# Instant Insanity Puzzle

# QAOA # Error mitigation

KHUbit

# Contents

# Introduction

Contemporary **Noisy Intermediate-Scale Quantum (NISQ)** devices
offer a new computational paradigm, but come with several limitations:

- **Limited qubit counts**
- **Short coherence times** and **gate errors**
- **Deep circuits** that rapidly accumulate noise

To work within these constraints while still leveraging quantum advantage, we propose:

- **Efficient Problem encoding & QAOA formulation**
- **Error-mitigation**

# Introduction

Why QAOA? :

- **Hardware efficiency:** QAOA uses only alternations of problem and mixer Hamiltonian evolutions, which can be tailored to the native gate set and connectivity of a given device.

- **Fixed, shallow depth:** By choosing small p, we keep circuit depth manageable.

Why Error mitigation? :

- **Accumulating gate and decoherence errors :** In QAOA, each layer p applies a large number of quantum gates. On real hardware, small errors at every gate (along with decoherence) build up, so the final measurement outcomes become heavily distorted.

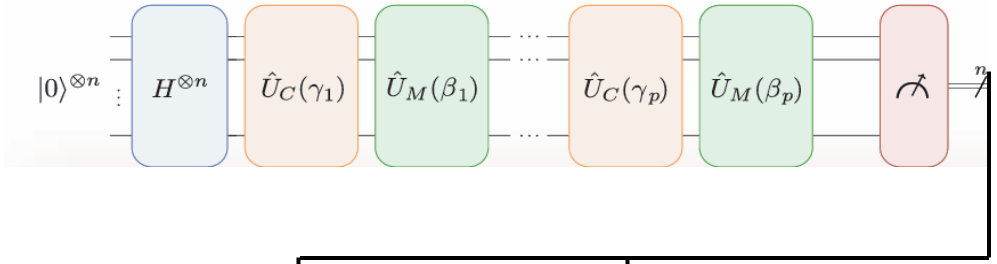
- **Need for precise expectation value estimation :** With high noise, the optimizer can veer off in the wrong direction or fail to converge altogether.

# Contents

# Our concept : (Using QAOA 3 times)

QAOA 1

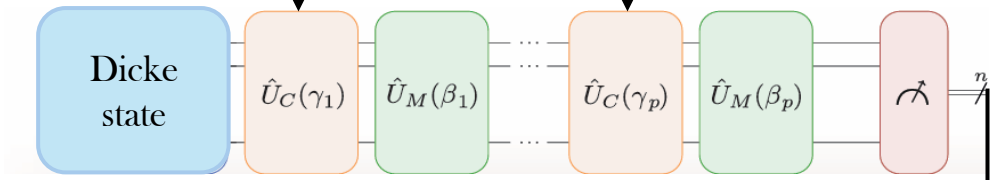
X-mixer




Stage 1: Run $QAOA_1$ to identify the **most promising subgraph structures**, then use its measurement outcomes to construct the cost Hamiltonian for $QAOA_2$.

QAOA 2

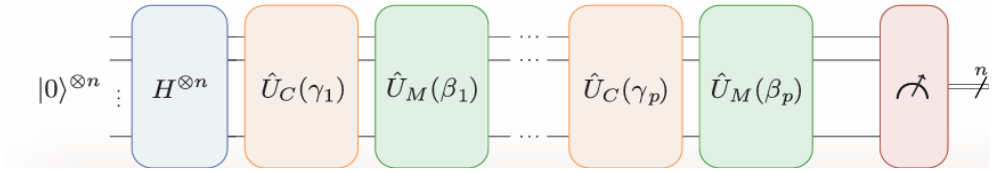
XY-mixer




Stage 2 : Run $QAOA_2$ to identify **the most promising edge**, then use its measurement outcomes to construct the cost Hamiltonian for $QAOA_3$.

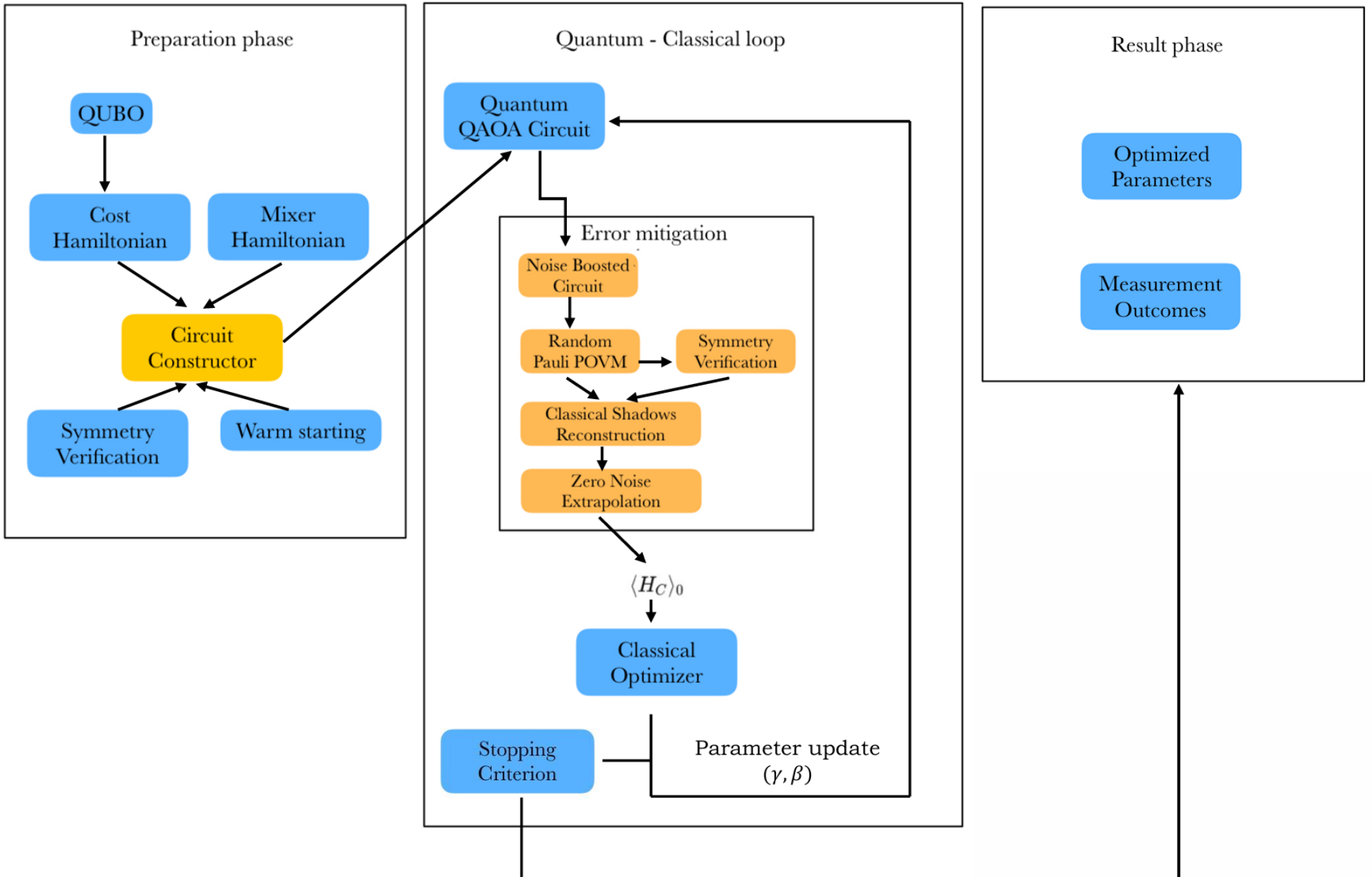
QAOA 3

X-mixer




Stage 3: Run $QAOA_3$ to identify **paired subgraph**, then we can find final solution for instant insanity puzzle.

# Our concept : (QAOA + Error mitigation)



# Contents

# How to solve Instant insanity puzzle
## – graph theory + QAOA

1. Draw a graph for each cube, connecting opposite faces

2. Put all 4 graphs from Step 1 into a total graph

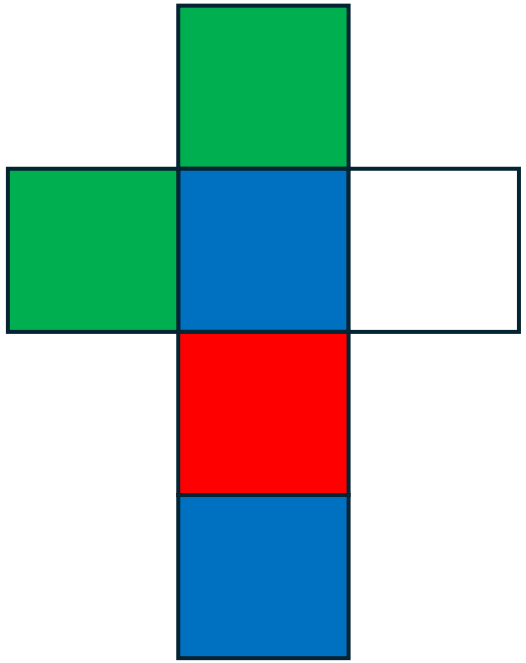
3. First QAOA to simplify total graph

4. Second QAOA to find subgraph

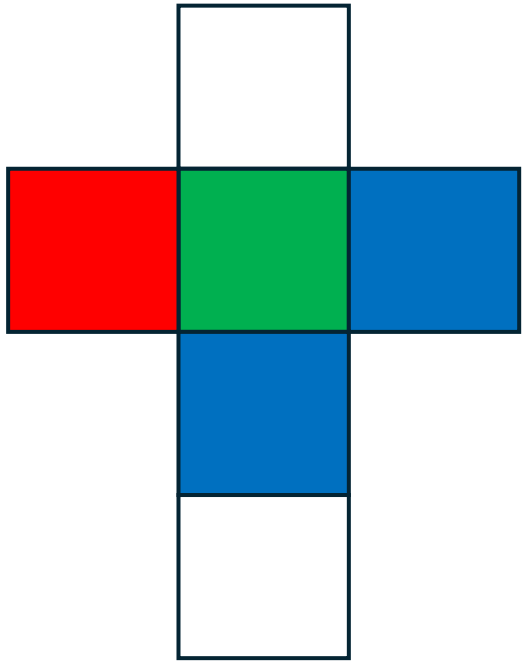
5. Third QAOA to find another subgraph

When we find two subgraphs (one for left/right, the other for top/bottom),
Puzzle is solved!

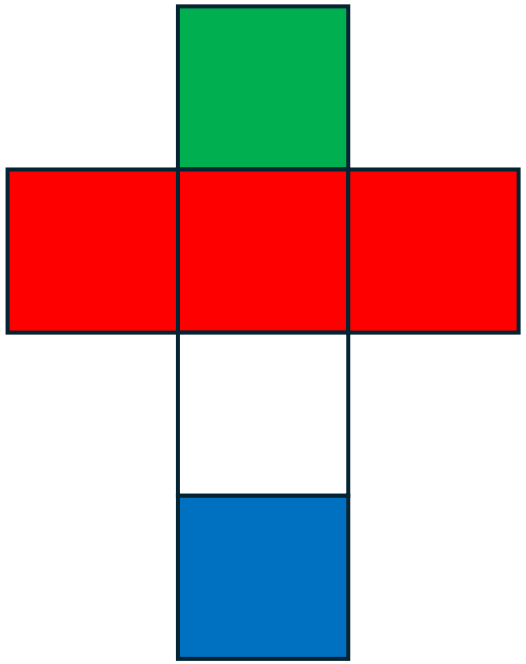
# Step 1 : Graph for each cube : O(n)



