Quantum Approximate Optimization Algorithm Part 2

Eunok Bae

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

2023. 8. 17.

고려대학교 양자대학원 2023 Special Summer Internship

구성

CONTENTS

01 Review: QAOA

02 Adiabatic Quantum Computation

03 Variants of QAOA

04 Hands-on

Variational Quantum Algorithms

- What is VQAs?
- VQE and **QAOA**

Quantum Approximate Optimization Algorithm

What is QAOA?

Level p-QAOA

1. Initialize the quantum processor in $|+\rangle^{\otimes N}$
2. Generate a variational wavefunction

$$|\psi_p(\vec{\gamma},\vec{\beta})\rangle = e^{-i\beta_p H_B} e^{-i\gamma_p H_C} \cdots e^{-i\beta_1 H_B} e^{-i\gamma_1 H_C} |+\rangle^{\otimes N}$$

by applying the problem Hamiltonian H_C and a mixing Hamiltonian $H_B = \sum_{j=1}^{N} X_j$

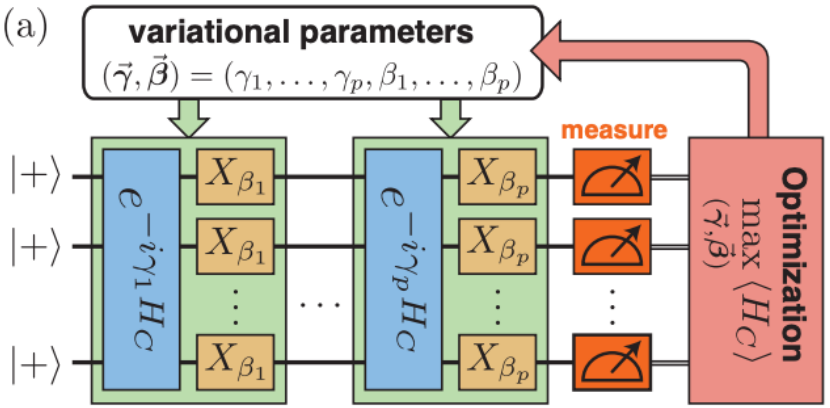
3. Determine the expectation value

$$F_p(\vec{\gamma},\vec{\beta}) = \langle \psi_p(\vec{\gamma},\vec{\beta}) | H_C | \psi_p(\vec{\gamma},\vec{\beta}) \rangle$$

4. Search for the optimal parameters

$$(\vec{\gamma}^*,\vec{\beta}^*) = \arg\max_{\vec{\gamma},\vec{\beta}} F_p(\vec{\gamma},\vec{\beta})$$

by a classical computer



[L. Zhou et al., Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices, Phys. Rev. X 10, 021067, 2020]

Approximation ratio $r = \dfrac{F_p(\vec{\gamma}^*,\vec{\beta}^*)}{C_{\max}}$

Variational Quantum Eigensolver (VQE)

Some variational ansatze – targeted at quantum simulation

✓ Hamiltonian Variational ansatz:

- Assume that: we want to find the ground state of $H = \sum_i H_i$

  we can write $H = H_B + H_C$

  ↑ easy to prepare the ground state of H_B

- Then: prepare the ground state of H_A

  For each of L layers l, implement $\prod_k e^{it_{lk}H_k}$ for some times $t_{lk} \in \mathbb{R}$

- Intuition comes from the **quantum adiabatic theorem**:

  As $L \to \infty$, this ansatz provably can represent the ground state of H.

Adiabatic Quantum Computing

Adiabatic Quantum Computing

= Quantum Annealing

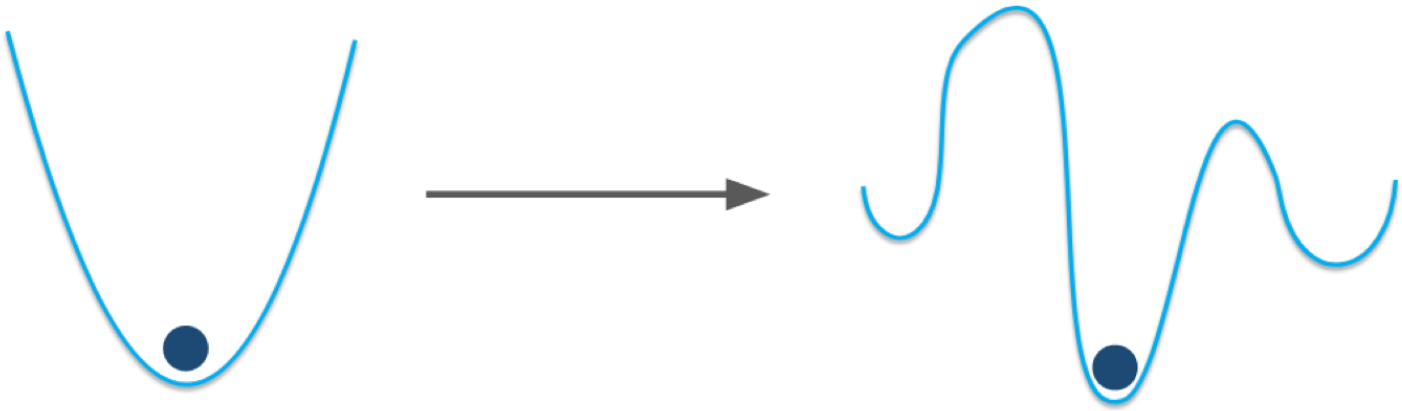


Figure 1. Schematic illustration of adiabatic quantum computing: by starting from the solution of a simple optimization problem (left) and slowly changing it to a complicated one (right), we are guaranteed by the adiabatic theorem to stay in the minimum during the whole evolution

Adiabatic Quantum Computing

Hamiltonian and time evolution

✓ **Schrödinger equation:**

- Time evolution of a quantum system with Hamiltonian H

$$H|\psi(t)\rangle = i\hbar \frac{\partial}{\partial t}|\psi(t)\rangle$$

- For time-independent H:

$$|\psi(t)\rangle = e^{-iHt/\hbar}|\psi(0)\rangle$$

✓ **Adiabatic theorem:**

If the Hamiltonian of a quantum system in its ground state is perturbed slowly enough, the system remains in its ground state.

