quantum computation, we replace each bit z_i by a spin-$\frac{1}{2}$ qubit labeled by $|z_i\rangle$:

$$\begin{aligned} 0 &\to |0\rangle \\ 1 &\to |1\rangle\,, \end{aligned} \tag{1.4}$$

where the quantum states $|0\rangle$ and $|1\rangle$, the eigenstates of the Pauli operator Z, are the core of the computational basis. Now, we would like to recast the SAT problem – *does a string z satisfying a boolean formula*

$$C_1 \wedge C_2 \wedge \ldots \wedge C_m \tag{1.5}$$

exist? – into an equivalent problem – *is the energy of the ground state of a Hamiltonian H equal to 0?* How to achieve this? We can construct a local Hamiltonian H_α to every clause C_α in such a way that

$$\begin{aligned} H_\alpha\,|z\rangle &= 0\,|z\rangle\,, \text{ if } C_\alpha(z) = 1, \\ H_\alpha\,|z\rangle &= 1\,|z\rangle\,, \text{ if } C_\alpha(z) = 0. \end{aligned} \tag{1.6}$$

In this way, the state which does not satisfy the clause receives an "energy penalty".[10] How to construct H_α for a general k-literal disjunctive clause $C_\alpha = z_i \vee z_j \vee \ldots \vee z_l$? The Hamiltonian should act nontrivialy only on the qubits included in the clause. If we take H_α as a tensor product of single-qubit Hamiltonians H_{α_i}, $i \in \{1, 2, \ldots, n\}$, in such a way that

- for each qubit corresponding to some variable z_i contained in C_α, we take

$$H_{\alpha_i} = \frac{1}{2}\big(\mathbb{I} + Z\big) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \tag{1.7}$$

- for each qubit corresponding to the negation of some variable $\neg z_i$, we take

$$H_{\alpha_i} = \frac{1}{2}\big(\mathbb{I} - Z\big) = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \tag{1.8}$$

and the identity matrix $\mathbb{I}$ for the rest of the qubits, we achieve exactly what we wanted. Now, we define the problem Hamiltonian C as the sum of the local Hamiltonians H_α[11]:

$$C = \sum_{\alpha=0}^{m} H_\alpha. \tag{1.9}$$

[10] For a disjunctive clause, there is exactly 1 state that violates the clause.

[11] They are *local* in the sense that each of them acts only on k qubit non-trivially.

This in-the-computational-basis-diagonal Hamiltonian has the following property:

$$C\left|z\right\rangle = (m - C(z))\left|z\right\rangle . \tag{1.10}$$

As we have just seen, we can easily reduce k-SAT into the k-local Hamiltonian problem.

Definition 1.2.2 (The k-local Hamiltonian problem)**.** *Given a k-local Hamiltonian $H = \sum_{\alpha=1}^{m} H_\alpha$ acting on n qubits which is composed of $m = poly(n)$ terms where each H_α acts on at most k qubits, and given two numbers a, $b \in \mathbb{R}$ such that $a < b$ and the separation $b - a$ is greater than (n^{-c}) for some constant c. The problem is to determine whether*

1. *the smallest eigenvalue of H does not exceed a, or*
2. *all eigenvalues of H are greater than b.*

We are guaranteed that either 1. or 2. is true.

The local Hamiltonian problem is, therefore, NP-hard. It may be NP-complete but may as well be much harder.

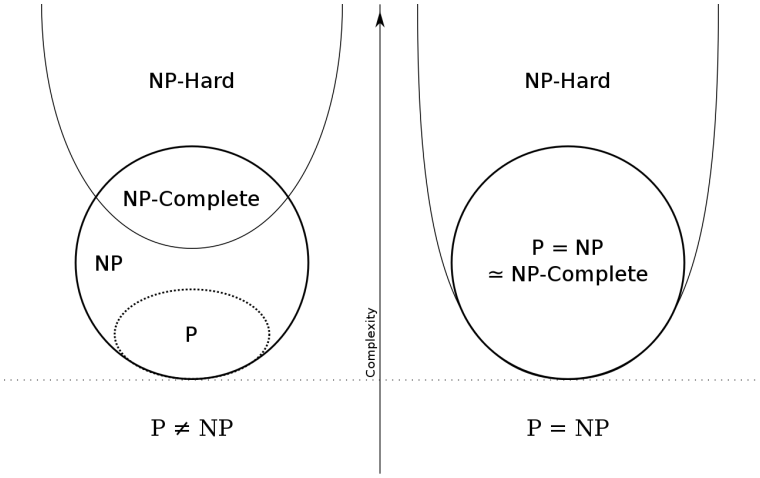


Figure 1.5: Euler diagram for P, NP, NP-complete, and NP-hard set of problems.[12]

.

1.2.1 QAA

The aspiration is to reach the ground state, the solution of a given instance of SAT. Without further ado, let us equip for this mission.

A quantum system evolves according to the Schrödinger equation

$$i\frac{\partial}{\partial t}\left|\psi(t)\right\rangle = H(t)\left|\psi(t)\right\rangle . \tag{1.11}$$

[12]Illustration by Behnam Esfahbod, distributed under a CC BY-SA 3.0 license, retrieved from https://commons.wikimedia.org/wiki/File:P_np_np-complete_np-hard.svg (on 4th May 2020).

The Hamiltonian $H(t)$ may evolve in time but it remains Hermitian. Therefore, we are guaranteed by the *spectral theorem* that the *instantaneous eigenvectors*,

$$H(\tau)\left|\psi_{n,\tau}\right\rangle = E_{n,\tau}\left|\psi_{n,\tau}\right\rangle, \tag{1.12}$$

form an orthonormal basis $\{|\psi_{n,\tau}\rangle\}_n$.

If the Hamiltonian does not evolve in time, the energy eigenstates are also time-independent, and we can solve the time-dependent Schrödinger equation pretty easily. We express the initial state $|\psi(0)\rangle$ in the orthonormal energy eigenbasis $\{|\psi_{n,0}\rangle\}_n$:

$$\left|\psi(0)\right\rangle = \sum_n c_n \left|\psi_{n,0}\right\rangle, \tag{1.13}$$

and the general solution to 1.11 is

$$\left|\psi(t)\right\rangle = \sum_n c_n \left|\psi_{n,0}\right\rangle e^{-iE_{n,0}t}. \tag{1.14}$$

Now, with the time-dependent Hamiltonian, we can write the general solution as

$$\left|\psi(t)\right\rangle = \sum_n c_n(t) \left|\psi_{n,t}\right\rangle e^{-i\int_0^t E_{n,t'}\mathrm{d}t'}. \tag{1.15}$$

Let us substitute this expression into the Schrödinger equation. We obtain

$$i\sum_n \left(\dot{c}_n \left|\psi_{n,t}\right\rangle + c_n \left|\dot{\psi}_{n,t}\right\rangle - \cancel{ic_n E_{n,t}\left|\psi_{n,t}\right\rangle}\right) e^{-i\int_0^t E_{n,t'}\mathrm{d}t'} = \cancel{\sum_n c_n H\left|\psi_{n,t}\right\rangle e^{-i\int_0^t E_{n,t'}\mathrm{d}t'}},$$

$$\sum_n \dot{c}_n \left|\psi_{n,t}\right\rangle e^{-i\int_0^t E_{n,t'}\mathrm{d}t'} = -\sum_n c_n \left|\dot{\psi}_{n,t}\right\rangle e^{-i\int_0^t E_{n,t'}\mathrm{d}t'}.$$

At this point, we may project onto $|\psi_{m,t}\rangle$ to express $\dot{c}_m$:

$$\dot{c}_m = -\sum_n c_n \left\langle \psi_{m,t}\middle|\dot{\psi}_{n,t}\right\rangle e^{-i\int_0^t (E_{n,t'}-E_{m,t'})\mathrm{d}t'}. \tag{1.16}$$

We can extract $\left\langle \psi_{m,t}\middle|\dot{\psi}_{n,t}\right\rangle$ from equation

$$\langle\psi_{m,t}|\dot{H}|\psi_{n,t}\rangle + \left\langle \psi_{m,t}\middle|H\middle|\dot{\psi}_{n,t}\right\rangle = \underbrace{\langle\psi_{m,t}|\,\dot{E}_{n,t}\,|\psi_{n,t}\rangle}_{\dot{E}_{n,t}\delta_{nm}} + \left\langle \psi_{m,t}\middle|E_{n,t}\middle|\dot{\psi}_{n,t}\right\rangle, \tag{1.17}$$

which was obtained by taking the inner product of an eigenstate $|\psi_{m,t}\rangle$ with differentiated time-independent Schrödinger equation (1.12), and get

$$\dot{c}_m = -c_m \left\langle \psi_{m,t}\middle|\dot{\psi}_{m,t}\right\rangle - \sum_{n\neq m} c_n \frac{\langle\psi_{m,t}|\,\dot{H}\,|\psi_{n,t}\rangle}{E_{n,t}-E_{m,t}} e^{-i\int_0^t (E_{n,t'}-E_{m,t'})\mathrm{d}t'}. \tag{1.18}$$

Let us take a closer look at the terms in the sum. Notice that if the rate of change of the Hamiltonian is small, then the terms are small as well. And presumably, a small term in a differential equation has little effect. Note that these terms are also oscillatory what might add to their irrelevance over long time intervals, as different

terms may take different phases, maybe even opposite and cancel each other. So, if these terms vanish, we are left only with

$$\dot{c}_m = -c_m \left\langle \psi_{m,t} \middle| \dot{\psi}_{m,t} \right\rangle, \tag{1.19}$$

which gives us

$$c_m = c_m(0) \cdot e^{-\int_0^t \langle \psi_{m,t'} | \dot{\psi}_{m,t'} \rangle \mathrm{d}t'}. \tag{1.20}$$

Since $\left\langle \psi_{m,t'} \middle| \dot{\psi}_{m,t'} \right\rangle$ is a pure imaginary number,[13]

$$|c_m(t)| = |c_m(0)|. \tag{1.21}$$

Hence, if we start in $|\psi_{n,0}\rangle$, for all $0 \leq t \leq T$, we will stay in $|\psi_{n,t}\rangle$ – we do not leave the instantaneous eigenstate. Generally, in this case, the solution would be

$$|\psi(t)\rangle = \sum_n c_n(0) \, |\psi_{n,t}\rangle \, e^{-i\int_0^t E_{n,t'}\mathrm{d}t'} e^{-\int_0^t \langle \psi_{n,t'} | \dot{\psi}_{n,t'} \rangle \mathrm{d}t'}. \tag{1.22}$$

Note that unless the evolution is cyclic the new phase factor $e^{-\int_0^t \langle \psi_{n,t'} | \dot{\psi}_{n,t'} \rangle \mathrm{d}t'}$ can be canceled out by an appropriate choice of eigenstates' complex phases.

Of course, in many practical settings, the "bad" terms do not vanish exactly. On the other hand, we can get a handle on them, and possibly show that they are small, or that their overall contribution over the progress of the evolution results in a near cancelation. That is the contents of the adiabatic theorem [5, 8, 9].

Theorem 1 (The adiabatic theorem[14]): *A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum.[5]*

In detail, it lower bounds the overlap of the evolved state with the instantaneous ground state in terms of spectral gaps, and various derivatives of the Hamiltonian with respect to the path taken. The approximation where the terms stay small is called *adiabatic*, and it represents the core building block, which the following quantum algorithm builds on.

In 2000, Farhi at al. [12] proposed an algorithm for solving instances of SAT relying on the adiabatic theorem. It gives us an idea for how to find a ground state of a local Hamiltonian C (which may not be easy to find). What we can do is the following.

[13] $0 = \frac{\partial}{\partial t} \langle \psi_{m,t} | \psi_{m,t} \rangle = \langle \dot{\psi}_{m,t} | \psi_{m,t} \rangle + \langle \psi_{m,t} | \dot{\psi}_{m,t} \rangle \implies \langle \psi_{m,t} | \dot{\psi}_{m,t} \rangle^* = - \langle \psi_{m,t} | \dot{\psi}_{m,t} \rangle$.

[14] Here stated in its original form (due to M. Born and V. Fock (1928)). Later analyses added formal guarantees of convergence which involve $||H||$, $\left|\left|\frac{\partial H}{\partial t}\right|\right|$, $\left|\left|\frac{\partial^2 H}{\partial t^2}\right|\right|$, the spectral gap, etc. If interested in a better formulation/analysis, you might take a look at [8] or [9], for example.

1. Start with an easily constructible ground state $|\psi_{0,0}\rangle$ of some Hamiltonian B with local structure. Our usual choice is

$$B = -\sum_{j=1}^{n} X_j. \tag{1.23}$$

As $|+\rangle$ is the highest energy eigenstate of the Pauli operator X and therefore the ground state of operator $-X$, the initial state $|\psi(0)\rangle$ is the uniform superposition over all the computational basis states

$$|s\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{z\in\{0,1\}^n} |z\rangle\,. \tag{1.24}$$

2. Continuously evolve the Hamiltonian B into the Hamiltonian C. There are many possibilities how to interpolate between the 2 Hamiltonians. A simple choice would be to take

$$H(t) = \left(1 - \frac{t}{T}\right) B + \left(\frac{t}{T}\right) C; \quad t \in [0, T]\,. \tag{1.25}$$

3. Measure $|\psi(T)\rangle$ in the computational basis to obtain the solution. If T is sufficiently large, and the gap between the two lowest energy levels is strictly greater than 0 for all $0 \le t \le T$, then $|\langle\psi_{0,T}|\psi(T)\rangle|$ will be (very near) 1.

It is worth noting that QAA was proven to be universal for quantum computation [2]. But algorithms that take too long are of no practical use to us. Hence we ask: *What exactly is a sufficiently large T for this algorithm*? A closer look at the adiabatic theorem would tell us that we should take

$$T \gg \frac{\mathcal{E}}{g_{\min}^2},$$

where $\mathcal{E} = \max_{0<t<T} \langle\psi_{1,t}|\,\dot{H}\,|\psi_{0,t}\rangle$ and $g_{\min} = \min_{0<t<T} (E_{1,t} - E_{0,t})$. If $\mathcal{E}$ is not too big, which is often the case, QAA takes $\mathcal{O}(g_{\min}^{-2})$ time.

Finally, we would like to mention that there is a quantum computing company, D-Wave Systems, which is building devices specifically designed to implement a noisy approximation to the adiabatic algorithm (called quantum annealing). However, they are using solely *stoquastic Hamiltonians*, which are often much easier to simulate on a classical computer compared to arbitrary Hamiltonians.

Definition 1.2.3 (Stoquastic Hamiltonian)**.** *A Hamiltonian H is stoquastic if all his off-diaginal matrix elements are in the standard basis real and non-positive.*

Notice that the Hamiltonians we have reduced the k-SAT problem to are also stoquastic. Stoquastic Hamiltonians arise naturally in the physical world – examples include

the quantum transverse Ising model, the ferromagnetic Heisenberg model, the Jaynes-Cummings model of light-matter interaction, and most Hamiltonians reachable using Josephson-junction flux qubits (the kind used by D-Wave). Unfortunately, it is not clear how much the restriction for all off-diagonal elements to be non-positive and real might reduce the complexity of local Hamiltonian (if at all).

1.2.2 QAOA

"The best is the enemy of the good."

—Voltaire

The goal of QAA was to find an exact solution for a given COP. Now we ask: *Could we use quantum computers for approximate optimization? How good would they be? Could they provide some advantage compared to classical algorithms, e.g., a better approximation ratio?*

In 2014, Farhi at al. introduced the Quantum approximate optimization algorithm (QAOA) [11]. They aimed to shed light on (and possibly disprove) the Unique Games Conjecture (UGC)[15] [28], (hopefully) clarifying the relationship between NP, approximations for NP, and the quantum complexity class BQP in the process. The algorithm has not yet succeeded at that goal but has opened an interesting discussion, motivated several classical algorithms, and basically started a whole industry of experiments on few-qubit quantum systems. Farhi et al. managed to show that QAOA_1 solves the MaxCut problem on 3-regular graphs with an approximation ratio 0.6924, which is better than random guessing (where $r^\star = 0.5$), but not as good as the best-known classical algorithms (Goemans-Williamson ($r^\star \approx 0.878$) [17], or for graphs of maximum degree at most 3, Halperin-Livnat-Zwick ($r^\star = 0.9326$) [20]), let alone better than those algorithms, which is what would be needed to refute the UGC [29]. Even though QAOA has not beaten the best-known classical approximation algorithm for MaxCut, it managed to outrun the best one for the Max E3LIN2 problem. However, only for a little while (see Section 1.3.2 for this story).

Before introducing the scheme of QAOA, let us define two unitary operators:

$$U(B,\beta) = e^{-i\beta B} = \prod_{j=1}^{n} e^{-i\beta B_j}, \tag{1.26}$$

$$U(C,\gamma) = e^{-i\gamma C} = \prod_{\alpha=1}^{m} e^{-i\gamma C_\alpha}, \tag{1.27}$$

[15] The UGC roughly says that the best approximations for hard problems are achieved by semidefinite programming, and anything better would basically result in solving NP-complete problems efficiently, which is unlikely.

which represent the quantum time evolution under the Hamiltonians B (1.23) and C (1.9) for time β and γ respectively.

Our goal is to end up in a low energy state of the (problem) Hamiltonian C. We will try to accomplish this by constructing a quantum state

$$\left|\vec{\gamma},\vec{\beta}\right\rangle = \underbrace{U(B,\beta_p)U(C,\gamma_p)}_{\text{pth round}} \dots \underbrace{U(B,\beta_2)U(C,\gamma_2)}_{\text{2nd round}} \underbrace{U(B,\beta_1)U(C,\gamma_1)}_{\text{1st round}} |s\rangle\,, \tag{1.28}$$

where the integer $p \geq 1$ and $2p$ angles $\gamma_1\gamma_2\dots\gamma_p \equiv \vec{\gamma}$ and $\beta_1\beta_2\dots\beta_p \equiv \vec{\beta}$ are parameters of QAOA (more precisely a p-level QAOA or QAOA$_p$). After the state $\left|\vec{\gamma},\vec{\beta}\right\rangle$ is prepared, we measure it in the computational basis to obtain a string z and evaluate $C(z)$ (the objective function). Multiple rounds of alternating these Hamiltonians aim to produce a string z, which gives $C(z)$ close to the maximum.

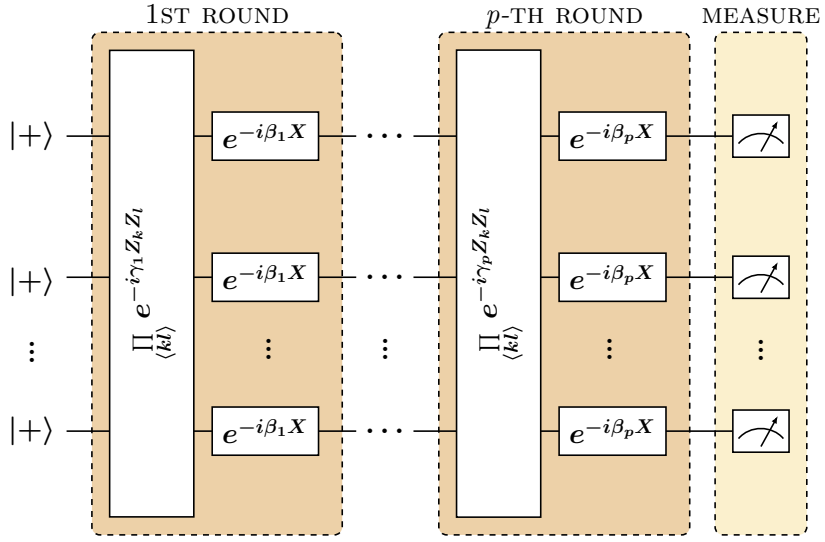


Figure 1.6: A crucial subprocess of QAOA. The preparation of the state $\left|\vec{\gamma},\vec{\beta}\right\rangle$ and the subsequent computational basis 0/1 measurement to get a candidate for the optimal/near-optimal solution. The state can be produced by a quantum circuit of depth at most $p(m+1)$.

One might imagine that the success of QAOA depends on the parameters. Therefore, now, the critical question is: *What parameters should we choose to achieve our goal?* This is a black-box optimization problem, which we grasp classically. When an analytical approach is not available, one may use (random-restart) hill climbing, grid search,[16] simulated annealing, or any other optimization method of choice.[17] Irrespective of the choice, the scenario goes like this. Start with some set of parameters $\{\vec{\gamma},\vec{\beta}\}$. Subsequently, use the quantum computer to construct $\left|\vec{\gamma},\vec{\beta}\right\rangle$ (Eqn. 1.28), measure it

[16]The size of the area one should investigate is discussed in Section 2.1. But the natural choice would (probably) be a fine grid on the compact set $[0,2\pi]^p \times [0,2\pi]^p$.

[17]One should, however, bear in mind that there is no free lunch in search [35] and optimization [36].

in the computational basis, and evaluate $C(z)$. Repeat this step several times to estimate the value of the cost function, usually taken to be the energy with respect to the problem Hamiltonian C (which is also the expected value of the number of unsatisfied clauses $\overline{m - C(z)}$)

$$F_p(\vec{\gamma}, \vec{\beta}) = \left\langle \vec{\gamma}, \vec{\beta} \right| C \left| \vec{\gamma}, \vec{\beta} \right\rangle . \tag{1.29}$$

Next, use the outer-loop classical optimization to get a new set of parameters. Repeat the whole process until you find (ideally) the set of parameters that minimizes $F_p(\vec{\gamma}, \vec{\beta})$. However, it is important to note that in order for QAOA to be useful for approximate optimization, efficient search strategies are needed.