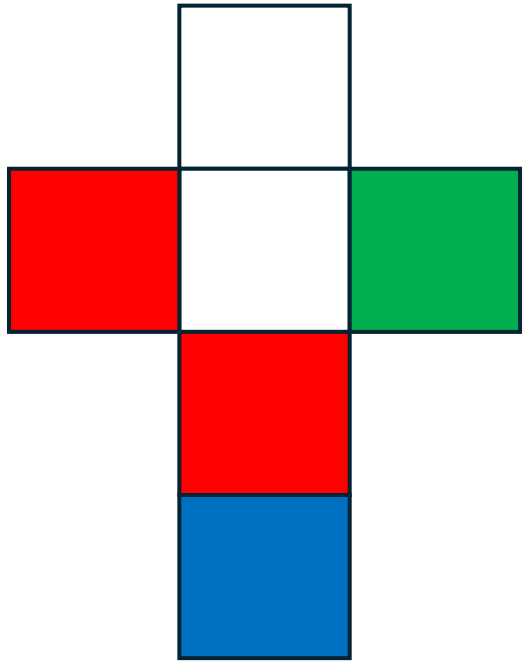
Cube 1



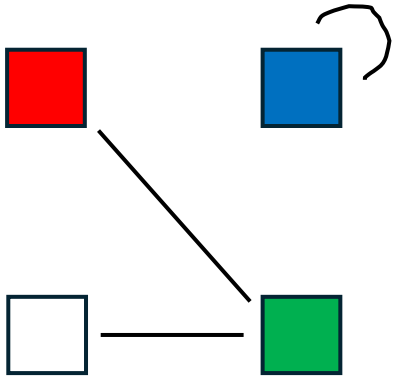
Cube 2

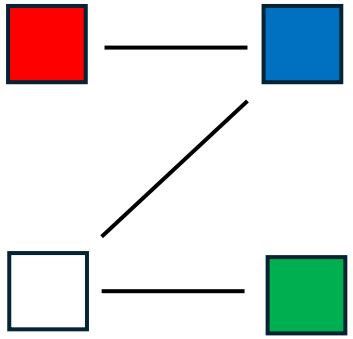


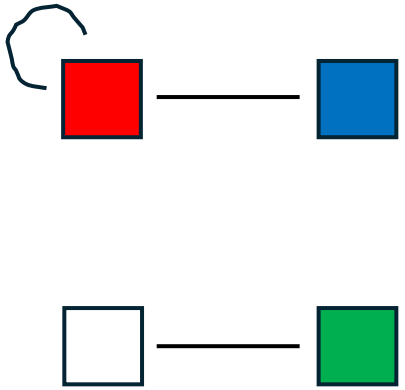
Cube 3

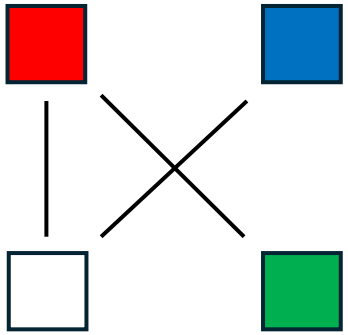


Cube 4

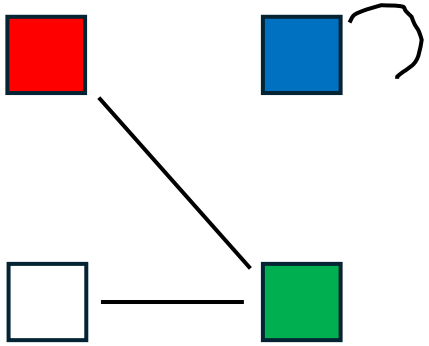




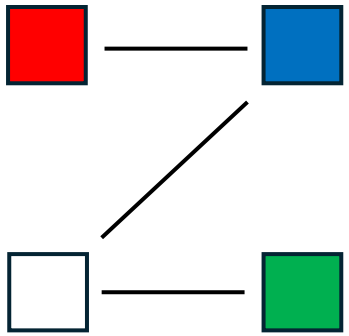




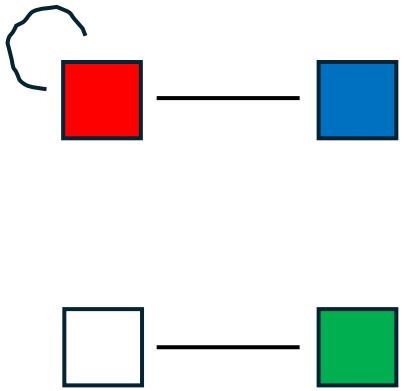
# Step 2 : Create 1 big graph : O(n)