Higher energy states

Adiabatic Quantum Computing

Hamiltonian and time evolution

- Consider the Hamiltonian $H = H_B + H_C$
- Time evolution operator

$$U(t) = e^{-\frac{iHt}{\hbar}} = e^{-\frac{i(H_B+H_C)t}{\hbar}}$$

- For commuting matrices H_B, H_C:

$$e^{H_B+H_C} = e^{H_B}e^{H_C}$$

✓ **Trotter Suzuki Formula:**

$$e^{-i(H_B+H_C)t} \approx \left(e^{-iH_Bt/r}e^{-iH_Ct/r}\right)^r$$

Adiabatic Quantum Computing

Adiabatic path or Annealing schedule

✓ **Adiabatic path or Annealing schedule:**

$$H(t) = \frac{t}{T} H_C + \left(1 - \frac{t}{T}\right) H_B \qquad t \in [0, T]$$

✓ **Discretizing AQC and QAOA:**

$$U(T) = U(T, 0) = U(T,$$

$$|\psi_p(\vec{\gamma}, \vec{\beta})\rangle = e^{-i\beta_p H_B} e^{-i\gamma_p H_C} \cdots e^{-i\beta_1 H_B} e^{-i\gamma_1 H_C} |+\rangle^{\otimes N}$$

$$\approx \prod_{j=1}^{p} e^{-iH(j\Delta t)\Delta t} = \prod_{j=1}^{p} e^{-i\left(\frac{j\Delta t}{T} H_C + \left(1 - \frac{j\Delta t}{T}\right) H_B\right)\Delta t} \approx \prod_{j=1}^{p} e^{-i\left(\frac{j\Delta t}{T}\right)\Delta t H_C} e^{-i\left(1 - \frac{j\Delta t}{T}\right)\Delta t H_B}$$

γ_j β_j

Variants of QAOA

arXiv > quant-ph > arXiv:2306.09198v2

Search...

Help | Advanced

Quantum Physics

[Submitted on 15 Jun 2023 (v1), last revised 26 Jun 2023 (this version, v2)]

A Review on Quantum Approximate Optimization Algorithm and its Variants

Kostas Blekos, Dean Brand, Andrea Ceschini, Chiao-Hui Chou, Rui-Hao Li, Komal Pandya, Alessandro Summer

The Quantum Approximate Optimization Algorithm (QAOA) is a highly promising variational quantum algorithm that aims to solve combinatorial optimization problems that are classically intractable. This comprehensive review offers an overview of the current state of QAOA, encompassing its performance analysis in diverse scenarios, its applicability across various problem instances, and considerations of hardware-specific challenges such as error susceptibility and noise resilience. Additionally, we conduct a comparative study of selected QAOA extensions and variants, while exploring future prospects and directions for the algorithm. We aim to provide insights into key questions about the algorithm, such as whether it can outperform classical algorithms and under what circumstances it should be used. Towards this goal, we offer specific practical points in a form of a short guide. Keywords: Quantum Approximate Optimization Algorithm (QAOA), Variational Quantum Algorithms (VQAs), Quantum Optimization, Combinatorial Optimization Problems, NISQ Algorithms

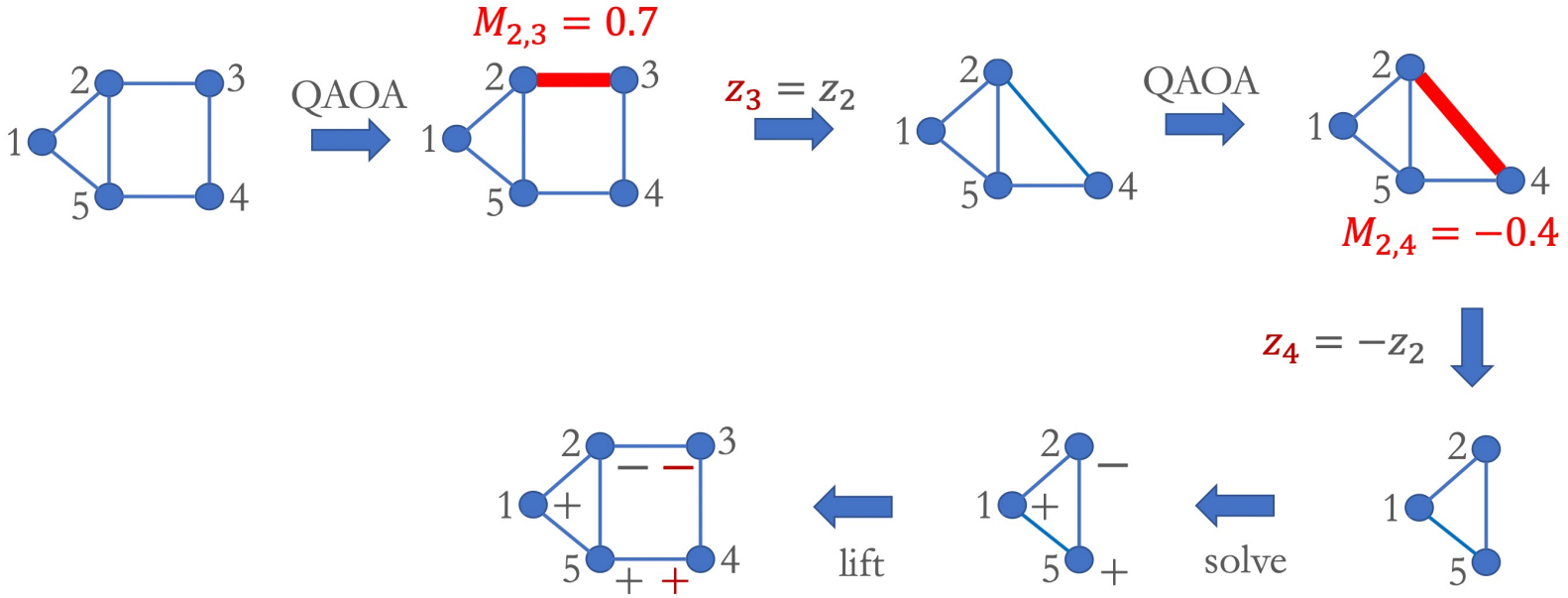
Comments: 67 pages, 9 figures, 9 tables; version 2 -- added more discussions and practical guides
Subjects: **Quantum Physics (quant-ph)**
Cite as: arXiv:2306.09198 **[quant-ph]**
(or arXiv:2306.09198v2 **[quant-ph]** for this version)
https://doi.org/10.48550/arXiv.2306.09198

