Quantum Approximate Optimization Algorithm

Performance on Max-Cut using Heuristic Parameter determination

Joost Bus

Delft University of Technology, the Netherlands

23 July 2020

Supervisors
Matthias Möller
Carmina G. Almudever

Contact: `j.c.p.bus@student.tudelft.nl`

Overview

Why Quantum Computing?

- Medicine
- Chemistry
- Cryptography
- Optimization

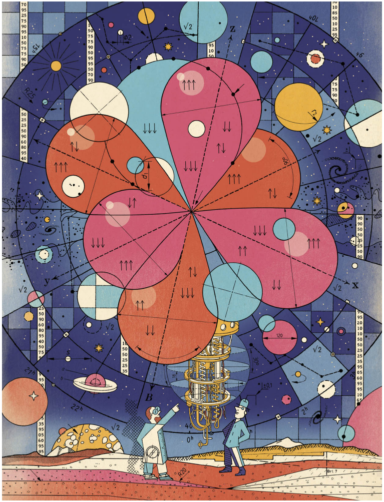


Illustration by Christian Gralingen

Are we there yet?

The current state of Quantum Computers

- $10 \sim 100$ qubits
- Limited connectivity
- Non-negligible error rates
- No error correction

Are we there yet?

The current state of Quantum Computers

- $10 \sim 100$ qubits
- Limited connectivity
- Non-negligible error rates
- No error correction

Do current devices still have useful applications?

Are we there yet?

Do current devices still have useful applications?

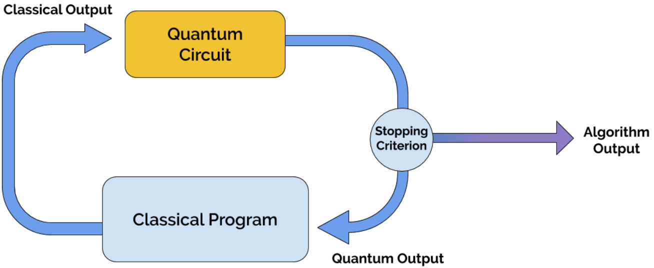


Figure from Entropica Labs

Quantum Computing - The basics

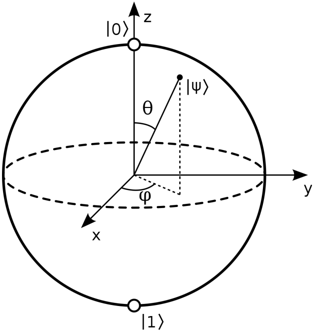


Figure from Wikipedia - Bloch sphere

Quantum Computing - The basics

One quantum bit or qubit can be described as a **superposition**, or (linear) combination of two states with corresponding **amplitudes**

$$|\psi\rangle = \alpha_1|0\rangle + \alpha_2|1\rangle \quad (1)$$

Quantum Computing - The basics

One quantum bit or qubit can be described as a **superposition**, or (linear) combination of two states with corresponding **amplitudes**

$$|\psi\rangle = \alpha_1|0\rangle + \alpha_2|1\rangle \tag{1}$$

Similarly, for a two qubit system we the system is described with four amplitudes.

$$\alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle \tag{2}$$

Quantum Computing - The basics

One quantum bit or qubit can be described as a **superposition**, or (linear) combination of two states with corresponding **amplitudes**

$$|\psi\rangle = \alpha_1|0\rangle + \alpha_2|1\rangle \tag{1}$$

Similarly, for a two qubit system we the system is described with four amplitudes.

$$\alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle \tag{2}$$

In general, we need 2^n amplitudes to describe an n qubit system and some say the system is in $\mathbf{2^n}$ **states** "at the same time". Hence the exponential power of the quantum computer.

$$\alpha_1|0\dots0\rangle + \dots + \alpha_{2^n}|1\dots1\rangle \tag{3}$$

Quantum Computing - The basics

However, we are not able to measure the amplitudes directly. Upon measurement, the system **collapses** into *one* classical state with probability related to the magnitude of the corresponding amplitude.