Now, let us denote

$$M_p = \min_{\vec{\gamma}, \vec{\beta}} F_p(\vec{\gamma}, \vec{\beta}). \tag{1.30}$$

As the minimization for the $(p-1)$-level algorithm can be viewed as a constrained minimization for the p-level algorithm with $\beta_p = \gamma_p = 0$, M_p can only decrease as p grows:

$$M_{p-1} \geq M_p. \tag{1.31}$$

Furthermore, the Suzuki-Trotter formula [32],

$$e^{A+B} = \lim_{n \to \infty} \left(e^{A/n} e^{B/n} \right)^n , \tag{1.32}$$

guarantees that there exists (in a $p \to \infty$ limit) a QAOA algorithm, which is equivalent to the adiabatic evolution, and as QAA promises optimal results for large enough T, it follows that

$$\lim_{p \to \infty} M_p = \min_z C(z). \tag{1.33}$$

Bear in mind that after all this (possibly classical) optimization for either the number of rounds p (this parameter is, however, often fixed) or the angles $\gamma_1 \gamma_2 \ldots \gamma_p \equiv \vec{\gamma}$ and $\beta_1 \beta_2 \ldots \beta_p \equiv \vec{\beta}$, one still needs to run the sequence of operations on a quantum computer. Any classical simulation of it is not expected to be efficient, as we know that sampling from the measurement results of states prepared by QAOA is computationally difficult [13].

1.3 QAOA analysis

1.3.1 QAOA on MaxCut

Best understanding often comes from examples. Therefore, let us apply QAOA to some simple instances of the MaxCut problem and see how close one can get towards

the actual ground state. To solve MaxCut, we need to find the ground state of a Hamiltonian

$$C = \sum_{\langle kl \rangle} C_{\langle kl \rangle} = \sum_{\langle kl \rangle} Z_k Z_l. \tag{1.34}$$

Remark. *To meet property 1.6 as well as 1.10, we should have used*

$$C_{\langle kl \rangle} = \frac{1}{2}\left(Z_k Z_l + 1\right)$$

instead of $C_{\langle kl \rangle} = Z_k Z_l$ *as*

$$\frac{1}{2}\left(Z_1 Z_2 + 1\right)|z_1 z_2\rangle = \begin{cases} |z_1 z_2\rangle & \text{if} \quad z_1 = z_2 \\ 0 & \text{if} \quad z_1 = \neg z_2 \end{cases}$$

and

$$Z_1 Z_2 |z_1 z_2\rangle = \begin{cases} |z_1 z_2\rangle & \text{if} \quad z_1 = z_2 \\ -|z_1 z_2\rangle & \text{if} \quad z_1 = \neg z_2 \end{cases}.$$

There are three things one should mention here. First, the only inconvenience brought by the new reward system:

- *as before, all agreeing edges receive the energy penalty (+1)*
- *but aditionally, a reward (-1) is given to those that disagree,*

is that now the ground energy is m minus twice the maximum cut value instead of nicely defined 0. But we believe it is not that much of a burden. Second, observe that the "$+\frac{1}{2}$*" part of* $C_{\langle kl \rangle}$ *would only add a global phase:*

$$\begin{aligned} \left|\vec{\gamma}, \vec{\beta}\right\rangle &= e^{-i\beta_p B} \prod_{\langle kl \rangle} e^{-i\gamma_p (Z_k Z_l + 1)/2} \dots e^{-i\beta_1 B} \prod_{\langle kl \rangle} \underbrace{e^{-i\gamma_1 (Z_k Z_l + 1)/2}}_{= \; e^{-i\gamma_1/2} e^{-i\gamma_1 Z_k Z_l/2}} |s\rangle \\ &= e^{-i\beta_p B} e^{-i\gamma_p m/2} \prod_{\langle kl \rangle} e^{-i\gamma_p Z_k Z_l/2} \dots e^{-i\beta_1 B} e^{-i\gamma_1 m/2} \prod_{\langle kl \rangle} e^{-i\gamma_1 Z_k Z_l/2} |s\rangle \\ &= \underbrace{e^{-i(\gamma_1 + \dots + \gamma_p)m/2}}_{= \; global \; phase} e^{-i\beta_p B} \prod_{\langle kl \rangle} e^{-i\gamma_p Z_k Z_l/2} \dots e^{-i\beta_1 B} \prod_{\langle kl \rangle} e^{-i\gamma_1 Z_k Z_l/2} |s\rangle, \end{aligned}$$

and we all know that the global phase is irrelevant. Third, we have also rescaled γ *as*

$$\gamma_{new} = \frac{\gamma_{original}}{2},$$

but rescaling is a matter of little consequence.
Overall, one must admit that the chosen change in C *significantly improves readability, and it does so without any cost.*

As the initial Hamiltonian B, we take (1.23) whose ground state is (1.24).

2-vertex line

The simplest graph to start with is a 2-vertex line. Thus, we have 2 vertices $\leftrightarrow$ 2 qubits, and 1 edge $\leftrightarrow$ 1 contributor to the problem Hamiltonian $C_{\langle 12\rangle} = Z_1Z_2$. Let us take a look at what state QAOA_1 prepares in the process.

$$\left|\vec{\gamma},\vec{\beta}\right\rangle = \underbrace{e^{i\beta X_2}e^{i\beta X_1}}_{U(B,\beta)}\underbrace{e^{-i\gamma Z_1Z_2}}_{U(C,\gamma)}|s\rangle\,, \tag{1.35}$$

where

$$|s\rangle = \frac{1}{2}\Big(\,|00\rangle + |01\rangle + |10\rangle + |11\rangle\,\Big) = \frac{1}{2}\begin{pmatrix}1\\1\\1\\1\end{pmatrix}. \tag{1.36}$$

As the Pauli matrices are both Hermitian ($P = P^\dagger$) and unitary ($PP^\dagger = P^\dagger P = \mathbb{I}$), they satisfy this analogue of Euler's formula $e^{i\varphi P} = (\cos\varphi\cdot\mathbb{I} + i\sin\varphi\cdot P)$. Thus, we can express $U(B,\beta)$ as

$$\begin{aligned} U(B,\beta) = e^{i\beta X_2}e^{i\beta X_1} &= \Big(\cos\beta\cdot\mathbb{I} + i\sin\beta\cdot X_2\Big)\Big(\cos\beta\cdot\mathbb{I} + i\sin\beta\cdot X_1\Big) \\ &= \cos^2\beta\cdot\mathbb{I} + i\sin\beta\cos\beta\cdot(X_2+X_1) - \sin^2\beta\cdot X_1X_2. \end{aligned} \tag{1.37}$$

Observe that $U(B,\beta)$ disperses any computational basis state $|i\rangle$ over the whole computational basis:

$$U(B,\beta)\,|i\rangle = \sum_{j\in\{0,1\}^n} a_B\,(\beta,j)\,|j\rangle = \sum_{j\in\{0,1\}^n} (\cos\beta)^{n-ham(i,j)}\,(i\sin\beta)^{ham(i,j)}\,|j\rangle\,.$$

Here $ham(i,j)$ is the hamming distance between the binary strings i and j. On the other hand, $U(C,\gamma)$ acts on a computational basis state $|j\rangle$ only by giving it a phase factor $e^{-i\gamma(m-2C(j))}$.[18] With this in mind, we can express the amplitudes $a(\beta,\gamma,j)$ from

$$\left|\vec{\gamma},\vec{\beta}\right\rangle = \sum_{j\in\{0,1\}^n} a\,(\beta,\gamma,j)\,|j\rangle \tag{1.38}$$

as

$$a\,(\beta,\gamma,j) = \frac{1}{\sqrt{2^n}}\sum_{k\in\{0,1\}^n} e^{-i\gamma(m-2C(k))}\,(\cos\beta)^{n-ham(j,k)}\,(i\sin\beta)^{ham(j,k)}\,. \tag{1.39}$$

[18] A small reminder: m is the number of edges in the graph, and C is the objective function (the number of edges that disagree).

So

$$\left|\vec{\gamma},\vec{\beta}\right\rangle = \begin{pmatrix} a(\beta,\gamma,00) \\ a(\beta,\gamma,01) \\ a(\beta,\gamma,10) \\ a(\beta,\gamma,11) \end{pmatrix} = \frac{1}{\sqrt{2^n}} \begin{pmatrix} e^{-i\gamma}(\cos^2\beta - \sin^2\beta) + 2ie^{i\gamma}\cos\beta\sin\beta \\ e^{i\gamma}(\cos^2\beta - \sin^2\beta) + 2ie^{-i\gamma}\cos\beta\sin\beta \\ e^{i\gamma}(\cos^2\beta - \sin^2\beta) + 2ie^{-i\gamma}\cos\beta\sin\beta \\ e^{-i\gamma}(\cos^2\beta - \sin^2\beta) + 2ie^{i\gamma}\cos\beta\sin\beta \end{pmatrix}, \tag{1.40}$$

$$\left|\vec{\gamma},\vec{\beta}\right\rangle = \frac{1}{2} \begin{pmatrix} e^{-i\gamma}\cos(2\beta) + ie^{i\gamma}\sin(2\beta) \\ e^{i\gamma}\cos(2\beta) + ie^{-i\gamma}\sin(2\beta) \\ e^{i\gamma}\cos(2\beta) + ie^{-i\gamma}\sin(2\beta) \\ e^{-i\gamma}\cos(2\beta) + ie^{i\gamma}\sin(2\beta) \end{pmatrix}. \tag{1.41}$$

Notice that

$$a(\beta,\gamma,00) = a(\beta,\gamma,11), \tag{1.42}$$

$$a(\beta,\gamma,01) = a(\beta,\gamma,10). \tag{1.43}$$

That is what one would expect – for the Max-Cut problem, there is no difference between strings z and $\neg z$. Let us call this property of QAOA "the NOT symmetry" and prove it.

Property 1.1 (The NOT symmetry): *Given an n-vertex graph G, where $n \in \mathbb{N}$, a problem Hamiltonian $C = \sum_{\alpha=1}^{m} C_\alpha$, an initial Hamiltonian $B = \sum_{j=1}^{n} B_j$, and (by B also) an initial state $|s\rangle$, if the following conditions are true:*

a. $\forall j \in \{1,2,\ldots,n\} : \ [B_j, X_1X_2\ldots X_n] = 0,$

b. $\forall \alpha \in \{1,2,\ldots,m\} : \ [C_\alpha, X_1X_2\ldots X_n] = 0,$

c. $X_1X_2\ldots X_n\,|s\rangle = |s\rangle$

then the amplitudes in the state

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

corresponding to the computational basis states $|z\rangle$ and $\mathrm{NOT}\,|z\rangle$ are the same:

$$a(\beta,\gamma,z) = a(\beta,\gamma,\neg z).$$

Proof. To start, observe that for any two operators A_1, A_2, such that $[A_1,A_2]=0$, the (bi)linearity of the commutator together with $[A_1,A_2]=0$ implies $\left[e^{-i\varphi A_1}, A_2\right] = 0$:

$$\left[e^{-i\varphi A_1}, A_2\right] = \left[\left(\mathbb{I} - i\varphi A_1 - \frac{\varphi^2}{2!}A_1^2 + i\frac{\varphi^3}{3!}A_1^3 + \frac{\varphi^4}{4!}A_1^4 - \ldots\right), A_2\right] \overset{[A_1,A_2]=0}{=} 0.$$

Hence, it should be clear that conditions a. and b. ensure that

A. $[U(B,\beta), X_1X_2\dots X_n] = [\prod_{j=1}^{n} e^{-i\beta B_j}, X_1X_2\dots X_n] = 0,$

B. $[U(C,\gamma), X_1X_2\dots X_n] = [\prod_{\alpha=1}^{m} e^{-i\beta C_\alpha}, X_1X_2\dots X_n] = 0.$

Now, using:

1. $|\neg z\rangle = \mathrm{NOT}\,|z\rangle = X_1X_2\dots X_n\,|z\rangle,$
2. $X = X^\dagger$ – the hermiticity of X,

we can write down the proof:

$$\begin{aligned} a(\beta,\gamma,\neg z) &= \langle \neg z|\, U(B,\beta_p)U(C,\gamma_p)\dots U(B,\beta_1)U(C,\gamma_1)\,|s\rangle \\ &\overset{1.\&2.}{=} \langle z|\, X_n\dots X_1 U(B,\beta_p)U(C,\gamma_p)\dots U(B,\beta_1)U(C,\gamma_1)\,|s\rangle \\ &\overset{A.\&B.}{=} \langle z|\, U(B,\beta_p)U(C,\gamma_p)\dots U(B,\beta_1)U(C,\gamma_1)X_n\dots X_1\,|s\rangle \\ &\overset{c.}{=} \langle z|\, U(B,\beta_p)U(C,\gamma_p)\dots U(B,\beta_1)U(C,\gamma_1)\,|s\rangle \\ &= a(\beta,\gamma,z). \end{aligned}$$

□

Note that in our case:

$$C = \sum_{\alpha=1}^{m} C_\alpha = \sum_{\langle kl\rangle} Z_kZ_l, \tag{1.44}$$

$$B = \sum_{j=1}^{n} B_j = -\sum_{j=1}^{n} X_j \quad \Longrightarrow \quad |s\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{z\in\{0,1\}^n} |z\rangle\,, \tag{1.45}$$

the conditions a., b., and c. are satisfied:

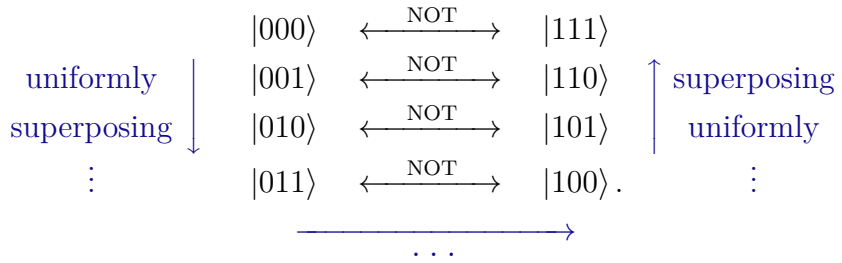
a. $\forall j \in \{1,2,\dots,n\}:\ [-X_j, X_1\dots X_n] = 0,$

b. $\forall k,l \in \{1,2,\dots,n\}$. If $k \neq l$:

$$\begin{aligned} [Z_kZ_l, X_1\dots X_n] &= Z_kZ_lX_1\dots X_n - X_1\dots X_nZ_kZ_l \\ &\overset{\{X,Z\}=0}{=} -Z_kX_k \\ &= Z_kZ_lX_1\dots X_n - \overbrace{X_kZ_k}\cdot\underbrace{X_lZ_l}\cdot X_1\dots \cancel{X_k}\dots\cancel{X_l}\dots X_n \\ &\qquad\qquad \underset{\{X,Z\}=0}{=} -Z_lX_l \\ &= Z_kZ_lX_1\dots X_n - Z_kZ_lX_1\dots X_n \\ &= 0. \end{aligned}$$

For $k = l$, it holds true trivially as $Z^2 = \mathbb{I}$.

c. $X_1X_2\dots X_n\,|s\rangle = |s\rangle$ – the uniform superposition of all computational basis states is the same uniform superposition after applying NOT on all the qubits. To illustrate, take a look at a special case of 3 qubits:



After applying NOT, the set of states, which we take a uniform superposition of, will be the same. Another possibility is to recall that $|+\rangle$ is the $+1$ eigenstate (= with eigenvalue 1) of the operator X, and therefore, $|s\rangle = |+\rangle^{\otimes n}$ is a $+1$ eigenstate of any combination of Xs.

Therefore, the NOT symmetry applies.

Let us continue with the QAOA_1 2-vertex-line example. The remaining question is: *Which parameters (angles γ and β) should we choose in order to obtain the string with the maximal cut?* In QAOA, we directly measure $\left|\vec{\gamma},\vec{\beta}\right\rangle$ in the computational basis. Thus, we can reformulate the question into: *Which parameters maximize the probability of obtaining the right result:*

$$\left|\left\langle 01\middle|\vec{\gamma},\vec{\beta}\right\rangle\right|^2 = \left|\left\langle 10\middle|\vec{\gamma},\vec{\beta}\right\rangle\right|^2 = \frac{1}{4}\cdot\left|e^{i\gamma}\cos(2\beta)+ie^{-i\gamma}\sin(2\beta)\right|^2? \tag{1.46}$$

To answer the question, let us first simplify the formula:

$$\begin{aligned}
\frac{1}{4}\cdot\left|e^{i\gamma}\cos(2\beta)+ie^{-i\gamma}\sin(2\beta)\right|^2 = \\
&= \frac{1}{4}\cdot\left|e^{i\gamma}\right|^2\cdot\left|\cos(2\beta)+ie^{-2i\gamma}\sin(2\beta)\right|^2 \\
&= \frac{1}{4}\Big(\cos(2\beta)+ie^{-2i\gamma}\sin(2\beta)\Big)\Big(\cos(2\beta)+ie^{-2i\gamma}\sin(2\beta)\Big)^* \\
&= \frac{1}{4}\left(\underbrace{\cos^2(2\beta)+\sin^2(2\beta)}_{=1} - i\sin(2\beta)\cos(2\beta)\cdot\underbrace{\Big(e^{2i\gamma}-e^{-2i\gamma}\Big)}_{=2i\sin(2\gamma)}\right) \\
&= \frac{1}{4}\Big(1+\sin(2\gamma)\sin(4\beta)\Big).
\end{aligned}$$

Now, finding the optimal parameters $\{\beta,\gamma\}$ is trivial as the optimal parameters $\{\beta,\gamma\}$ are those that maximize $\sin(2\gamma)\sin(4\beta)$. The function achieves its maximum (value 1) when

- $\sin 2\gamma = \sin 4\beta = 1$ – this is when $\beta = \frac{\pi}{8}+k\frac{\pi}{2};\ k\in\mathbb{Z}$ and $\gamma = \frac{\pi}{4}+l\pi;\ l\in\mathbb{Z}$, or
- $\sin 2\gamma = \sin 4\beta = -1$ – this is when $\beta = \frac{3\pi}{8}+k\frac{\pi}{2};\ k\in\mathbb{Z}$ and $\gamma = \frac{3\pi}{4}+l\pi;\ l\in\mathbb{Z}$.

To conclude, if we want the optimal MaxCut solution, we choose

- either $\beta = \frac{\pi}{8}+k\frac{\pi}{2};\ k\in\mathbb{Z}$ and $\gamma = \frac{\pi}{4}+l\pi;\ l\in\mathbb{Z}$

- or $\beta = \frac{3\pi}{8} + k\frac{\pi}{2};\ k \in \mathbb{Z}$ and $\gamma = \frac{3\pi}{4} + l\pi;\ l \in \mathbb{Z}$.

The corresponding state $\left|\vec{\gamma}, \vec{\beta}\right\rangle$ prepared in QAOA is (up to a global phase) the Bell state

$$\left|\Psi^+\right\rangle = \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right), \tag{1.47}$$

which when measured in the computational basis gives either 01 or 10, both with equal, $\frac{1}{2}$, probability. But either way, we achieved the desired result.

Now, what if we would rather want to increase the probability of measuring 00 or 11 instead? We already know that $a(\beta, \gamma, 00) = a(\beta, \gamma, 11)$. Therefore, the best we can hope for is reaching the Bell state

$$\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right). \tag{1.48}$$

Are we able to achieve this? By a very similar calculation, we would find out that we can. We reach the state when using

- either $\beta = \frac{\pi}{8} + k\frac{\pi}{2};\ k \in \mathbb{Z}$ and $\gamma = \frac{3\pi}{4} + l\pi;\ l \in \mathbb{Z}$
- or $\beta = \frac{3\pi}{8} + k\frac{\pi}{2};\ k \in \mathbb{Z}$ and $\gamma = \frac{\pi}{4} + l\pi;\ l \in \mathbb{Z}$

in the preparation.

As a side note, not connected to Max-Cut, we can also ask: *What about the remaining Bell states:*

$$\left|\Psi^-\right\rangle = \frac{1}{\sqrt{2}}\left(|01\rangle - |10\rangle\right), \tag{1.49}$$

$$\left|\Phi^-\right\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right)? \tag{1.50}$$

Can we prepare them, too? Unfortunately, the NOT symmetry of the Hamiltonians we use does not allow it. At least not when taking $|s\rangle = \frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}|z\rangle$ as the initial state, together with the X's and ZZ's on interactions. But of course, if we already have one Bell state, we can easily reach also the others, simply by applying locally X or Z.

$$\begin{array}{ccc}
\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) & \xleftrightarrow[(Z_1 \text{ or } Z_2)]{Z} & \left|\Phi^-\right\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right) \\
X \updownarrow (X_1 \text{ or } X_2) & & (-X_1 \text{ or } X_2) \updownarrow X \\
\left|\Psi^+\right\rangle = \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right) & \xleftrightarrow[Z]{(Z_1 \text{ or } -Z_2)} & \left|\Psi^-\right\rangle = \frac{1}{\sqrt{2}}\left(|01\rangle - |10\rangle\right)
\end{array}$$

Interestingly enough, this additional step creates a new, valid, QAOA protocol.

First, let us apply the Z gate:

$$Z_1 \left|\vec{\gamma},\vec{\beta}\right\rangle = Z_1 e^{i\beta X_2} e^{i\beta X_1} e^{-i\gamma Z_1 Z_2} |s\rangle = \overset{\{X_1,Z_1\}=0}{=} e^{i\beta X_2} e^{-i\beta X_1} e^{-i\gamma Z_1 Z_2} Z_1 |+\rangle \otimes |+\rangle = \frac{1}{2} e^{i\beta X_2} e^{-i\beta X_1} e^{-i\gamma Z_1 Z_2} |-\rangle \otimes |+\rangle .$$

Now, observe that $Z_1 \left|\vec{\gamma},\vec{\beta}\right\rangle$ is a new QAOA$_1$ $\left|\vec{\gamma},\vec{\beta}\right\rangle$ state, only this time with the initial Hamiltonian $B = X_2 - X_1$. What's more, $|-\rangle \otimes |+\rangle$ is a ground state of B as desired. And the best part is that now, instead of NOT symmetry, we have NOT antisymmetry! Therefore, as one can observe,

$$Z_1 \left|\vec{\gamma},\vec{\beta}\right\rangle = \frac{1}{2} \begin{pmatrix} e^{-i\gamma}\cos(2\beta) + ie^{i\gamma}\sin(2\beta) \\ e^{i\gamma}\cos(2\beta) + ie^{-i\gamma}\sin(2\beta) \\ -e^{i\gamma}\cos(2\beta) - ie^{-i\gamma}\sin(2\beta) \\ -e^{-i\gamma}\cos(2\beta) - ie^{i\gamma}\sin(2\beta) \end{pmatrix}$$

is exactly the $\left|\vec{\gamma},\vec{\beta}\right\rangle$ given in equation 1.41 with the symmetry adjusted. Hence, we already know, which angles to use to obtain $|\Phi^-\rangle$ and $|\Psi^-\rangle$.