Submission history

From: Rui-Hao Li [view email]
[v1] Thu, 15 Jun 2023 15:28:12 UTC (5,133 KB)
[v2] Mon, 26 Jun 2023 19:41:01 UTC (5,922 KB)

Variants of QAOA

✓ **Recursive QAOA** (2019) : iteratively reduces the problem size



[S. Bravyi *et al.*, Obstacles to State Preparation and Variational Optimization from Symmetry Protection, Phys. Rev. Lett. 125, 260505 (2019)]

Variants of QAOA

- ✓ Recursive QAOA (2019) : iteratively reduces the problem size
- ✓ **Warm starting QAOA** (2021)

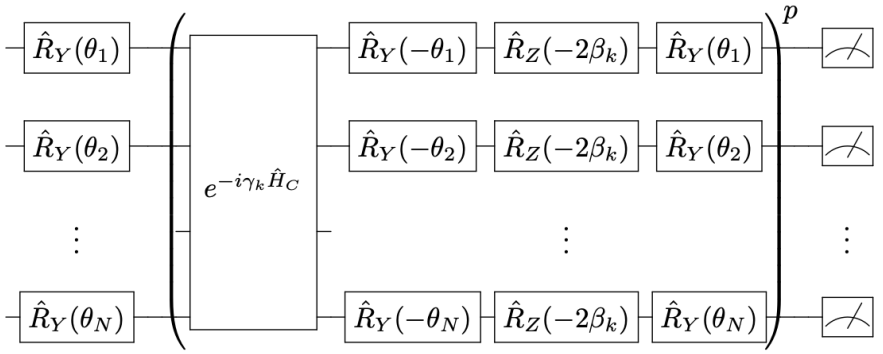
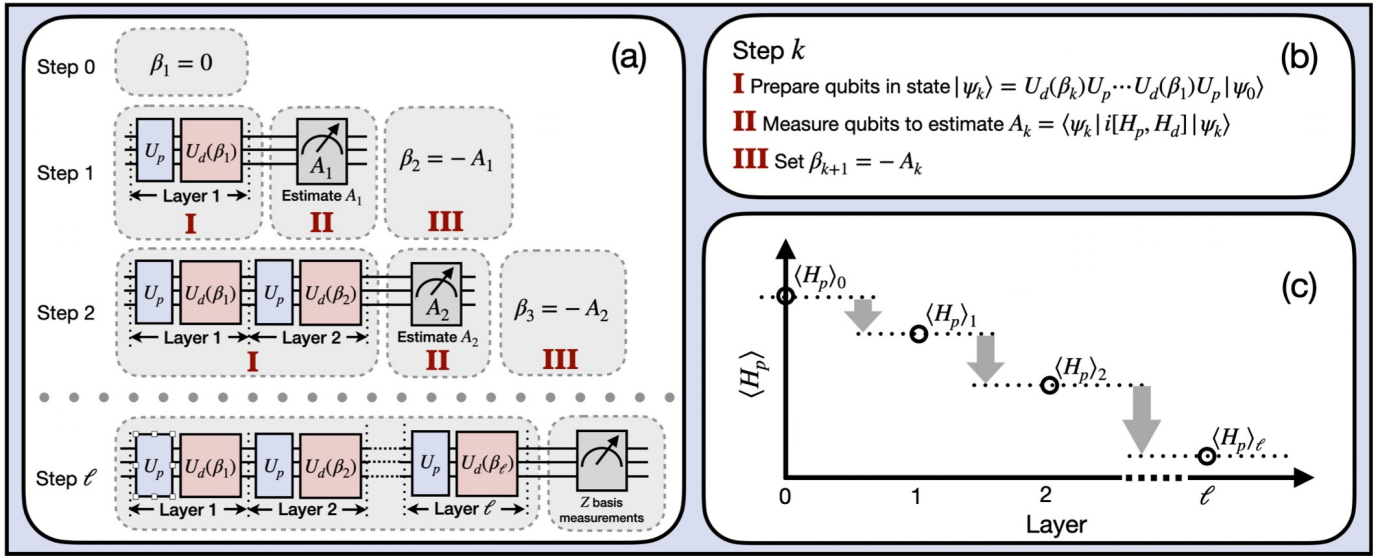


Figure 2: Quantum circuit for WS-QAOA. The first $\hat{R}_Y$ rotations prepare the initial state $|\phi^*\rangle$. The mixer operator, i.e. $\hat{R}_Y(\theta_i)\hat{R}_Z(-2\beta_k)\hat{R}_Y(-\theta_i)$, is applied after the time-evolved problem Hamiltonian $\hat{H}_C$.