Quantum Computing - The basics

Example with 2 qubits

Given the following system

$$\alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle \tag{4}$$

we find the following results with their corresponding probabilities

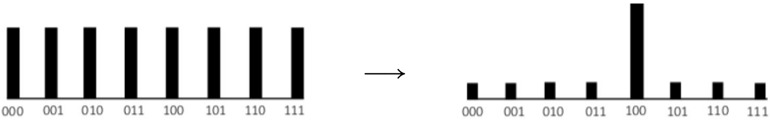
State	Amplitude	Probability
00	α_1	$\lvert\alpha_1\rvert^2$
01	α_2	$\lvert\alpha_2\rvert^2$
10	α_3	$\lvert\alpha_3\rvert^2$
11	α_4	$\lvert\alpha_4\rvert^2$

Quantum Computing - The basics

So the amplitudes have a special meaning: **the squared magnitude signifies the probability of a particular state.** We can change the amplitudes by applying **gates**, which are descibed by (unitary) matrices.

Quantum Computing - The basics

So the amplitudes have a special meaning: **the squared magnitude signifies the probability of a particular state.** We can change the amplitudes by applying **gates**, which are descibed by (unitary) matrices.



The Max-Cut problem

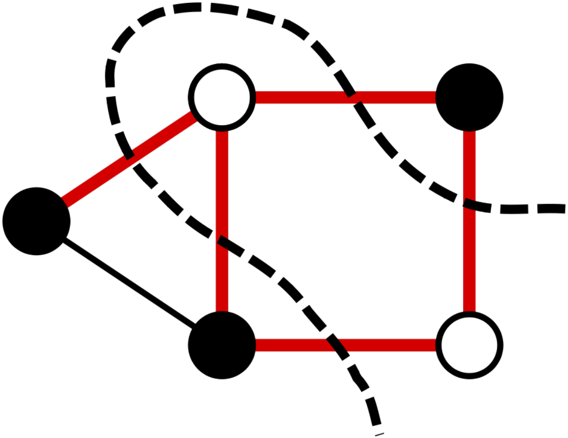


Figure from Wikipedia - Maximum cut

We would like to find a **bipartition** that maximizes the following **cost function**

$$C = \sum_{i \in S, j \in \bar{S}} w_{i,j} \tag{5}$$

We would like to find a **bipartition** that maximizes the following **cost function**

$$C = \sum_{i \in S, j \in \bar{S}} w_{i,j} \tag{5}$$

Equivalently, we can use a **binary string** to represent the bipartition

$$C = \sum_{\{i,j\}} \frac{w_{i,j}}{2}(1 - z_i z_j) \tag{6}$$

where $\boldsymbol{z} \in \{-1, 1\}^n$ and $z_i = \begin{cases} 1, & \text{if} \quad i \in S \\ -1, & \text{if} \quad i \in \bar{S} \end{cases}$

The Quantum Approximate Optimization Algorithm

The Quantum Approximate Optimization Algorithm is designed to tackle combinatorial optimization problems. In general, these can be specified with n bits and m clauses. The aim is to satisfy as many clauses as possible

$$C = \sum_{\alpha=1}^{m} C_\alpha \tag{7}$$

where $C_\alpha = \begin{cases} 1, & \text{if clause } C_\alpha \text{ is satisfied} \\ 0, & \text{if clause } C_\alpha \text{ is } \textit{not} \text{ satisfied} \end{cases}$

Schematic of the QAOA circuit

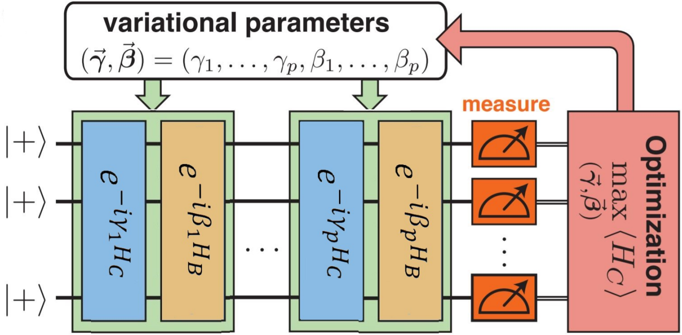


Figure adapted from Zhou et al. Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices (2018)

