

Qiskit Team PiQAOAsso:

Quantum Approximate Optimization Algorithm for constrained optimization problems — Graph coloring

Riley Chien¹, Will Pol², Alex Pozas³, Zhihui Wang⁴

1 Dartmouth

2 UChicago

3 ICFO

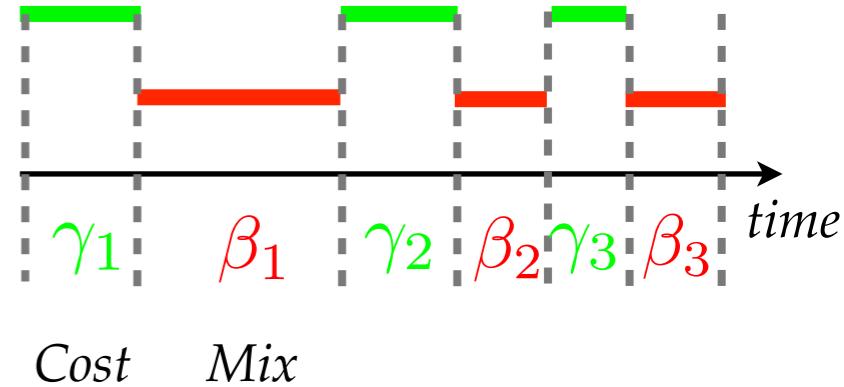
4 NASA Ames Center, USRA, Quantum AI Lab

One leading Candidate of quantum heuristics: Quantum Approximate Optimization Algorithms (QAOA) → Quantum Alternating Operator Ansatz

- One-line summary of the algorithm

$$U = e^{-i\beta_p H_M} e^{-i\gamma_p H_C} \dots e^{-i\beta_2 H_M} e^{-i\gamma_1 H_C}$$

[Farhi, Goldstone, and Gutmann, arXiv:1411.4028]

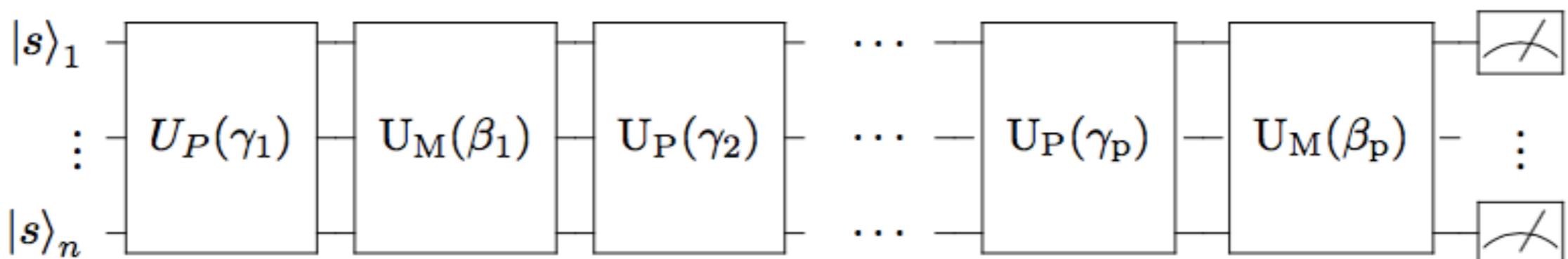


$$H_C = \sum_{j,j',\dots} (C_j \sigma_j^z + C_{j,j'} \sigma_j^z \sigma_{j'}^z + C_{j,j',j''} \sigma_j^z \sigma_{j'}^z \sigma_{j''}^z, + \dots)$$

$$H_M = \sum_j \sigma_j^x$$

Approach: $QAOA_p$ circuit:

- ▶ Prepare initial state
- ▶ Loop p times, and on iteration i apply Hamiltonians
 - ▶ (Phase separation) Cost-function-based H_C diag. in Z basis, for time γ_i
 - ▶ (Mixing) Hamiltonian H_M , for time β_i
- ▶ Measure in computational basis



QAOA for constrained optimization

[Hadfield et al., from QAOA to QAOA, arXiv 1709.03489];

- How are constrained problems approached?

Encode the constraints as **penalty in the cost function.** — Lagrange multipliers

Commonly practiced in quantum annealing.

Alternative: Use a **mixer** that contains the quantum evolution in the subspace that satisfies the constraints.