[D. J. Egger *et al.*, Warm-starting quantum optimization, Quantum 5, 479 (2021)]

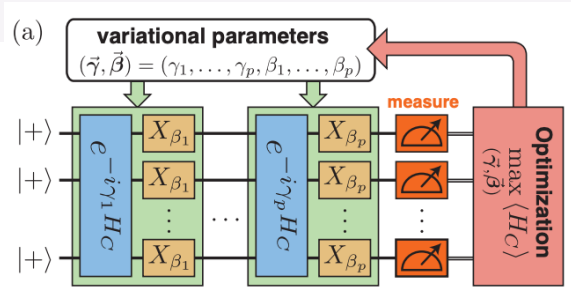
Variants of QAOA

- ✓ Recursive QAOA (2019) : iteratively reduces the problem size
- ✓ Warm starting QAOA (2021)
- ✓ **Feedback-based ALgorithm for Qauntum OptimizatioN (FALQON)** (2021)

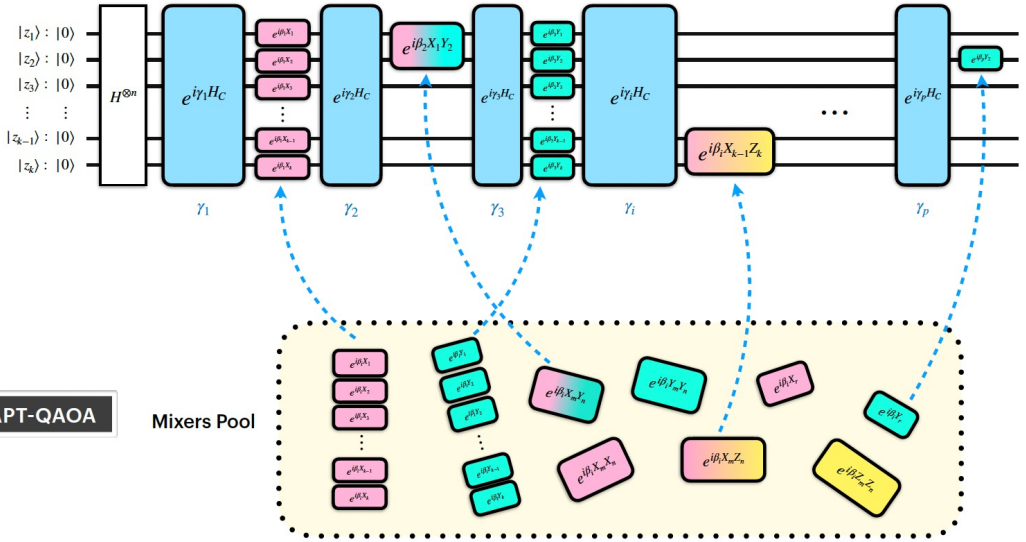


Variants of QAOA

- ✓ **Recursive QAOA** (2019) : iteratively reduces the problem size
- ✓ Warm starting QAOA (2021)
- ✓ FALQON (2021)
- ✓ **Adaptive QAOA** (2022) ⟶
  - different mixer Hamiltonian at each level



[Y. Liao *et al.*, Quantum Optimization for Training Quantum Neural Networks, arXiv:2103.17047 (2021)]

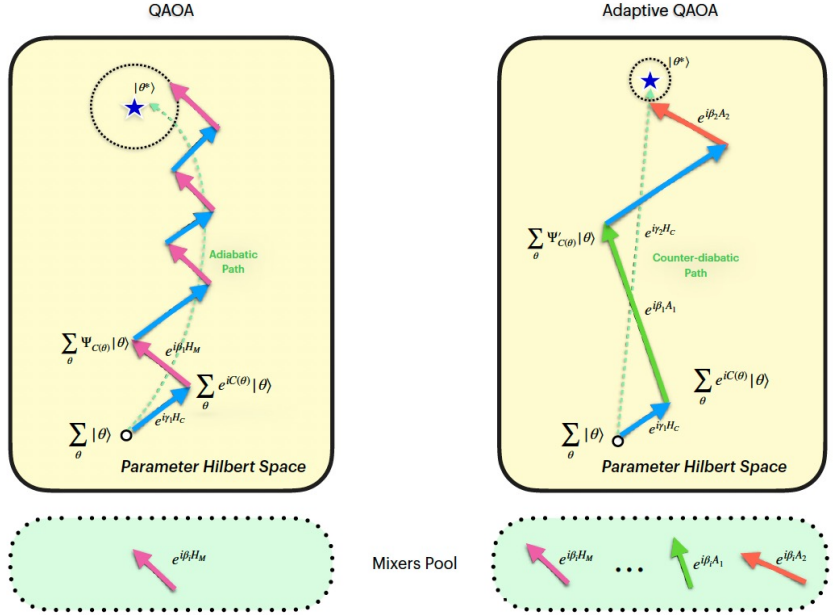


[L. Zhu *et al.*, Adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer, Phys. Rev. Research **4**, 033029 (2022)]

Variants of QAOA

- ✓ **Recursive QAOA** (2019) : iteratively reduces the problem size
- ✓ Warm starting QAOA (2021)
- ✓ FALQON (2021)
- ✓ **Adaptive QAOA** (2022)
  - shortcuts to adiabaticity

[Y. Liao *et al.*, Quantum Optimization for Training Quantum Neural Networks, arXiv:2103.17047 (2021)]



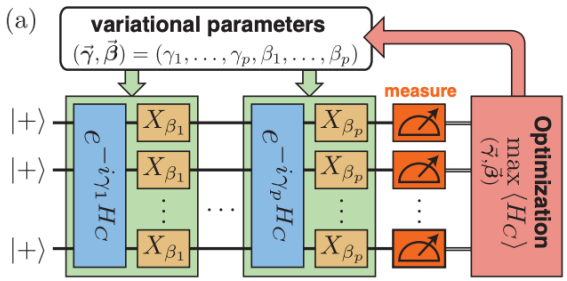
[L. Zhu *et al.*, Adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer, Phys. Rev. Research **4**, 033029 (2022)]