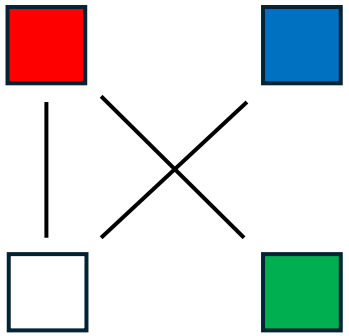
Cube 1



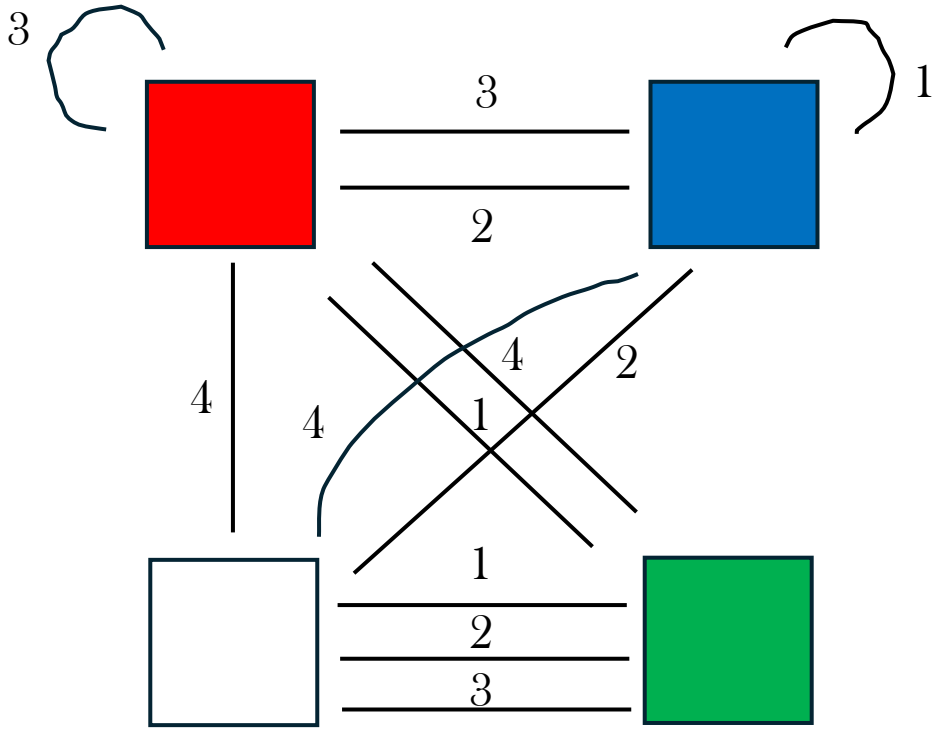
Cube 2



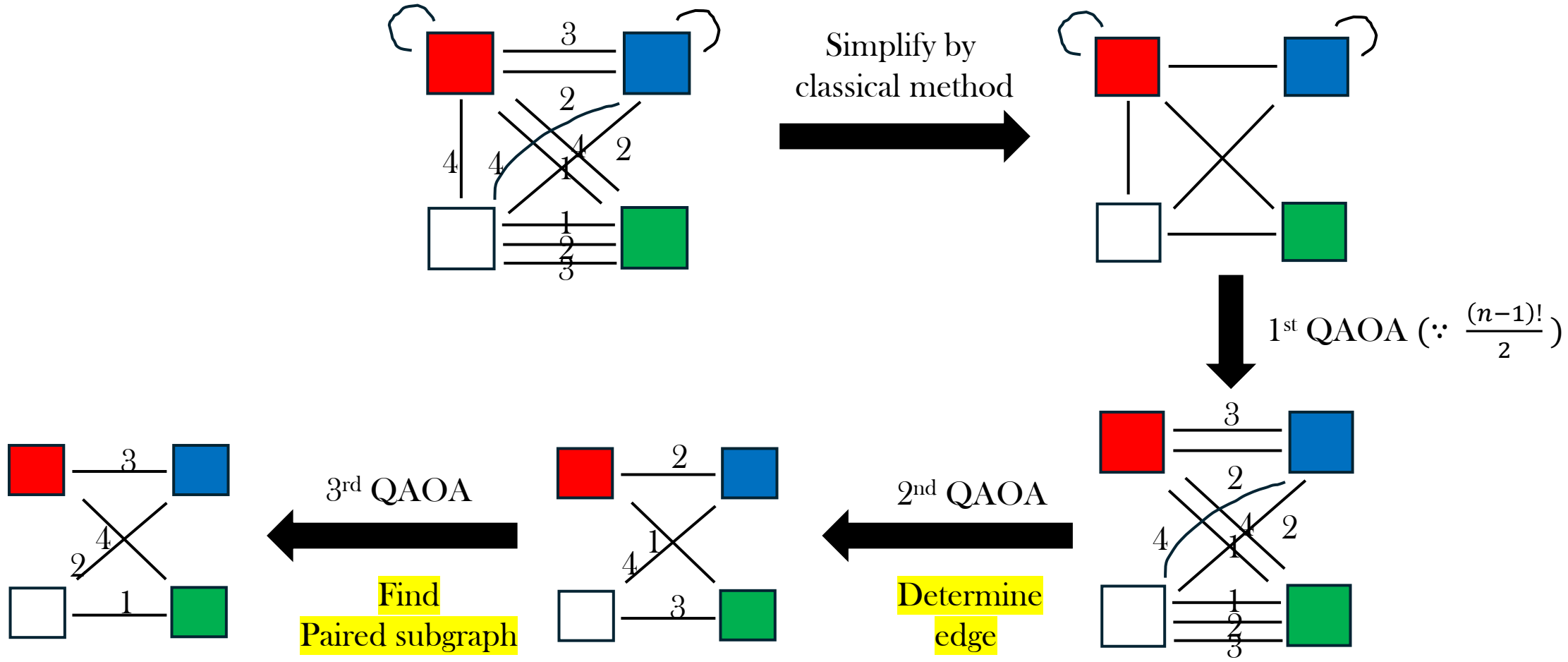
Cube 3



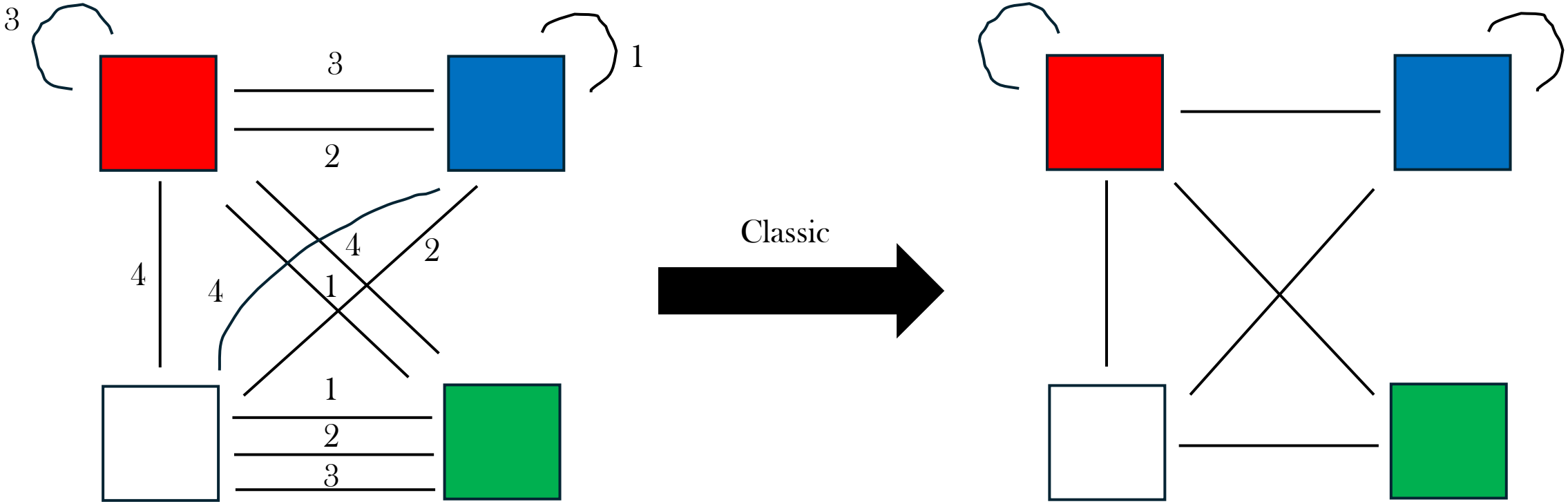
Cube 4



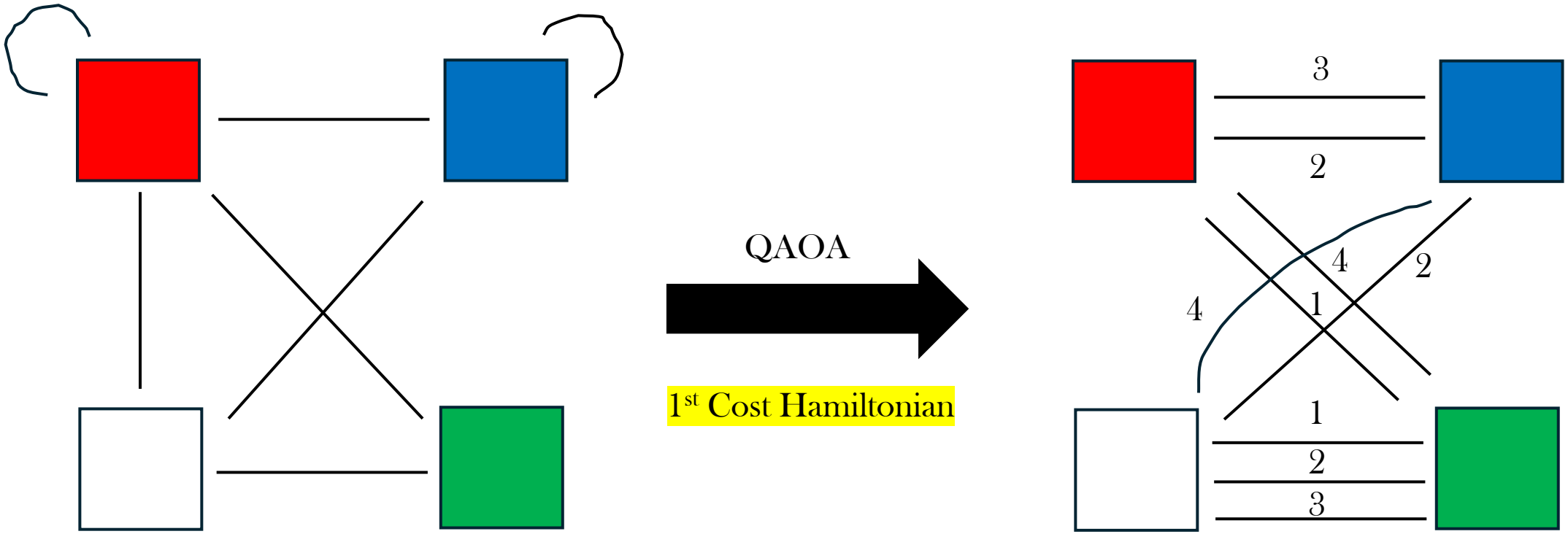
# # Our Idea : Overview



# # Simplify total graph

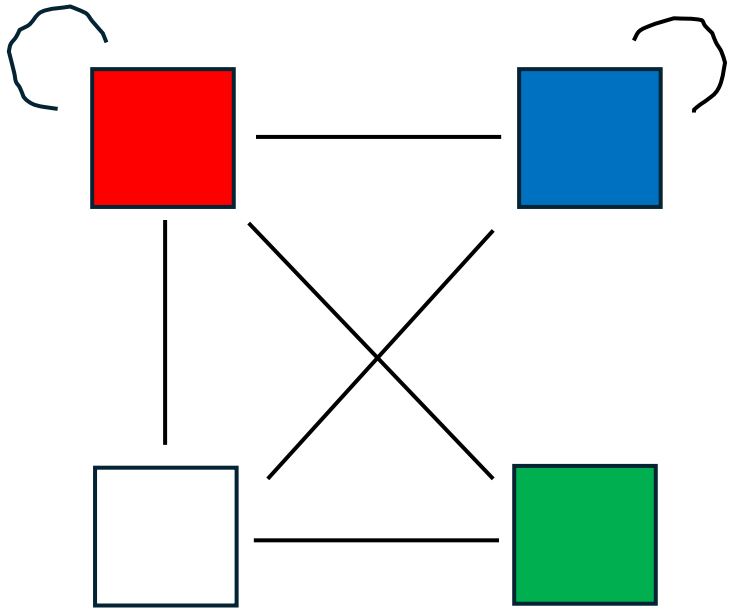


# # First QAOA




See Appendix A for details

# # 1st Cost Hamiltonian



$n \times (n-1)\ qubits \rightarrow$ $edge : 1\ , no\ edge : 0$

| <R> | | | <B> | | | <G> | | | <W> | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| B | G | W | R | G | W | R | B | W | R | B | G |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

Case 1 ) invalid edge : penalty

$$\sum_{e \notin E} \frac{1}{2}(I - Z_e)$$

Ex. G - B : penalty

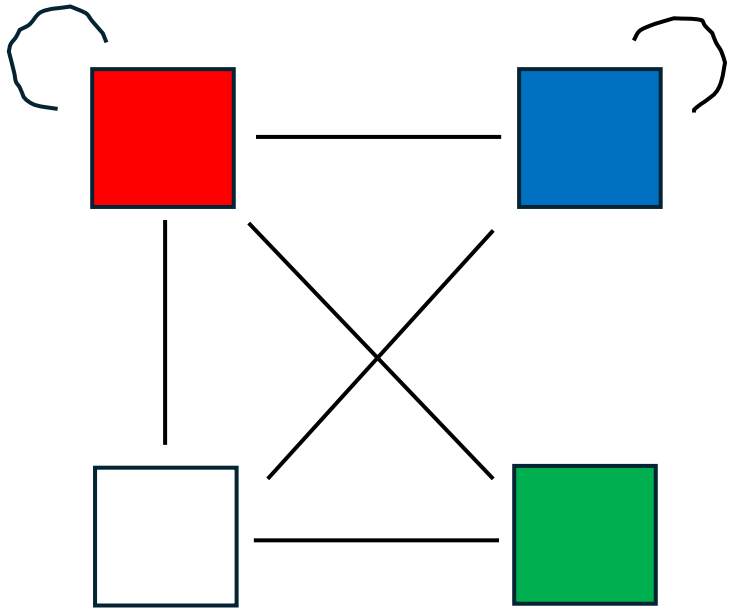
Case 2 ) nodes with a degree not equal to two : penalty

$$\left(\sum_{i=1}^{n-1} \frac{1}{2}(I - Z_i) - 2\right)^2 = \frac{1}{4}\left(\sum_{i=1}^{n-1}(I - Z_i)\right)^2 - 2\sum_{i=1}^{n-1}(I - Z_i) + 4$$

Ex. 000 : 4 , 001 : 1 , 010 : 1 , 011 : 0

$$= \frac{1}{4}\left(\sum_{i=1}^{n-1}(I - Z_i)^2 + \sum_{i \neq j=1}^{n-1}(I - Z_i)(I - Z_j)\right) - 2\sum_{i=1}^{n-1}(I - Z_i) + 4 = -\frac{3}{2}\sum_{i=1}^{n-1}(I - Z_i) + \frac{1}{2}\sum_{i \neq j=1}^{n-1}(I - Z_i)(I - Z_j) + 4$$

# # 1st Cost Hamiltonian



$n \times (n-1)\ qubits \rightarrow$ $edge : 1\ , no\ edge : 0$

| <R> | | | <B> | | | <G> | | | <W> | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| B | G | W | R | G | W | R | B | W | R | B | G |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

Case 3 ) Same pair, different value : penalty (graph is undirected)

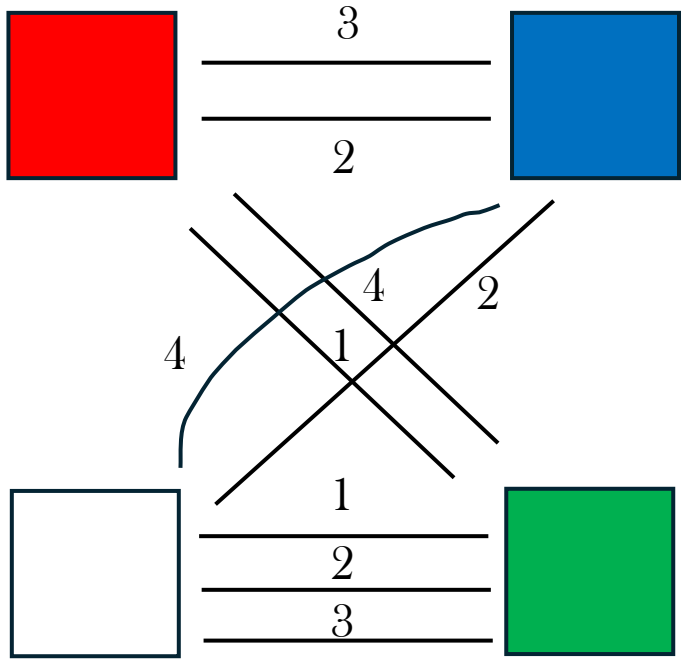
$$\sum_{i \neq j=1}^{4} \frac{1}{2}(I - Z_i Z_j)$$

Ex. R - G : 1 , G - R : 0

Circuit Depth of 1st Cost Hamiltonian:

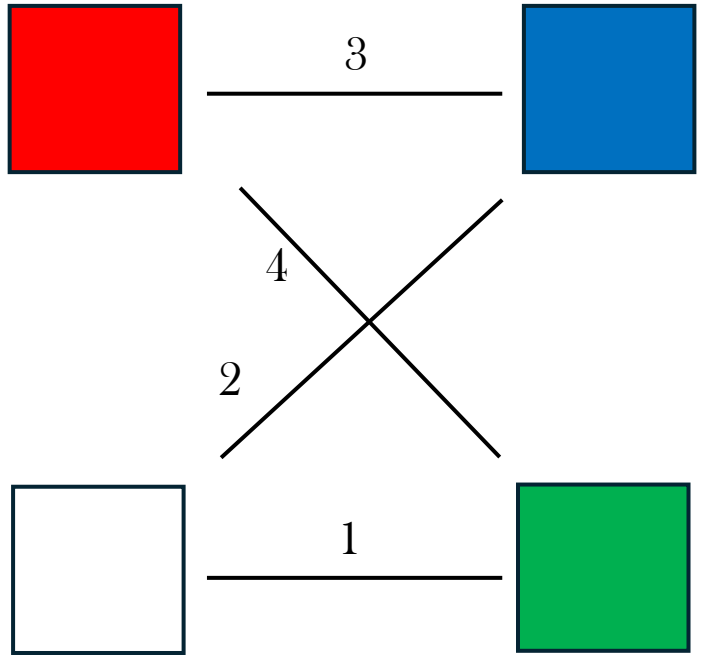
Case 1 : O(1) , Case 2 : O($n^2$) , Case 3 : O($n^2$)

# # Second QAOA



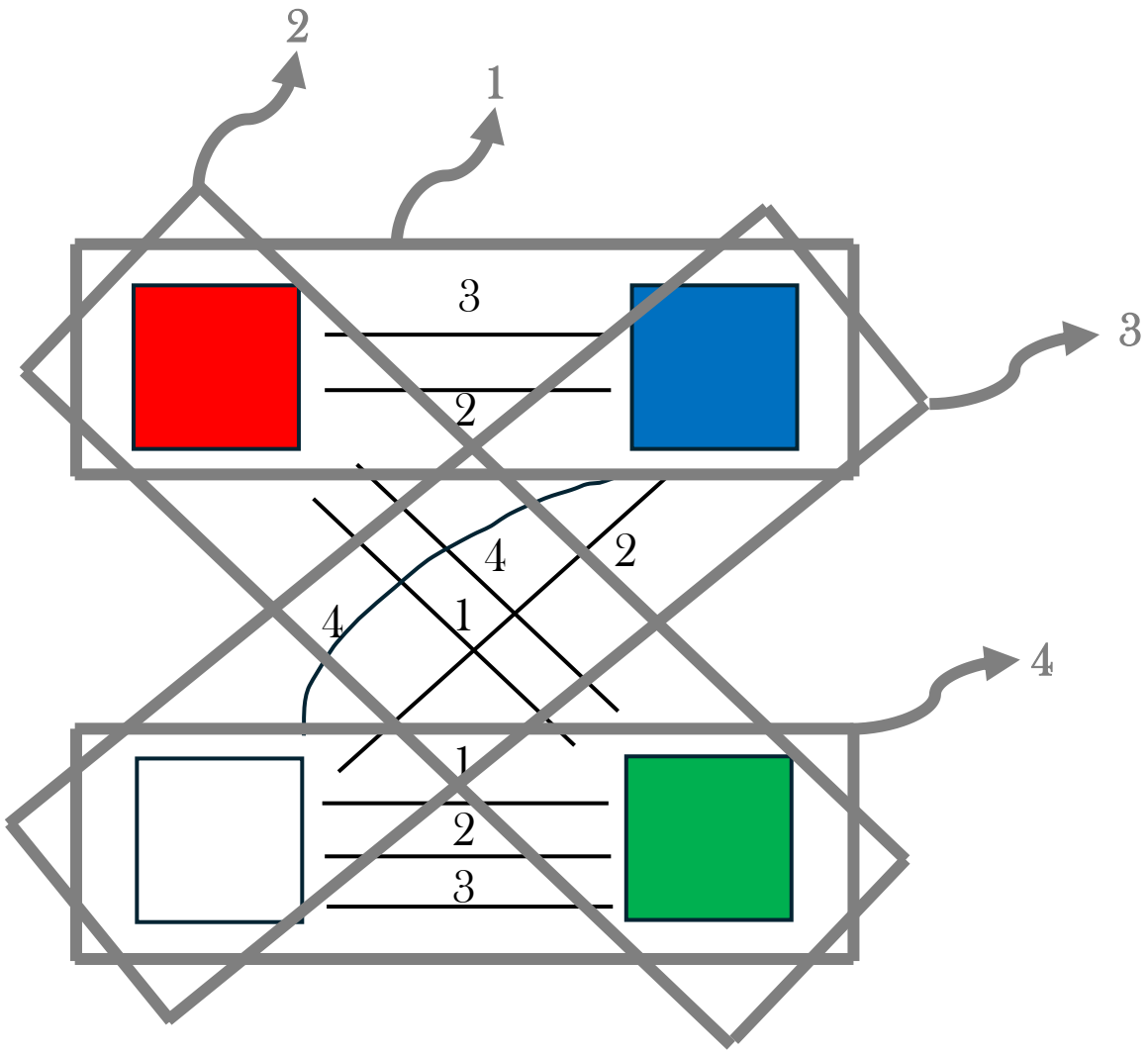

QAOA (Finding single subgraph)

2nd Cost Hamiltonian




See Appendix B for details

# # Second QAOA – Qubit encoding




1 2 3 4

( )|( )|( )|( )