[Hen & Spedalieri, '16]

Original motivation: Alleviate embedding burden

Another Advantage: **Smaller search space!**

QAOA for graph coloring problem

Theory and Simulation:

Wang, Rubin, Dominy, Rieffel. XY mixer for QAOA on graph coloring problems. (to-appear)

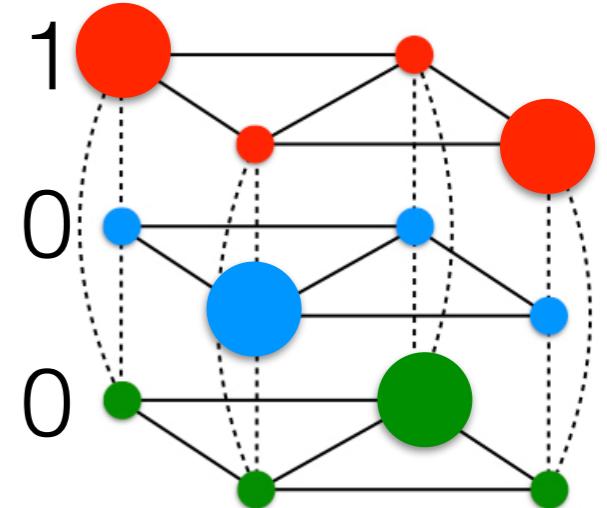
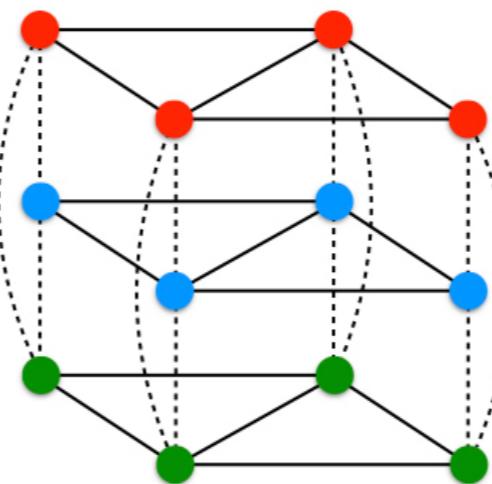
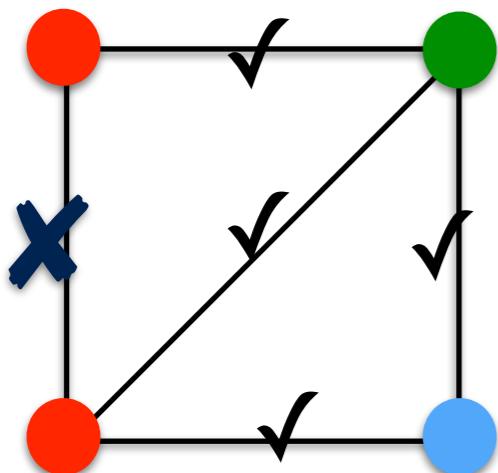
- Goal: Assign colors to vertices to maximize properly-colored edges (connecting two vertices of different color) – NP hard

Encoding:

Binary: $x_{v,c} = 1$ whether vertex v is assigned color-c

Constraints:

Each vertex should have exactly one color: $\sum_{c=1}^k x_{v,c} = 1 \iff \sum_{c=1}^k \sigma_{v,c}^z = k - 2$



QAOA for graph coloring problem

Constraints:

Each vertex should have exactly one color:

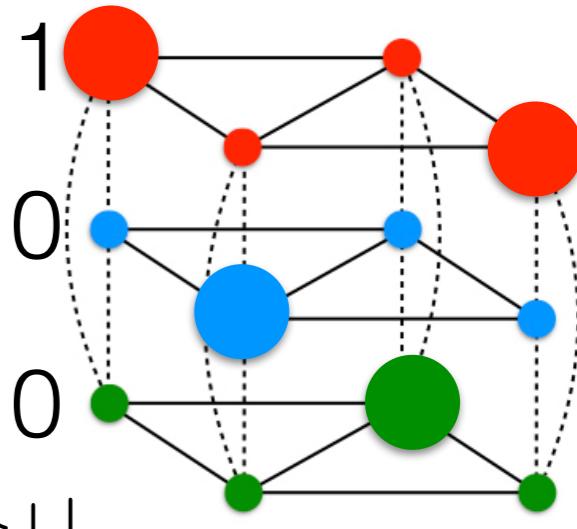
$$\sum_{c=1}^k \sigma_{v,c}^z = k - 2$$

Mixer to Stay in the feasible subspace:

XY-model

$$|\uparrow\downarrow\rangle\langle\downarrow\uparrow| + |\downarrow\uparrow\rangle\langle\uparrow\downarrow|$$

$$\propto \sigma_{v,c}^x \sigma_{v,c}^x + \sigma_{v,c}^y \sigma_{v,c}^y$$



- Size of search space

Penalty + X-mixer

Full Hilbert space

$$2^{nk}$$

XY mixer

Feasible subspace

$$k^n$$

Ratio: $\left(\frac{k}{2^k}\right)^n$

The feasible space **shrinks exponentially** with n .