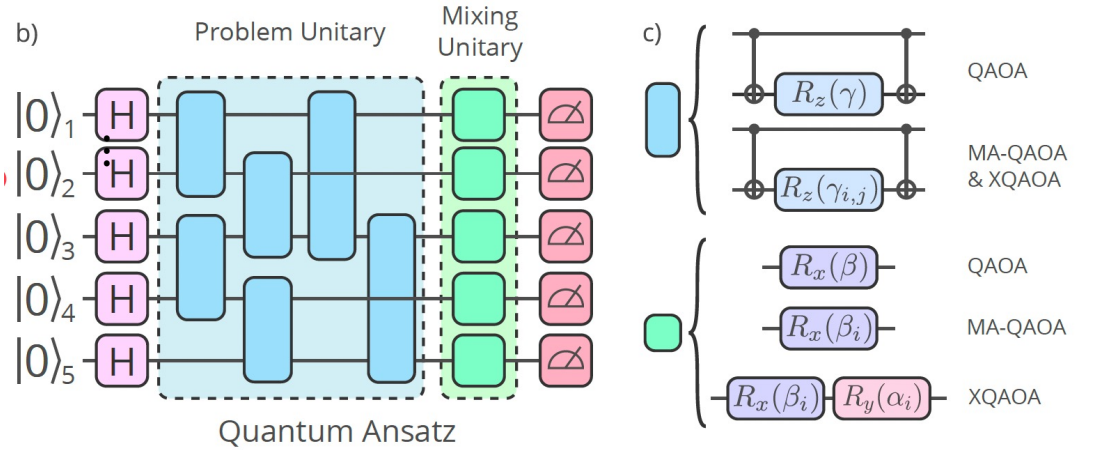
Variants of QAOA

- ✓ **Recursive QAOA** (2019) : iteratively reduces the problem size
- ✓ Warm starting QAOA (2021)
- ✓ Adaptive QAOA (2022)

- ✓ **Multi-angle QAOA** (2022)
- ✓ **eXpressive QAOA** (2023)

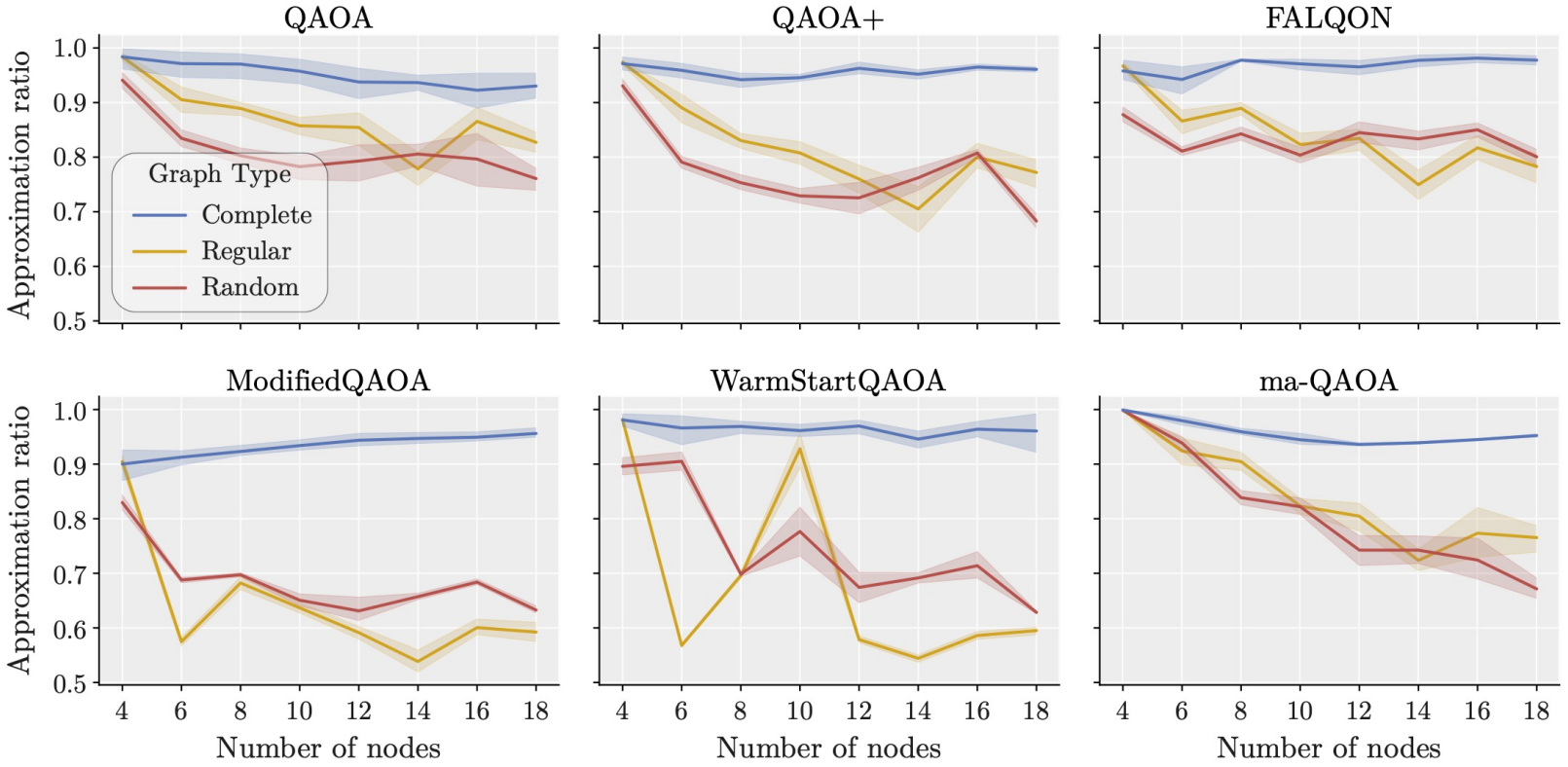
⋮





[V. Vijendran *et al.*, An Expressive Ansatz for Low-Depth Quantum Optimisation, arXiv:2302.04479 (2023)]

Variants of QAOA



[K. Blekos *et al.*, A Review on Quantum Approximate Optimization Algorithm and its Variants, arXiv:2306.09198 (2023)]

Implementing QAOA Hands-on

Quantum Approximate Optimization Algorithm

What is QAOA?