Second, let us apply the X gate:

$$X_1 \left|\vec{\gamma},\vec{\beta}\right\rangle = X_1 e^{i\beta X_2} e^{i\beta X_1} e^{-i\gamma Z_1 Z_2} |s\rangle \overset{\{X_1,Z_1\}=0}{=} e^{i\beta X_2} e^{i\beta X_1} e^{i\gamma Z_1 Z_2} \underbrace{X_1 |s\rangle}_{=|s\rangle}.$$ [19]

We see that the X gate changed only the problem we solve. The new problem Hamiltonian is: $C = -Z_1 Z_2$. Now, we receive the energy penalty when the neighbors disagree and the reward when they agree. We will call this problem "$\mathcal{O}(1)$-MinCut" as it is pretty trivial to solve. But somehow, in this alternating Hamiltonians setting, the original (hard) MaxCut problem and this new (easy) "$\mathcal{O}(1)$-MinCut" problem are similar. Let us explain. Instead of giving the minus sign to C, we could have hand it to the parameter γ and claim that γ is the one that has changed (to $-\gamma$). Hence,

$$\left|\vec{\gamma},\vec{\beta}\right\rangle_{\text{MaxCut}} = \left|(-\vec{\gamma}),\vec{\beta}\right\rangle_{\mathcal{O}(1)\text{-MinCut}}. \tag{1.51}$$

But does this mean that we have discovered a viable reduction from MaxCut to $\mathcal{O}(1)$-MinCut?

Unfortunately not. This result (on its own) does not guarantee that finding the optimal parameters is easy,[20] nor does it ensure that the optimal parameters correspond to a high probability of measuring the optimal solution[21]. It does not even guarantee

[19] We would obtain exactly the same result for X_2.

[20] As it turns out, the $\mathcal{O}(1)$-MinCut problem is no longer effortless in this setting (see Section 2.3).

[21] By *optimal parameters*, we mean those that minimize the cost function.

that the parameters which are optimal for one problem are optimal for the other one as well. For illustration, take a look at the usual cost function:

$$F_p(\vec{\gamma},\vec{\beta}) = \left\langle \vec{\gamma},\vec{\beta}\right| C \left|\vec{\gamma},\vec{\beta}\right\rangle .$$

The cost function is practically the same, but it differs in the sign as

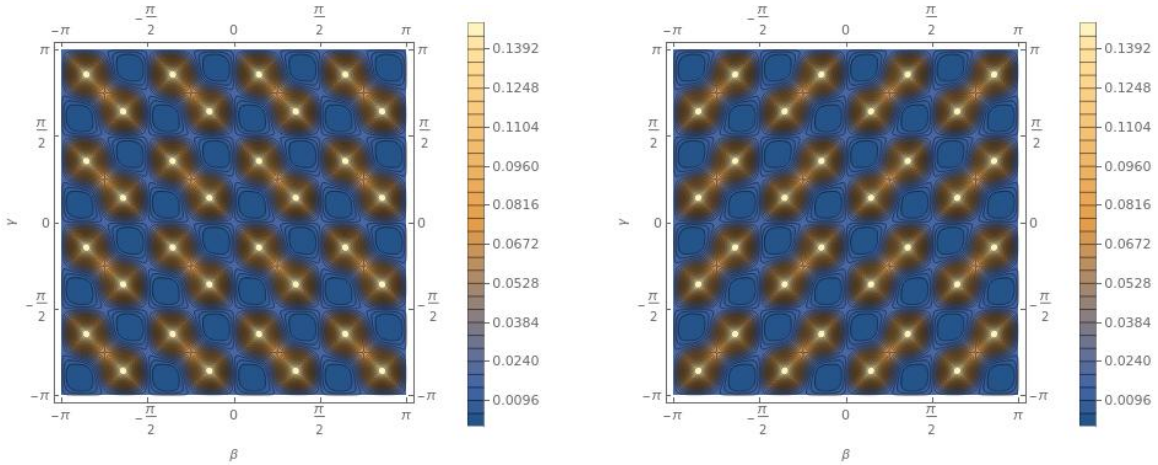
$$C_{\text{MaxCut}} = -C_{\mathcal{O}(1)-\text{MinCut}}.$$

Hence, for MaxCut, we are searching for the minimum of $\left\langle \vec{\gamma},\vec{\beta}\right| C_{\text{MaxCut}} \left|\vec{\gamma},\vec{\beta}\right\rangle$, while for $\mathcal{O}(1)$-MinCut, we wish to find its maximum. And these might be two very different problems. To sum up, we have not found a reduction from one problem to the other. All that 1.51 tells us is that the set of states one can prepare using QAOA with Hamiltonian for one of the problems is the same one can reach using the other problem's Hamiltonian (as C).

Nevertheless, as it turns out, the parameters which are optimal for $\mathcal{O}(1)$-MinCut are optimal (with the slight change: $\gamma \to -\gamma$) also for MaxCut, but only when the graph is bipartite (in other words, there exists a string z, such that $C_{\text{MaxCut}}(z) = m\,(= \#\text{edges})$). Let us underline that for these instances, it is not hard to find out whether the optimal solution exists (as we have already mentioned, 2-SAT is in P). To illustrate, take a look at the following contour plots depicting the probability of measuring a computational basis state $|\psi\rangle$:

$$p_\psi(\vec{\gamma},\vec{\beta}) = \left|\left\langle \psi\middle|\vec{\gamma},\vec{\beta}\right\rangle\right|^2,$$

where $|\psi\rangle = |000000\rangle$ ($\mathcal{O}(1)$-MinCut solution) for the graphs on the left and $|010101\rangle$ (MaxCut solution) for the ones on the right.[22] For the 6-vertex ring, there exists optimal solution. And as we can see, the contour plots really are mirror images of one another:



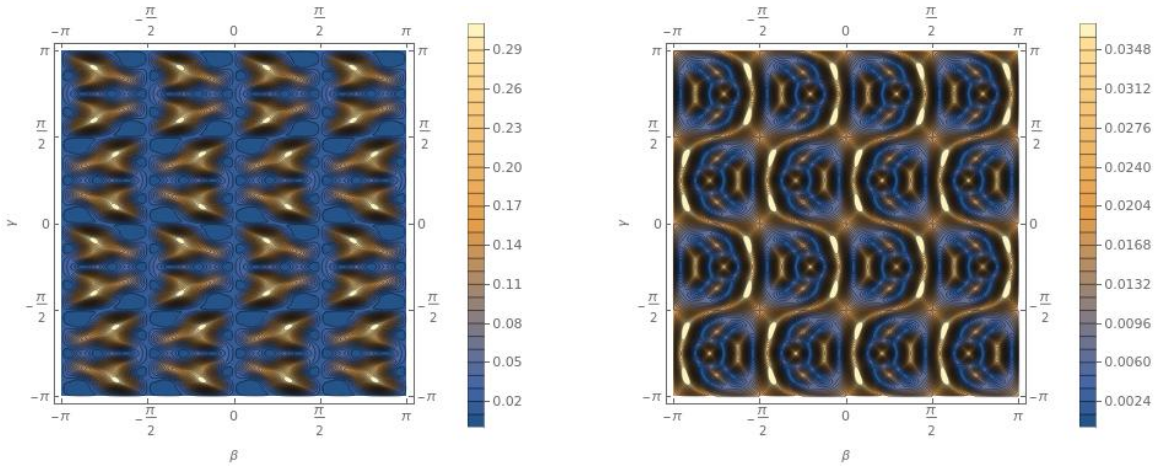
(a) A contour plot of $p_{000000}(\vec{\gamma},\vec{\beta})$. (b) A contour plot of $p_{010101}(\vec{\gamma},\vec{\beta})$.

Figure 1.7: MaxCut on 6-vertex ring.

[22] The choice of $n = 6$ was arbitrary. We could have, as well, chosen any other n.

The same is true for all n-even rings, for all lines[23] (= rings to whom 1 edge was stolen), and for many other graphs. One may observe that the optimal angles for generating $|\Phi^+\rangle$ and $|\Psi^+\rangle$, which we have calculated earlier in this section, are interchangeable by the sign change for γ.

However, e.g., for complete graphs (= graphs where each vertex shares an edge with all other vertices), this is no longer true. For instance, take a look at this 6-vertex example:



(a) A contour plot of $p_{000000}(\vec{\gamma},\vec{\beta})$. (b) A contour plot of $p_{010101}(\vec{\gamma},\vec{\beta})$.

Figure 1.8: MaxCut on 6-vertex complete graph.

The contour plots are very different.

Now, we all probably ask ourselves: *Why is that? Why are these probabilities symmetric only when the graph is bipartite?*

Let us first take a look at a concrete example – a 4-vertex ring (a square).

$$p_{0101}(\vec{\gamma},\vec{\beta}) = \left|\left\langle 0101 \middle| \vec{\gamma},\vec{\beta}\right\rangle\right|^2 =$$

$$= \langle 0101| \prod_{j=1}^{4} e^{i\beta X_j} \prod_{\langle kl\rangle \in \square} e^{-i\gamma Z_k Z_l} |s\rangle \langle s| \prod_{\langle kl\rangle \in \square} e^{i\gamma Z_k Z_l} \prod_{j=1}^{4} e^{-i\beta X_j} |0101\rangle =$$

$$= \langle 0000| X_2X_4 e^{i\beta X_4} e^{i\beta X_3} e^{i\beta X_2} e^{i\beta X_1} e^{-i\gamma Z_1Z_2} e^{-i\gamma Z_2Z_3} e^{-i\gamma Z_3Z_4} e^{-i\gamma Z_4Z_1} |s\rangle \cdot$$

$$\cdot \langle s| e^{i\gamma Z_1Z_2} e^{i\gamma Z_2Z_3} e^{i\gamma Z_3Z_4} e^{i\gamma Z_4Z_1} e^{-i\beta X_1} e^{-i\beta X_2} e^{-i\beta X_3} e^{-i\beta X_4} X_2X_4 |0000\rangle \overset{\{X_k,Z_k\}=0}{=}$$

$$\overset{\{X_k,Z_k\}=0}{=} \langle 0000| e^{i\beta X_4} e^{i\beta X_3} e^{i\beta X_2} e^{i\beta X_1} e^{i\gamma Z_1 \mathbf{Z_2}} e^{i\gamma \mathbf{Z_2} Z_3} e^{i\gamma Z_3 \mathbf{Z_4}} e^{i\gamma \mathbf{Z_4} Z_1} X_4X_2 |s\rangle \cdot$$

$$\cdot \langle s| X_4X_2 e^{-i\gamma Z_1 \mathbf{Z_2}} e^{-i\gamma \mathbf{Z_2} Z_3} e^{-i\gamma Z_3 \mathbf{Z_4}} e^{-i\gamma \mathbf{Z_4} Z_1} e^{-i\beta X_1} e^{-i\beta X_2} e^{-i\beta X_3} e^{-i\beta X_4} |0000\rangle \overset{X_2X_4|s\rangle=|s\rangle}{=}$$

$$\overset{X_2X_4|s\rangle=|s\rangle}{=} \langle 0000| \prod_{j=1}^{4} e^{i\beta X_j} \prod_{\langle kl\rangle \in \square} e^{i\gamma Z_k Z_l} |s\rangle \langle s| \prod_{\langle kl\rangle \in \square} e^{-i\gamma Z_k Z_l} \prod_{j=1}^{4} e^{-i\beta X_j} |0000\rangle =$$

$$\left|\left\langle 0000 \middle| (-\vec{\gamma}),\vec{\beta}\right\rangle\right|^2 = p_{0000}\left((-\vec{\gamma}),\vec{\beta}\right).$$

[23] Hence, also the 2-vertex line example.

So, the main reason for the equality of these two probabilities is exactly the existence of optimal solution. In other words, it is because for each pair Z_kZ_l from $C = \sum Z_kZ_l$, there is exactly one X to anticommute through. Thus, the transition between the $\mathcal{O}(1)$-MinCut and MaxCut optimal states – pulling out the Xs and anticommuting/-commuting them to $|s\rangle$ where they disappear (because $|s\rangle = |+\rangle^{\otimes n}$) – causes precisely the $\gamma \leftrightarrow -\gamma$ mirroring.

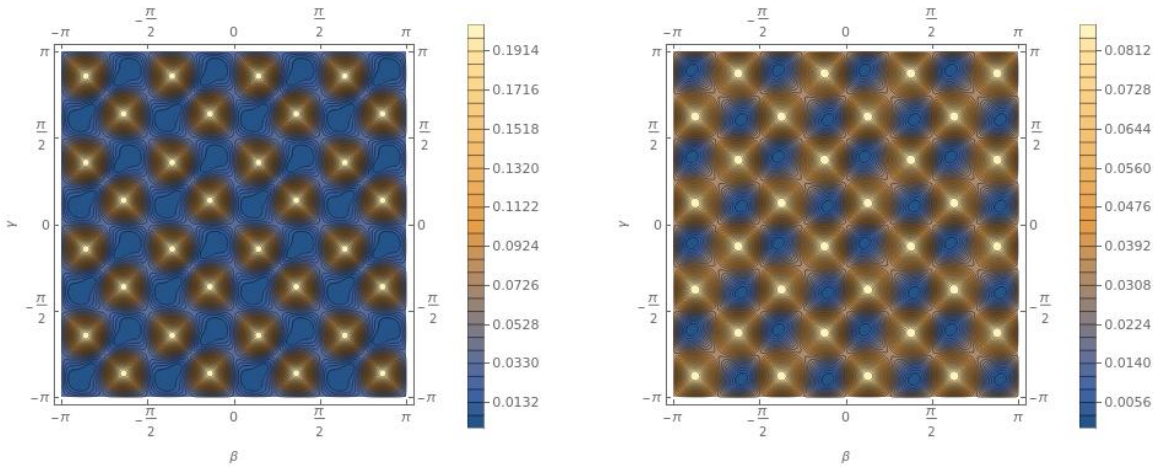
Now, let us take a look at a 5-vertex ring. Here $C = Z_1Z_2 + Z_2Z_3 + Z_3Z_4 + Z_4Z_5 + Z_5Z_1$. There is no way how to get a set of Xs such that for each Z_kZ_l there would be exactly one X in the set which anticommutes with Z_kZ_l. There simply exists no all-clauses-satisfying solution for MaxCut. So, in contrast to the 4-vertex case, where

$$\begin{aligned} a(\vec{\gamma},\vec{\beta},0101) &= \langle 0101|\, e^{i\beta X_4}e^{i\beta X_3}e^{i\beta X_2}e^{i\beta X_1}e^{-i\gamma Z_1Z_2}e^{-i\gamma Z_2Z_3}e^{-i\gamma Z_3Z_4}e^{-i\gamma Z_4Z_1}\,|s\rangle = \\ &= \langle 0000|\, e^{i\beta X_4}e^{i\beta X_3}e^{i\beta X_2}e^{i\beta X_1}e^{i\gamma Z_1Z_2}e^{i\gamma Z_2Z_3}e^{i\gamma Z_3Z_4}e^{i\gamma Z_4Z_1}\,|s\rangle = a((-\vec{\gamma}),\vec{\beta},0000), \end{aligned}$$

for this 5-vertex ring, we have:

$$\begin{aligned} &a(\vec{\gamma},\vec{\beta},01010) = \\ &= \langle 01010|\, e^{i\beta X_5}e^{i\beta X_4}e^{i\beta X_3}e^{i\beta X_2}e^{i\beta X_1}e^{-i\gamma Z_1Z_2}e^{-i\gamma Z_2Z_3}e^{-i\gamma Z_3Z_4}e^{-i\gamma Z_4Z_5}e^{-i\gamma Z_5Z_1}\,|s\rangle = \\ &= \langle 00000|\, e^{i\beta X_5}e^{i\beta X_4}e^{i\beta X_3}e^{i\beta X_2}e^{i\beta X_1}e^{i\gamma Z_1Z_2}e^{i\gamma Z_2Z_3}e^{i\gamma Z_3Z_4}e^{i\gamma Z_4Z_5}\boldsymbol{e^{-i\gamma Z_5Z_1}}\,|s\rangle \neq \\ &\neq a((-\vec{\gamma}),\vec{\beta},00000). \end{aligned}$$

Correspondingly, as we may observe, the contour plots are slightly different:



(a) A contour plot of $p_{00000}(\vec{\gamma},\vec{\beta})$.

(b) A contour plot of $p_{01010}(\vec{\gamma},\vec{\beta})$.

Figure 1.9: MaxCut on 5-vertex ring.

To conclude, let us note that the optimal solution guarantees also equality of the cost functions:

$$\Big(F_p(\vec{\gamma},\vec{\beta})\Big)_{\text{MaxCut}} = \Big(F_p(\vec{\gamma},\vec{\beta})\Big)_{\mathcal{O}(1)-\text{MinCut}}. \tag{1.52}$$

That is because:

1. we can pull out any set of Xs from $|s\rangle$ since

$$\forall k \in \{1, 2, \ldots, n\} : X_k |+\rangle^{\otimes n} = |+\rangle^{\otimes n} = |s\rangle ,$$

2. the existence of optimal solution guarantees that there exists $\{X_a, X_b, \ldots, X_c\}$ such that for each term $Z_k Z_l$ from $C = \sum Z_k Z_l$ there is exactly one X in the set which will anticommute and all others will commute.

Hence, we can pull out the operator $X_a X_b \cdots X_c$ from $|s\rangle$ at one side of $F_p(\vec{\gamma}, \vec{\beta})$ and anticommute/commute it to the other side, where it will dissappear, changing each C to $-C$ in the process:

$$\begin{aligned}
\left(F_p\left(\vec{\gamma}, \vec{\beta}\right)\right)_{\text{MaxCut}} &= \left\langle \vec{\gamma}, \vec{\beta} \right| C_{\text{MaxCut}} \left| \vec{\gamma}, \vec{\beta} \right\rangle = \\
&= \langle s| e^{i\gamma_1 C_{\text{MaxCut}}} U^\dagger(B, \beta_1) \cdots e^{i\gamma_p C_{\text{MaxCut}}} U^\dagger(B, \beta_p) \cdot C_{\text{MaxCut}} \cdot \\
&\qquad \cdot U(B, \beta_p) e^{-i\gamma_p C_{\text{MaxCut}}} \cdots U(B, \beta_1) e^{-i\gamma_1 C_{\text{MaxCut}}} |s\rangle = \\
&= \langle s| \boldsymbol{X_a X_b \cdots X_c} e^{i\gamma_1 C_{\text{MaxCut}}} U^\dagger(B, \beta_1) \cdots e^{i\gamma_p C_{\text{MaxCut}}} U^\dagger(B, \beta_p) \cdot C_{\text{MaxCut}} \cdot \\
&\qquad \cdot U(B, \beta_p) e^{-i\gamma_p C_{\text{MaxCut}}} \cdots U(B, \beta_1) e^{-i\gamma_1 C_{\text{MaxCut}}} |s\rangle = \\
&= \langle s| e^{-i\gamma_1 C_{\text{MaxCut}}} U^\dagger(B, \beta_1) \cdots e^{-i\gamma_p C_{\text{MaxCut}}} U^\dagger(B, \beta_p) \cdot \overbrace{-C_{\text{MaxCut}}}^{=\boldsymbol{C_{\mathcal{O}(1)-\text{MinCut}}}} \cdot \\
&\qquad \cdot U(B, \beta_p) e^{+i\gamma_p C_{\text{MaxCut}}} \cdots U(B, \beta_1) e^{+i\gamma_1 C_{\text{MaxCut}}} \boldsymbol{X_a X_b \cdots X_c} |s\rangle = \\
&= \langle s| e^{i\gamma_1 C_{\mathcal{O}(1)-\text{MinCut}}} U^\dagger(B, \beta_1) \cdots e^{i\gamma_p C_{\mathcal{O}(1)-\text{MinCut}}} U^\dagger(B, \beta_p) \cdot C_{\mathcal{O}(1)-\text{MinCut}} \cdot \\
&\qquad \cdot U(B, \beta_p) e^{-i\gamma_p C_{\mathcal{O}(1)-\text{MinCut}}} \cdots U(B, \beta_1) e^{-i\gamma_1 C_{\mathcal{O}(1)-\text{MinCut}}} |s\rangle = \\
&= \left(F_p(\vec{\gamma}, \vec{\beta})\right)_{\mathcal{O}(1)-\text{MinCut}} .
\end{aligned}$$

Let us now move on to more complex and interesting graphs.

3-regular graphs

In this example – MaxCut on graphs of fixed degree 3 (= 3-regular graphs) – we present some of the observations and results from the original paper [11] by Farhi et al. There, they suggested an elegant approach for classical preprocessing, which enables us to find the optimal angles, for fixed p (or, more generally, for p not growing with n) and graphs of bounded degree, efficiently – possibly even analytically, but mainly without any exhaustive optimization (at least for small p).

The key observation is the following. Recall that our goal is to minimize

$$F_p(\vec{\gamma}, \vec{\beta}) = \left\langle \vec{\gamma}, \vec{\beta} \right| C \left| \vec{\gamma}, \vec{\beta} \right\rangle = \sum_{\langle kl \rangle} \langle s| \underbrace{U^\dagger(C, \gamma_1) \ldots U^\dagger(B, \beta_p) C_{\langle kl \rangle} U(B, \beta_p) \ldots U(C, \gamma_1)}_{=: \boldsymbol{F^p_{\langle kl \rangle}(\vec{\gamma}, \vec{\beta})}} |s\rangle . \tag{1.53}$$

The operator associated with edge $\langle kl \rangle$, which we denote $F^p_{\langle kl \rangle}(\vec{\gamma}, \vec{\beta})$, involves only qubits k and l and those qubits whose distance on the graph from k or l is less than or equal to p. *Why?* Let us work our way through the computation to see this.