The Quantum Part

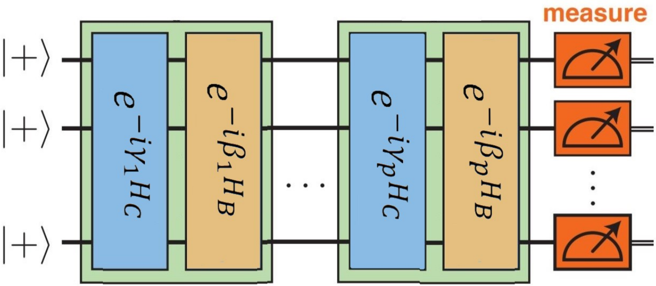
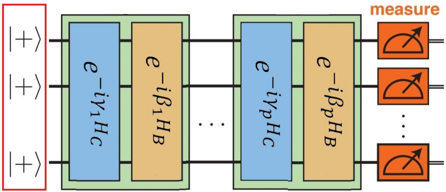
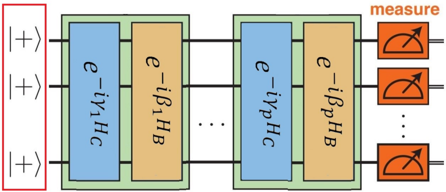


Figure adapted from Zhou et al. Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices (2018)



We start from $|+\rangle^{\otimes n}$ which is the **equal superposition** over all 2^n bit strings (or equivalently bipartitions)

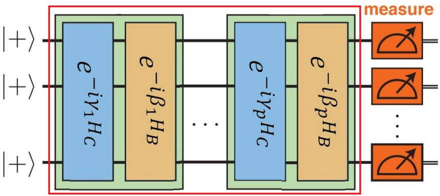
$$|+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{\boldsymbol{x} \in \{0,1\}^n} |\boldsymbol{x}\rangle \tag{8}$$



We start from $|+\rangle^{\otimes n}$ which is the **equal superposition** over all 2^n bit strings (or equivalently bipartitions)

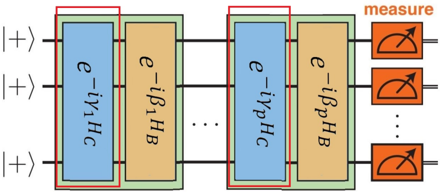
$$|+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{\boldsymbol{x}\in\{0,1\}^n} |\boldsymbol{x}\rangle \tag{8}$$

So the amplitude of every bit string is the same



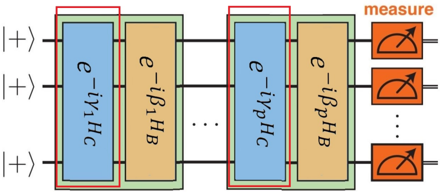
Next we alternately apply two gates, the **cost unitary** and the **mixer unitary**, derived from two Hamiltonians.

We repeat this p times



First of the two being the **cost unitary**. This unitary is derived from the **cost Hamiltonian** that encodes the objective function C

$$H_C \equiv \hat{C} = \begin{bmatrix} C(0,\ldots,0) & & \\ & \ddots & \\ & & C(1,\ldots,1) \end{bmatrix} \tag{9}$$



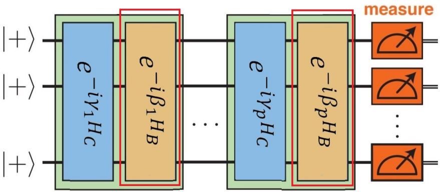
From the cost Hamiltonian we derive the **cost unitary**

$$H_C \longrightarrow U(H_C, \gamma) \tag{10}$$

with

$$U(H_C, \gamma) = e^{-i\gamma H_C} \tag{11}$$

for some real parameter $\gamma \in \mathbb{R}$

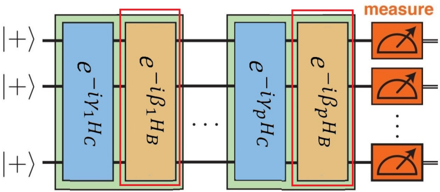


Secondly, the **mixer unitary**, derived from the **mixer Hamiltonian**

$$H_B = \sum_{k=1}^{n} \sigma_x^{(k)} \tag{12}$$

with $\sigma_x^{(k)}$ is the Pauli-X gate applied to the kth qubit, which is also called the quantum NOT-gate