Level p-QAOA

1. Initialize the quantum processor in $|+\rangle^{\otimes N}$
2. Generate a variational wavefunction

$$|\psi_p(\vec{\gamma},\vec{\beta})\rangle = e^{-i\beta_p H_B} e^{-i\gamma_p H_C} \cdots e^{-i\beta_1 H_B} e^{-i\gamma_1 H_C} |+\rangle^{\otimes N}$$

by applying the problem Hamiltonian H_C and a mixing Hamiltonian $H_B = \sum_{j=1}^{N} X_j$

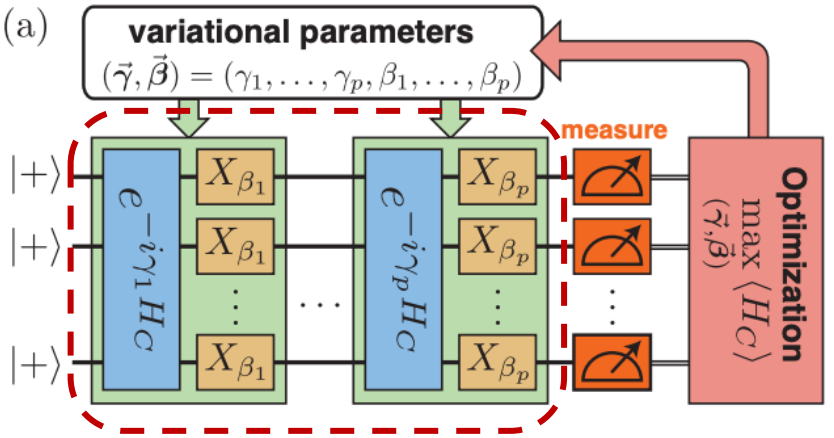
3. Determine the expectation value

$$F_p(\vec{\gamma},\vec{\beta}) = \langle \psi_p(\vec{\gamma},\vec{\beta}) | H_C | \psi_p(\vec{\gamma},\vec{\beta}) \rangle$$

4. Search for the optimal parameters

$$(\vec{\gamma}^*,\vec{\beta}^*) = \arg\max_{\vec{\gamma},\vec{\beta}} F_p(\vec{\gamma},\vec{\beta})$$

by a classical computer



[L. Zhou et al., Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices, Phys. Rev. X 10, 021067, 2020]

Approximation ratio $r = \dfrac{F_p(\vec{\gamma}^*,\vec{\beta}^*)}{C_{\max}}$

Quantum Approximate Optimization Algorithm

What is QAOA?

Level p-QAOA

1. Initialize the quantum processor in $|+\rangle^{\otimes N}$
2. Generate a variational wavefunction

$$|\psi_p(\vec{\gamma},\vec{\beta})\rangle = e^{-i\beta_p H_B} e^{-i\gamma_p H_C} \cdots e^{-i\beta_1 H_B} e^{-i\gamma_1 H_C} |+\rangle^{\otimes N}$$

by applying the problem Hamiltonian H_C and a mixing Hamiltonian $H_B = \sum_{j=1}^{N} X_j$

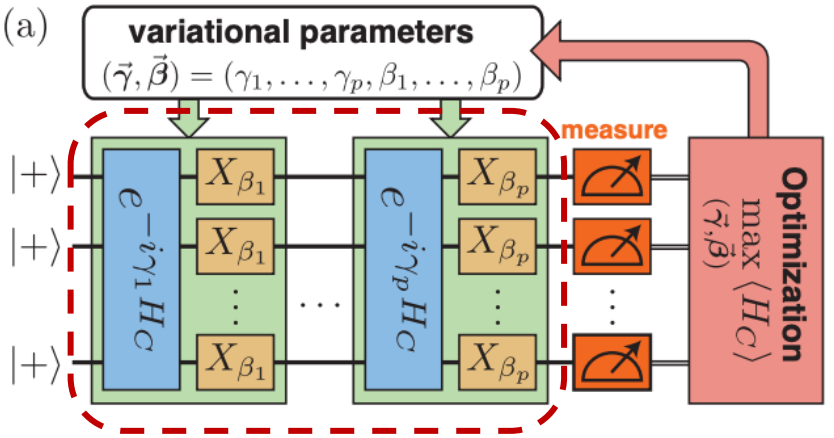
3. Determine the expectation value

$$F_p(\vec{\gamma},\vec{\beta}) = \langle \psi_p(\vec{\gamma},\vec{\beta}) | H_C | \psi_p(\vec{\gamma},\vec{\beta}) \rangle$$

4. Search for the optimal parameters

$$(\vec{\gamma}^*,\vec{\beta}^*) = \arg\max_{\vec{\gamma},\vec{\beta}} F_p(\vec{\gamma},\vec{\beta})$$

by a classical computer



[L. Zhou et al., Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices, Phys. Rev. X 10, 021067, 2020]

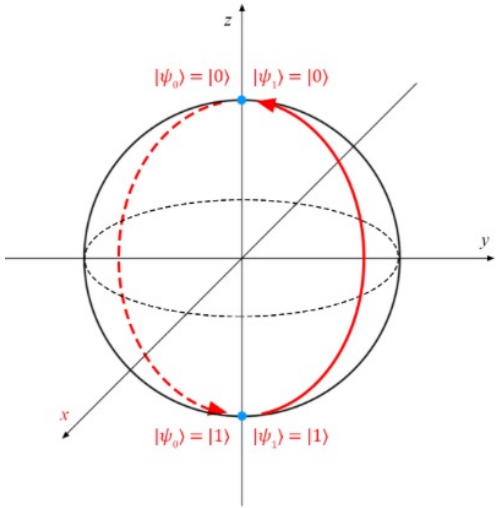
Approximation ratio $r = \dfrac{F_p(\vec{\gamma}^*,\vec{\beta}^*)}{C_{\max}}$