First, consider only the middle part of $F^p_{\langle kl \rangle}(\vec{\gamma}, \vec{\beta})$:

$$U^\dagger(B, \beta_p) C_{\langle kl \rangle} U(B, \beta_p) =$$
$$= e^{-i\beta_p X_n} \dots e^{-i\beta_p X_l} \dots e^{-i\beta_p X_k} \dots e^{-i\beta_p X_1} C_{\langle kl \rangle} e^{i\beta_p X_1} \dots e^{i\beta_p X_k} \dots e^{i\beta_p X_l} \dots e^{i\beta_p X_n}.$$

All factors $e^{i\beta_p X_j}$ in the operator $U(B, \beta_p)$ which do not involve qubits k or l commute through $C_{\langle kl \rangle}$ and annihilate each other: $e^{-i\beta_p X_j} e^{i\beta_p X_j} = \mathbb{I}$. In the result, we are left only with

$$e^{-i\beta_p X_l} e^{-i\beta_p X_k} C_{\langle kl \rangle} e^{i\beta_p X_k} e^{i\beta_p X_l}.$$

Second, let us take a look at the $p = 1$ case:

$$\begin{aligned} F^1_{\langle kl \rangle}(\gamma_1, \beta_1) &= U^\dagger(C, \gamma_1) U^\dagger(B, \beta_1) C_{\langle kl \rangle} U(B, \beta_1) U(C, \gamma_1) \\ &= U^\dagger(C, \gamma_1) e^{-i\beta_1 X_l} e^{-i\beta_1 X_k} C_{\langle kl \rangle} e^{i\beta_1 X_k} e^{i\beta_1 X_l} U(C, \gamma_1) \\ &= \Big(\prod\nolimits_{\langle ab \rangle} e^{i\gamma_1 C^\dagger_{\langle ab \rangle}} \Big) e^{-i\beta_1 X_l} e^{-i\beta_1 X_k} C_{\langle kl \rangle} e^{i\beta_1 X_k} e^{i\beta_1 X_l} \Big(\prod\nolimits_{\langle ab \rangle} e^{-i\gamma_1 C_{\langle ab \rangle}} \Big). \end{aligned}$$

All factors $e^{i\gamma_1 C^\dagger_{\langle ab \rangle}}$ (or $e^{-i\gamma_1 C_{\langle ab \rangle}}$) corresponding to edges that include neither qubit k nor l will commute through and cancel out. So, in this case, the operator $U(C, \gamma_1)$ can be restricted only to those factors that involve the edge $\langle kl \rangle$ or adjacent edges to $\langle kl \rangle$. Third, let us consider the operator

$$U^\dagger(B, \beta_1) F^1_{\langle kl \rangle}(\gamma_2, \beta_2) U(B, \beta_1).$$

We already know that $F^1_{\langle kl \rangle}$ involves only qubits k and l and qubits that lie on the adjacent edges to the edge $\langle kl \rangle$ (= those qubits whose distance on the graph from k or l is less than or equal to 1). Thus, the only factors in $U(B, \beta_1)$ that cannot commute through are those involving qubits k and l and their neighbors. Now, we can take a look at $F^2_{\langle kl \rangle}(\vec{\gamma}, \vec{\beta})$.

$$F^2_{\langle kl \rangle}(\gamma_1\gamma_2, \beta_1\beta_2) = U^\dagger(C, \gamma_1) U^\dagger(B, \beta_1) F^1_{\langle kl \rangle}(\gamma_2, \beta_2) U(B, \beta_1) U(C, \gamma_1).$$

$U(C, \gamma_1)$ adds qubits at the distance 2 from k or l. Hence, in a $p = 3$ algorithm, $U(C, \gamma_2)$ (from the 2nd round) and $U(B, \beta_1)$ (from round 3) already involve qubits whose distance from k or l is less than or equal to 2. $U(C, \gamma_1)$ (from round 3) increases the distance to 3. In the fourth round, $U(B, \beta)$ keeps the maximal distance 3 and $U(C, \gamma)$ increases the distance to 4, and so on. $F^p_{\langle kl \rangle}(\vec{\gamma}, \vec{\beta})$, therefore, involves only edges at most p steps away from $\langle kl \rangle$ and the qubits at the ends of those edges.

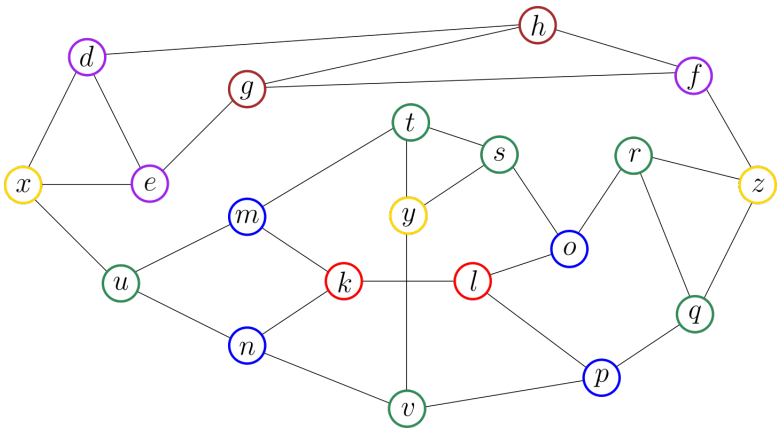


Figure 1.10: An example of a 3-regular graph. We are analyzing $F^p_{\langle kl\rangle}(\vec{\gamma},\vec{\beta})$ (Eqn. 1.53) – operator associated with edge $\langle kl\rangle$. The operator involves only qubit k and l (the red ones) and those qubits whose distance on the graph from k or l is less than or equal to p. In this graph, qubits whose distance from k or l is **1** are the blue ones – $\{m,n,o,p\}$, qubits whose distance is **2** are the green ones – $\{q,r,s,t,u,v\}$, **3** – the yellow ones – $\{x,y,z\}$, **4** – the violet ones – $\{d,e,f\}$, and **5** – the brown ones – $\{g,h\}$. Thus, $F^p_{\langle kl\rangle}(\vec{\gamma},\vec{\beta})$ for $p\geq 5$ already involves all qubits.

Let us take a look at a concrete example. Consider the edge $\langle kl\rangle$ from the graph depicted in Figure (1.10). Then

$$F^1_{\langle kl\rangle}(\gamma,\beta) = e^{i\gamma\left(C^\dagger_{\langle kl\rangle}+C^\dagger_{\langle km\rangle}+C^\dagger_{\langle kn\rangle}+C^\dagger_{\langle lo\rangle}+C^\dagger_{\langle lp\rangle}\right)}e^{-i\beta(X_k+X_l)}C_{\langle kl\rangle}\cdot$$
$$\cdot e^{i\beta(X_k+X_l)}e^{-i\gamma\left(C_{\langle kl\rangle}+C_{\langle km\rangle}+C_{\langle kn\rangle}+C_{\langle lo\rangle}+C_{\langle lp\rangle}\right)},$$

$$F^2_{\langle kl\rangle}(\gamma_1\gamma_2,\beta_1\beta_2) =$$
$$= e^{i\gamma_1\left(C^\dagger_{\langle kl\rangle}+C^\dagger_{\langle km\rangle}+C^\dagger_{\langle kn\rangle}+C^\dagger_{\langle lo\rangle}+C^\dagger_{\langle lp\rangle}+C^\dagger_{\langle mu\rangle}+C^\dagger_{\langle mt\rangle}+C^\dagger_{\langle nu\rangle}+C^\dagger_{\langle nv\rangle}+C^\dagger_{\langle os\rangle}+C^\dagger_{\langle or\rangle}+C^\dagger_{\langle pv\rangle}+C^\dagger_{\langle pq\rangle}\right)}\cdot$$
$$\cdot e^{-i\beta_1(X_k+X_l+X_m+X_n+X_o+X_p)}F^1_{\langle kl\rangle}(\gamma_2,\beta_2)e^{i\beta_1(X_k+X_l+X_m+X_n+X_o+X_p)}\cdot$$
$$\cdot e^{i\gamma_1\left(C_{\langle kl\rangle}+C_{\langle km\rangle}+C_{\langle kn\rangle}+C_{\langle lo\rangle}+C_{\langle lp\rangle}+C_{\langle mu\rangle}+C_{\langle mt\rangle}+C_{\langle nu\rangle}+C_{\langle nv\rangle}+C_{\langle os\rangle}+C_{\langle or\rangle}+r_{\langle pv\rangle}+C_{\langle pq\rangle}\right)},$$

and so on.

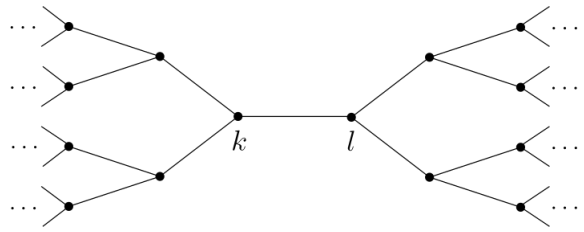
So far, we have shown that $F^p_{\langle kl\rangle}(\vec{\gamma},\vec{\beta})$ depends only on the subgraph (G) involving qubits k and l and those at a distance no more than p away. To continue in the analysis, let us return to equation (1.53). Recall that

$$|s\rangle = |+\rangle_1 |+\rangle_2 \ldots |+\rangle_n . \tag{1.54}$$

The contribution from qubits at a distance greater than p is solely a multiplication by $\langle +|+\rangle = 1$. Hence, we can restrict $|s\rangle$ to the subgraph G, too.

$$|s\rangle \quad \rightarrow \quad |s,G\rangle = \sum_{j\in G}|+\rangle_j . \tag{1.55}$$

To summarize, as well as $F^p_{\langle kl\rangle}(\vec{\gamma},\vec{\beta})$, each term in equation (1.53) depends only on the subgraph composed of the qubits at a distance no more than p away from k or l, too. The number of qubits forming these subgraphs is, for graphs with bounded degree d, independent of n. The worst (maximal number of qubits in the subgraph) that can happen is when the subgraph is a tree:



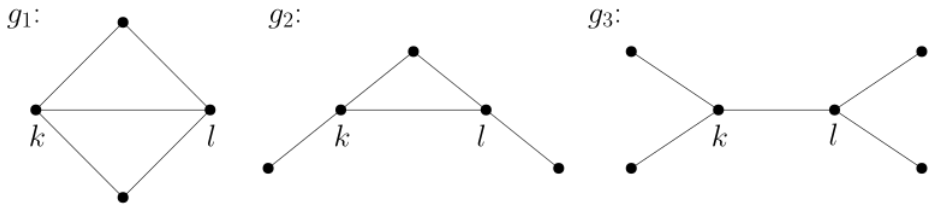
In that case, the graph involves

$$n_G = 2\left[(d-1)^0 + (d-1)^1 + \cdots + (d-1)^p\right] = 2\left[\frac{(d-1)^{p+1}-1}{(d-1)-1}\right] \tag{1.56}$$

qubits. In our case of 3-regular graphs, $d = 3$ and

$$n_G \leq 2\cdot\left(2^{p+1}-1\right). \tag{1.57}$$

Let us also mention that for each p there are only finitelly many subgaph types. To illustrate, for $p = 1$, these are all the possibilities:



We can simplify the sum in equation (1.53) into a sum over the possible subgraph types:

$$F_p(\vec{\gamma},\vec{\beta}) = \sum_g w_g f_g(\vec{\gamma},\vec{\beta}), \tag{1.58}$$

where

- $f_g(\vec{\gamma},\vec{\beta}) = \langle s, g(k,l)|\, U^\dagger(C_{g(k,l)},\gamma_1)\ldots U^\dagger(B_{g(k,l)},\beta_p) C_{\langle kl\rangle} U(B_{g(k,l)},\beta_p)\ldots U(C_{g(k,l)},\gamma_1)\,|s,g(k,l)\rangle$,
  - $g(k,l)$ is a subgraph of type g whose "core" vertices are labeled k and l,
  - C_G is defined as C restricted to a graph G:

$$C_G = \sum_{\langle ll'\rangle\in G} C_{\langle ll'\rangle}, \tag{1.59}$$

– B_G is defined as B restricted to G:

$$B_G = \sum_{j \in G} X_j, \tag{1.60}$$

- w_g is the number of occurrences of the subgraph g in the original edge sum. While the functions $f_g(\vec{\gamma}, \vec{\beta})$ do not depend on n and m, the weights w_g do. But they are easily evaluated. They are just a simple read of the original graph.

Let us summarize the results on an example of finding the optimal QAOA$_1$ parameters $\{\beta, \gamma\}$ for the graph from Figure (1.10). So, we wish to minimize

$$F_1(\gamma, \beta) = \left\langle \vec{\gamma}, \vec{\beta} \right| C \left| \vec{\gamma}, \vec{\beta} \right\rangle = \sum_{\langle kl \rangle} \langle s | \, U^\dagger(C, \gamma) U^\dagger(B, \beta) C_{\langle kl \rangle} U(B, \beta) U(C, \gamma) \, |s\rangle \, . \tag{1.61}$$

Each term in (1.61) depends only on a subgraph composed of the qubit k and l and their direct neighbors. For 3-regular graphs, there are only 3 possible subgraph types: g_1, g_2, and g_3. Hence, we can rewrite (1.61) as a sum of only 3 terms:

$$F_1(\vec{\gamma}, \vec{\beta}) = \sum_{j=1}^{3} w_{g_j} f_{g_j}(\gamma, \beta). \tag{1.62}$$

Now, let us assign subgraphs to the edges (by reading the graph (1.10)):

$$\begin{aligned}
g_1 \quad &\rightarrow \quad \emptyset, \\
g_2 \quad &\rightarrow \quad \Big\{ \langle de \rangle, \langle dx \rangle, \langle ex \rangle, \langle fg \rangle, \langle qr \rangle, \langle qz \rangle, \langle rz \rangle, \langle st \rangle, \langle sy \rangle, \langle ty \rangle \Big\}, \\
g_3 \quad &\rightarrow \quad \Big\{ \langle dh \rangle, \langle eg \rangle, \langle fz \rangle, \langle gh \rangle, \langle hl \rangle, \langle kl \rangle, \langle km \rangle, \langle kn \rangle, \langle lo \rangle, \langle lp \rangle, \\
&\qquad\qquad \langle mt \rangle, \langle mu \rangle, \langle nu \rangle, \langle nv \rangle, \langle or \rangle, \langle os \rangle, \langle pq \rangle, \langle pv \rangle, \langle ux \rangle, \langle vy \rangle \Big\}.
\end{aligned}$$

We see that $w_{g_1} = 0$, $w_{g_2} = 10$, and $w_{g_3} = 20$. So, we only need to minimize

$$\begin{aligned}
&F_1(\vec{\gamma}, \vec{\beta}) = 10 f_{g_2}(\gamma, \beta) + 20 f_{g_3}(\gamma, \beta) = \\
&= 10 \, \langle s, g_2(d,e) | \, e^{i\gamma \left(C^\dagger_{\langle de \rangle} + C^\dagger_{\langle dx \rangle} + C^\dagger_{\langle dh \rangle} + C^\dagger_{\langle ex \rangle} + C^\dagger_{\langle eg \rangle} \right)} e^{-i\beta(X_d + X_e)} C_{\langle de \rangle} \cdot \\
&\qquad \cdot e^{i\beta(X_d + X_e)} e^{-i\gamma \left(C_{\langle de \rangle} + C_{\langle dx \rangle} + C_{\langle dh \rangle} + C_{\langle ex \rangle} + C_{\langle eg \rangle} \right)} \, |s, g_2(d,e)\rangle + \\
&+ 20 \, \langle s, g_3(d,h) | \, e^{i\gamma \left(C^\dagger_{\langle dh \rangle} + C^\dagger_{\langle dx \rangle} + C^\dagger_{\langle de \rangle} + C^\dagger_{\langle hf \rangle} + C^\dagger_{\langle hg \rangle} \right)} e^{-i\beta(X_d + X_h)} C_{\langle dh \rangle} e^{i\beta(X_d + X_h)} \\
&\qquad e^{-i\gamma \left(C_{\langle dh \rangle} + C_{\langle dx \rangle} + C_{\langle de \rangle} + C_{\langle hf \rangle} + C_{\langle hg \rangle} \right)} \, |s, g_3(d,h)\rangle \, .
\end{aligned} \tag{1.63}$$

To conclude, all we need to build the cost function $F_p(\vec{\gamma}, \vec{\beta})$ (Eqn. 1.58) is to determine all subgraph types g for the given p, where p (the number of rounds in QAOA) is the maximal distance one can walk from the subgraph's core vertices to visit all other subgraph's vertices, together with their corresponding weights w_g, as

these give us the cost function's terms $w_g \cdot f_g(\gamma, \beta)$ (1 subgraph type $\leftrightarrow$ 1 term). Each of these terms shares its locality with the subgraph. QAOA is in this sense a local optimization algorithm, relying on the local structure of the underlying graph. It is its strength (efficiency) as well as a weakness (can't capture global properties). In our own research in Chapter 2, we aim to build specific graphs with properties that can't be assessed by local optimization, but moreover, we claim that QAOA can't possibly faithfully or even approximately prepare a class of states (possible solutions) with only a few rounds. After our own work has progressed, we have learned about the paper of Hastings [22], where he has raised this exact objection.

1.3.2 QAOA vs. classical algorithms

Max E3LIN2

So far, we have considered QAOA applied only to MaxCut. However, in order not to miss out on many interesting results, let us consider the Max E3LIN2 problem (E3 $\rightarrow$ each clause contains exactly 3 variables, LIN2 $\rightarrow$ each constraint is a linear equation mod 2). That is, given n variables ($z = z_1 z_2 \dots z_n$; $z_i = 0, 1$) and m equations of the type

$$(z_i + z_j + z_k) \mod 2 = \begin{cases} 0 \\ 1 \end{cases},$$

which string maximizes the number of satisfied equations?

First, suppose we would only ask: *Is there is a solution*[24]*?* Then, we could simply solve these equations by Gaussian elimination and determine whether the solution exists or not. This decision version of E3LIN2 is in P. However, what if there are many more equations then the variables and there is no solution? What is the maximal number of satisfied equations then?

Let us take a look at this simple algorithm: *Guess a random string.* A random string will satisfy, on average, 1/2 of the equations. That is because each equation is satisfied by four of the eight possible settings of the variables. Now, here is an interesting point to add: it was proven that if there is a efficient classical approximation algorithm for general instances with $r^\star = \left(\frac{1}{2} + \varepsilon\right)$ than P=NP [26]. So, unless P=NP, we can not improve (much) on simple guessing.

But let us simplify the problem a little bit. Let us add a restriction that every bit is in no more than D equations. Here is a storyline of interesting results regarding this bounded-occurence version (as introduced in [10]):

[24] a string satisfying all the equations

2000 Håstad [25] gave a classical algorithm that achieves $r^\star = \left(\frac{1}{2} + \frac{constant}{D}\right)$.

2001 Trevisan [34] showed that the existence of an efficient $r^\star = \left(\frac{1}{2} + \frac{constant}{D^{1/2}}\right)$ classical approximation algorithm, for a sufficiently large constant, would imply P=NP.

2014 First version of [10] by Farhi et al. showed that QAOA_1 applied to an instance of E3LIN2 with m equations produces a string z that satisfies $\left(\frac{1}{2} + \frac{constant}{22D^{3/4}}\right) m$ equations. As far as we know, this is the first time that a quantum algorithm has been proved to achieve a better approximation ratio for a natural NP-hard optimization problem than the best known classical algorithm achieves!

2015 → A classical algorithm was found [4, **?**] that efficiently finds a string satisfying $\left(\frac{1}{2} + \frac{constant}{D^{1/2}}\right) m$ equations.
→ In the second version [10], Farhi et al. improved their earlier analysis and showed that QAOA_1 produces z satisfying $\left(\frac{1}{2} + \frac{constant}{101D^{1/2}\ln D}\right) m$ equations. They also showed that in the typical case (where *typical* means that when specifying the instance of E3LIN2, each triple was said to sum to 0 or 1 with (equal) 50% probability), the quantum algorithm outputs z that satisfies $\left(\frac{1}{2} + \frac{constant}{2\sqrt{3e}D^{1/2}}\right) m$ equations.

So, if nothing else, QAOA at least inspired an improvement in classical algorithms for this bounded occurence E3LIN2.

While this is all very much interesting, we are more interested in results from [22]. In [22], Hastings defined a class of classical optimization algorithms, called *local tensor algorithms*, and showed that a single step version of these algorithms can achieve the same performance as QAOA_1 on MAX-3-LIN-2, and outperform QAOA_1 on all triangle-free instances of MaxCut. Moreover, he showed that for some problems, increasing the depth of QAOA to any bounded amount will not lead to significant improvements in performance.

1.3.3 Further results

Farhi and Harrow [13] shown that efficient sampling from the output of even the lowest depth version of QAOA would collapse the Polynomial Hierarchy. This result suggests that QAOA is a promising candidate for establishing quantum supremacy. As well, this result is directly related to our work, as we have been investigating the output distribution of the low-depth versions of QAOA using numerical simulations. It tells

us we cannot do this efficiently.

In [6], Bravyi et al. argue that the QAOA's performance is severely limited by QAOA's locality and by the NOT symmetry[25]. Motivated by these findings, they proposed a non-local version of QAOA, named RQAOA (R for *recursive*). They numerically compared a level-1 version of RQAOA to the standard QAOA_1 on random 3-regular graphs, using Ising type Hamiltonians[26], and observed that RQAOA_1 significantly outperformed the standard QAOA_1.

Lastly, we would like to mention that a more general approach [19] to QAOA was introduced as well. This extension, the Quantum Alternating Operator Ansatz, suggests that we should consider general parametrized families of unitaries rather than only those unitaries that correspond to the time evolution with respect to local Hamiltonians. So, the Quantum Alternating Operator Ansatz circuit is characterized by two parametrized families of unitary operators:

- A family of *mixing operators* $U_{\text{M}}(\beta)$. The operators should preserve the feasible subspace (hence also the set of feasible solutions), and provide transitions between all pairs of states that correspond to feasible solutions (in other words, transitions between all feasible computational-basis states).
- A family of *mixing operators* $U_{\text{P}}(\gamma)$ that depends on the objective function C.. The operators are required to be diagonal in the computational basis.

Alternating operators from these two families, we create

$$\left|\vec{\gamma},\vec{\beta}\right\rangle = U_{\text{M}}(\beta_p)U_{\text{P}}(\gamma_p)\cdots U_{\text{M}}(\beta_1)U_{\text{P}}(\gamma_1)\left|s\right\rangle .$$

$|s\rangle$ is required only to be trivial to implement. The Quantum Alternating Operator Ansatz consists of the states $\left|\vec{\gamma},\vec{\beta}\right\rangle$. This generalization should allow us to create larger, and potentially more useful, set of states. Hence, it could lead to more promising results. In addition, it may make practical implementation more efficient.