$n \log n$ *qubits*

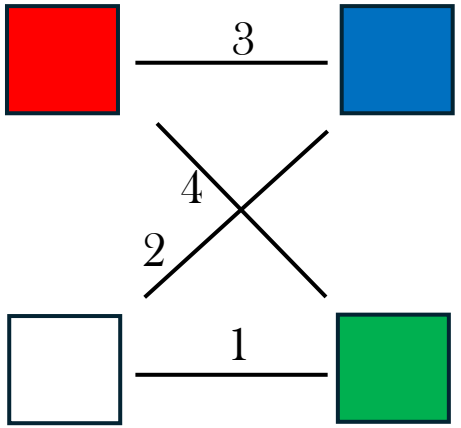
$00 \rightarrow 1\ cube\ (edge)$

$01 \rightarrow 2\ cube\ (edge)$

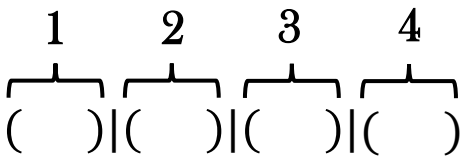
$10 \rightarrow 3\ cube\ (edge)$

$11 \rightarrow 4\ cube\ (edge)$

*If result is* ( 10 )|( 11 )|( 01 )|( 00 ),



# # Second QAOA – Using Dicke State



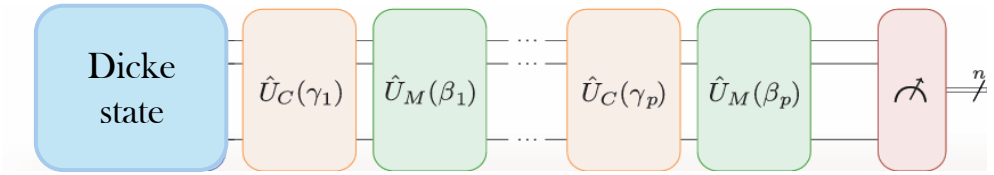

$00 \rightarrow 1\ cube\ (edge)$

$01 \rightarrow 2\ cube\ (edge)$

$10 \rightarrow 3\ cube\ (edge)$

$11 \rightarrow 4\ cube\ (edge)$

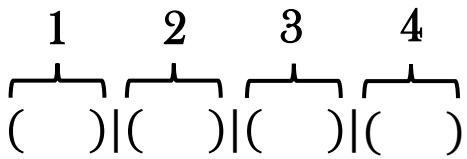
QAOA 2




Because our 8-qubit must always contain exactly four 1's and four 0's, we initialize the QAOA in the Dicke state that ensures all subsequent evolution remains within that fixed-weight subspace.

$$|D_k^n\rangle = \frac{1}{\sqrt{\binom{n}{k}}} \sum_{\substack{x\in\{0,1\}^n \\ |x|=k}} |x\rangle \quad \longrightarrow \quad |D_4^8\rangle = \frac{1}{\sqrt{70}} \sum_{|x|=4} |x\rangle.$$

# # Second QAOA – Using Dicke State




$00 \rightarrow 1\ cube\ (edge)$

$01 \rightarrow 2\ cube\ (edge)$

$10 \rightarrow 3\ cube\ (edge)$

$11 \rightarrow 4\ cube\ (edge)$

In our Dicke-state QAOA, we no longer use the "standard" single qubit X–mixer :

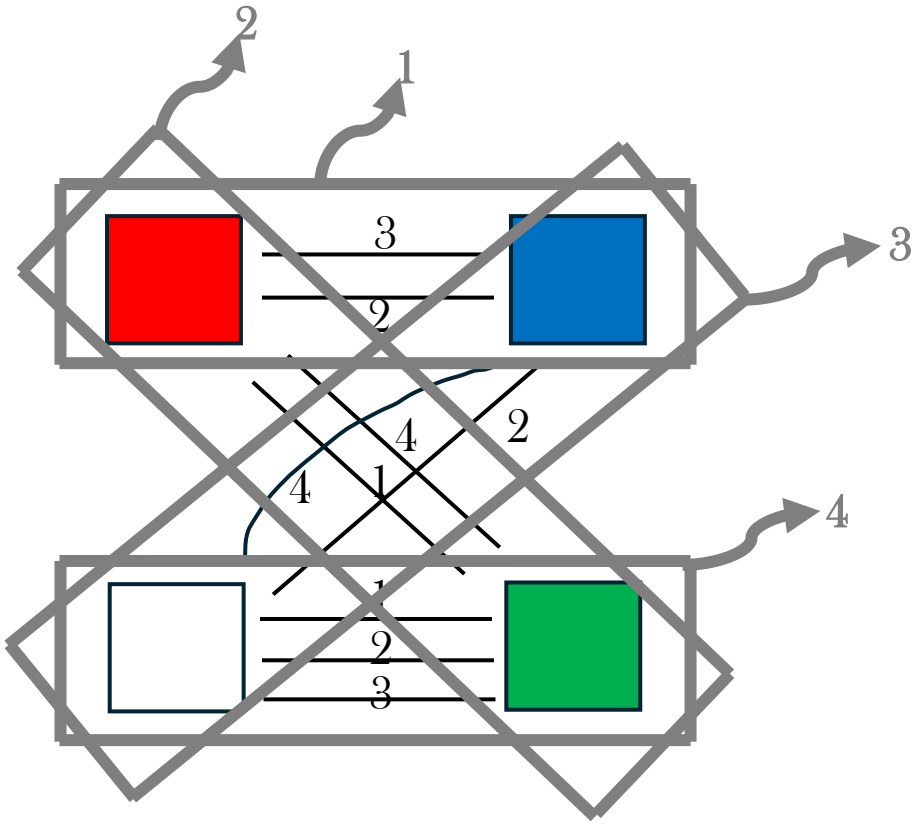
$$H_M^{(X)} = \sum_i X_i,$$

Single qubit X-mixer does not preserve Hamming weight and would leak us out of our Dicke subspace.

Instead, we choose an "XY-mixer" whose Hamiltonian commutes with the total-Z operator

$$H_M^{(\mathrm{XY})} = \sum_{i<j} (X_i X_j + Y_i Y_j)$$

# # 2nd Cost Hamiltonian




Case 1 ) For each group, invalid edge : penalty

$$H_{C1} = \underbrace{\frac{1}{2}(I + Z_1Z_2)}_{\text{Group 1 Penalty}} + \underbrace{\frac{1}{2}(I - Z_3Z_4)}_{\text{Group 2 Penalty}} + \underbrace{\frac{1}{2}(I + Z_6)}_{\text{Group 3 Penalty}} + \underbrace{\frac{1}{4}(I - Z_7)(I - Z_8)}_{\text{Group 4 Penalty}}$$

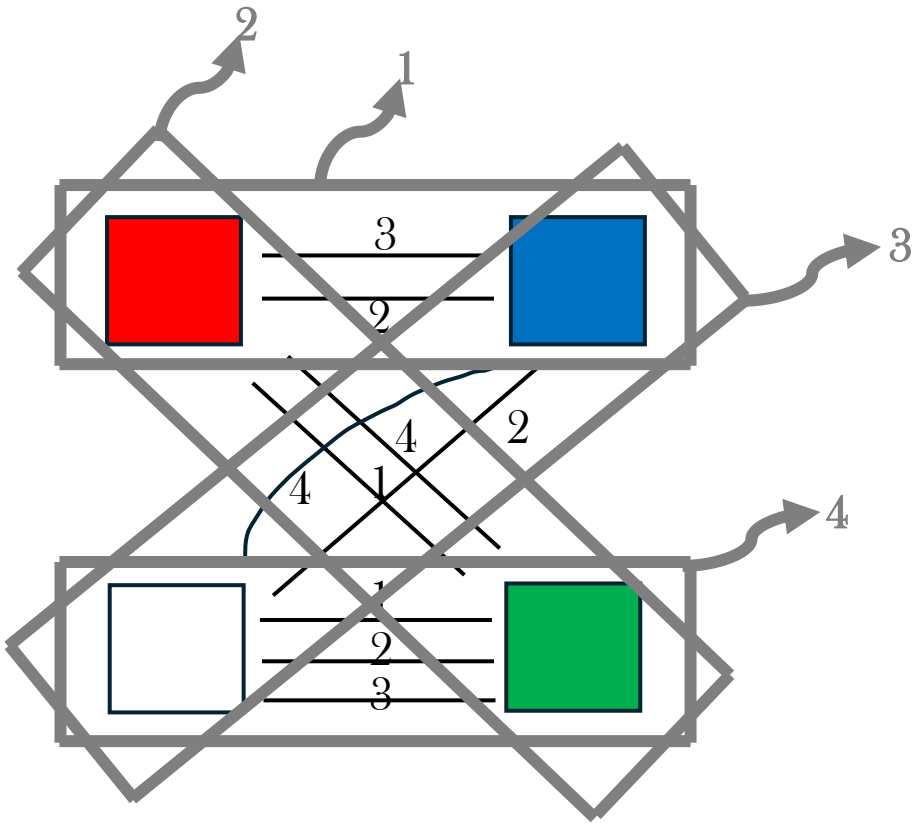
In group 1, valid edge is <2> and <3>
So, edge 1 (00) and edge (11) should be penalized.

In group 2, valid edge is <1> and <4>
So, edge 2 (01) and edge (10) should be penalized.

In group 3, valid edge is <2> and <4>
So, edge 1 (00) and edge 3 (10) should be penalized.

In group 4, valid edge is <1> , <2> and <3>
So, edge 4 (11) should be penalized.

# # 2nd Cost Hamiltonian




Case 2 ) Duplicated edge : penalty

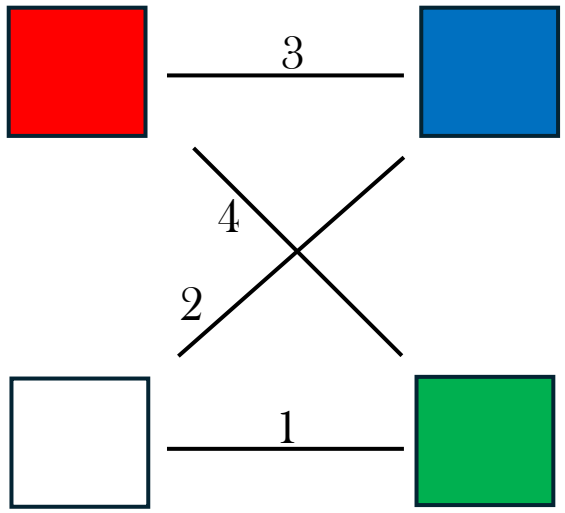
$$H_{C2} = \sum_{0 \le i < j \le 3} \frac{1}{4}\left(I + Z_{2i}Z_{2j}\right)\left(I + Z_{2i+1}Z_{2j+1}\right)$$

It can penalize any two group being equal.

Circuit Depth of 2nd Cost Hamiltonian:

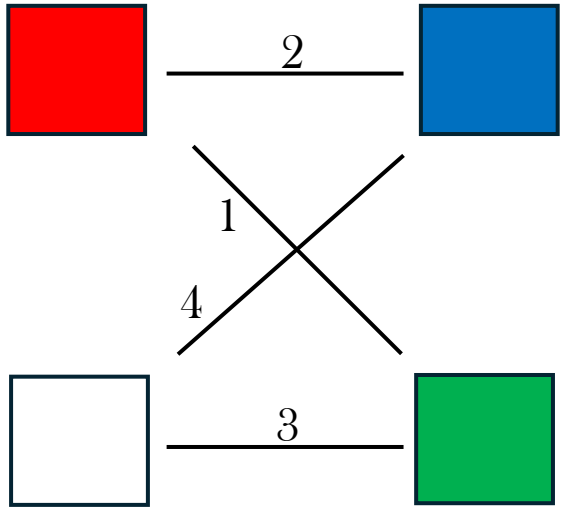
Case 1 : O(1) , Case 2 : O($n^2$)

# # Third QAOA

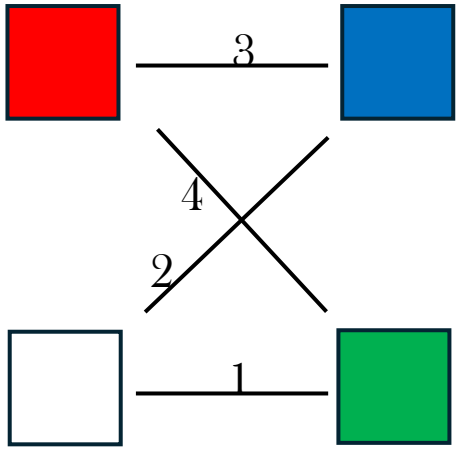



QAOA (Finding paired subgraph)

3rd Cost Hamiltonian



# # 3rd Cost Hamiltonian

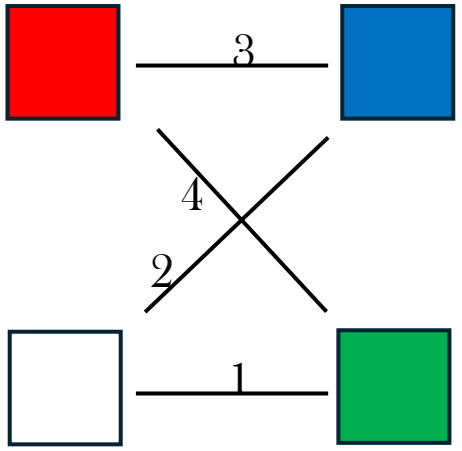



The basic cost Hamiltonian is same with 2nd cost Hamiltonian.
Because Both QAOA 2 and QAOA 3 find subgraph.