Implementing QAOA

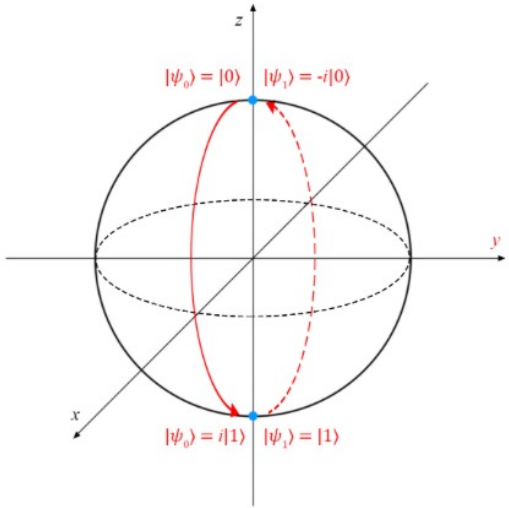
Pauli operators X, Y, and Z

Pauli operators (single qubit operations)

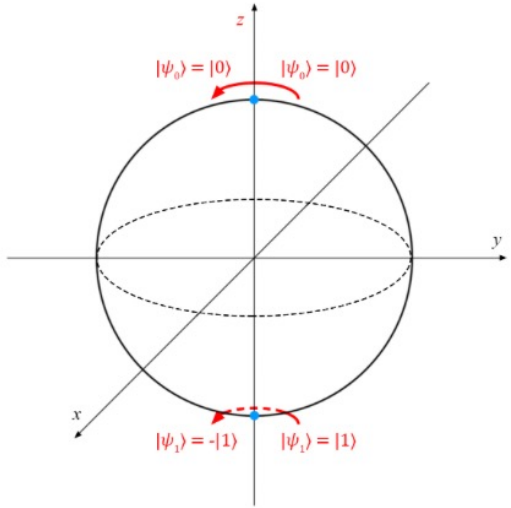
$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$



Pauli X



Pauli Y



Pauli Z

Implen

Pauli operat

Single-Qubit Gates

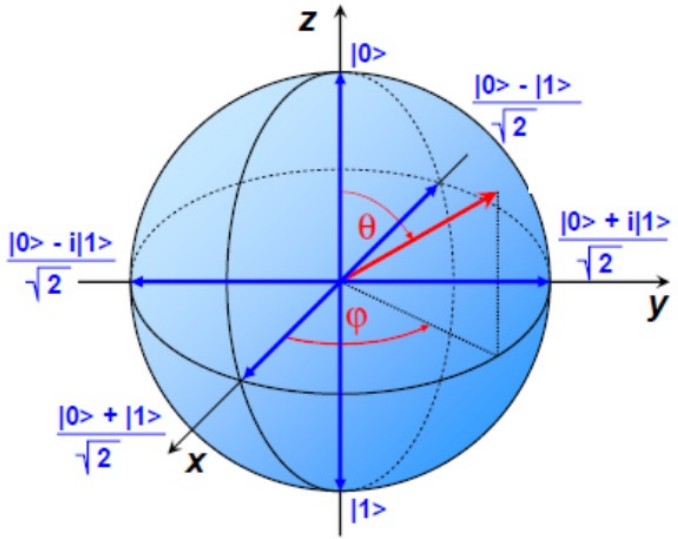
$$|\psi> = \cos(\theta/2)\,|0> + e^{i\phi}\sin(\theta/2)\,|1> = \begin{pmatrix} \cos(\theta/2) \\ e^{i\phi}\sin(\theta/2) \end{pmatrix}$$

Rotation matrices:

$$\hat{R}_x(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

$$\hat{R}_y(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

$$\hat{R}_z(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$$



- The rotation matrices are a linear combination of the Pauli operators: $\hat{\sigma}_x$, $\hat{\sigma}_y$, $\hat{\sigma}_z$ and the identity operator ($\hat{I}$).

$$\hat{R}_x(\theta) = \begin{pmatrix} cos\frac{\theta}{2} & -isin\frac{\theta}{2} \\ -isin\frac{\theta}{2} & cos\frac{\theta}{2} \end{pmatrix} = cos\frac{\theta}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - isin\frac{\theta}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = cos\frac{\theta}{2}\hat{I} - isin\frac{\theta}{2}\hat{\sigma}_x = e^{-i\frac{\theta}{2}X}$$

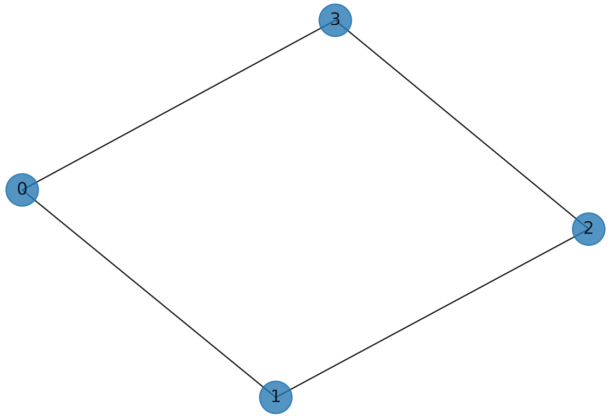
$$\hat{R}_y(\theta) = \begin{pmatrix} cos\frac{\theta}{2} & -sin\frac{\theta}{2} \\ sin\frac{\theta}{2} & cos\frac{\theta}{2} \end{pmatrix} = cos\frac{\theta}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - isin\frac{\theta}{2}\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = cos\frac{\theta}{2}\hat{I} - isin\frac{\theta}{2}\hat{\sigma}_y = e^{-i\frac{\theta}{2}Y}$$

$$\hat{R}_z(\theta) = \begin{pmatrix} e^{-\frac{i\theta}{2}} & 0 \\ 0 & e^{\frac{i\theta}{2}} \end{pmatrix} = \begin{pmatrix} cos\frac{\theta}{2} - isin\frac{\theta}{2} & 0 \\ 0 & cos\frac{\theta}{2} + isin\frac{\theta}{2} \end{pmatrix}$$

$$= cos\frac{\theta}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - isin\frac{\theta}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = cos\frac{\theta}{2}\hat{I} - isin\frac{\theta}{2}\hat{\sigma}_z = e^{-i\frac{\theta}{2}Z}$$