[25] They called this symmetry $\mathbb{Z}_2$-symmetry and the definitions were:

- A local Hamiltonian is said to be $\mathbb{Z}_2$-symmetric if all interaction terms commute with $X^{\otimes n}$.
- A quantum circuit U is said to be $\mathbb{Z}_2$-symmetric if it obeys $UX^{\otimes n} = X^{\otimes n}U$.
- A state ψ is said to be $\mathbb{Z}_2$-symmetric if $X^{\otimes n}\psi = \psi$.

[26] Hamiltonians of type $H = \sum_{k,l} J_{kl}Z_kZ_l$, where $J_{kl} \in \mathbb{R}$. Hence, examples include C_{MaxCut}.

2 Low-order QAOA on symmetric graphs

Now that we are familiar with QAOA and some of its properties, we may proceed with looking for answers to the following questions: *In general, how does QAOA cope with simple MaxCut instances? Is it able to solve them? Does the observation provide some useful insights into QAOA? For example, is it so that the locality of QAOA keeps it from finding the optimal solutions? What are the limits of QAOA? How many rounds do we need to solve these simple instances? Can QAOA be useful even for small p?*

Recall that the goal of QAOA is to find a near-optimal ("good enough") solution. Hence, it might be a good idea to take a wider look at QAOA and ask: *What is the maximal probability of measuring/reaching an arbitrary computational-basis state $|z\rangle$:*

$$\max_{\vec{\gamma},\vec{\beta}} p_z(\vec{\gamma},\vec{\beta}) = \left| \langle z| \underbrace{\prod_j e^{i\beta_p X_j} \prod_{\langle kl\rangle} e^{-i\gamma_p Z_k Z_l} \cdots \prod_j e^{i\beta_1 X_j} \prod_{\langle kl\rangle} e^{-i\gamma_1 Z_k Z_l} |s\rangle}_{=\left|\vec{\gamma},\vec{\beta}\right\rangle} \right|^2 ?$$

For this purpose, we built a set of tools for numerical simulations of QAOA for MaxCut on rings, lines and complete graphs. Among (few) other things, it calculates $\left|\vec{\gamma},\vec{\beta}\right\rangle$'s computational basis amplitudes[1] (for $p = 1$ and $p = 2$) and creates Mathematica notebooks, where we can play with the amplitudes (e.g., find $\max_{\vec{\gamma},\vec{\beta}} p_z(\vec{\gamma},\vec{\beta})$). We explain the idea behind the algorithms as well as list the main things the toolbox provides within Appendix B. The set of simulations is available at https://github.com/dendaxD/QAOA-MaxCut-amplitudes.

Before we move on to our observations, let us eliminate degeneracies in the parameter space.

[1] It generates $\left\langle z\middle|\vec{\gamma},\vec{\beta}\right\rangle$ for all $z \in \{0,1\}^n$.

2.1 Symmetries of the parameter space

We aim to explore the probability of reaching an arbitrary state $|z\rangle$

$$p_z(\vec{\gamma},\vec{\beta}) = \left|\left\langle z\middle|\vec{\gamma},\vec{\beta}\right\rangle\right|^2.$$

For instance, we would like to find the angles that maximize $p_z(\vec{\gamma},\vec{\beta})$. As we are not guaranteed for this probability not to have many local maxima, a relatively safe choice is using grid search with a reasonable step size. This choice is however quite time-consuming. Hence, it would be helpful to restrict the area $(\gamma \times \beta)$ we need to investigate even more than the natural $[0, 2\pi] \times [0, 2\pi]$ region. In 1 round of QAOA, by observing the shape of $p(\gamma, \beta)$, we noticed the following useful properties, which enable us to do this.

- For *each type of graph*[2], there is a $\pi \times \frac{\pi}{2}$ unit cell (see Figure 2.2(a)). Translating it by $k \cdot \pi$ in the γ-direction and by $l \cdot \frac{\pi}{2}$ in the β-direction, we gain the whole cover of $p\,(\gamma, \beta)$. In other words, the problem has translational symmetry

$$p\,(\gamma,\beta) = p\left(\gamma + k\pi, \beta + l\frac{\pi}{2}\right), \qquad k, l \in \mathbb{Z}.$$

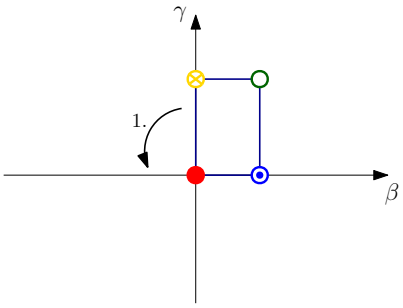
- The cell is symmetric about its center. As depicted in Figure 2.1, this is a consequence of the translational symmetry and the symmetry about the origin $[\beta = 0, \gamma = 0]$.

- All rings and the n-odd complete graphs are periodic in the γ-direction with period $\frac{\pi}{2}$ (half the general period). Their unit cell size is, therefore, $\frac{\pi}{2} \times \frac{\pi}{2}$ (see Figure 2.2(b)) and the overall translational symmetry dictates that

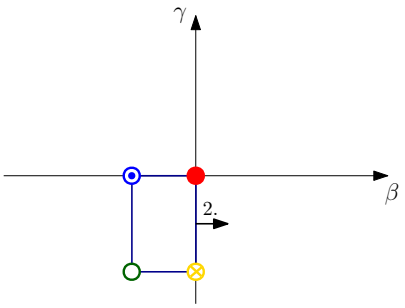
$$p\,(\gamma,\beta) = p\left(\gamma + k\frac{\pi}{2}, \beta + l\frac{\pi}{2}\right), \qquad k, l \in \mathbb{Z}.$$

Now, let us reorder and prove these properties for 1 round of QAOA and discuss their generalization to more rounds.

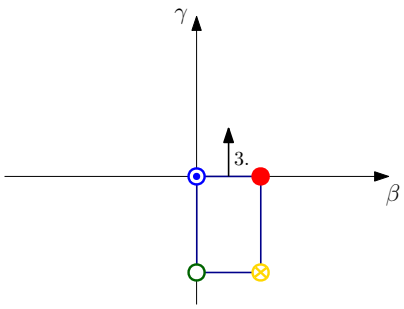
[2]Each type of graph. Not just the rings, lines, and complete graphs we observed.



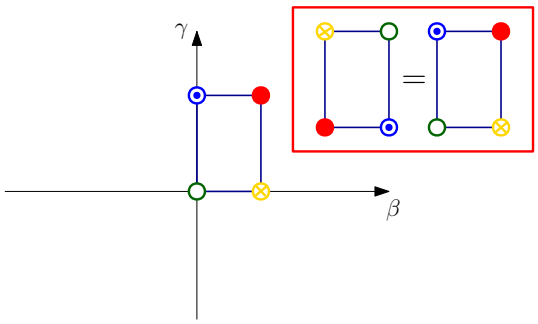
(a) Step 1: Use the symmetry about the origin.



(b) Step 2: Use periodicity in β.

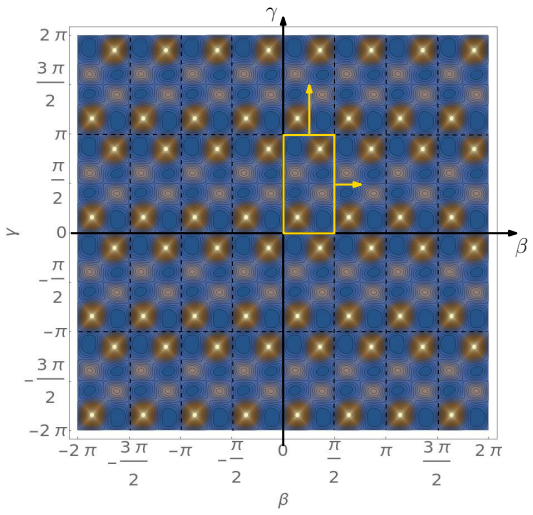


(c) Step 3: Use periodicity in γ.

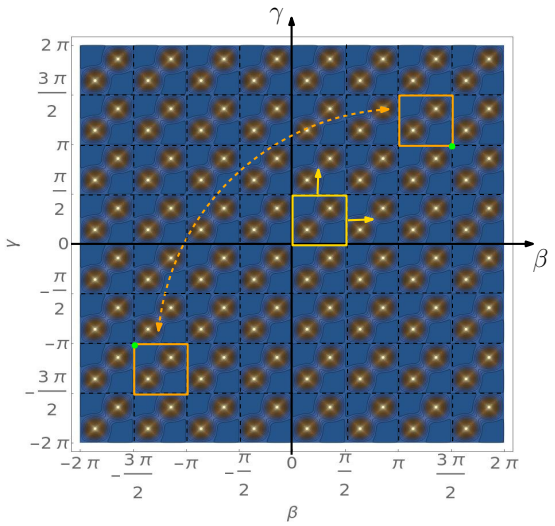


(d) The original unit cell at this place must be equal to the new one. The cell is therefore symmetric about its centre.

Figure 2.1: The centre symmetry of the cell is a consequence of the symmetry about the origin and the transational symmetry.



(a) The probability of reaching the state $|00\ldots0\rangle$. A contour plot for 6-vertex line.



(b) The probability of reaching the state $|00\ldots0\rangle$. A contour plot for a 9-vertex ring.

Figure 2.2: Illustration of the $p\,(-\gamma, \beta)$:[3](a) The symmetry about the origin as well as the periodicity in β with period $\frac{\pi}{2}$ and in γ with period π. (b) For rings and the n-odd complete graphs, the period in γ is halved to $\frac{\pi}{2}$.

Property 2.1: *Periodicity in β, with period $\frac{\pi}{2}$:*

$$p(\gamma,\beta) = p\left(\gamma,\beta+k\frac{\pi}{2}\right); \quad \forall k\in\mathbb{Z}.$$

Proof. First, observe these three properties:

1. $X = \pm ie^{\mp i\frac{\pi}{2}X}$, as $e^{\mp i\frac{\pi}{2}X} = \cos\left(\frac{\pi}{2}\right)\cdot\mathbb{I} \mp i\sin\left(\frac{\pi}{2}\right)\cdot X = \mp iX$,
2. $|s\rangle = X_1X_2\ldots X_n|s\rangle$ – simply because $|s\rangle = |+\rangle^{\otimes n}$ and $X|+\rangle = |+\rangle$,
3. $\forall i,j\in\{1,2,\ldots,n\}:\ [e^{-i\gamma Z_iZ_j}, X_1X_2\ldots X_n] = 0$ – each term of the expansion of $e^{-i\gamma Z_iZ_j}$ contains an even number of Z's and we can commute/anticommute X's in $Z_iZ_jZ_iZ_j\ldots Z_iZ_jX_1X_2\ldots X_n$ into the front without sign change.

They enable us to rewrite the amplitude $\langle z|\prod_j e^{i\beta X_j}\prod_{\langle kl\rangle}e^{-i\gamma Z_kZ_l}|s\rangle$ as

$$\begin{aligned}
\langle z|\prod_j e^{i\beta X_j}\prod_{\langle kl\rangle}e^{-i\gamma Z_kZ_l}|s\rangle &= \langle z|\prod_j e^{i\beta X_j}\prod_{\langle kl\rangle}e^{-i\gamma Z_kZ_l}X_1X_2\ldots X_n|s\rangle\\
&= \langle z|\prod_j e^{i\beta X_j}X_1X_2\ldots X_n\prod_{\langle kl\rangle}e^{-i\gamma Z_kZ_l}|s\rangle\\
&= (\pm i)^n\langle z|\prod_j e^{i\beta X_j}\prod_m e^{\mp i\frac{\pi}{2}X_m}\prod_{\langle kl\rangle}e^{-i\gamma Z_kZ_l}|s\rangle\\
&= (\pm i)^n\langle z|\prod_j e^{i\left(\beta\pm\frac{\pi}{2}\right)X_j}\prod_{\langle kl\rangle}e^{-i\gamma Z_kZ_l}|s\rangle.
\end{aligned}$$

Omitting a global phase, the amplitudes are unchanged when we shift β by $\pm\frac{\pi}{2}$. We can repeat this trick as many times as we like. After m repetitions, we obtain

$$\langle z|\prod_j e^{i\beta X_j}\prod_{\langle kl\rangle}e^{-i\gamma Z_kZ_l}|s\rangle = (\pm i)^{mn}\langle z|\prod_j e^{i\left(\beta\pm m\frac{\pi}{2}\right)X_j}\prod_{\langle kl\rangle}e^{-i\gamma Z_kZ_l}|s\rangle.$$

Finally, we have the proof of Property 2.1 for 1 round of QAOA:

$$\begin{aligned}
p(\gamma,\beta) &= \left|\langle z|\prod_j e^{i\beta X_j}\prod_{\langle kl\rangle}e^{-i\gamma Z_kZ_l}|s\rangle\right|^2 = \left|(\pm i)^{mn}\langle z|\prod_j e^{i\left(\beta\pm m\frac{\pi}{2}\right)X_j}\prod_{\langle kl\rangle}e^{-i\gamma Z_kZ_l}|s\rangle\right|^2 =\\
&= \left|\langle z|\prod_j e^{i\left(\beta\pm m\frac{\pi}{2}\right)X_j}\prod_{\langle kl\rangle}e^{-i\gamma Z_kZ_l}|s\rangle\right|^2 = p\left(\gamma,\beta+k\frac{\pi}{2}\right); \quad \forall k\in\mathbb{Z}.
\end{aligned}$$

What will happen, if we consider more than 1 round of QAOA? As the $X_1X_2\ldots X_n$ operator commutes with all the inner operators (namely $e^{i\beta_iX_j}$ and $e^{-i\gamma_iZ_kZ_l}$ for all i,

[3] Originally, throughout our research we used $B = \sum_j X_j$ instead of the currently used $B = -\sum_j X_j$. Hence, the contour plots depict $p(-\gamma,\beta) = \left|\langle z|\prod_j e^{-i\beta X_j}\prod_{\langle kl\rangle}e^{-i\gamma_pZ_kZ_l}|s\rangle\right|^2$, a mirrored image to the expected $p(\gamma,\beta) = \left|\langle z|\prod_j e^{i\beta X_j}\prod_{\langle kl\rangle}e^{-i\gamma_pZ_kZ_l}|s\rangle\right|^2$. However, for our purposes, the mirrored contour plots have exactly the same informative value.

j, k, l), we can move this operator to any desired location. Therefore, we can prove this property for any round, and thus this property holds for all the parameters β_i independently!

□

Property 2.2: *Periodicity in γ, with period:*

(a) *π in general.*

(b) *$\frac{\pi}{2}$ for all rings and complete graphs with n odd.*

(a) *Proof.* Observe that

$$-\mathbb{I} = \cos\pi \cdot \mathbb{I} \pm i \sin\pi \cdot ZZ = e^{\pm i\pi ZZ}.$$

As we are free to insert an identity $\mathbb{I} = -e^{\pm i\pi ZZ}$ anywhere, we can shift any γ by $\pm\pi$ without changing the probability. So once again, we created a simple trick:

$$\begin{aligned}\langle z| \prod_j e^{i\beta X_j} \prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l} |s\rangle &= \langle z| \prod_j e^{i\beta X_j} \cdot \prod_{\langle pq\rangle} \mathbb{I} \cdot \prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l} |s\rangle = \\ &= (-1)^{\#\text{ edges}} \langle z| \prod_j e^{i\beta X_j} \prod_{\langle pq\rangle} e^{\pm i\pi Z_p Z_q} \prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l} |s\rangle = \\ &= (-1)^{\#\text{ edges}} \langle z| \prod_j e^{i\beta X_j} \prod_{\langle kl\rangle} e^{-i(\gamma\mp\pi) Z_k Z_l} |s\rangle .\end{aligned}$$

This trick shifts the parameter γ by $\pm\pi$ and leaves the amplitude unchanged up to a global phase. We can repeat the trick as many times as we like, also for QAOA with multiple rounds. Thus, Property 2.2(a) is satisfied for each γ independently. □

(b) *Proof.* Consider the following facts:

1. Z is a projector ($Z^2 = \mathbb{I}$). This enables us to create many useful identities such as
   - $\mathbb{I} = Z_1 Z_1 Z_2 Z_2 \dots Z_n Z_n$
   - $\mathbb{I} = Z_i^k$ for k even.
2. $[Z_i, Z_j] = 0$ as well as $[e^{-i\gamma Z_i Z_j}, Z_k] = 0$ for all i, j, k,
3. $Z_i Z_j = \pm i e^{\mp i \frac{\pi}{2} Z_i Z_j}$.

Let us now use these when investigating QAOA on rings and complete graphs.

Rings. The explicit formula for $\prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l}$ is

$$\prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l} = e^{-i\gamma Z_1 Z_2} e^{-i\gamma Z_2 Z_3} \dots e^{-i\gamma Z_{n-1} Z_n} e^{-i\gamma Z_n Z_1}.$$

We can combine it with the identity $\mathbb{I} = Z_1 Z_1 Z_2 Z_2 \dots Z_n Z_n$ and obtain

$$\prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l} = \mathbb{I} \prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l} =$$
$$= \Big(Z_1(Z_1Z_2)(Z_2Z_3)\dots(Z_{n-1}Z_n)Z_n\Big) e^{-i\gamma Z_1 Z_2} e^{-i\gamma Z_2 Z_3} \dots e^{-i\gamma Z_{n-1} Z_n} e^{-i\gamma Z_n Z_1} =$$
$$= (Z_1Z_2) e^{-i\gamma Z_1 Z_2} (Z_2Z_3) e^{-i\gamma Z_2 Z_3} \dots (Z_{n-1}Z_n) e^{-i\gamma Z_{n-1} Z_n} (Z_nZ_1) e^{-i\gamma Z_n Z_1} =$$
$$= (\pm i)^n e^{-i(\gamma \pm \frac{\pi}{2}) Z_1 Z_2} e^{-i(\gamma \pm \frac{\pi}{2}) Z_2 Z_3} \dots e^{-i(\gamma \pm \frac{\pi}{2}) Z_{n-1} Z_n} e^{-i(\gamma \pm \frac{\pi}{2}) Z_n Z_1} =$$
$$= (\pm i)^n \prod_{\langle kl\rangle} e^{-i(\gamma \pm \frac{\pi}{2}) Z_k Z_l}.$$

This means a shift of γ by $\frac{\pi}{2}$ only changes the global phase. Of course, we could insert there the identity multiple times, w.l.o.g. $|m|$ times ($m \in \mathbb{Z}$). In that case,

$$\prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l} = \mathbb{I}^{|m|} \prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l} =$$
$$= \mathbb{I}^{|m|-1} (\pm i)^n \prod_{\langle kl\rangle} e^{-i(\gamma \pm \frac{\pi}{2}) Z_k Z_l} = (\pm i)^{|m|\cdot n} \prod_{\langle kl\rangle} e^{-i(\gamma + m\frac{\pi}{2}) Z_k Z_l}.$$

Now, we can prove the $\frac{\pi}{2}$-periodicity in γ_q for any round q of a multiround QAOA:

$$p(\vec{\gamma}, \vec{\beta}) =$$
$$= \left| \langle z | \prod_j e^{i\beta_p X_j} \dots \prod_j e^{i\beta_q X_j} \prod_{\langle kl\rangle} e^{-i\gamma_q Z_k Z_l} \dots \prod_{\langle kl\rangle} e^{-i\gamma_1 Z_k Z_l} |s\rangle \right|^2 =$$
$$= \left| \langle z | \prod_j e^{i\beta_p X_j} \dots \prod_j e^{i\beta_q X_j} \left((\pm i)^{|m|\cdot n} \prod_{\langle kl\rangle} e^{-i(\gamma_q + m\frac{\pi}{2}) Z_k Z_l} \right) \dots \prod_{\langle kl\rangle} e^{-i\gamma_1 Z_k Z_l} |s\rangle \right|^2 =$$
$$= \left| (\pm i)^{|m|\cdot n} \right|^2 \cdot \left| \langle z | \prod_j e^{i\beta_p X_j} \dots \prod_j e^{i\beta_q X_j} \prod_{\langle kl\rangle} e^{-i(\gamma_q + m\frac{\pi}{2}) Z_k Z_l} \dots \prod_{\langle kl\rangle} e^{-i\gamma_1 Z_k Z_l} |s\rangle \right|^2 =$$
$$= p\Big(\begin{pmatrix} \gamma_1 & \gamma_2 & \dots & \gamma_q + m\frac{\pi}{2} & \dots & \gamma_p \end{pmatrix}, \vec{\beta} \Big); \quad \forall m \in \mathbb{Z}.$$

Complete graphs. The explicit formula for $\prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l}$ is

$$\prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l} = e^{-i\gamma Z_1 Z_2} \dots e^{-i\gamma Z_1 Z_n} e^{-i\gamma Z_2 Z_3} \dots e^{-i\gamma Z_{n-1} Z_n}.$$

As each of the n vertices of any complete graph has $n-1$ neighbours, the formula has exactly $n-1$ operators ($e^{-i\gamma ZZ}$) which contain Z_i (let us emphasize that this holds for each vertex i). So the identity[4] we need to insert into the formula should be $Z_1^{n-1} Z_2^{n-1} \dots Z_n^{n-1}$ with $n-1$ even (for $n-1$ odd the term is not the identity).

[4]The only operator we are free to insert into the formula is the identity.