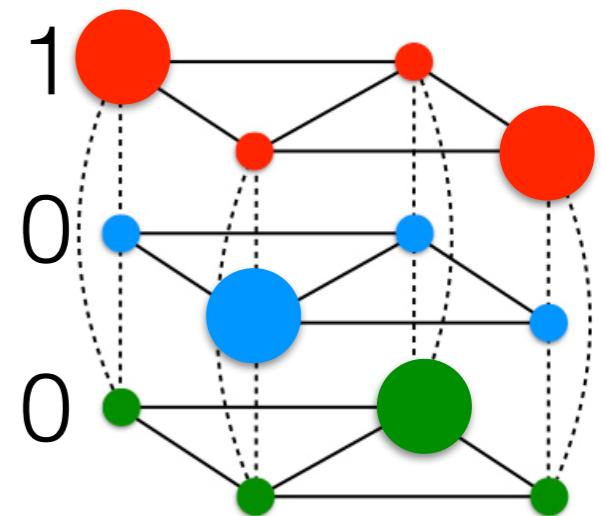
QAOA for graph coloring problem

Cost function

$$f_C = m - \sum_{c=1}^k \sum_{\{v,v'\} \in E} x_{v,c} x_{v',c}$$

Cost Hamiltonian

$$H_C = \sum_{c=1}^k \sum_{\{v,v'\} \in E} \sigma_{v,c}^z \sigma_{v',c}^z$$



Pseudo code for qiskit to incorporate arbitrary mixer for QAOA

Pseudo-Code

```
class constrainedQAOA(VQE):