But we need to penalty QAOA 2's result.
Because two subgraph should not be overlapped.

More details, next page.

# # 3rd Cost Hamiltonian




If the Second QAOA's output is 10110100 (single subgraph),
We construct cost Hamiltonian based on that output. (We need to find two subgraph)

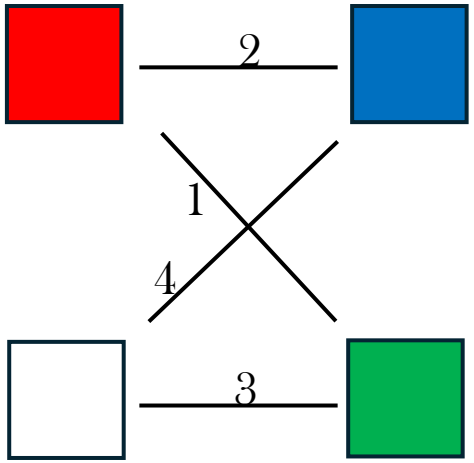
Case 1) Overlap with 10110100 : penalty

$$H_C = \frac{1}{4}(I - Z_1)(I + Z_2) + \frac{1}{4}(I - Z_3)(I + Z_4) + \frac{1}{4}(I + Z_5)(I - Z_6) + \frac{1}{4}(I + Z_7)(I + Z_8)$$

Circuit Depth of 3rd Cost Hamiltonian:

$O(n^2)$

# # 3rd Cost Hamiltonian




If the Second QAOA's output is 01001110 (single subgraph),
We construct cost Hamiltonian based on that output. (We need to find two subgraph)

Case 2) Overlap with 01001110 : penalty

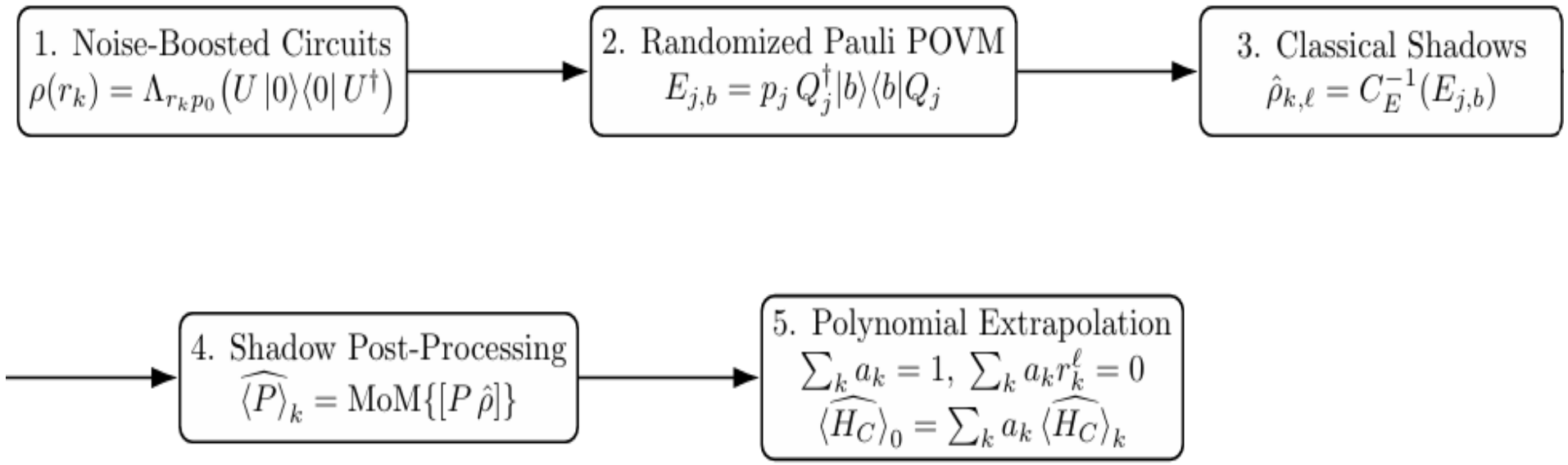
$$H_C = \frac{1}{4}(I + Z_1)(I - Z_2) + \frac{1}{4}(I + Z_3)(I + Z_4) + \frac{1}{4}(I - Z_5)(I - Z_6) + \frac{1}{4}(I - Z_7)(I + Z_8)$$

Circuit Depth of 3rd Cost Hamiltonian:

$O(n^2)$

# Contents

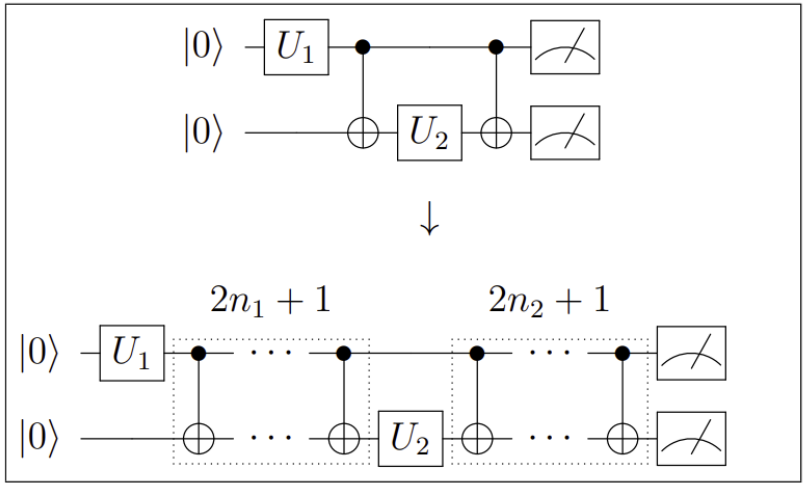
# Our Error mitigation details



# # 1. Noise-Boosted Circuits

· Step 1 : Prepare our QAOA circuit

$$\mathcal{U}(\gamma,\beta)=\prod_{l=1}^{p}e^{-i\beta_l H_M}e^{-i\gamma_l H_C}$$




· Step 2 : Determine Noise Boosting method (By CNOT gate)

- **By inserting multiple CNOT gates into circuit, the inherent hardware noise can be boosted.**
- **The number of inserted CNOT gates determines the noise scaling factor $r_k$.**

· Step 3 : Noise Boosting

$$p_0 \to r_k p_0, \quad (r_0 = 1 < r_1 < r_2 < \cdots < r_m)$$

· Step 4 : Prepare the state by applying Boosted noise

$$\rho(r_k) = \Lambda_{r_k p_0}\left(\mathcal{U}(\gamma,\beta)|0\rangle\langle 0|\mathcal{U}^{\dagger}(\gamma,\beta)\right)$$

# # 2. Randomized Pauli POVM

· Step 1 : For each shot, select $Q_j$ randomly

Ex. Qubit number (N) : Possible Pauli Operator set will be selected randomly

- **Shot 1 :** $Q_j = X \otimes Y \otimes \cdots \otimes Z$
- **Shot 2 :** $Q_j = Z \otimes Z \otimes \cdots \otimes X$
- …

· Step 2 : Pauli basis measurement after our QAOA circuit

- **Add rotation at the end of the circuit according to the selected Pauli basis**
- **Obtain the measurement results for each qubit like** $(b = b_1 b_2 \ldots b_N)$

· Step 3 : After measurement, represent the elements of the POVM measurement operators.

$$E_{j,b} = p_j Q_j^\dagger |b\rangle\langle b| Q_j$$

# # 3. Classical Shadows

· Step 1 : Prepare the results obtained in step 2

The measurement results are stored for each individual shot.

Ex.
- **Shot 1 :** $E_{j1,b1}$
- **Shot 2 :** $E_{j2,b2}$

· Step 2 : Define the Classical Shadow Channel

$C_E(\rho) = \sum_l \mathrm{Tr}[\rho E_l] E_l$ The channel is invertible if $\{E_l\}$ spans the observable space.

· Step 3 : Shadow reconstruction via the inverse channel

$\hat{\rho}_{k,l} = C_E^{-1}(E_{j,b})$ By using the inverse channel, the real state can be approximately reconstructed from the measurement state.

# # 4. Shadow Post-Processing

· Step 1 : Prepare the result of classical shadow

Noise-boosted circuit → Randomized Pauli measurement → The set of states obtained after classical shadow

$\{\hat{\rho}_{k,1}, \ldots, \hat{\rho}_{k,L}\}$ This state is prepared for each noise level $r_k$

· Step 2 : Prepare Our Cost Hamiltonian (Construct this using Pauli terms)

$$\hat{H}_C = \sum_{i<j} J_{ij}\hat{Z}_i\hat{Z}_j + \sum_i h_i\hat{Z}_i \qquad P = Z_iZ_j \text{ or } Z_i$$

· Step 3 : Using Classical shadow state to calculate expectation value

$\mathrm{Tr}[P\hat{\rho}_{k,l}]$

· Step 4 : Make an estimator of the expectation value of P (Cost Hamiltonian's Pauli term) – Using median of means

$$\langle\hat{P}\rangle_k = \text{Median of Mean}\{\mathrm{Tr}[P\hat{\rho}_{k,l}]\}_{\in S_k^{sym}}$$

# # 5. Polynomial Extrapolation

The expectation values obtained through the preceding steps are influenced by noise.

In practice, directly measuring the ideal quantum state is difficult or often impossible.

Therefore, Polynomial Extrapolation is used to mathematically estimate the noise-free result.

· Step 1 : Obtain expectation values that calculate various noise level

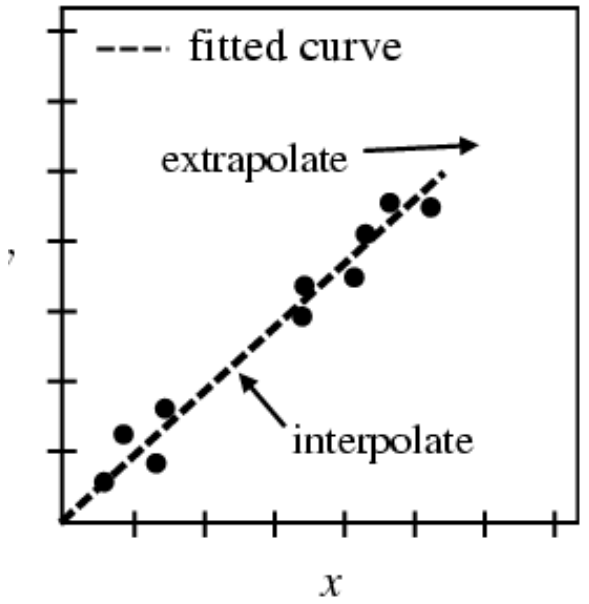
$$\{\langle \hat{H}_C \rangle_1, \langle \hat{H}_C \rangle_2, \ldots, \langle \hat{H}_C \rangle_K\}$$