Why? This is the reason:

$$\prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l} = \mathbb{I}\prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l} =$$
$$= \left(Z_1^{n-1} Z_2^{n-1} \dots Z_n^{n-1}\right) e^{-i\gamma Z_1 Z_2} \dots e^{-i\gamma Z_1 Z_n} e^{-i\gamma Z_2 Z_3} \dots e^{-i\gamma Z_{n-1} Z_n} =$$
$$= \left((Z_1 Z_2)\dots(Z_1 Z_n)(Z_2 Z_3)\dots(Z_{n-1} Z_n)\right) e^{-i\gamma Z_1 Z_2} \dots e^{-i\gamma Z_{n-1} Z_n} =$$
$$= (Z_1 Z_2) e^{-i\gamma Z_1 Z_2} \dots (Z_1 Z_n) e^{-i\gamma Z_1 Z_n} (Z_2 Z_3) e^{-i\gamma Z_2 Z_3} \dots (Z_{n-1} Z_n) e^{-i\gamma Z_{n-1} Z_n} =$$
$$= (\pm i)^{\frac{n(n-1)}{2}} e^{-i(\gamma\pm\frac{\pi}{2}) Z_1 Z_2} \dots e^{-i(\gamma\pm\frac{\pi}{2}) Z_1 Z_n} e^{-i(\gamma\pm\frac{\pi}{2}) Z_2 Z_3} \dots e^{-i(\gamma\pm\frac{\pi}{2}) Z_{n-1} Z_n} =$$
$$= (\pm i)^{\frac{n(n-1)}{2}} \prod_{\langle kl\rangle} e^{-i(\gamma\pm\frac{\pi}{2}) Z_k Z_l}.$$

More repetitions ($|m|$, $m \in \mathbb{Z}$) would lead us to

$$\prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l} = (\pm i)^{|m|\frac{n(n-1)}{2}} \prod_{\langle kl\rangle} e^{-i(\gamma+m\frac{\pi}{2}) Z_k Z_l}.$$

However, as $Z_1^{n-1} Z_2^{n-1} \dots Z_n^{n-1} = \mathbb{I}$ only if n is odd, the periodicity in γ is satisfied only for n-odd complete graphs. Moreover, for odd n, proving the periodicity for any round of QAOA is analogous to the above proof for rings. □

Property 2.3: *Symmetry about the origin* $[\beta = 0, \gamma = 0]$*:*

$$p(\gamma, \beta) = p(-\gamma, -\beta).$$

Proof. This time, we are going to use a trick that transposing has no effect on numbers (1x1 matrices): a number = a number$^{\mathrm{T}}$. With this in mind, we can write down the whole proof (for 1 round of QAOA) directly:

$$p(\gamma,\beta) = \left|\langle z| \prod_j e^{i\beta X_j} \prod_{\langle kl\rangle} e^{i\gamma Z_k Z_l} |s\rangle\right|^2 =$$
$$= \overbrace{\langle z| \prod_j e^{i\beta X_j} \prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l} |s\rangle}^{\textbf{a (complex) number } c} \underbrace{\langle s| \left(\prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l}\right)^\dagger \left(\prod_j e^{i\beta X_j}\right)^\dagger |z\rangle}_{\textbf{its conjugate } c^*} =$$
$$= \overbrace{cc^*}^{\textbf{number}} = \underbrace{(cc^*)^{\mathrm{T}}}_{\textbf{number}^{\mathrm{T}}} =$$
$$= \overbrace{\langle z| \left(\prod_j e^{i\beta X_j}\right)^\star \left(\prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l}\right)^\star |s\rangle}^{(c^*)^{\mathrm{T}}} \underbrace{\langle s| \left(\prod_{\langle kl\rangle} e^{-i\gamma Z_k Z_l}\right)^{\mathrm{T}} \left(\prod_j e^{i\beta X_j}\right)^{\mathrm{T}} |z\rangle}_{(c)^{\mathrm{T}}} =$$

$$= \langle z| \prod_j e^{-i\beta X_j} \prod_{\langle kl \rangle} e^{i\gamma Z_k Z_l} |s\rangle \langle s| \left(\prod_{\langle kl \rangle} e^{i\gamma Z_k Z_l} \right)^\dagger \left(\prod_j e^{-i\beta X_j} \right)^\dagger |z\rangle =$$

$$= \left| \langle z| \prod_j e^{-i\beta X_j} \prod_{\langle kl \rangle} e^{i\gamma Z_k Z_l} |s\rangle \right|^2 = p(-\gamma, -\beta).$$

Note that this would work for more rounds, too. Only this time not for each round independently, but as an overall time-reversal symmetry: $p(\vec{\gamma}, \vec{\beta}) = p(-\vec{\gamma}, -\vec{\beta})$. □

What speedup do these symmetries (hence some area restrictions) bring us? Let us start with the $[0, \pi] \times \left[0, \frac{\pi}{2}\right]$ area, which can be used for any graph. In comparison to the original $[0, 2\pi] \times [0, 2\pi]$ area, one needs to search only $\frac{1}{8}$ of the original space (see Fig. 2.2(a)). The cell symmetry about its center further restricts this area to half (to a triangle). So, we need to walk only through $\frac{1}{16}$ of the original space, which takes $\frac{1}{16}$ of the original time. However, this was only for $p = 1$. How does this speedup scale with the number of rounds?

Restricting each γ independently always halves the previous time. Restricting any β reduces the time to a quarter. Hence, if we restrict all γs and βs, we get $(1/8)^p$ of the original space. On the other hand, the time-reversal symmetry does not hold for each parameter independently. Therefore, this symmetry enables us only to halve the $(1/8)^p$ once. To conclude, we managed to shorten the time to $\dfrac{1}{2 \cdot 8^p}$ of the original one. What about the further restricted area to $\left[0, \frac{\pi}{2}\right]^p \times \left[0, \frac{\pi}{2}\right]^p$ for rings and n-odd complete graphs? In this case, analogically, we achieved a reduction to $\dfrac{1}{2 \cdot 4^{2p}}$ of the original time.

2.2 GHZ state preparation

While studying QAOA_1 for MaxCut on a 2-vertex line (in Subsection 1.3.1), we have observed that we can efficiently use QAOA_1 for Bell state preparation. Now, we are going to demonstrate that efficient preparation of GHZ (Greenberger-Horne-Zeilinger) states, nontrivial entangled quantum states of $n > 2$ d-dimensional subsystems of the form

$$|\mathrm{GHZ}\rangle = \frac{1}{\sqrt{d}} \sum_{i=0}^{d-1} \underbrace{|i\rangle \otimes \cdots \otimes |i\rangle}_{n} = \frac{1}{\sqrt{d}} \left(|0\rangle^{\otimes n} + |1\rangle^{\otimes n} + \cdots + |d-1\rangle^{\otimes n} \right), \tag{2.1}$$

for the case where each subsystem is a qubit ($d = 2$) and thus

$$|\mathrm{GHZ}_n\rangle = \frac{|0\rangle^{\otimes n} + |1\rangle^{\otimes n}}{\sqrt{2}}, \tag{2.2}$$

is in the QAOA setting also possible.

More precisely, what we are going to show here is the following. When using an n-vertex complete graph (all to all interactions), then for QAOA_1 on MaxCut

$$p_{|GHZ_n\rangle}(\gamma, \beta = \frac{\pi}{4}) = \begin{cases} 1 & \text{for } n \text{ odd} \quad \& \quad \gamma = \frac{\pi}{4} \\ \frac{1}{2} & \text{for } n \text{ even} \quad \& \quad \gamma = \frac{\pi}{8} \end{cases}.$$

For $p = 2$, we have numerically observed (https://github.com/dendaxD/QAOA-MaxCut-amplitudes/tree/master/complete-graphs/2rounds/even-n-ghz-state-preparation) that for even n up to 18, $\max_{\beta_1,\gamma_1,\beta_2,\gamma_2}\left(p_{|GHZ_n\rangle}\right)$ is already 1, too. Assuming it holds for greater n as well, it led us to believe that for $p = 2$, for all n:

$$\max_{\beta_1,\gamma_1,\beta_2,\gamma_2}\left(p_{|GHZ_n\rangle}(\gamma_1\gamma_2, \beta_1\beta_2)\right) = 1.$$

After the work which we present here was finished, we have noticed that this has already been shown in [24] and [23], too. Concerning $p = 2$ case, [24] extended our knowledge regarding the parameters as well as with the proof that

$$p_{|GHZ_n\rangle}\left(\beta_1 = \frac{3\pi}{4n}, \gamma_1 = \frac{\pi}{4}, \beta_2 = \frac{\pi}{4}, \gamma_2 = \frac{\pi}{8}\right) = 1.$$

Hence, we present here only our orignal proofs for the observations regarding $p = 1$. It is also different from what Ho et al. presented.

1 round

Let us start with ***n*-odd complete graphs**. It is enough to show that

$$a := \left\langle GHZ \middle| \gamma = \frac{\pi}{4}, \beta = \frac{\pi}{4} \right\rangle = \frac{\langle 00\cdots0| + \langle 11\cdots1|}{\sqrt{2}} \prod_j e^{i\frac{\pi}{4}X_j} \prod_{\langle kl\rangle} e^{-i\frac{\pi}{4}Z_kZ_l} \underbrace{|s\rangle}_{=\frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}|z\rangle} = e^{-i\varphi},$$

where $\varphi \in \mathbb{R}$, as $p_{|GHZ_n\rangle}(\frac{\pi}{4}, \frac{\pi}{4}) = aa^*$. To show 2.2 holds, let us draw a vertical line between the operators $U(B, \beta)$ and $U(C, \gamma)$ and apply each operator to the corresponding vector on that side:

$$a = \frac{1}{\sqrt{2^n}} \sum_{z\in\{0,1\}^n} \underbrace{\frac{\langle 00\cdots0| + \langle 11\cdots1|}{\sqrt{2}} \overbrace{\prod_j e^{i\frac{\pi}{4}X_j}}^{\prod_j\left(\cos\frac{\pi}{4}\mathbb{I} + i\sin\frac{\pi}{4}X_j\right)}}_{=\frac{1}{\sqrt{2^n}}\sum_{m\in\{0,1\}^n}\langle m|\left(\frac{i^{\#_1(m)}+i^{\#_0(m)}}{\sqrt{2}}\right)} \Bigg| \underbrace{\prod_{\langle kl\rangle} e^{-i\frac{\pi}{4}Z_kZ_l}|s\rangle}_{=e^{-i\frac{\pi}{4}(\#\text{edges}-2\cdot CUT(z))}|z\rangle}$$

$$= \frac{1}{2^n} \sum_{m\in\{0,1\}^n} \sum_{z\in\{0,1\}^n} \left(\frac{i^{\#_1(m)}+i^{\#_0(m)}}{\sqrt{2}}\right) e^{-i\frac{\pi}{4}\left(\frac{n(n-1)}{2}-2\cdot\#_0(z)\cdot\#_1(z)\right)} \underbrace{\langle m|z\rangle}_{\delta_{mz}}$$

$$= \frac{1}{2^n} e^{-i\frac{\pi}{8}\cdot n(n-1)} \sum_{z\in\{0,1\}^n} \underbrace{\left(\frac{i^{\#_1(z)}+i^{\#_0(z)}}{\sqrt{2}}\right) e^{i\frac{\pi}{2}\cdot\#_0(z)\cdot\#_1(z)}}_{=:\, e^{-if(z)}}.$$ [5]

Now, all we need to show is that all the 2^n terms $e^{-if(z)}$ point in the same direction, as this would imply the existence of such $\alpha \in [0, 2\pi]$ that $\forall z : e^{-if(z)} = e^{-i\alpha}$ and with such α, we will achieve what we want:

$$a = \frac{1}{2^n} e^{-i\frac{\pi}{8}\cdot n(n-1)} \sum_{z\in\{0,1\}^n} e^{-i\alpha} = e^{-i\left(\frac{\pi}{8}\cdot n(n-1)+\alpha\right)} = e^{-i\varphi}.$$

To show it, let us first rewrite the terms:

$$\begin{aligned}
e^{-if(z)} &= \left(\frac{i^{\#_1(z)} + i^{\#_0(z)}}{\sqrt{2}}\right) e^{i\frac{\pi}{2}\cdot\#_0(z)\cdot\#_1(z)} = \left(\frac{i^{\#_0(z)} + i^{\#_1(z)}}{\sqrt{2}}\right) i^{\#_0(z)\#_1(z)} \\
&= \left(\frac{1 + i^{\#_1(z)-\#_0(z)}}{\sqrt{2}}\right) i^{\#_0(z)} i^{\#_0(z)\#_1(z))} \\
&\overset{\#_0(z)=n-\#_1(z)}{=} \left(\frac{1 + i^{-n}\cdot i^{2\cdot\#_1(z)}}{\sqrt{2}}\right) i^n \cdot i^{-\#_1(z)} \cdot i^{n\cdot\#_1(z)} \cdot i^{-(\#_1(z))^2} \\
&\overset{\substack{n=2k+1\\ k\in\mathbb{N}}}{=} i^n \cdot \left(\frac{1 + (-i)\cdot(-i)^{2k}\cdot(-1)^{\cdot\#_1(z)}}{\sqrt{2}}\right) \cdot i^{-\#_1(z)} \cdot i^{\#_1(z)} \cdot i^{2k\cdot\#_1(z)} \cdot (-i)^{(\#_1(z))^2} \\
&= i^n \cdot \left(\frac{1 + (-i)\cdot(-1)^k\cdot(-1)^{\#_1(z)}}{\sqrt{2}}\right) \cdot (-1)^{k\cdot\#_1(z)} \cdot (-i)^{(\#_1(z))^2}.
\end{aligned}$$

Now, the only terms that depend on z are: $(-1)^{\#_1(z)}$, $(-1)^{k\cdot\#_1(z)}$, and $(-i)^{(\#_1(z))^2}$. Observe that depending on the value of $\#_1(z)$, each of these terms has only two possible values:[6,7]

	$(-1)^{\#_1(z)}$	$(-1)^{k\cdot\#_1(z)}$	$(-i)^{(\#_1(z))^2}$
$\#_1(z)$ even	1	1	1
$\#_1(z)$ odd	-1	$(-1)^k$	$-i$

Hence, the term $e^{-if(z)}$ has only 2 possible values, too. Now, we need to check if they are equal:

$$\begin{aligned}
&i^n \cdot \left(\frac{1 + (-i)\cdot(-1)^k\cdot(-1)^{\#_1(z)}}{\sqrt{2}}\right) \cdot (-1)^{k\cdot\#_1(z)} \cdot (-i)^{(\#_1(z))^2} = \\
&\qquad \overset{\#_1(z)\text{ even:}}{=} i^n \cdot \left(\frac{1 + (-i)\cdot(-1)^k}{\sqrt{2}}\right) \\
&\qquad \overset{\#_1(z)\text{ odd:}}{=} i^n \cdot \left(\frac{1 + i\cdot(-1)^k}{\sqrt{2}}\right) \cdot (-1)^k \cdot (-i) = i^n \cdot \left(\frac{1 + (-i)\cdot(-1)^k}{\sqrt{2}}\right).
\end{aligned}$$

[5] As $n = \#_0(z) + \#_1(z)$ is odd, either $\#_0(z)$ is even and $\#_1(z)$ is odd or vice versa. Hence, the term $\frac{i^{\#_1(z)}+i^{\#_0(z)}}{\sqrt{2}}$ has only four possible values: $\frac{\pm1\pm i}{\sqrt{2}}$. All of these possibilities have magnitude one. They can be expressed as $e^{i\alpha}$, where $\alpha = \frac{\pi}{4} + k\frac{\pi}{2}$, $k \in \mathbb{Z}$.

[6] k is given by n. Therefore, we do not touch it.

[7] Even the term $(-i)^{(\#_1(z))^2}$ has only two possible values. Of course, $(-i)^m$ has four possible values, depending on $m \mod 4$. However, $m^2 \mod 4$ can be only 0 or 1. That is because for any even number $2l$ ($l \in \mathbb{Z}$), it follows that $(2l)^2 \mod 4 = 4l^2 \mod 4 = 0$, and if we take any odd number $2l+1$, then $(2l+1)^2 \mod 4 = 4l^2 + 4l + 1 \mod 4 = 1$ (independently of l).

And equal they are.

Let us proceed to ***n*-even complete graphs**. Here,

$$\left\langle GHZ\middle|\tfrac{\pi}{8},\tfrac{\pi}{4}\right\rangle = \frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}\frac{\langle 00\cdots0|+\langle 11\cdots1|}{\sqrt{2}}\prod_j e^{i\frac{\pi}{4}X_j}\Bigg|\prod_{\langle kl\rangle}e^{-i\frac{\pi}{8}Z_kZ_l}|s\rangle$$

$$= \frac{1}{2^n}\sum_{z\in\{0,1\}^n}\left(\frac{i^{\#_1(z)}+i^{\#_0(z)}}{\sqrt{2}}\right)e^{-i\frac{\pi}{8}\left(\frac{n(n-1)}{2}-2\cdot\#_0(z)\cdot\#_1(z)\right)}$$

$$= \frac{1}{2^n}e^{-i\frac{\pi}{8}\cdot\frac{n(n-1)}{2}}\sum_{z\in\{0,1\}^n} i^n\cdot\underbrace{i^{-\#_1(z)}\left(\frac{1+i^{-n}\cdot i^{2\#_1(z)}}{\sqrt{2}}\right)e^{i\frac{\pi}{4}\cdot n\cdot\#_1(z)}e^{-i\frac{\pi}{4}(\#_1(z))^2}}_{=:\,f(z)}.$$

Now, we may take a closer look at $f(z)$:

$$f(z)\overset{\substack{n=2k\\k\in\mathbb{Z}}}{=} i^{-\#_1(z)}\cdot e^{i\frac{\pi}{2}\cdot k\cdot\#_1(z)}\cdot e^{-i\frac{\pi}{4}(\#_1(z))^2}\left(\frac{1+(-1)^k\cdot(-1)^{\#_1(z)}}{\sqrt{2}}\right)=$$

$$= (-i)^{\#_1(z)}\cdot i^{k\cdot\#_1(z)}\cdot e^{-i\frac{\pi}{4}(\#_1(z))^2}\cdot\left(\frac{1+(-1)^k\cdot(-1)^{\#_1(z)}}{\sqrt{2}}\right).$$

This time, the z-dependent terms are: $(-i)^{\#_1(z)}$, $i^{k\cdot\#_1(z)}$, $e^{-i\frac{\pi}{4}(\#_1(z))^2}$, and $(-1)^{\#_1(z)}$. For the first two, we are interested in $\#_1(z) \mod 4$. $(-1)^{\#_1(z)}$ depends purely on the parity of $\#_1(z)$. Last, $e^{-i\frac{\pi}{4}(\#_1(z))^2}$, depends on $(\#_1(z))^2 \mod 8$. However, $(\#_1(z))^2 \mod 8$ can be only 0, 1, or 4. That is because any $x\in\mathbb{Z}$ can be writen either as $x=4l$, $x=4l+1$, $x=4l+2$, or $x=4l+3$, for some $l\in Z$, and independently of l:

- $(4l)^2 \mod 8 = 8\cdot 2l^2 \mod 8 = 0$,
- $(4l+1)^2 \mod 8 = 8\cdot 2l^2+8l+1 \mod 8 = 1$,
- $(4l+2)^2 \mod 8 = 8\cdot 2l^2+8*2l+4 \mod 8 = 4$,
- $(4l+3)^2 \mod 8 = 8\cdot 2l^2+8*3l+9 \mod 8 = 1$.

Hence, these are all the possibilities we have:

	$(-i)^{\#_1(z)}$	$i^{k\cdot\#_1(z)}$	$e^{-i\frac{\pi}{4}(\#_1(z))^2}$	$(-1)^{\#_1(z)}$
$\#_1(z) \mod 4 = 0$	1	1	1	1
$\#_1(z) \mod 4 = 1$	$-i$	i^k	$(1-i)/\sqrt{2}$	-1
$\#_1(z) \mod 4 = 2$	-1	$(-1)^k$	-1	1
$\#_1(z) \mod 4 = 3$	i	$(-i)^k$	$(1-i)/\sqrt{2}$	-1

Correspondingly, all the possible values of $f(z)$ are:

$$f(z) = (-i)^{\#_1(z)} \cdot i^{k\cdot\#_1(z)} \cdot e^{-i\frac{\pi}{4}(\#_1(z))^2} \cdot \left(\frac{1+(-1)^k \cdot (-1)^{\#_1(z)}}{\sqrt{2}}\right) =$$
$$\overset{\#_1(z)=4l}{=} \frac{1+(-1)^k}{\sqrt{2}}$$
$$\overset{\#_1(z)=4l+1}{=} -i \cdot i^k \cdot \frac{1-i}{\sqrt{2}} \cdot \frac{1-(-1)^k}{\sqrt{2}}$$
$$\overset{\#_1(z)=4l+2}{=} \frac{1+(-1)^k}{\sqrt{2}}$$
$$\overset{\#_1(z)=4l+3}{=} i \cdot (-i)^k \cdot \frac{1-i}{\sqrt{2}} \cdot \frac{1-(-1)^k}{\sqrt{2}}.$$

A reasonable next step could be to observe that for k even, we have:

$$f(z) = \begin{cases} \sqrt{2} & \text{if } (\#_1(z) \mod 2) = 0 \\ 0 & \text{if } (\#_1(z) \mod 2) = 1 \end{cases}$$

and for k odd:

$$f(z) = \begin{cases} 0 & \text{if } (\#_1(z) \mod 2) = 0 \\ -i^k \cdot \frac{1+i}{\sqrt{2}} \cdot \sqrt{2} & \text{if } (\#_1(z) \mod 2) = 1 \end{cases}.$$

Hence, for even k, it follows that

$$\left\langle GHZ \middle| \frac{\pi}{8}, \frac{\pi}{4} \right\rangle = \underbrace{i^n \cdot e^{-i\frac{\pi}{8}\cdot\frac{n(n-1)}{2}}}_{=e^{-i\varphi_1}} \cdot \frac{\sqrt{2}}{2^n} \cdot \sum_{j=0}^{n/2} \binom{n}{2j},$$

where $\sum_{j=0}^{n/2} \binom{n}{2j}$ is the number of ways how to place an *even* number of 1s in n possible positions, and for odd k,

$$\left\langle GHZ \middle| \frac{\pi}{8}, \frac{\pi}{4} \right\rangle = \underbrace{-i^n \cdot e^{-i\frac{\pi}{8}\cdot\frac{n(n-1)}{2}} \cdot i^k \cdot \frac{1+i}{\sqrt{2}}}_{=e^{-i\varphi_2}} \cdot \frac{\sqrt{2}}{2^n} \cdot \sum_{j=0}^{(n-2)/2} \binom{n}{2j+1},$$

where $\sum_{j=0}^{(n-2)/2} \binom{n}{2j+1}$ is the number of ways how to place an *odd* number of 1s in n possible positions.