Implementing QAOA

The circuits of H_C and H_B



Pauli matrices

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

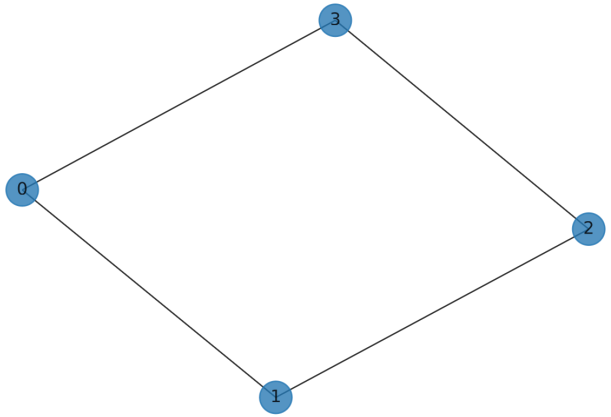
$$X|0\rangle = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = |1\rangle, \qquad X|1\rangle = |0\rangle$$

$$R_X(\theta) = e^{-i\frac{\theta}{2}X}$$

$$H_B = \sum_{j=1}^{N} X_j \qquad U(\beta) = e^{-i\beta H_B} = e^{-i\beta \sum_{j=1}^{N} X_j} = \prod_{j=1}^{N} e^{-i\beta X_j} = \prod_{j=1}^{N} R_{X_j}(2\beta)$$

Implementing QAOA

The circuits of H_C and H_B

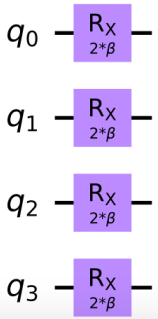


$$H_B = \sum_{j=1}^{N} X_j \qquad U(\beta) = e^{-i\beta H_B}$$

The Mixing Unitary

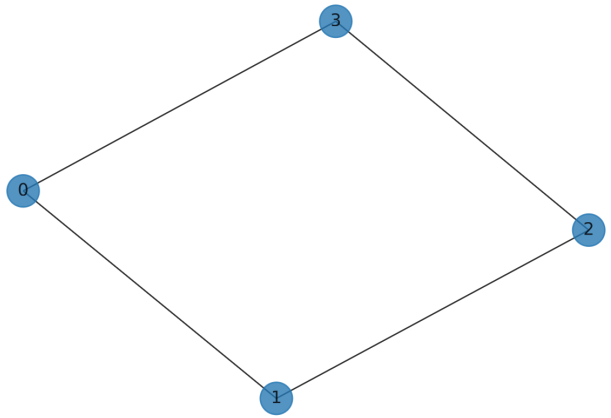
```
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister
from qiskit import Aer, execute
from qiskit.circuit import Parameter

# Adjacency is essentially a matrix which tells you which nodes are
# connected. This matrix is given as a sparse matrix, so we need to
# convert it to a dense matrix
adjacency = nx.adjacency_matrix(G).todense()

nqubits = 4

beta = Parameter("$\\beta$")
qc_mix = QuantumCircuit(nqubits)
for i in range(0, nqubits):
    qc_mix.rx(2 * beta, i)

qc_mix.draw()
```

try



Implementing QAOA

The circuits of H_C and H_B



$$H_C = \frac{1}{2}\sum_{\{i,j\}\in E}(1 - Z_i Z_j) \qquad U(\gamma) = e^{-i\gamma H_C}$$

Pauli matrices

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

$$Z|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle, \qquad Z|1\rangle = -|1\rangle$$

Note that

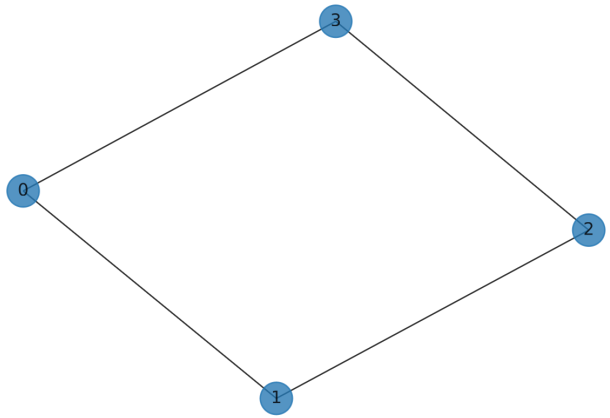
$$e^A|v\rangle = e^\lambda|v\rangle \qquad \text{if } A|v\rangle = \lambda|v\rangle$$

Since $Z|x\rangle = (-1)^x|x\rangle$,

$$e^{-i\gamma Z_i Z_j}|x_i x_j\rangle = e^{-i\gamma(x_i \oplus x_j)}|x_i x_j\rangle$$

Implementing QAOA

The circuits of H_C and H_B

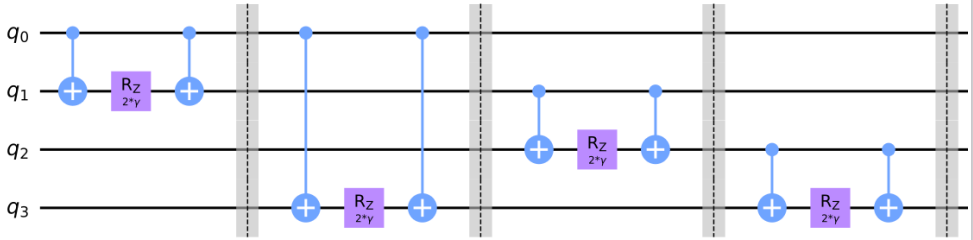


$$H_C = \frac{1}{2} \sum_{\{i,j\} \in E} (1 - Z_i Z_j) \qquad U(\gamma) = e^{-i\gamma H_C}$$

The Problem Unitary

```
gamma = Parameter("$\\gamma$")
qc_p = QuantumCircuit(nqubits)
for pair in list(G.edges()):  # pairs of nodes
    qc_p.rzz(2 * gamma, pair[0], pair[1])
    qc_p.barrier()

qc_p.decompose().draw()
```

try



Thank you!

—

eobae@kias.re.kr