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{13}$$



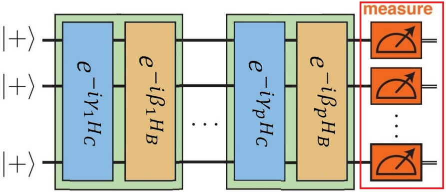
Analogous to the cost unitary we derive the **mixer unitary** is derived from the mixer Hamiltonian

$$H_B \longrightarrow U(H_B, \beta) \tag{14}$$

the mixer unitary is given by

$$U(H_B, \beta) = e^{-i\beta H_B} \tag{15}$$

for some real parameter $\beta \in \mathbb{R}$



In the end we prepared the parametrized state

$$|\gamma, \beta\rangle = \underbrace{U(H_B, \beta_p)U(H_C, \gamma_p) \dots \underbrace{U(H_B, \beta_1)U(H_C, \gamma_1)}_{\text{one layer}}}_{p \text{ layers}} |+\rangle^{\otimes n} \tag{16}$$

after which we measure the outcome.

The goal: maximize the expectation value

Our aim is to prepare a state such that the **expectation value of the cost Hamiltonian** is maximized

$$F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \boldsymbol{\gamma}, \boldsymbol{\beta} | H_C | \boldsymbol{\gamma}, \boldsymbol{\beta} \rangle \tag{17}$$

for some sequences of parameters

$$\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$$

$$\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$$

Applying QAOA to Max-Cut

We **translate** the objective function into a **Hamiltonian**

$$C = \sum_{\{i,j\}} w_{i,j}(1 - z_i z_j) \tag{18}$$

$$\downarrow$$

$$H_C = \sum_{\{i,j\}} w_{i,j}\left(I - \sigma_z^{(i)} \sigma_z^{(j)}\right) \tag{19}$$

Applying QAOA to Max-Cut

We **translate** the objective function into a **Hamiltonian**

$$C = \sum_{\{i,j\}} w_{i,j}(1 - z_i z_j) \tag{18}$$

$$\downarrow$$

$$H_C = \sum_{\{i,j\}} w_{i,j}\left(I - \sigma_z^{(i)}\sigma_z^{(j)}\right) \tag{19}$$

Using the fact σ_z has eigenvalues 1 and -1. Here $\sigma_z^{(i)}$ denotes the Pauli-Z matrix applied to the ith qubit.

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{20}$$

Applying QAOA to Max-Cut

We can construct the necessary unitaries with local operators

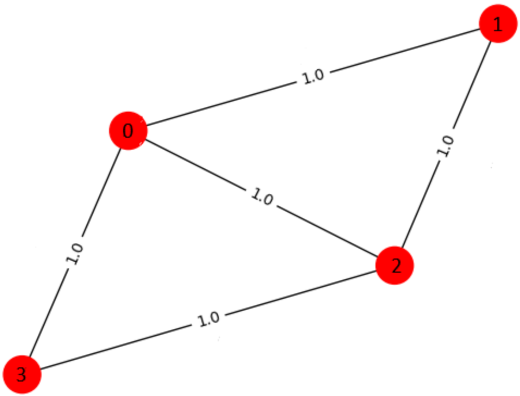
$$U(H_C,\gamma) = e^{-i\gamma H_C} = \prod_{\{i,j\}\in E} e^{-i\gamma w_{i,j}(1-\sigma_z^{(i)}\sigma_z^{(j)})} \tag{21}$$

$$U(H_B,\beta) = e^{-i\beta H_B} = \prod_{k\in V} e^{-i\beta\sigma_x^{(k)}} \tag{22}$$

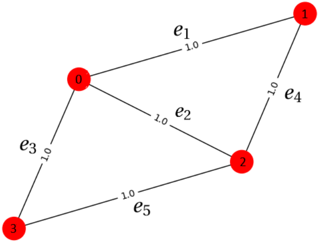
Note that the circuit is dependent on the graph $G = (V, E)$

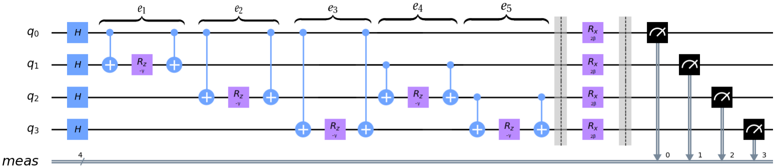
Now we are set to construct the circuit for Max-Cut!