Recall that what we are trying to prove is:

$$p_{|GHZ\rangle}(\frac{\pi}{8}, \frac{\pi}{4}) = \left\langle GHZ \middle| \frac{\pi}{8}, \frac{\pi}{4} \right\rangle \left\langle \frac{\pi}{8}, \frac{\pi}{4} \middle| GHZ \right\rangle = \frac{1}{2}.$$

Observe that we only need to show that

$$\frac{\sqrt{2}}{2^n} \cdot \sum_{j=0}^{n/2} \binom{n}{2j} = \frac{\sqrt{2}}{2^n} \cdot \sum_{j=0}^{(n-2)/2} \binom{n}{2j+1} = \frac{1}{\sqrt{2}},$$

or equivalently:

$$\sum_{j=0}^{n/2}\binom{n}{2j} = \sum_{j=0}^{(n-2)/2}\binom{n}{2j+1} = \frac{2^n}{2} = 2^{n-1}. \tag{2.3}$$

Is it really true that having n positions, the number of ways we can pick an even number of positions from these, and the number of ways we can choose an odd number of positions are equal? Let us consider the case where we are choosing an even number of positions. In general, we may or may not pick the first position. But it must be true that the number of ways how to select an even number of positions from the n given including the first one and the number of ways how to choose an even number of positions excluding the first one must sum to all the possible ways how to select an even number of positions. If we include the first one, we still need to pick an odd number of positions from the remaining $n-1$ positions, and all of the ways to do this are valid. If we exclude the first one, in the remaining set of $n-1$ positions, there is still an even number of positions we need to pick, and all the possibilities are valid. Hence, the number of ways how to choose an even number of positions from n positions is equal to the sum of the number of ways how to select an even number of positions from $n-1$ positions and the number of ways how to choose an odd number of positions from $n-1$ positions. Notice that these are all the possibilities of how to select any number of elements from a set of size $n-1$. And there are 2^{n-1} of them. That is precisely the result we have wanted. Mathematically:

$$\sum_{j=0}^{n/2}\binom{n}{2j} = \sum_{j=0}^{(n-1)/2}\binom{n-1}{2j} + \sum_{j=0}^{(n-3)/2}\binom{n-1}{2j+1} = \sum_{j=0}^{n}\binom{n-1}{j} = 2^{n-1}.$$

By applying the same thought process for the case of selecting an odd number of positions, we obtain the same result. Hence, Equation 2.3 is satisfied and we have proved that $p_{|GHZ_n\rangle}\left(\frac{\pi}{8},\frac{\pi}{4}\right) \overset{\text{for } n \text{ even}}{=} \frac{1}{2}$.

But what does the fact that $p_{|GHZ_n\rangle}\left(\frac{\pi}{8},\frac{\pi}{4}\right) \overset{\text{for } n \text{ even}}{=} \frac{1}{2}$ mean? Note that it certainly does not mean that we have achieved the pure $|GHZ_n\rangle$ state with 50% probability. How we would like for you to interpret this result is the following way.

$$p_{|GHZ_n\rangle}(\gamma,\beta) = p_{00\cdots0}(\gamma,\beta) + p_{11\cdots1}(\gamma,\beta) = p_{\mathcal{O}(1)\text{-MinCut}}(\gamma,\beta).$$

That is true because:

$$p_{|GHZ_n\rangle}(\gamma,\beta) = \frac{\langle 00\cdots0| + \langle 11\cdots1|}{\sqrt{2}}|\gamma,\beta\rangle\langle\gamma,\beta|\frac{|00\cdots0\rangle + |11\cdots1\rangle}{\sqrt{2}} =$$

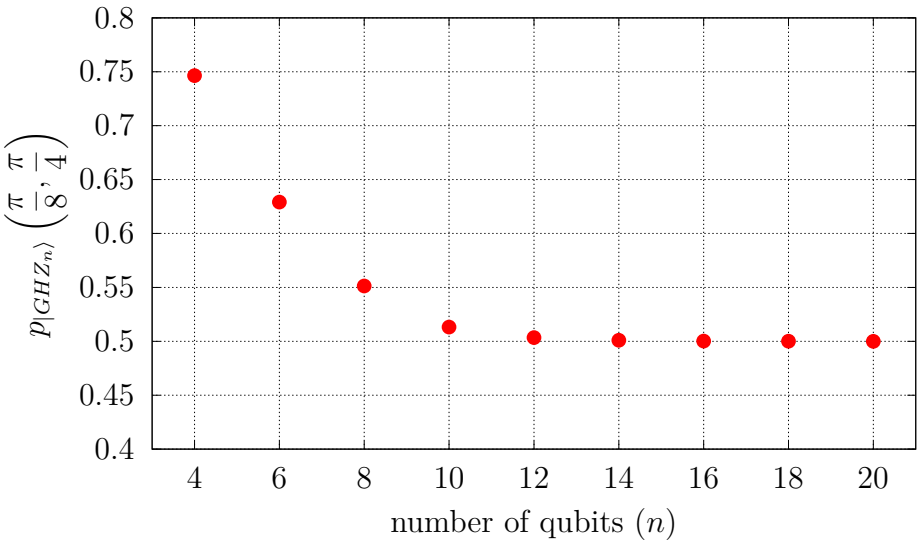
$$= \frac{1}{2}\Big(\langle 00\cdots0|\gamma,\beta\rangle\langle\gamma,\beta|00\cdots0\rangle + \overbrace{\langle 11\cdots1|\gamma,\beta\rangle}^{\text{NOT symmetry}=\langle 00\cdots0|\gamma,\beta\rangle}\langle\gamma,\beta|00\cdots0\rangle +$$

$$+ \underbrace{\langle 00\cdots0|\gamma,\beta\rangle}_{\text{NOT symmetry}=\langle 11\cdots1|\gamma,\beta\rangle}\langle\gamma,\beta|11\cdots1\rangle + \langle 11\cdots1|\gamma,\beta\rangle\langle\gamma,\beta|11\cdots1\rangle\Big) =$$

$$= p_{00\cdots0}(\gamma,\beta) + p_{11\cdots1}(\gamma,\beta).$$

As we can see $p_{|GHZ_n\rangle}(\gamma, \beta)$ is the probability of measuring $00\cdots0$ or $11\cdots1$ – the $\mathcal{O}(1)$-MinCut solutions. So, the result $p_{|GHZ_n\rangle}\left(\frac{\pi}{8}, \frac{\pi}{4}\right) \overset{\text{for } n \text{ even}}{=} \frac{1}{2}$ tells us that if we choose $\gamma = \frac{\pi}{8}$ and $\beta = \frac{\pi}{4}$ for QAOA$_1$ with this all to all geometry, then in the 0/1 measurement part, there is 25% probability of measuring $00\cdots0$ and 25% probability of the measurement outcome being $11\cdots1$.

To conclude, let us note that we have not proved that, for n-even complete graphs, the $\frac{1}{2}$ probability is the maximal probability of obtaining some of the $\mathcal{O}(1)$-MinCut solutions. This result only tells us that the maximal probability is not smaller – as this one is guaranteed. Using our numerical simulation, these are the maximal probabilities:

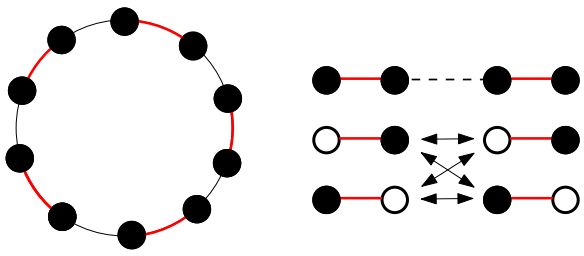


For n-odd complete graphs the maximal probability of reaching the $O(1)$-MinCut states is, of course, 1 (or $\frac{1}{2}$ for each of them independently).

2.3 QAOA$_1$ applied to rings

We are aware that QAOA is, in some sense, local. Can we find some graph where this locality prevents QAOA from finding the optimal solution?

Illustratively, let us take a look at rings. To start, let us consider a highly local version of QAOA, which we would have achieved if we considered for $U(C, \gamma)$ only every other edge:



It is possible to choose such parameters that all the considered edges would disagree.[8] Regardless of the parameters, the edges we skipped would disagree only with 50% probability, as all four combinations are equally likely. Overall, in this QAOA version, we have, in the average case, 3/4 of the edges satisfied and hence $r^* = 3/4$. However, a very simple classical algorithm achieves $r^\star = 1$.

Now, let us take a look at the probability of reaching the MaxCut solution:

- the state $|0101\ldots01\rangle$ or $|1010\ldots10\rangle$ for even n,
- or $|0101\ldots010\rangle$, $|1010\ldots101\rangle$, or their rotations for odd n.

In other words, we ask: *What is the probability that all edges will "point" in the right direction?* Let us start with even n. The probability of reaching $|0101\ldots01\rangle$, as well as $|1010\ldots10\rangle$, is, of course, $2^{-n/2}$. So together:

$$p_{\text{MaxCut}} \overset{\text{even } n}{=} 2 \cdot 2^{-\frac{n}{2}}.$$

For odd n, this is a little bit more complicated. We are guaranteed that all considered edges disagree. This property reduces the number of MaxCut solutions we can reach. Instead of $2n$, there are only $n+1$ reachable MaxCut solutions now, and as each of these solutions can be obtained with probability $2^{-(n+1)/2}$,

$$p_{\text{MaxCut}} \overset{\text{odd } n}{=} (n+1) \cdot 2^{-\frac{n+1}{2}}.$$

Thus, the probability of reaching the solution decreases exponentially with n. This means that QAOA needs exponential time (number of repetitions) to find the solution with a reasonable probability. This is not what we want – in order for QAOA to be useful for this problem, it must be able to solve it in polynomial time. Of course, the goal of QAOA is only to find a near-optimal solution, but contrast this with a classical algorithm, which can solve this problem in linear time.

Now, let us return to the unmodified QAOA$_1$, where $U(C,\gamma)$ contains all edges. To gain the algorithm's best possible approximation ratio,

$$r^* = \frac{\overline{C(z)}}{m},$$ [9]

we need to find the minimum of

$$F_p(\gamma,\beta) = \langle\gamma,\beta|\, C\,|\gamma,\beta\rangle\,,$$

[8] On the 2-vertex line examle, we have demonstrated that we can create the Bell state $|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$ with fidelity 1. And this is precisely a set of $n/2$ 2-vertex lines (for even n; for odd n, the state of the left-out qubit is $|+\rangle$).

as $F_p(\gamma,\beta)$ can be interpreted as the expectation value of $m-2C(z)$ (in general, the expectation of $m-C(z)$, but recall that we have rescalled the problem Hamiltonian: $C_{\langle kl\rangle}=\frac{1}{2}\left(Z_kZ_l+1\right)\rightarrow C_{\langle kl\rangle}=Z_kZ_l$). Observe that

$$\min_{\gamma,\beta}F_p(\gamma,\beta)=\min\left(\sum_z p_z\left(m-2C(z)\right)\right)=m-2\cdot\max\overline{C(z)}.$$

Hence,

$$\max\overline{C(z)}=\frac{1}{2}\left(m-\min_{\gamma,\beta}F_p(\gamma,\beta)\right).$$

Following the result from Section 1.3.1, where we have analyzed 3-regular graphs, we need to minimize only

$$f_{4\text{vline}}(\gamma,\beta)=\langle+|^{\otimes 4}e^{+i\gamma Z_1Z_2}e^{+i\gamma Z_2Z_3}e^{+i\gamma Z_3Z_4}e^{-i\beta X_1}e^{-i\beta X_2}e^{-i\beta X_3}e^{-i\beta X_4}\cdot \\ \cdot Z_2Z_3\cdot e^{i\beta X_4}e^{i\beta X_3}e^{i\beta X_2}e^{i\beta X_1}e^{-i\gamma Z_1Z_2}e^{-i\gamma Z_2Z_3}e^{-i\gamma Z_3Z_4}|+\rangle^{\otimes 4}.$$

That is because in rings, there is only one subgraph type – (for $n\geq 4$) a 4-vertex line. Thus,

$$F_p(\gamma,\beta)=\#_{\text{edges}}\cdot f_{4\text{vline}}(\gamma,\beta).$$

Using https://github.com/dendaxD/QAOA-MaxCut-amplitudes/blob/master/rings/RINGS:optimal-parameters-by-minimizing-F.nb we have found the minimum:

$$\min_{\gamma,\beta}f_{4\text{vline}}(\gamma,\beta)=f_{4\text{vline}}(\frac{\pi}{8},\frac{\pi}{8})=-0.5.$$

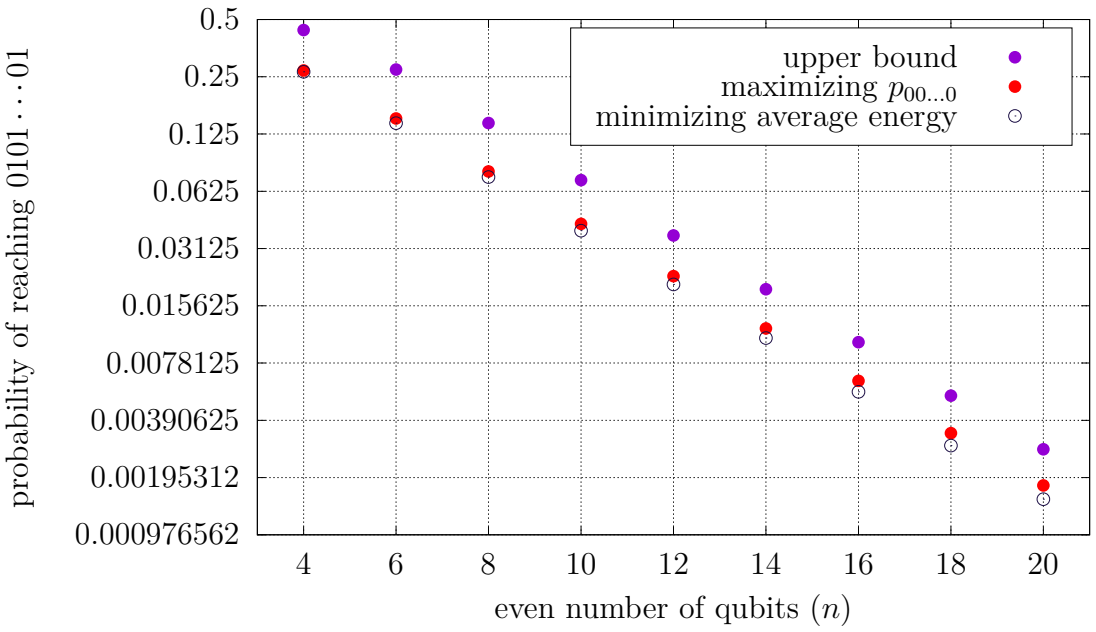
Hence, now the approximation ratio is

$$r^*=\frac{\overline{C(z)}}{m}=\frac{3}{4},$$

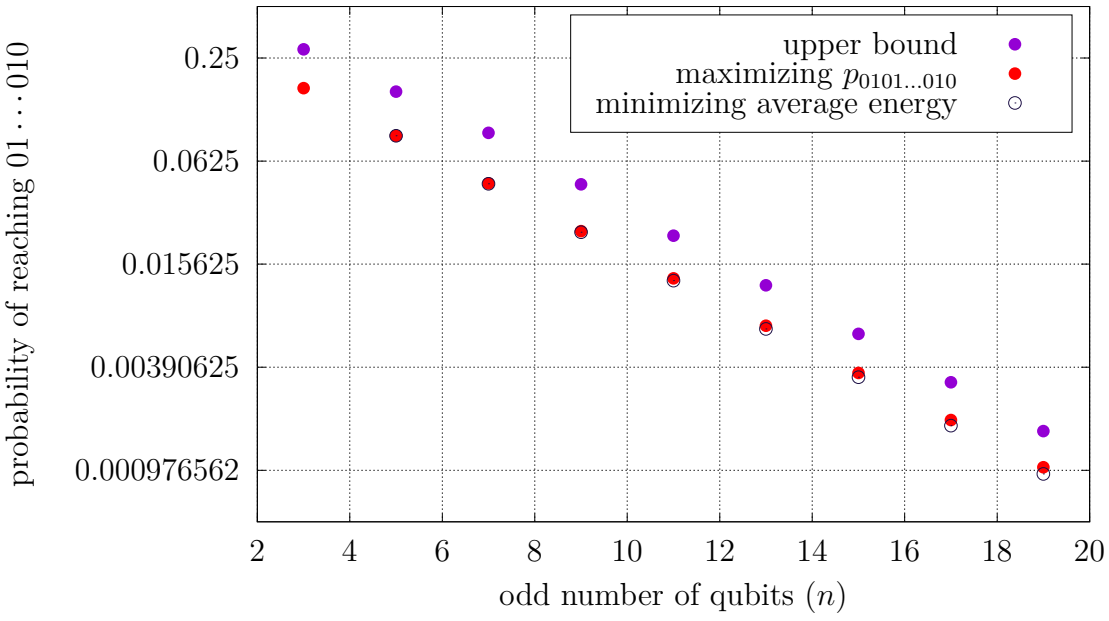
as the number of edges for an n-vertex ring is n. Notice that this is exactly the same result which we have achieved for the highly local version of QAOA. However, while the local version cannot improve its performance with increasing number of rounds p, the standard QAOA can. And due to the original Farhi et al's article [11]:

$$r^*=\frac{2p+1}{2p+2}.$$

Now, we may ask: *What is the maximal probability of reaching optimal/near-optimal solutions (using $p=1$ QAOA)?* We have numerically observed that the maximal probability of reaching all the computational-basis states decreases exponentially with n. Hence, also the maximal probability of reaching the optimal MaxCut solutions, which we present in the following graphs:



Graph 2.1: The maximal probability (red) of reaching $0101\cdots01$ (or $1010\cdots10$, but as well $0000\cdots00$ or $1111\cdots11$, as for even n we have (1.51)) for even n. As the value is for each state independently, the real p_{Maxcut} probability is twice the value in the graph.



Graph 2.2: The maximal probability (red) of reaching any of the MaxCut optimal solutions for odd n. The rotational symmetry of the rings gives us properties analogical to the NOT symmetry for all rotations of the state - they all have the same amplitude $a(\gamma,\beta)$. Together, there are $2n$ optimal MaxCut solutions. So the real p_{Maxcut} probability is $2n$ times the value in the graph.

Remark. *The "minimizing average energy" are the parameters that maximimize $r^\star$ (the name because $F_p(\gamma,\beta)$ is the expected energy with respect to C). Notice that the corresponding probability value of the optimal MaxCut solutions (the empty circles) is not that different from the maximal possible (the red circles). However, they do differ.*

We have also found an upper bound that decays exponentially as well. Recall that we can express the amplitude as

$$a(\beta,\gamma,j) = \frac{1}{\sqrt{2^n}} \sum_{k\in\{0,1\}^n} e^{-i\gamma(m-2C(k))} (\cos\beta)^{n-ham(j,k)} (i\sin\beta)^{ham(j,k)} .$$

The upper bound is the case, when we group all terms with the same phase $e^{-i\gamma(m-2C(k))}$ together, hence, we gain:

$$a(\beta,\gamma,j) = term_1 \cdot e^{-i\gamma n} + term_2 \cdot e^{-i\gamma(n-2)} + \cdots ,$$

maximize these terms independetly, and then suppose that all the local amplitudes $e^{-i\gamma\varphi}$ will point in the same direction (upper bound $= \sum_i \max(term_i)$). Hence, we can conclude, that the local amplitudes do not affect the exponential decline. The decline is caused by the combination of the terms that gained the same amlitude[10].

And as even for this simple MaxCut instances that can be solved by a classical computer in linear time, the maximal probability of reaching the MaxCut solutions (the same is true for all computational basis states) decays exponentially with n, QAOA when limited only to few rounds does not seem very promising for approximate optimization.[11]

Further observations

For $p = 1, 2, 3, 4$ QAOA, we have observed that the maximal probability of reaching the optimal MaxCut solution is one until the subgraph type for the cost function $F_p(\gamma,\beta)$ starts to be a line (instead of a ring or an overlapping ring what is the case before the number of vertices reaches $2p+2$).[12] As $n = 2p+1$ would be the last case when we reach $p_{\text{MaxCut}} = 1$, such result would imply that we need

$$p = \frac{n-1}{2}$$

to find the exact solution with probability 1. So, it would mean that QAOA can solve these simple MaxCut instances in polynomial time after all.[13]

[10] However, there are still some local amplitudes in this terms, too. There is the factor $i^{ham(j,k)}$

[11] $r^* = 3/4$ is not very good, in comparison to $r^* = 1$.

[12] Unfortunatelly, we prove it here only by "hasty generalization" (http://jwilson.coe.uga.edu/EMT668/EMAT6680.F99/Challen/proof/proof.html) from numerical observations.

[13] Let us note that previously by QAOA limited only to few rounds, we really meant few rounds (as constants). Not polynomials.

Conclusion

The goal of this thesis was to investigate QAOA, a promising application for near-term quantum computers. We were curious about whether even low-complexity versions of this algorithm can be useful, or if we could observe some patterns of what could make this algorithm seem promising in general. Hopefully, even ask whether QAOA could motivate some local classical algorithm that could make some improvement in the area of approximate optimization for some COP. A thing analogical to what Ewin Tang [33] managed to do for recommendation systems when she introduced a quantum-inspired classical algorithm that was only polynomially slower than the Kerenidis and Prakash's quantum algorithm [27], previously suspected of giving an exponential speedup over classical algorithms.

Our goal was probably too ambitious as currently, there is no known problem instance where QAOA would surpass the best-known classical algorithm. As well, we were outperformed by Hastings [22], who has introduced a class of local classical optimization algorithms and suggested that they are at least as promising as QAOA for approximate optimization. However, not much is currently understood about QAOA beyond its lowest depth versions. So, it might outrun them, after all. One may even take into account the generalization to Quantum alternating operator ansatz [19]. Overall, it is still an open question.

While persuing the goal by investigating QAOA's properties on simple MaxCut instances - rings, lines, and complete graphs – we have observed that even a low-round QAOA can be useful. We can use $p = 1$ ($p = 2$) QAOA for efficient preparation of n-odd (n-even) GHZ states (Section 2.2). For $p = 1$, we present the proof, while for $p = 2$, we provide only numerical evidence.