· Step 2 : By Richardson extrapolation, determine coefficient

$$\sum_k a_k = 1, \qquad \sum_k a_k (r_k)^l = 0$$

· Step 3 : Estimate Final noise-free expectation value

$$\langle \hat{H}_C \rangle_{r=0} = \sum_k a_k \langle \hat{H}_C \rangle_k$$



# # Summary of our error mitigation

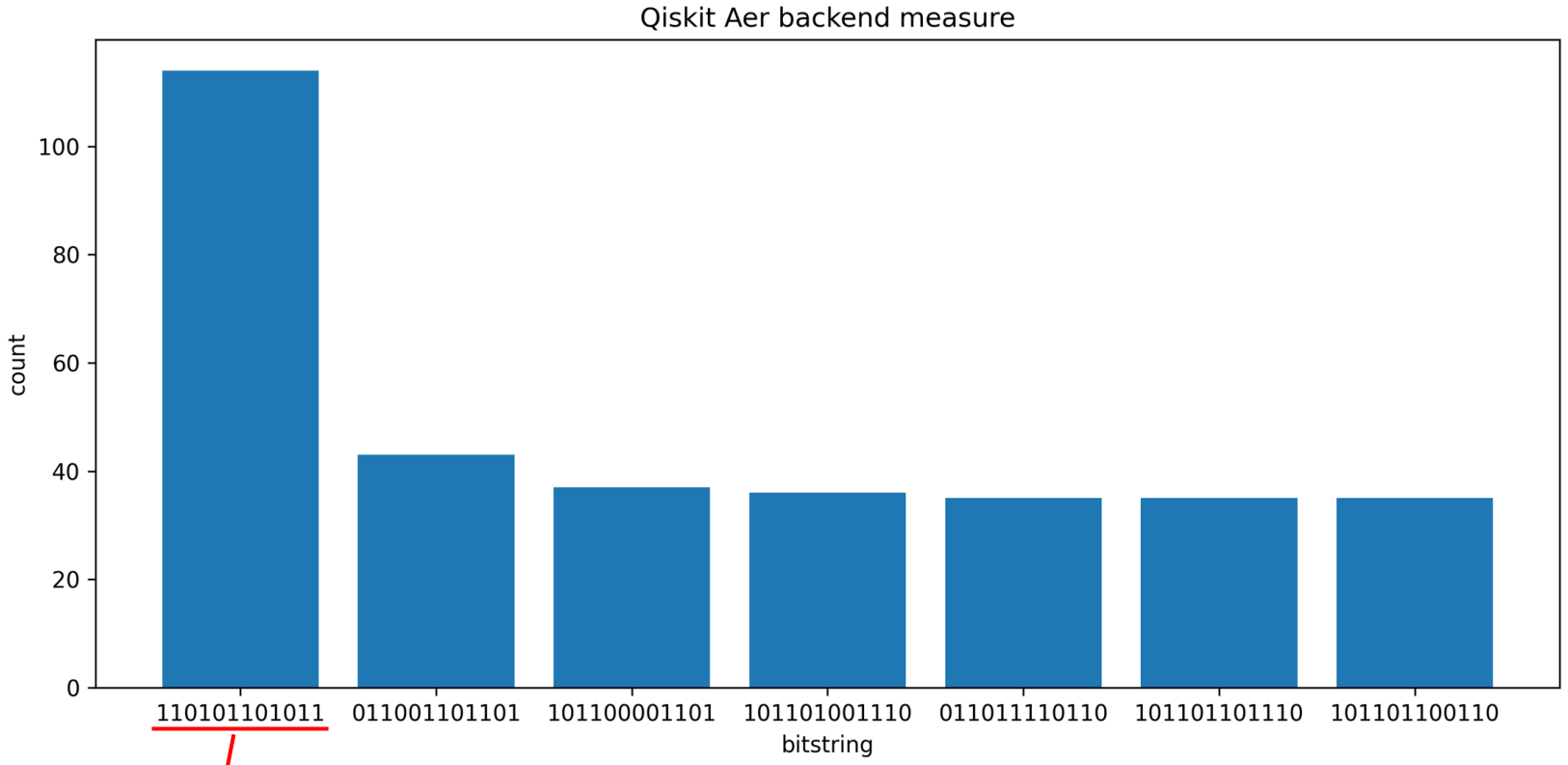
The goal is to estimate the QAOA cost Hamiltonian expectation value with high accuracy, despite hardware noise, and recover results that closely resemble those from an ideal quantum device.

Even though “all the measurements are taken from noisy quantum circuits”,
the combination of two idea( ZNE & Classical shadow) enables us to effectively recover
the **ideal expectation value** of the QAOA cost Hamiltonian.

As a result, QAOA optimization can proceed **accurately and reliably** even on **noisy hardware**, as if it were running on a noise-free quantum device.

# Contents

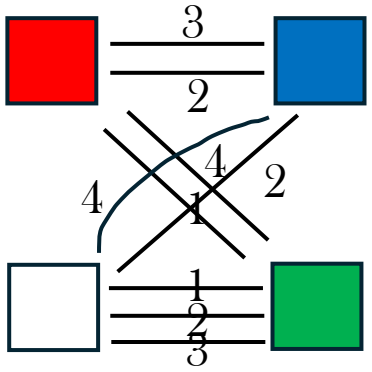
# First QAOA result



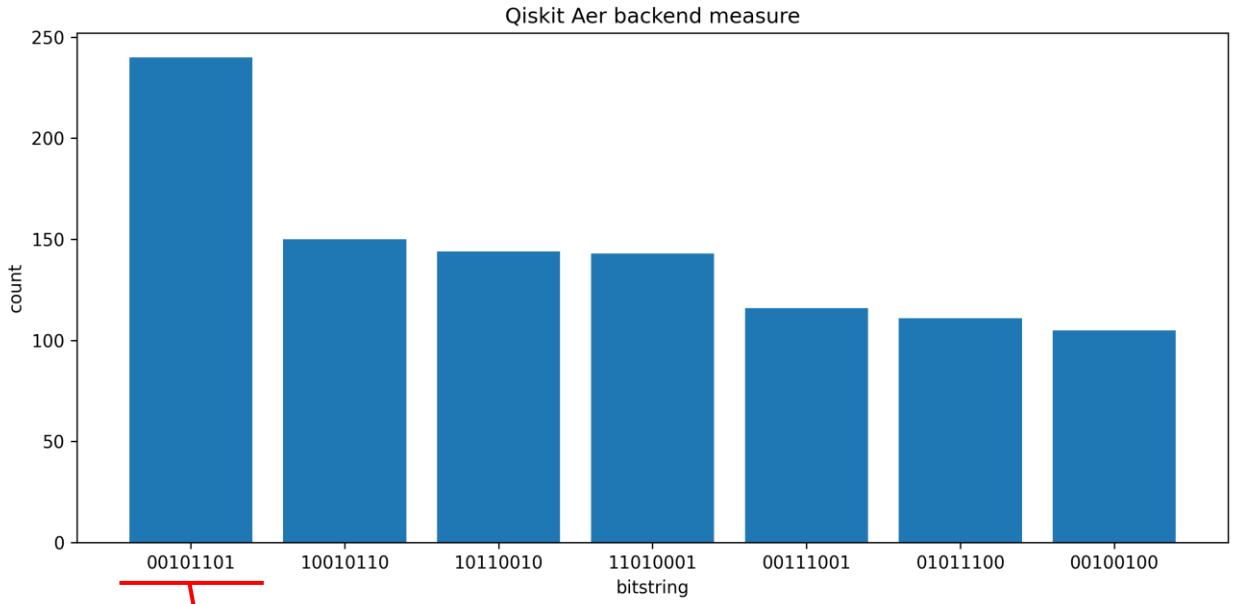

Determine
Subgraph “shape”

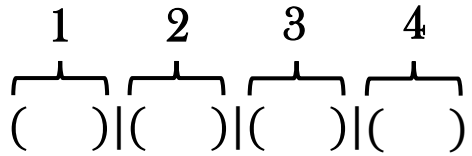
110 101 101 011

| <R> | <B> | <G> | <W> |
|---|---|---|---|
| B G W | R G W | R B W | R B G |



# # Second QAOA result (2-1)



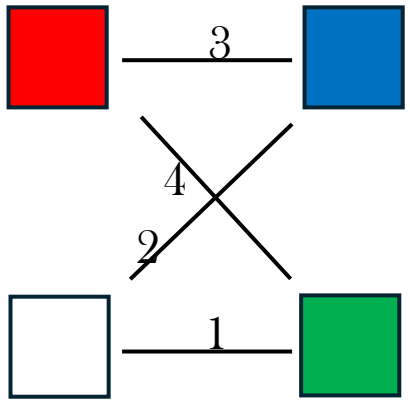

$00 \rightarrow 1\ cube\ (edge)$

$01 \rightarrow 2\ cube\ (edge)$

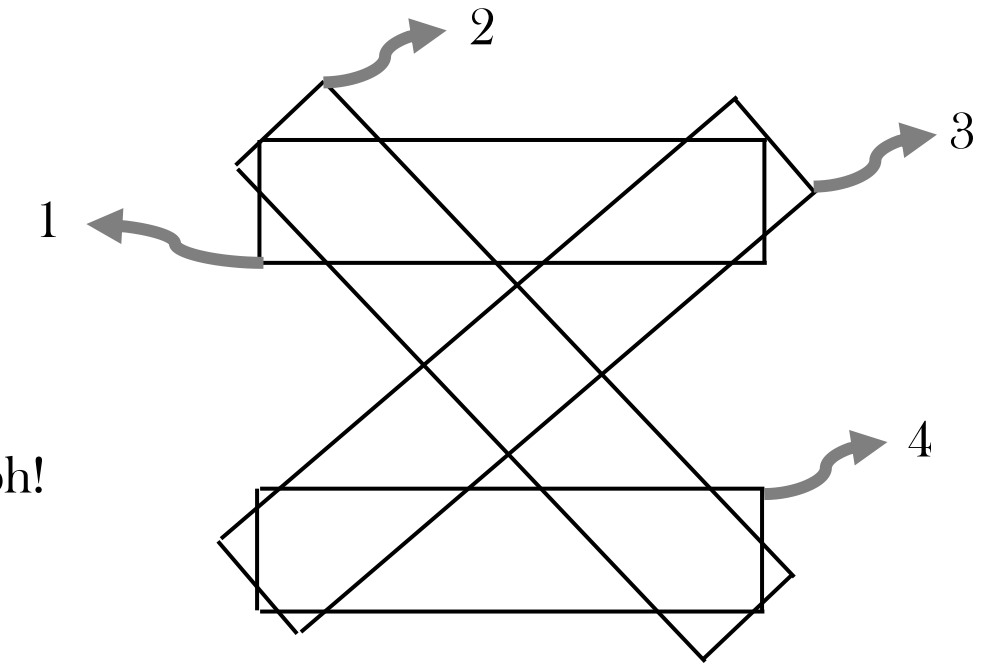
$10 \rightarrow 3\ cube\ (edge)$

$11 \rightarrow 4\ cube\ (edge)$

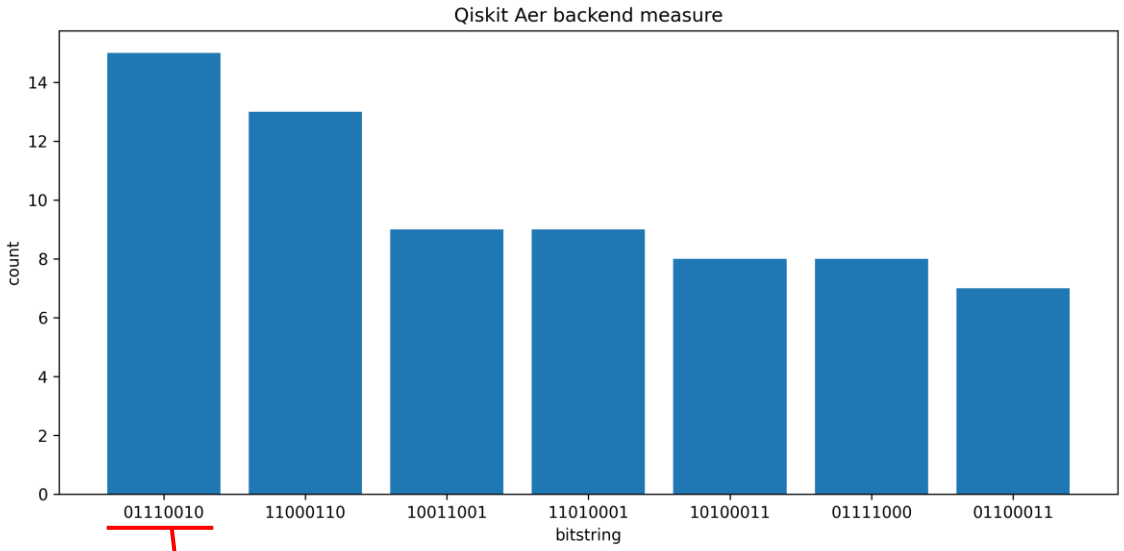
10 11 01 00 →

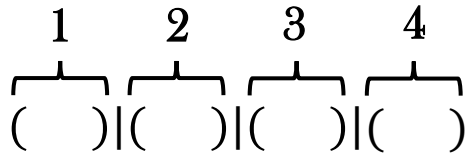



First subgraph!