    def __init__(cost, mixer, optimizer, initial_state, p, ...):

        # remove old mixer construction

        var_form = constrainedQAOAVarForm(cost, p, mixer, initial_state)

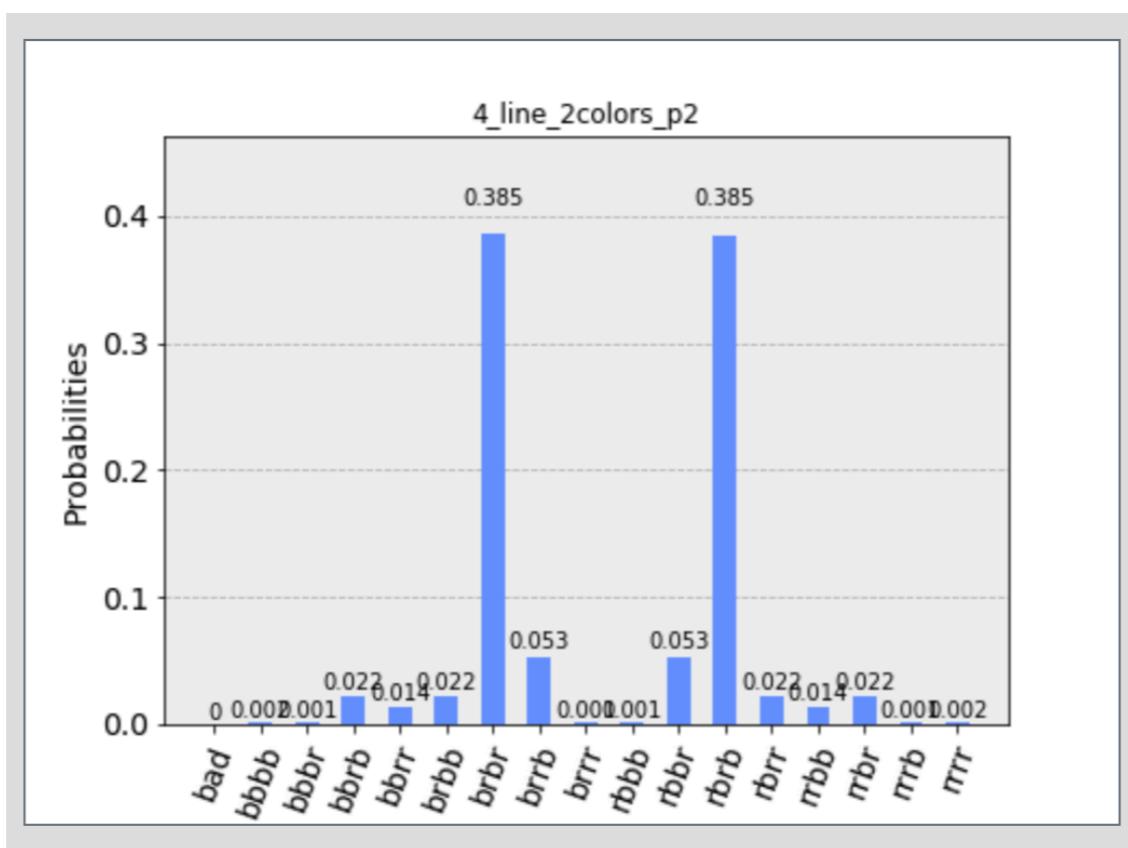
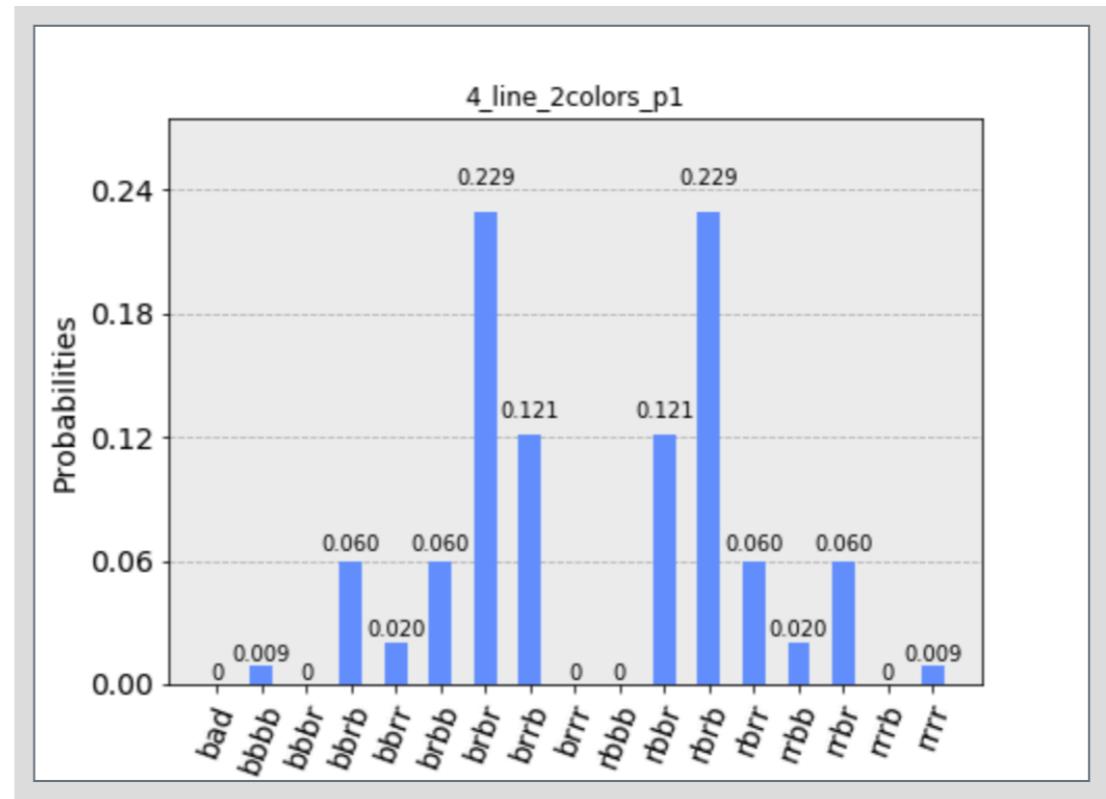
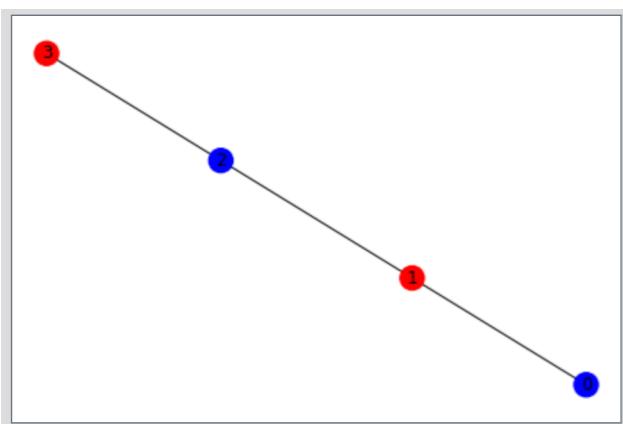
        super().__init__(cost, var_form, optimizer)
```