On the other hand, we suspect a low-round QAOA of having a very poor performance for approximate optimization as these low-complexity QAOA versions applied to rings and lines, simple MaxCut instances that can be solved by a classical computer in linear time, cannot reach particular computational basis states (e.g., the optimal MaxCut solution). The maximal probability of reaching these states decreases exponentially with the number of qubits/vertices. This part is discussed in Section 2.3.

For the purpose of QAOA investigation, we have built a set of numerical simulations one can use to investigate (maximal) probabilities of reaching computational basis

states using low-round ($p = 1$, $p = 2$) QAOA, more in Appendix B. Moreover, in order to make a grid search for optimal QAOA parameters more effective, we have also proved several symmetries of the parameter space (see Section 2.1).

Bibliography

[1] S. Aaronson. NP-complete Problems and Physical Reality. *arXiv:quant-ph/0502072*, February 2005.

[2] Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic Quantum Computation is Equivalent to Standard Quantum Computation. *arXiv:quant-ph/0405098v2*, May 2004.

[3] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters 8(3):121–123*, 1979.

[4] 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:1505.03424v2 [cs.CC]*, 2015.

[5] M. Born and V. A. Fock. Beweis des adiabatensatzes. *Zeitschrift für Physik A, 51(3–4):165–180*, March 1928.

[6] S. Bravyi, A. Kliesch, R. Koenig, and E. Tang. Obstacles to State Preparation and Variational Optimization from Symmetry Protection. *arXiv:1910.08980 [quant-ph]*, October 2019.

[7] S. Bringsjord and J. Taylor. P=NP. *arXiv:cs/0406056v1 [cs.CC]*, June 2004.

[8] D. Cheung, P. Hoyer, and N. Wiebe. Improved Error Bounds for the Adiabatic Approximation. *arXiv:1103.4174 [quant-ph]*, March 2011.

[9] R. Duan. Quantum Adiabatic Theorem Revisited. *arXiv:2003.03063 [quant-ph]*, March 2020.

[10] E. Farhi, J. Goldstone, and S. Gutmann. A Quantum Approximate Optimization Algorithm Applied to a Bounded Occurrence Constraint Problem. *arXiv:1412.6062v2 [quant-ph]*, June 2015.

[11] E. Farhi, J. Goldstone, and S. Gutmann. A Quantum Approximate Optimization Algorithm. *arXiv:1411.4028 [quant-ph]*, November 2014.

[12] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Quantum Computation by Adiabatic Evolution. *arXiv:quant-ph/0001106*, January 2000.

[13] E. Farhi and A. W. Harrow. Quantum Supremacy through the Quantum Approximate Optimization Algorithm. *arXiv:1602.07674 [quant-ph]*, February 2016.

[14] M. Garey, D. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theor. Comput. Sci. 1:237–267*, 1976.

[15] M. R. Garey, R. L. Grajam, and D. S. Johnson. Some NP-complete geometric problems. *Proceedings of the eighth annual ACM symposium on Theory of computing, pages 10-22*, May 1976.

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

[17] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM), 42(6):1145*, November 1995.

[18] Lov K. Grover. A fast quantum mechanical algorithm for database search. *arXiv:quant-ph/9605043*, May 1996.

[19] S. Hadfield, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas. From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. *Algorithms 2019, 12(2), 34. (Special issue "Quantum Optimization Theory, Algorithms, and Applications"), https://doi.org/10.3390/a12020034*, February 2019.

[20] E. Halperin, D. Livnat, and U. Zwick. MAX CUT in cubic graphs. *Journal of Algorithms, 53(2):169-185*, 2004.

[21] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum Algorithm for Linear Systems of Equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009.

[22] M. B. Hastings. Classical and Quantum Bounded Depth Approximation Algorithms. *arXiv:1905.07047v2 [quant-ph]*, August 2019.

[23] Wen Wei Ho and Timothy H. Hsieh. Efficient variational simulation of non-trivial quantum states. *arXiv:1803.00026v4 [cond-mat.str-el]*, 2019.

[24] Wen Wei Ho, Cheryne Jonay, and Timothy H. Hsieh. Ultrafast Variational Simulation of Non-trivial Quantum States with Long Range Interactions. *arXiv:1810.04817v2 [cond-mat.str-el]*, May 2019.

[25] J. Håstad. On bounded occurrence constraint satisfaction. *Information Processing Letters, 74(1):1-6*, 2000.

[26] J. Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM), 48(4):798-859*, July 2001.

[27] I. Kerenidis and A. Prakash. Quantum Recommendation Systems. *arXiv:1603.08675v3 [quant-ph]*, March 2016.

[28] S. Khot. On the power of unique 2-prover 1-round games. *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing, pages 767-775*, May 2002.

[29] S. Khot, G. Kindler, E. Mossel, and R. O'Donnel. Optimal Inapproximability Results for MAX-CUT and Other 2-Variable CSPs? *SIAM Journal on Computing, 37(1):319-357*, 2007.

[30] D. Knuth. Teach Calculus with Big O. *Notices of the American Mathematical Society, 45(6):687*, 1998.

[31] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[32] M. Suzuki. Generalized Trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems. *Commun. Math. Phys. 51:183–190*, 1976.

[33] E. Tang. A quantum-inspired classical algorithm for recommendation systems. *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing - STOC 2019. pages 217–228*, 2019.

[34] L. Trevisan. Non-approximability results for optimization problems on bounded degree instances. *Proceedings of the thirty-third annual ACM symposium on Theory of computing, pages 453–461*, July 2001.

[35] D. H. Wolpert and W. G. Macready. No Free Lunch Theorems for Search. *Technical Report SFI-TR-95-02-010, Santa Fe Institute*, 1995.

[36] D. H. Wolpert and W. G. Macready. No Free Lunch Theorems for Optimization. *IEEE transactions on evolutionary computation 1(1):67-82*, 1997.

[37] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices. *arXiv:1812.01041v2 [quant-ph]*, December 2018.

Appendices

A Big $\mathcal{O}$ notation

Big $\mathcal{O}$ notation is a mathematical notation that describes the asymptotical behavior of a function. This notation, along with Big Θ (theta) and Big Ω (omega), is widely used in computer science for classification of algorithms according to how their run time or space requirements grow as the input size grows.

Let us define them formally. Let f and g be two increasing functions on natural numbers. Then:

- $f \in \mathcal{O}(g)$ if and only if there exists a constant c, and some $n_0 \in \mathbb{N}$, such that for all $n \geq n_0$ it follows that: $f(n) \leq c \cdot g(n)$. Hence, g is an upper bound for f.
- $f \in \Omega(g)$ if and only if $g \in \mathcal{O}(f)$. Ω, therefore, represents o lower bound.
- $f \in \Theta(g)$ if and only if $f \in \mathcal{O}(g)$ and at the same time $f \in \Omega(g)$. In other words, the theta notation bounds a functions from both: above and below. It defines the exact asymptotic behavior of the function.

Remark. *It is more common to write "$f = \mathcal{O}(g)$" ("f is $\mathcal{O}(g)$") instead of $f \in \mathcal{O}(g)$. Strictly mathematically, it is incorrect, but as Donald Knuth pointed out: mathematicians customarily use the = sign as they use the word "is" in English: Aristotle is a man, but a man isn't necessarily Aristotle. [30]*

Motivation

Even though the computational resources are getting stronger and stronger, the amount of data we need to process is growing as well, and even today, we simply cannot run an enormous number of steps quickly. If we need some results, it would be useful to have a good guess about how long the algorithm will take to give you the desired results. And having an asymptotic estimation on how many steps an algorithm needs to run depending on the input size may be a good measure. Just take a look at the following table:

time/complexity	n	n^2	n^3	2^n	$n!$
millisecond	1 000 000	1 000	100	20	10
second	1 000 000 000	30 000	1 000	30	12
minute	∞	250 000	4 000	35	14
hour	∞	2 000 000	15 000	41	15
day	∞	9 000 000	44 000	46	16
month	∞	51 000 000	130 000	51	17
year	∞	170 000 000	310 000	54	18
millennium	∞	∞	3 100 000	64	21

Table A.1: The greatest n for which a program with the given time complexity finishes in the given time. (The values are rounded. The values over 1 000 000 000 are substituted by the symbol ∞.) Of course, the real values depend on the power of the computer. This table is only illustrative.

As we may observe, it does not really matter whether the time complexity of the given algorithm is n^2, or n^2+100n. For n big enough, the value of n^2 is much more significant than $100n$, and we can neglect the $100n$. Structurally, the entries of the table would not change (up to division by 5), if we would have taken $5n^3$ instead of n^3. Therefore, from an asymptotic time estimation point of view, all we need to know about the algorithm is the term of the highest magnitude. When analyzing algorithms, the most commonly encountered classes of functions are the following.

- $\mathcal{O}(1)$ constant
- $\mathcal{O}(\log n)$ logarithmic
- $\mathcal{O}(n)$ linear
- $\mathcal{O}(n^2)$ quadratic
- $\mathcal{O}(n^c)$ polynomial
- $\mathcal{O}(c^n)$ exponential

How to determine the complexity of an algorithm?

Let us start with an example. How many stars ($*$) will the following python program output?

```
for i in range(n):
    for j in range(i,n):
        print("*")
```

For $i = 0$, the algorithm will sequentially print out n lines, each containing "$*$". For $i = 1$, there will be $n - 1$ lines with "$*$", for $i = 2$, $n - 2$ lines with "$*$", and so on until $i = n - 1$, when the last "$*$" will be outputed. In total, there were $n + (n - 1) + (n - 2) + \cdots + 1 = \frac{n(n-1)}{2}$ steps performed. The term of the highest magnitude is

$\frac{n^2}{2}$. We also cross out the constant. Now, we can say that the time complexity of this algorithm is $\mathcal{O}(n^2)$.

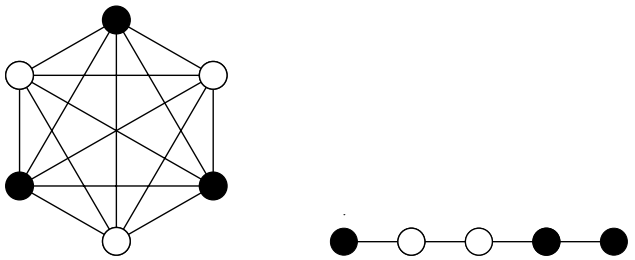
Of course, when analyzing the time complexity, we need not be that precise. We could simply tell that those are two nested for-loops, each executed at most n times, the print out of the "$*$" is $\mathcal{O}(1)$, hence $\mathcal{O}(n \cdot n) = \mathcal{O}(n^2)$.

In general, a sequence of independent block codes gives us the sum of the steps performed in each block code. A for-loop that iterates n times over a block of code takes n times the number of steps performed by the block code. And in case of if-else statements, we take the worst possible case.

For a non-trivial example, you might take a look at the program example in Appendix B.

B QAOA-MaxCut-amplitudes

In this text, we explain the idea and scheme of our QAOA-analyzing toolbox's content, which one can find at https://github.com/dendaxD/QAOA-MaxCut-amplitudes. Its purpose is to produce all the amplitudes (when expressed in the computational basis) in low-round QAOA-MaxCut-produced $\left|\vec{\gamma},\vec{\beta}\right\rangle$ (Eqn. 1.28), as well as the corresponding maximal probabilities, for a given graph. The graph types we investigated are the following:



(a) *A complete graph (6v3).* (b) *A line (5v1 2 2).* (c) *A ring (9v3 1 1 2 1 1).*

Complete graphs. *Graphs where each vertex shares an edge with all other vertices.*
Rings. *2-regular (each vertex has exactly 2 neighbors), connected graphs.*
Lines. *Rings to whom one edge was stolen.*

The desired output for each graph type, should include:

- a file **result.txt** which contains a filled table with columns: *the identification of the state – the maximal probability of reaching the state – the optimal parameters*,
- in **mathematica-files** directory, for each state, there should be a Mathematica notebook containing the state's amplitude to play with,
- in **amplitudes** directory, there should be **amplitudes.pdf** file containing all the amplitudes for $n \leq 6$, together with the value of the maximal probability of reaching the corresponding state.

We use the python code, {*the graph type*}.*py*, the achieve this. In a high-level overview, the algorithm does the following:

```
for number_of_vertices in range(3,15):
    state_to_amplitude = {}
    for z in {0,1}^n:
        state_to_amplitude[z] = create_amplitude(z)
        create Mathematica notebook to find max p_z
        run the Mathematica notebook
        remove the Mathematica notebook
        create the Mathematica file to play with
    pick up he ugly formated Mathematica outputs (located in .txt files)
    create the desired output (result.txt and amplitudes.pdf)
```

For maximization, the launched Mathematica notebook does a grid search, and subsequently, it uses its built-in NMaximize function:

```
NMaximize[{ probability[β, γ],
 β_f - step < β < β_f + step,γ_f - step < γ < γ_f + step}, {β, γ},
 Method → {"SimulatedAnnealing", "PerturbationScale" → 3}]
```

to improve the value – to search the surrounding area of the grid-search-found result $p_f(\gamma_f, \beta_f)$ – as the steps were relatively large.[1]

Now, let us note, that besides the NOT symmetry which makes the amplitudes of $|z\rangle$ and $|\neg z\rangle$ equal, the symmetries of the graph enlarge this group of states with equal amplitudes as well. Therefore, calculating the amplitude for each state, separately, would be wasteful. Hence, we label each group of states with the sam amplitude with a "label" and calculate the amplitude only for a representant from each group. Of course, all states sharing a group have the same value of the *cut* (the objective function).

Labels for 2-colored graph configurations

So, how should we label the groups? Of course, the first thing that interests us is: *How many vertices does the graph have?* Therefore, each label starts with a string 'nv'.

Complete graphs. Highly symmetric graphs – we can interchange any two vertices. Hence, from the graph's symmetries, there are only $n + 1$ different groups. All that matters is how many of the vertices are labeled by '1'. We will label a string with $\#_1$

[1] Please note that as an implementation detail, the code has opposite order of the parameters γ, β.

ones: $n\text{v}\#_1$. For instance, 0000 is labeled by 4v0, 010101 (as well as all $\binom{6}{3}-1$ other permutations of three zeroes and three ones) is labeled 6v3. If we take into account the NOT symmetry, there are only $\lfloor n/2 \rfloor$ groups as the amplitude of states from $n\text{v}\#_1$ is equal to the amplitude of a state from a group $n\text{v}\#_0 = n\text{v}(n - \#_1)$. Hence, we will take $n\text{v}\{\min(\#_1, \#_0)\}$.

Rings. We can rotate them as well as mirror them. To create the label, choose your favorite jump (edge connecting 0 and 1) and your preferred direction. Now, walk through the whole graph, counting the number of subsequent vertices of the same color. When there is a jump, write down the number and continue with another group, unless you have reached the jump where you have started. For example, a possible label for 110000110 is 9v2 4 2 1, for 11000011 a label is 8v4 4, for 10101 5v2 1 1 1.

Lines. We are left only with the mirroring. For the label, choose your favorite side of the line and walk to the other side analogously as you do for the rings. Some examples of line labels are: 11000011 $\rightarrow$ 8v2 4 2, 111 $\rightarrow$ 3v3, 1011010 $\rightarrow$ 7v1 1 2 1 1 1.

How to create the amplitude

Let us construct an algorithm whose goal is to output each label with the corresponding amplitude (in the Mathematica form).

In section 1.3.1 (the 2-vertex line example), we observed that the amplitude corresponding to the computational basis state j can be expressed as

$$a\left(\beta,\gamma,j\right) = \frac{1}{\sqrt{2^n}} \sum_{k\in\{0,1\}^n} e^{-i\gamma(m-2C(k))} \left(\cos\beta\right)^{n-ham(j,k)} \left(i\sin\beta\right)^{ham(j,k)}.$$

Howevever, we do not want to blindly follow this equation. We wish to organize the output somehow. Hence, we wil group the terms with the same phase ($e^{-i\gamma(m-2C(k))}$) together. Additional requirement would be to avoid unnecessary duplicities. By this we mean that instead of $\sin^2(x)\cos^3(x)+\sin^2(x)\cos^3(x)$, we prefer to write $2\sin^2(x)\cos^3(x)$.

The scheme of the algorithm is the following.

1. Group the computational basis states according to their label. In the following code, this is done by the function *create_label_to_states_dict(n)*.

2. We wish to gain the amplitude for each label group. Hence, we iterate through all possible labels. For each:

   a. We choose a representant j for whom the amplitude will be calculated.

   b. As we have mentioned, we want the amplitude in the form:

   $$a_{\text{label}} = \frac{1}{\sqrt{2}} \sum_{\text{cut}} e^{-i\gamma\text{cut}} \sum_{\text{ham}} \text{coef} \cdot \left(\cos\beta\right)^{n-\text{ham}} \left(i\sin\beta\right)^{\text{ham}}.$$

   Hence, we will compare the representant to every $z \in \{0,1\}^n$, creating a nested dictionary $ampl\_info = \{\text{cut} \rightarrow \{\text{ham} \rightarrow \text{coef}\}\}$ in the process.

c. All we are left to do, is to "loop" through the *ampl_info* to create the amplitude. (*create_amplitude(n, ampl_info)*)

Here is our version of the algorithm:

```
# An O(n^2) function which generates the set of edges in n-vertex CG.
def edges(n):
    return [(x + 1, y + 1) for x in range(n) for y in range(n) if x < y]

# This one takes O(n).
def hamming_distance(string1, string2):
    diff = 0
    for c1, c2 in zip(string1, string2):
        if c1 != c2:
            diff += 1
    return diff

# O(1)
def cut(l, n):
    return n * (n - 1) / 2 - 2 * l * (n - l)

# O(n)
def label(state):
    return min(state.count("1"), state.count("0"))

# O(2^n · n)
def create_label_to_states_dict(n):
    label_to_states_dict = {}
    # iterate through all the states that start with 0
    # the rest is included by 'not_s'
    for i in range(2 ** (n - 1)):
        # Each computational basis state is a binary representation
        # of some i (e.g. '01001' is binary form of 9).
        s = str(bin(i))[2:]
        s = (n - len(s)) * "0" + s
        # It takes O(n) to obtain 'not_s'.
        not_s = ""
        for symbol in s:
            not_s += "0" if symbol == "1" else "1"
        l = label(s)  # O(n)
        c = cut(l, n)
        # A look-up to a dictionary takes, on average, only O(1).
        # In the worst case O(n).
        if l in label_to_states_dict:
            label_to_states_dict[l].extend([s, not_s])
        else:
            label_to_states_dict[l] = [s, not_s]
```

```
43      return label_to_states_dict
44
45  # O(n^2 · 2^n)
46  def create_ampl_info(n, representant, label_to_states, labels):
47      ampl_info = {}  # cut to {hamming_distance to coefficients}
48      for l in labels:
49          if cut(l, n) not in ampl_info:
50              ampl_info[cut(l, n)] = {}
51          # O(n · 2^n) - This bound could be improved as it will never
52          # happen for all states to have the same label.
53          for state in label_to_states[l]:
54              ham = hamming_distance(representant, state)  # O(n)
55              if ham in ampl_info[cut(l, n)]:
56                  ampl_info[cut(l, n)][ham] += 1
57              else:
58                  ampl_info[cut(l, n)][ham] = 1
59      return ampl_info
60
61  # O(n^3)
62  def create_amplitude(n, ampl_info):
63      ampl = ""  # ampl for amplitude
64      for c in ampl_info: #c for cut
65          ampl += f"Exp[{-c} I y] ("
66          # O(n)
67          for ham, coef in ampl_info[c].items():
68              if coef != 1:
69                  ampl += f"{coef} "
70              if ham == n:
71                  ampl += f"(I Sin[x])^{n}"
72              elif ham == 0:
73                  ampl += f"Cos[x]^{n}"
74              else:
75                  ampl += f"(I Sin[x])^{ham} Cos[x]^{n - ham}"
76              ampl += " + "
77          ampl = ampl[:-3] + ") + "
78          # '[:-3]' to remove the additional ' + '
79      return f"({ampl[:-3]})/Sqrt[2^n]"
80
81  # O(n^3 · 2^n)
82  def create_label_to_amplitude_dict(n):
83      label_to_amplitude_dict = {}
84      label_to_states = create_label_to_states_dict(n)  # O(2^n · n)
85      labels = label_to_states.keys()
86      # O(n^3 · 2^n)
87      for label in labels:  # number of labels = round up n/2
88          representant = label_to_states[label][0]
89          # O(n^2 · 2^n)
```

```
90          ampl_info = create_ampl_info(n, representant, label_to_states, labels)
91          label_to_amplitude_dict[label] = create_amplitude(n, ampl_info)
92      return label_to_amplitude_dict
93
94  for label, amplitude in create_label_to_amplitude_dict(3).items():
95      print(str(label), ": ", amplitude)
```

Observe that it is not hard to extend the algorithm to more rounds. First let us write the term for the $p = 2$ amplitude:

$$a_2\left(\beta_1\beta_2, \gamma_1\gamma_2, j\right) = \sum_{k\in\{0,1\}^n} a(\beta_1, \gamma_1, k) e^{-i\gamma_2(m-2C(k))} \left(\cos\beta_2\right)^{n-ham(j,k)} \left(i\sin\beta_2\right)^{ham(j,k)},$$

where $a(\beta_1, \gamma_1, k)$ is the amplitude in front of the computational basis state k after 1 round. Now, observe that we can write the equation also generally (we gain a recursive equation):

$$a_p\left(\beta_1\cdots\beta_p, \gamma_1\cdots\gamma_p, j\right) = \sum_{k\in\{0,1\}^n} a_{p-1}(\beta_1\cdots\beta_{p-1}, \gamma_1\cdots\gamma_{p-1}, k)\cdot \\ \cdot e^{-i\gamma_p(m-2C(k))} \left(\cos\beta_p\right)^{n-ham(j,k)} \left(i\sin\beta_p\right)^{ham(j,k)},$$

But let us consider only $p = 2$. From the point of view of this calculation, the amplitude is simply an extension of the coeficient $e^{-i\gamma_2(m-2C(k))}$. As in this special case of complete graphs, we are also lucky that the cut is in one to one correspondence with a group, having already *ampl_info* and *label_to_amplitude_dict*, we can easily extend the code. How to do this and how to generalize this approach to more general cases, we leave as an exercise.