# # Second QAOA result (2-2)



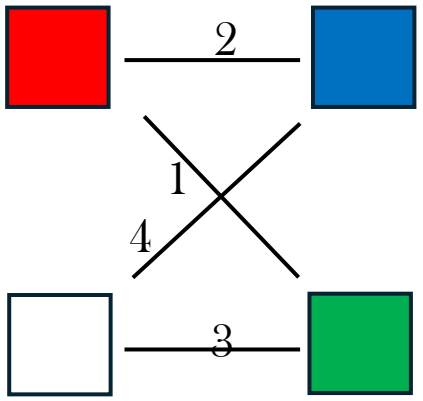

$00 \rightarrow 1\ cube\ (edge)$

$01 \rightarrow 2\ cube\ (edge)$

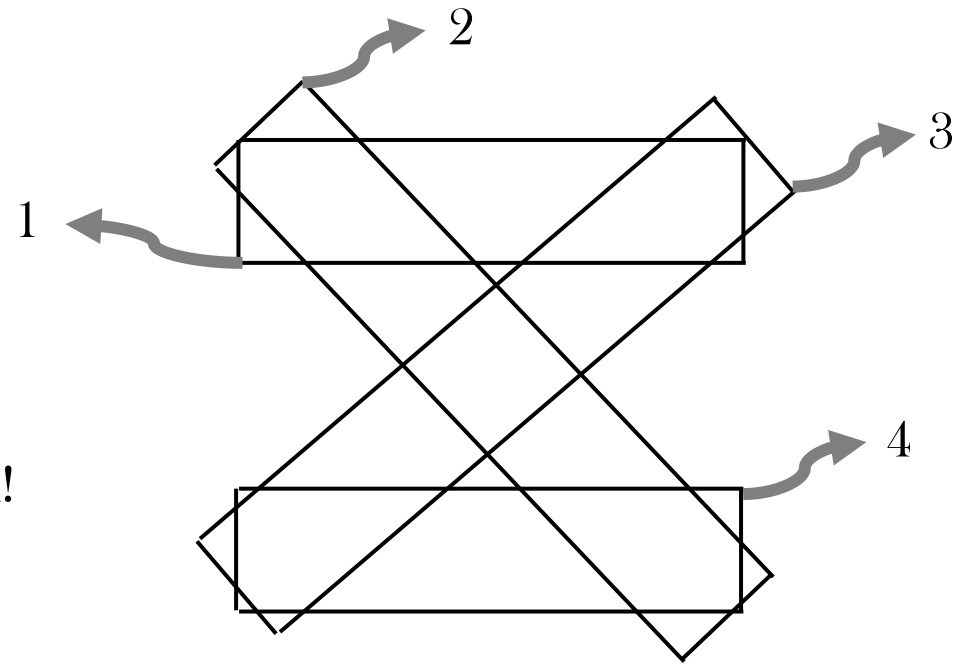
$10 \rightarrow 3\ cube\ (edge)$

$11 \rightarrow 4\ cube\ (edge)$

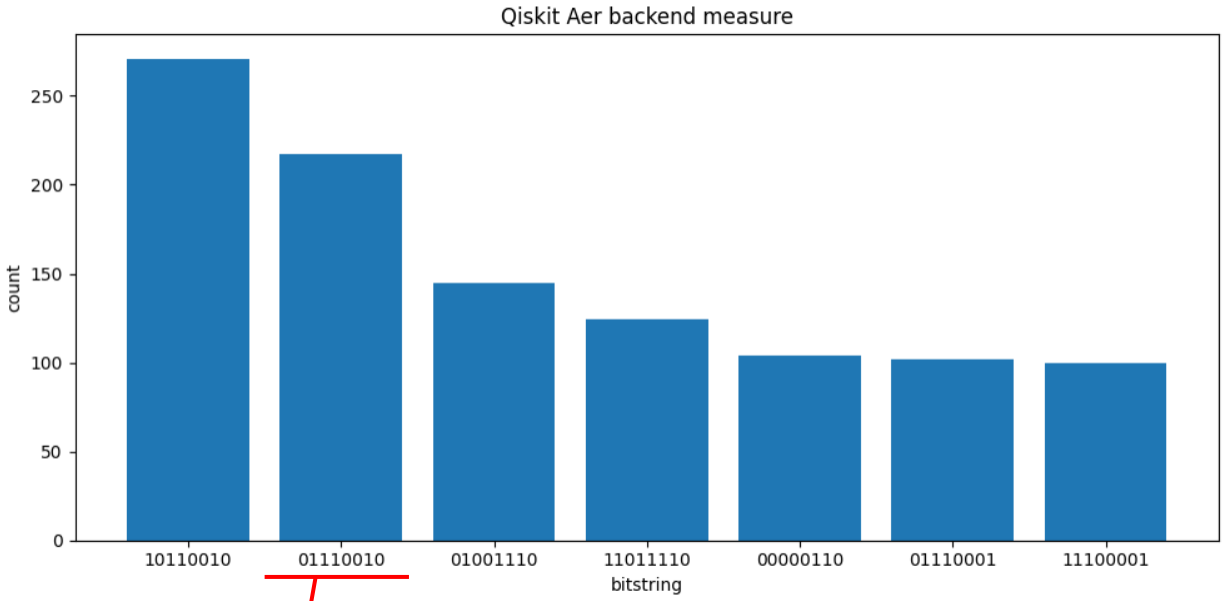
01 00 11 10 →

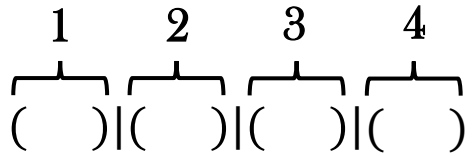



First subgraph!



# # Third QAOA result (3-1) : penalty 2-1 result



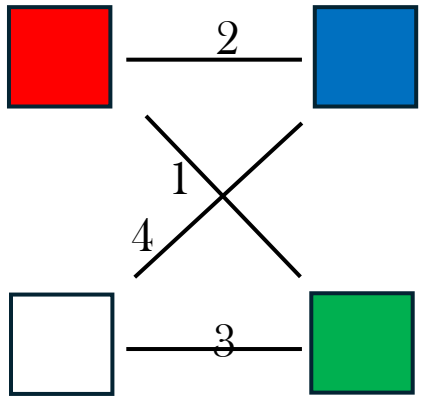

$00 \rightarrow 1\ cube\ (edge)$

$01 \rightarrow 2\ cube\ (edge)$

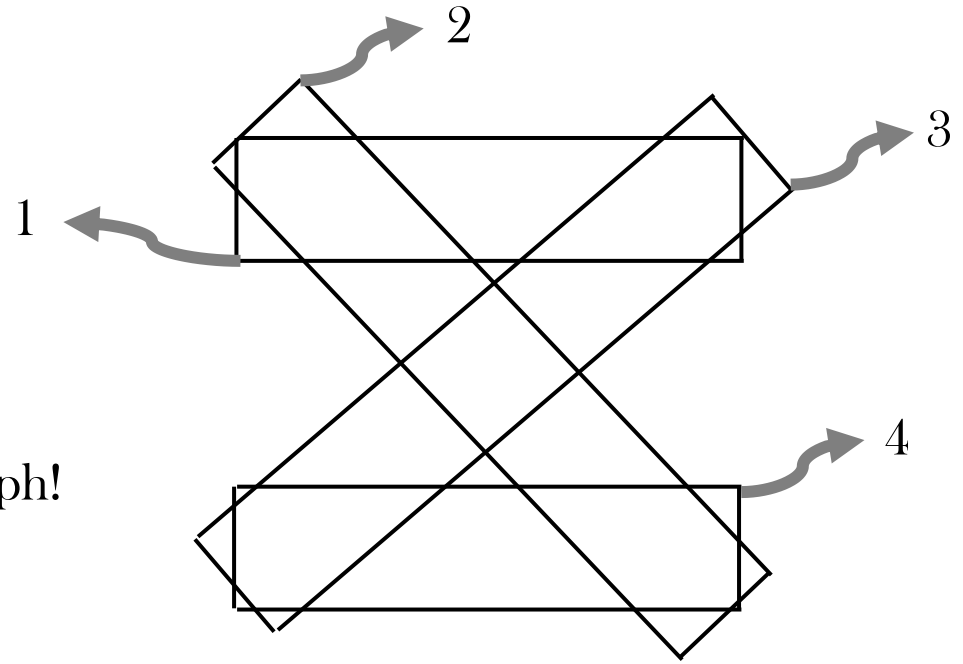
$10 \rightarrow 3\ cube\ (edge)$

$11 \rightarrow 4\ cube\ (edge)$

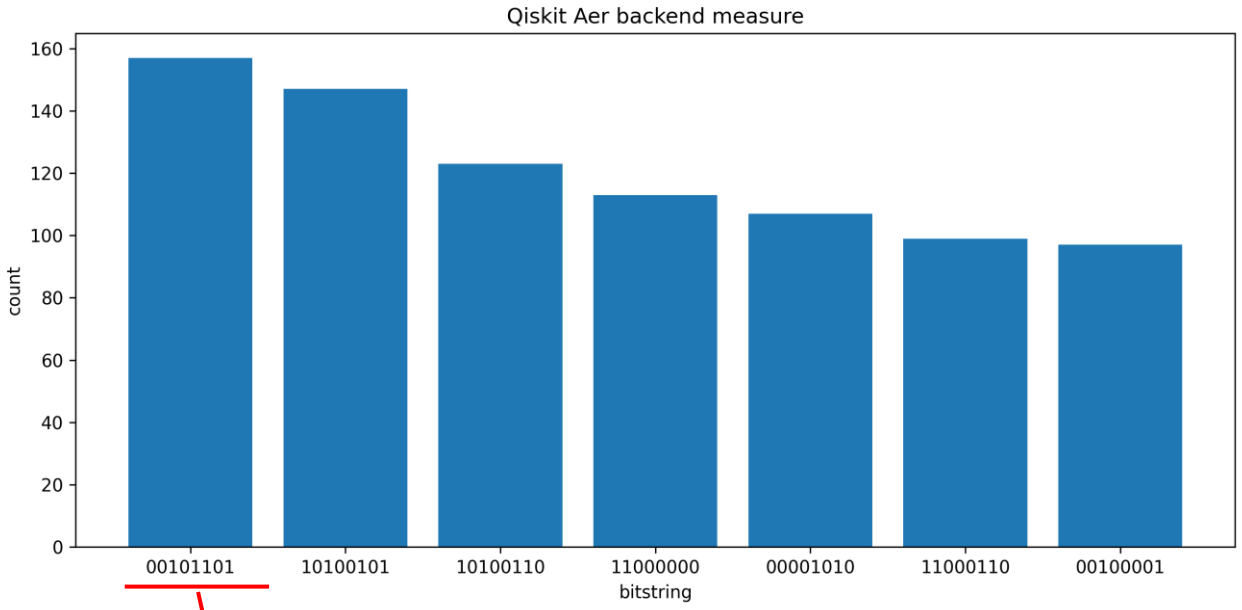
01 00 11 10 →

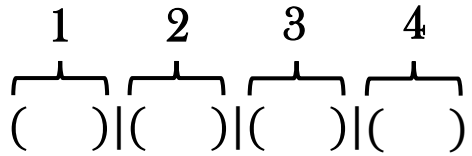



Another subgraph!



# # Third QAOA result (3-2) : penalty 2-2 result



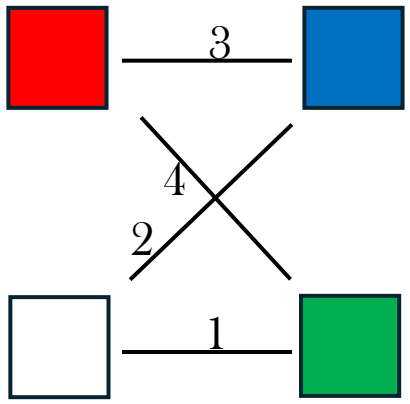

$00 \rightarrow 1\ cube\ (edge)$

$01 \rightarrow 2\ cube\ (edge)$

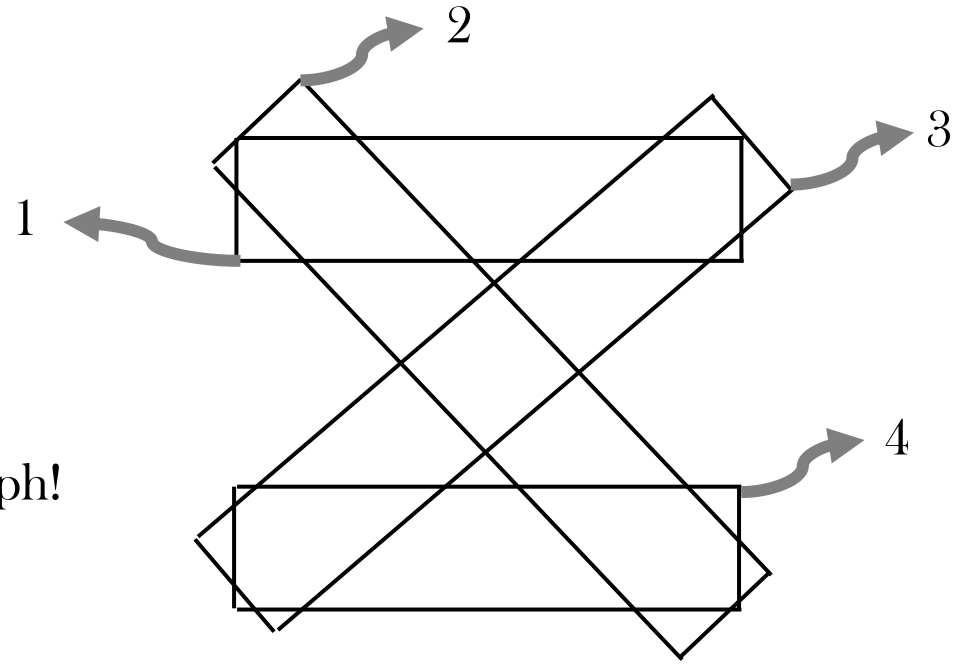
$10 \rightarrow 3\ cube\ (edge)$

$11 \rightarrow 4\ cube\ (edge)$

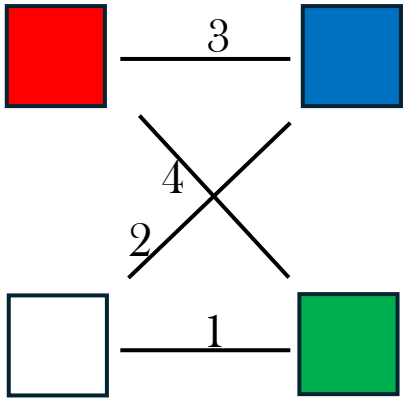
10 11 01 00 →

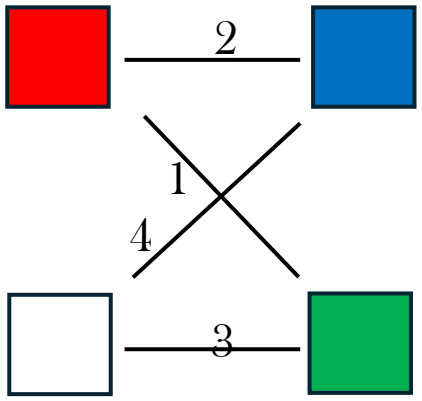



Another subgraph!