Example - the Diamond graph

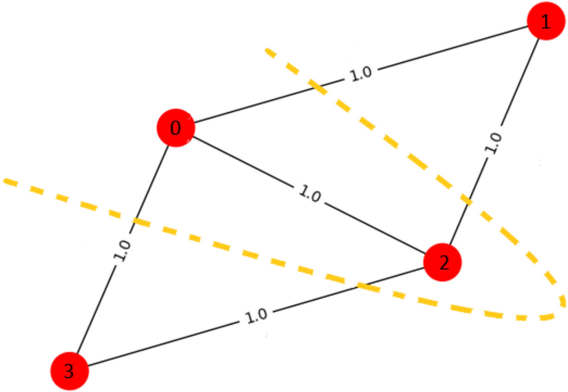


Example - the circuit



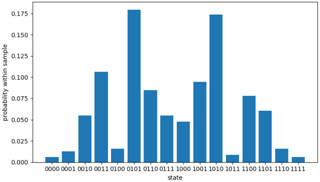


Example - the optimal cut

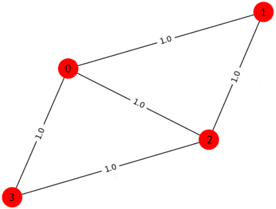


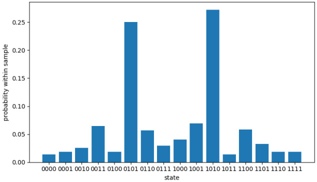
Note that the optimal cut is $S = \{0, 2\}, \bar{S} = \{1, 3\}$ or in terms of a binary string 0101 or 1010