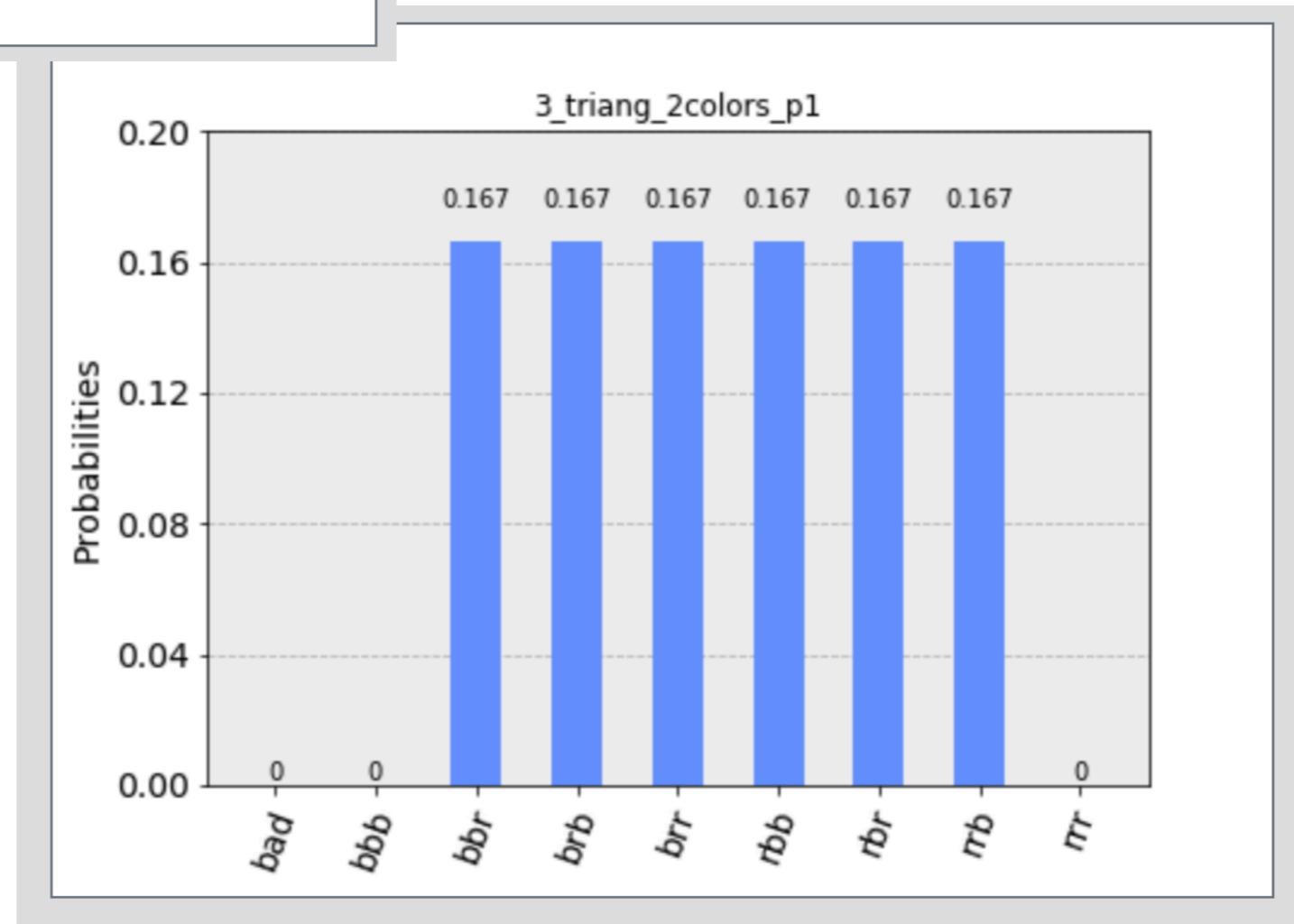
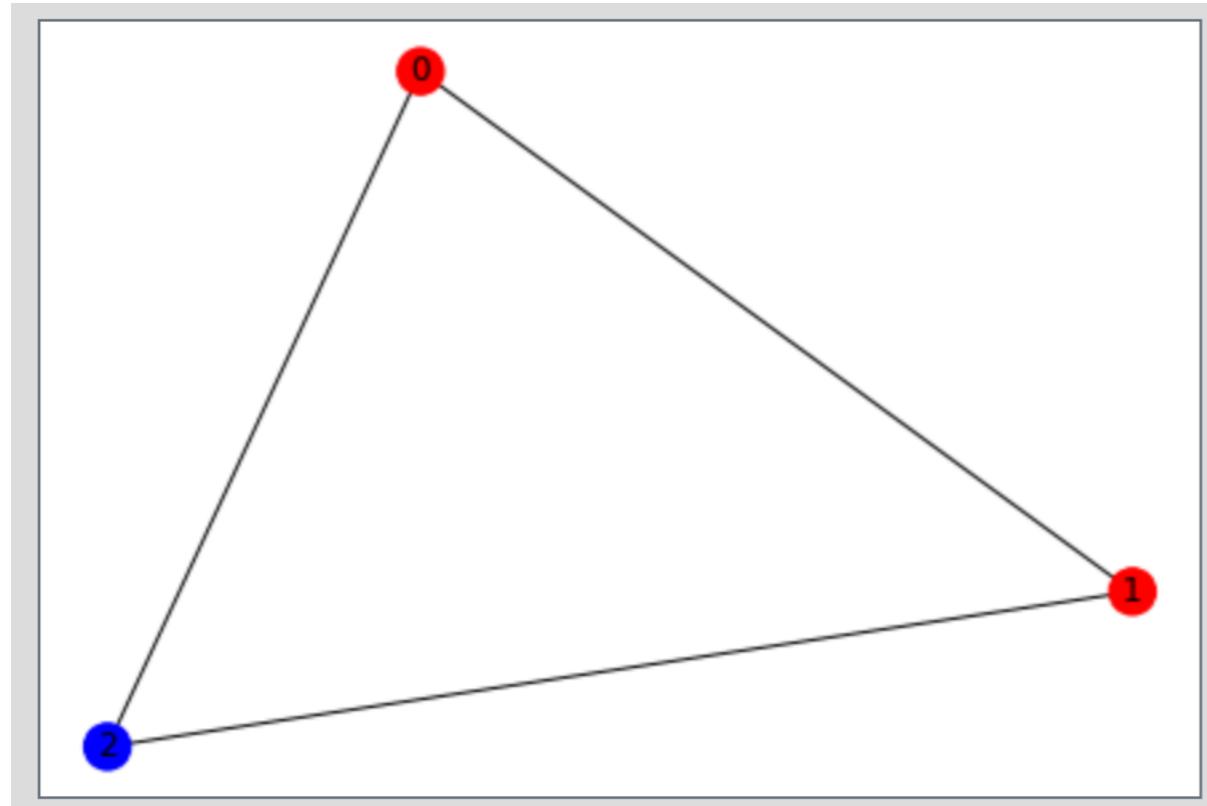
Pseudo-Code

```
class constrainedQAOAVarForm(QAOAVarForm):
    def __init__(self, cost, mixer, initial_state, p, optimizer):
        # remove old mixer construction

        U_c = cost.evolve(gamma)
        U_m = mixer.evolve(beta)
        U_qaoa = U_c*U_m

        VQE(initial_state, U_qaoa, optimizer)
```