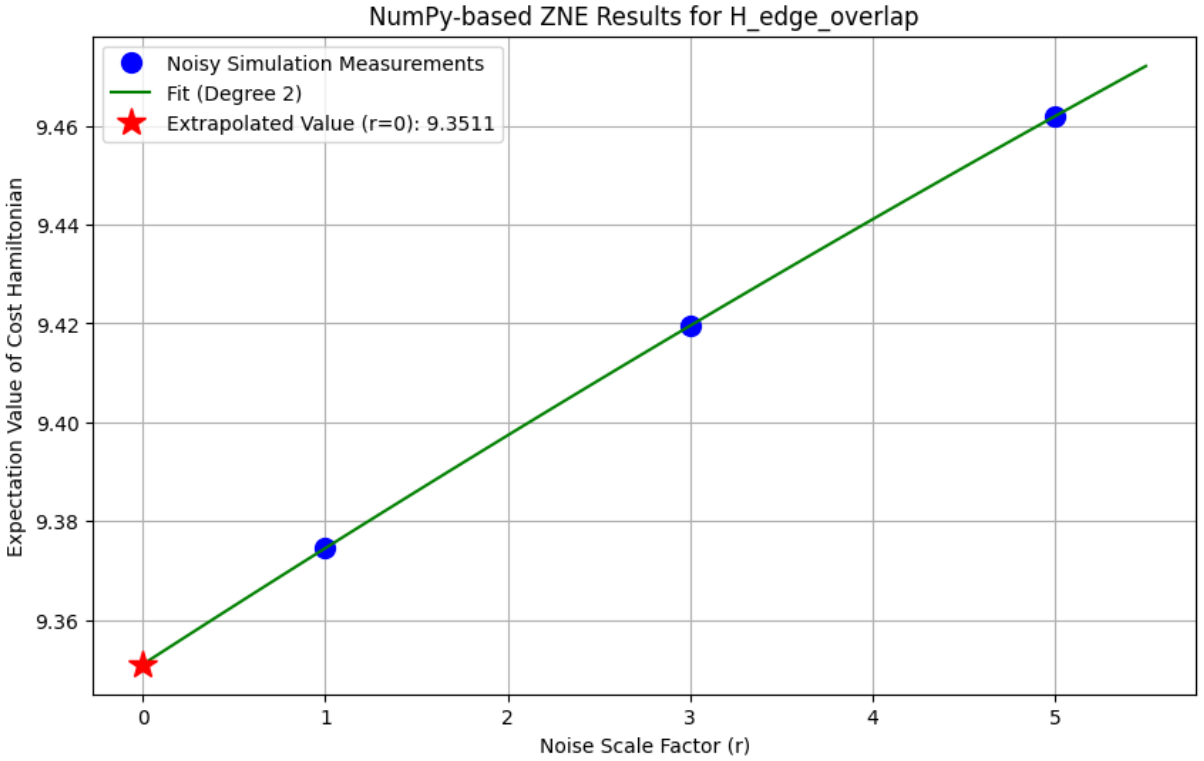
# Final Result

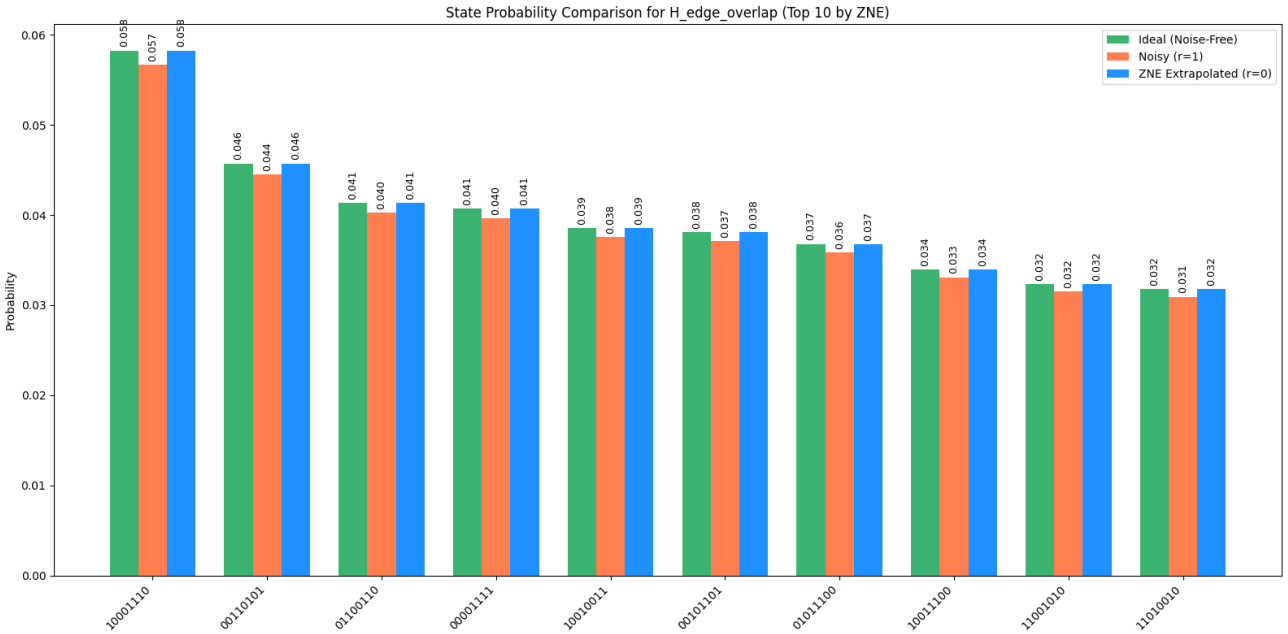



We find two subgraph!

# # Error mitigation result



# Reference

1. K. Blekos et al, A review on Quantum Approximate Optimization Algorithm and its variants, Phys. Rep 03.002(2024).

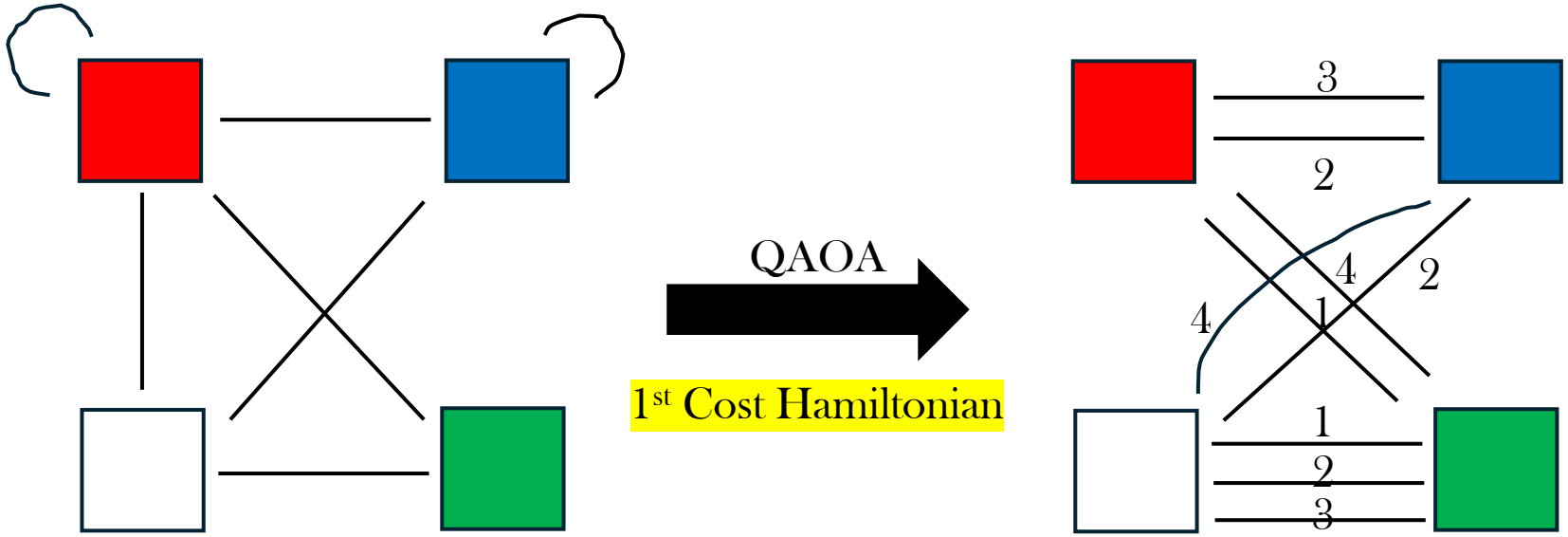
2. D. Egger et al, Warm starting quantum optimization, Quantum 5, 479 (2021).

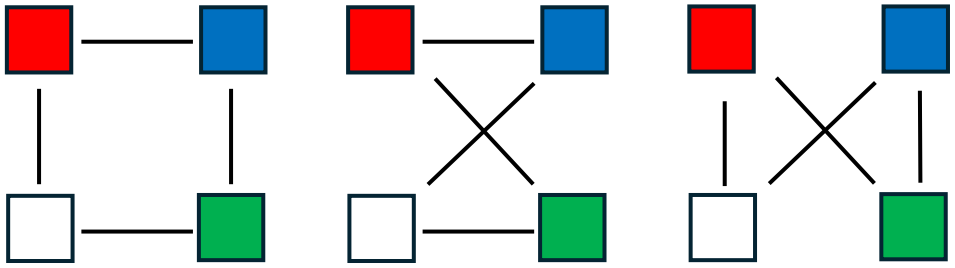
3. A. Weidinger et al, Error Mitigation for Quantum Approximate Optimization, Phys. Rev. A 3, 032408(2023).

4. H. Jnane et al, Quantum Error Mitigated Classical Shadows, PRX Quantum 5, 010324(2024).

5. V.Pascuzzi et al, Computationally Efficient Zero Noise Extrapolation for Quantum Gate Error Mitigation, Phys. Rev. A 105, 042406(2022)

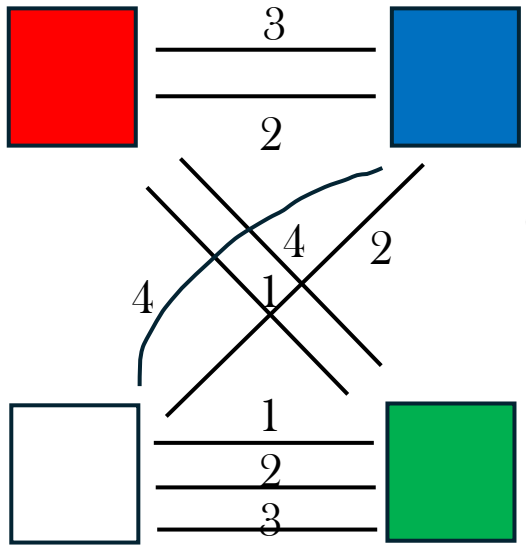
# # Appendix A. Why we apply first QAOA?



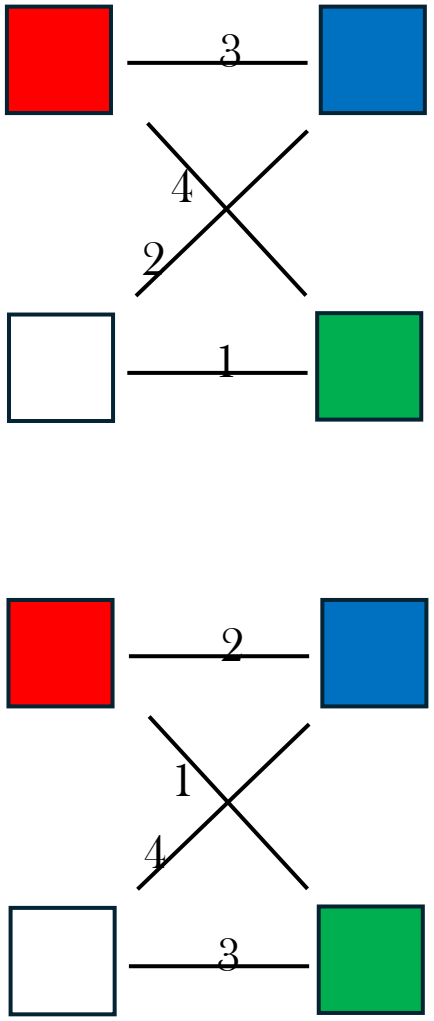
There are three possible subgraph patterns– i.e. $\frac{(n-1)!}{2}$ total configurations–
So, we first use QAOA to project the full graph
down onto one of these subgraph shapes.
At that stage, we recast the problem as a CSP
by adding a penalty term for every vertex whose degree is not exactly two.

# # Appendix B. Why we apply Second QAOA?




QAOA (Finding 2 subgraph)

2nd Cost Hamiltonian




Having used the first QAOA to pin down the allowed subgraph shape,
we now need to decide which edges to include.
We again frame this as a CSP and introduce two cost Hamiltonians:

One penalty for invalid edge,
the other penalty for duplicated edge.