Example - Maximizing the expectation value

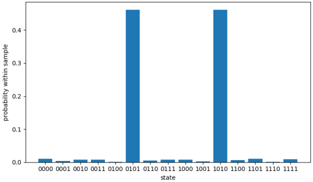


p=1, $F_1 = 3.24$



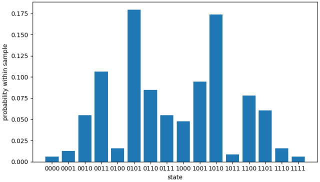


p=2, $F_2 = 3.39$

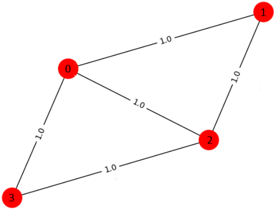


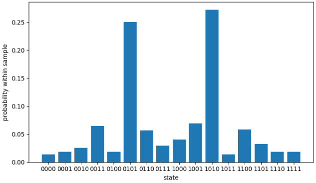
p=3, $F_3 = 3.87$

Example - Maximizing the expectation value

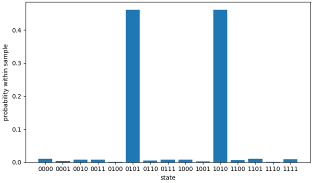


p=1, F_1 = 3.24



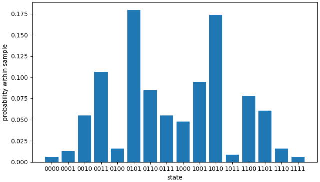


p=2, F_2 = 3.39

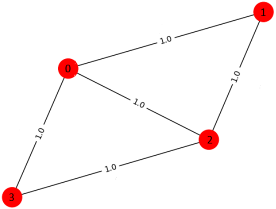


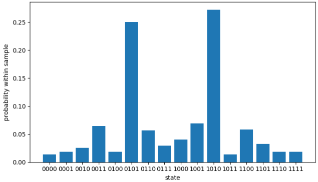
p=3, F_3 = 3.87

Example - Maximizing the expectation value

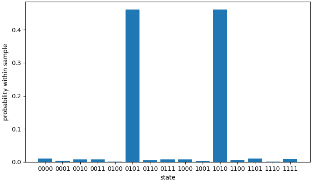


p=1, F_1 = 3.24





p=2, F_2 = 3.39



p=3, F_3 = 3.87

Determining the parameters γ, β

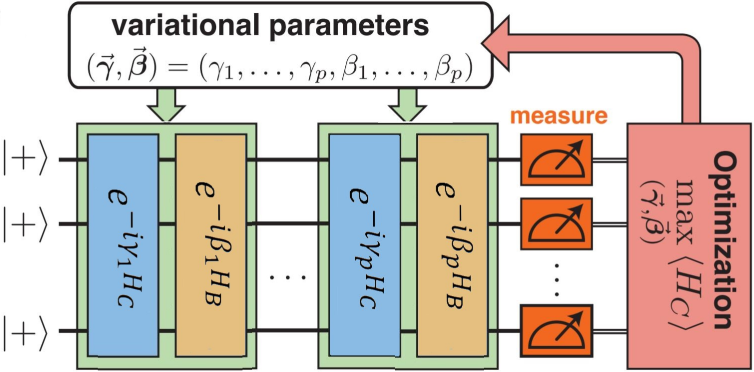


Figure: Schematic of the QAOA circuit using VQE optimization.[1]

[1] Figure adapted from Zhou et al. Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices (2018)

Relation to Quantum Adiabatic Algorithm

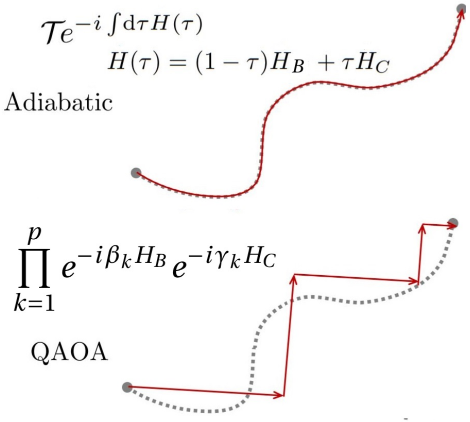


Figure adapted from Verdon et al. A quantum algorithm to train neural networks using low-depth circuits (2017)

Relation to Quantum Adiabatic Algorithm

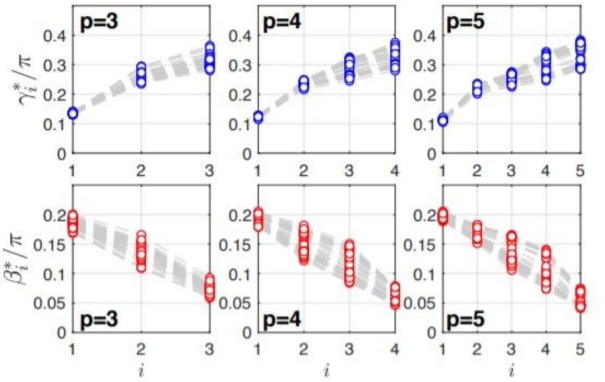
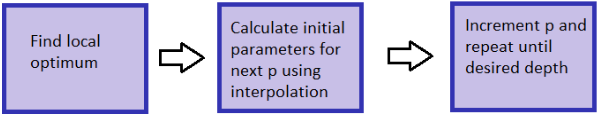


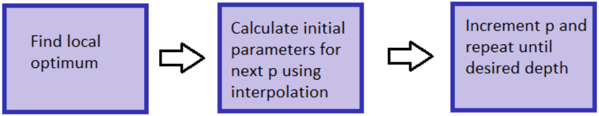
Figure: Optimal parameter patterns for unweighted 3-regular graphs with 16 nodes[2]

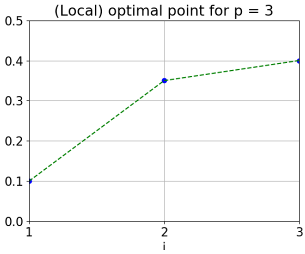
[2]Zhou et al. Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices (2018)

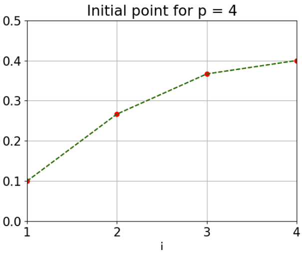
Exploiting the relation to QAA - the INTERP method