Results

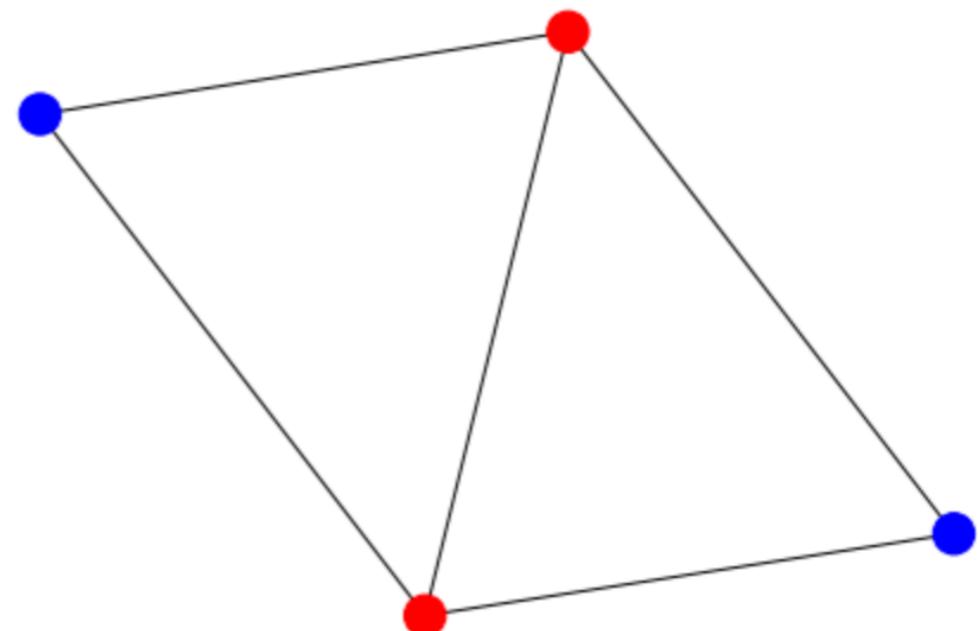
Graphs with 2 colors

```
In [4]: for graph, structure in zip(to_graph_2colors, structures_2colors):
    probs = get_probs('results/' + graph)
    print('-----' + graph[:-11] + '-----')
    draw_best_coloring_2colors(probs, graphs[structure])
    plt.show()
```

```
-----4_square1diag_2colors_p1-----
Probability of coloring: 0.2202
```

```
-----4_square1diag_2colors_p2-----
Probability of coloring: 0.4327
```

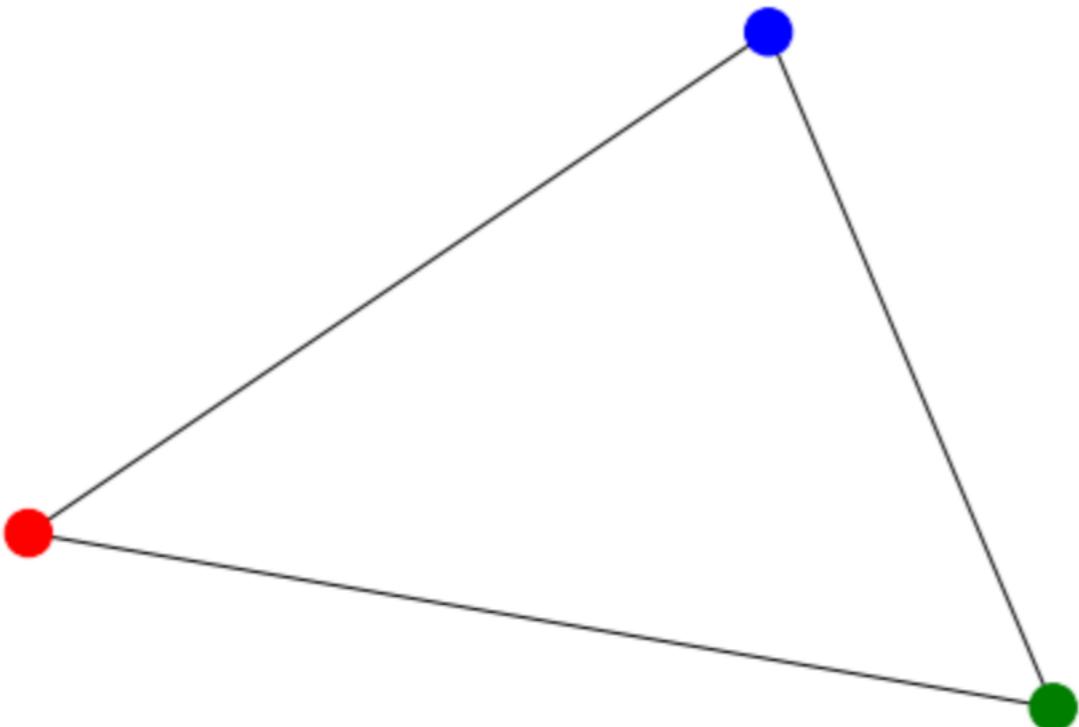
```
-----4_square1diag_2colors_p3-----
Probability of coloring: 0.4644
```



Graphs with 4 colors

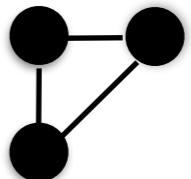
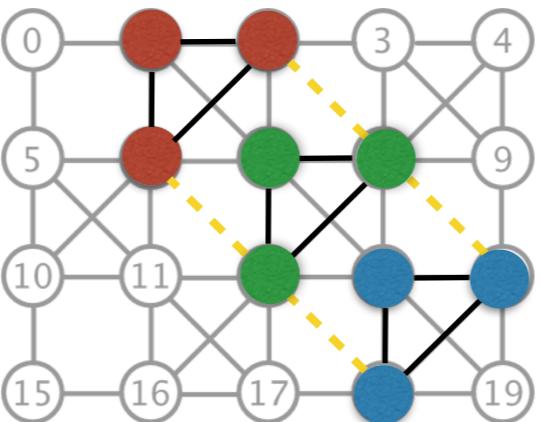
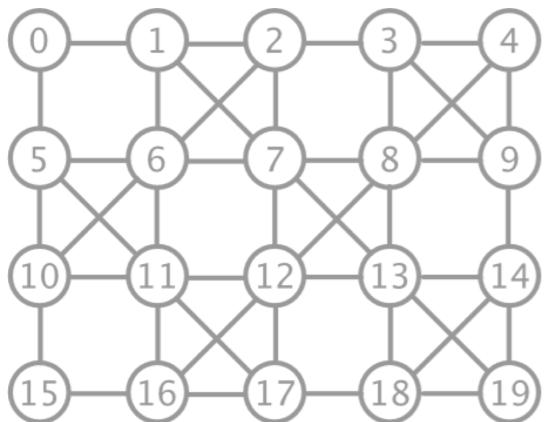
```
In [5]: for graph, structure in zip(to_graph_4colors, structures_4colors):
    probs = get_probs('results/' + graph)
    print('-----' + graph[:-11] + '-----')
    draw_best_coloring_4colors(probs, graphs[structure])
    plt.show()
```

-----3_triang_4colors_p1-----
Probability of coloring: 0.0278



On the hardware IBM Q 20 Tokyo:

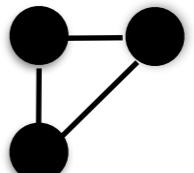
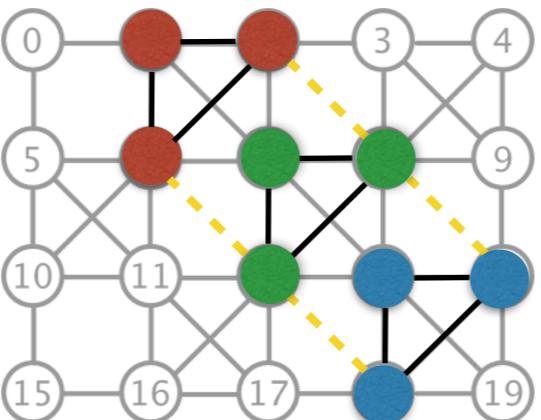
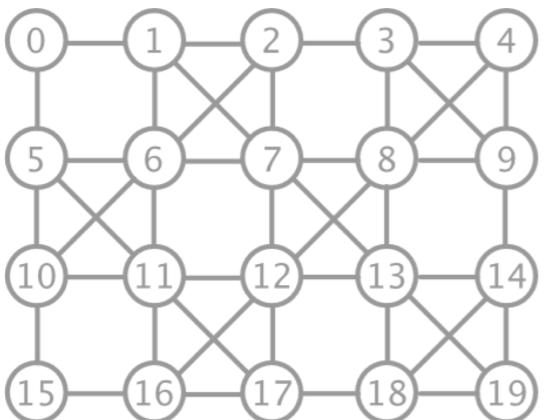
3-coloring of a triangle



missing connections: SWAP gates needed

On the hardware IBM Q 20 Tokyo:

3-coloring of a triangle



missing connections: SWAP gates needed

Hi. We get "ERROR_RUNNING_JOB". It seems that your circuit is too long.

I see there are so many CNOT gates. They result in bad fidelity because of CNOT gate errors.

What we could be seeing: (previous simulation results)

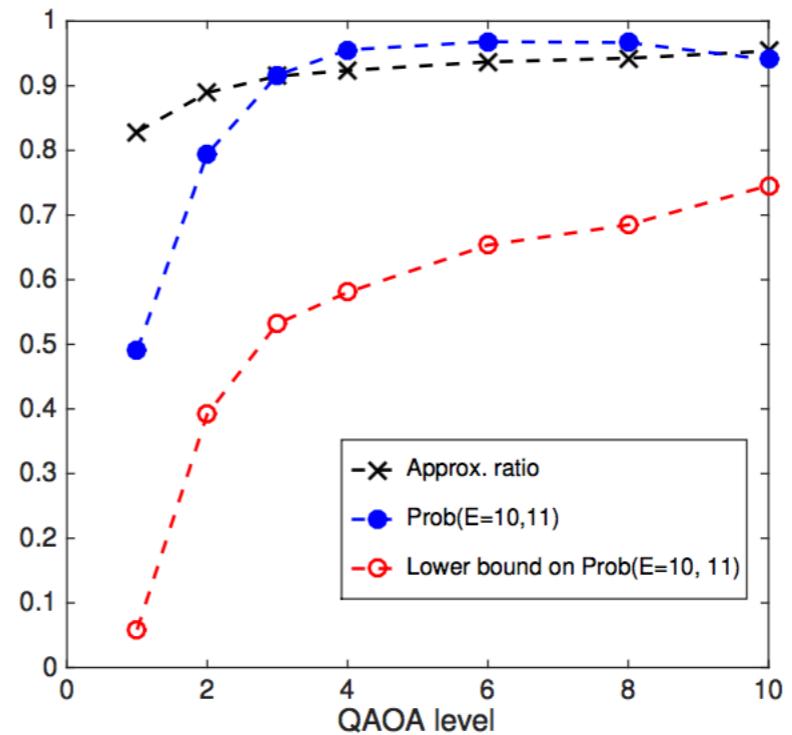
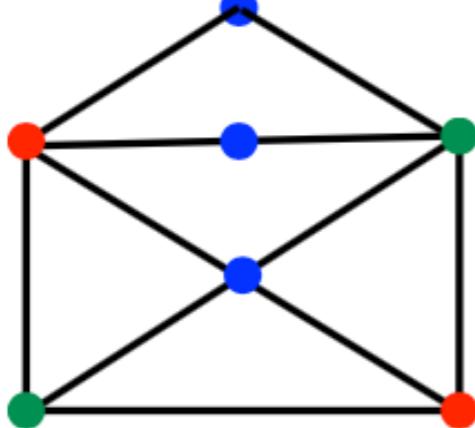


Figure 8. 3-coloring of Envelope graph (11 edges). QAOA with simultaneous ring-mixer. For each QAOA level, the probability of getting the top two highest approximate results (cost 11 and 10) is shown in comparison to the bound given by Eq. (1) with the observed approximation ratio as parameter.

Wang, Rubin, Dominy, Rieffel. XY mixer for QAOA on graph coloring problems. (to-appear)

Conclusion

- QAOA on constraint problems — Explore symmetries in the system, use a smart mixer!