Exploiting the relation to QAA - the INTERP method



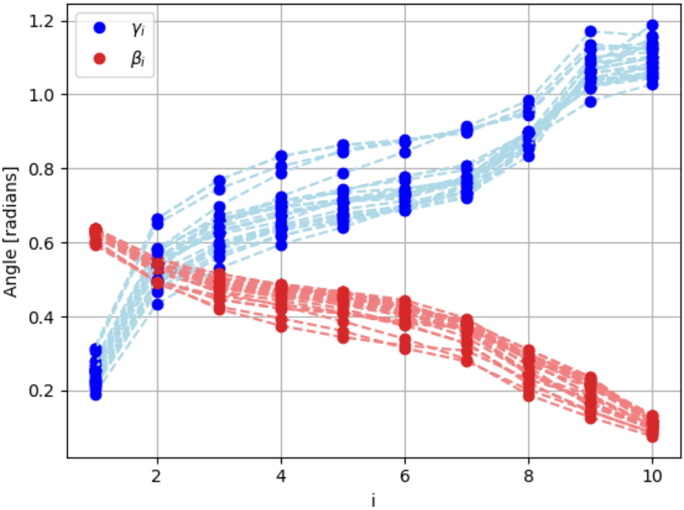




My work

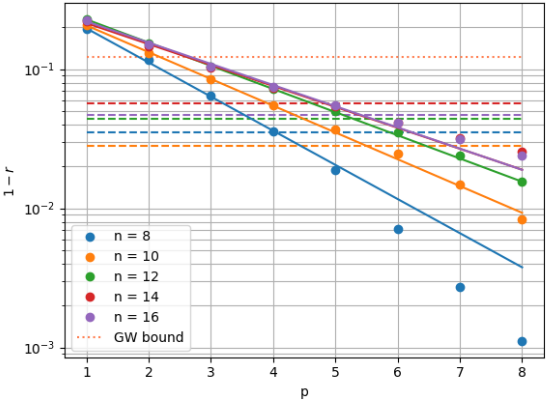
1. analysis on **INTERP method** on different graphs
   - cyclic graphs
   - 3-regular graphs, weighted and unweighted
   - Erdős-Rényi graphs
2. benchmark against **Goemans-Williamson**, the best known classical approximation algorithm
3. polynomial time

Found patterns



Unweighted 3-regular graphs

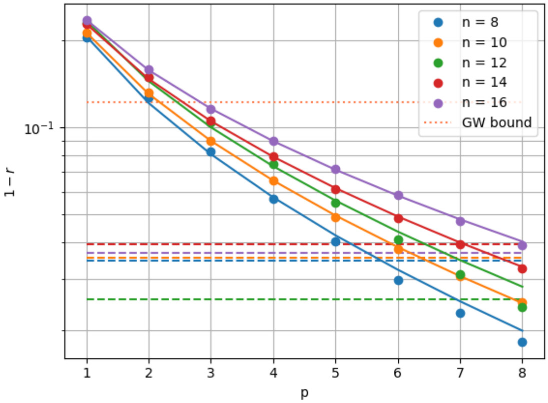
Fractional error $1 - r$ decays exponentially with p



The horizontal lines indicate the average performance of the classical Goemans-Williamson

Weighted 3-regular graphs

Fractional error $1 - r$ decays exponentially with $\sqrt{p}$



The horizontal lines indicate the average performance of the classical Goemans-Williamson

Similar relations were found for the Erdős-Rényi graphs

Conclusions

- INTERP beats the classical Goemans-Williamson for relatively low $p \approx 7$ for small graphs
- The method needs a lot of function evaluations to determine good angles
- However, this number increases polynomially with p and n so it might offer advantages for large graphs

Conclusions

- INTERP beats the classical Goemans-Williamson for relatively low $p \approx 7$ for small graphs
- The method needs a lot of function evaluations to determine good angles
- However, this number increases polynomially with p and n so it might offer advantages for large graphs

Conclusions

- INTERP beats the classical Goemans-Williamson for relatively low $p \approx 7$ for small graphs
- The method needs a lot of function evaluations to determine good angles
- However, this number increases polynomially with p and n so it might offer advantages for large graphs

Thank you!

Questions?

Time complexity

In Zhou et al. (2018) it was claimed the INTERP method is polynomial in p, and since the quantum circuit has depth $3m + n$, we find that the complete INTERP method is polynomial in both n and p, but is this true in practice?