Quantum Information and Quantum Computing Final

Project 10: Variational Quantum Algorithms

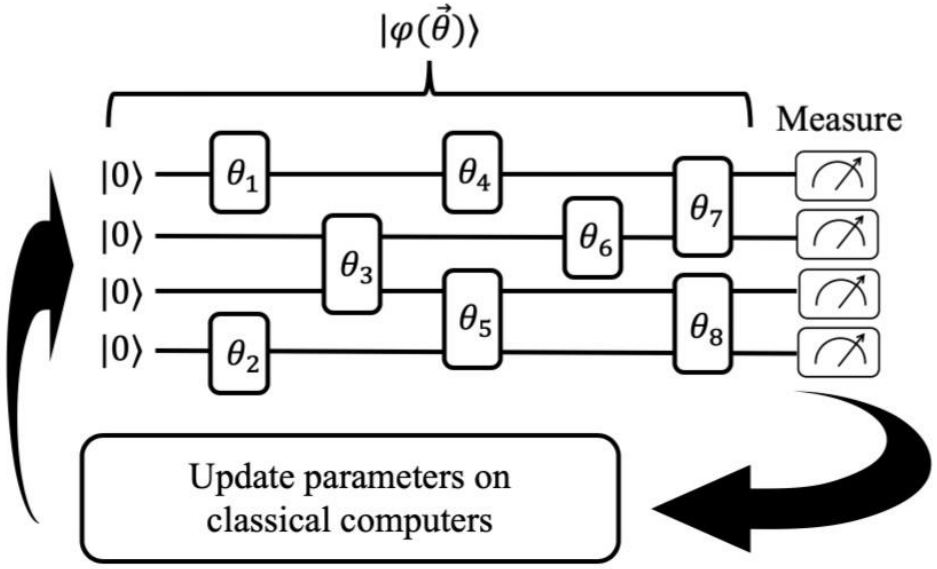
Zeynepnur Sahinel

31.01.2023

Ansätz

$$|\varphi(\vec{\theta})\rangle = U(\vec{\theta})\,|\varphi_{ref}\rangle$$

- $|\varphi(\vec{\theta})\rangle$ → ansatz state
- $U(\vec{\theta})$ → $U_N(\theta_N) \dots U_k(\theta_k) \dots U_1(\theta_1)$ (ansatz circuit) (shallow quantum circuit)
- $|\varphi_{ref}\rangle$ → initial state



From: Hybrid classical-Quantum Algorithms

VQA
- VQO (Variational Quantum Optimisation): Optimize parameters under a cost function, e.g. Variational Quantum Eigensolver
- VQS (Variational Quantum Simulation): simulate dynamical process such as Schrödinger time evolution

Variational Quantum Eigensolver (VQE)

Rayleigh-Ritz variational principle:

$$\min_{\vec{\theta}} \langle \psi(\vec{\theta}) | H | \psi(\vec{\theta}) \rangle \geq E_G$$

E_G → ground-state energy

energy

$$E(\vec{\theta}) = \langle \psi(\vec{\theta}) | H | \psi(\vec{\theta}) \rangle \rightarrow \text{Cost Function to be Minimised}$$

How to measure $E(\vec{\theta})$?

$$H = \sum_{\alpha} f_\alpha P_\alpha \longrightarrow P_\alpha \in \{I, X, Y, Z\}^{\otimes N_q}$$

f_α → real coefficients

N_q → # of qubits

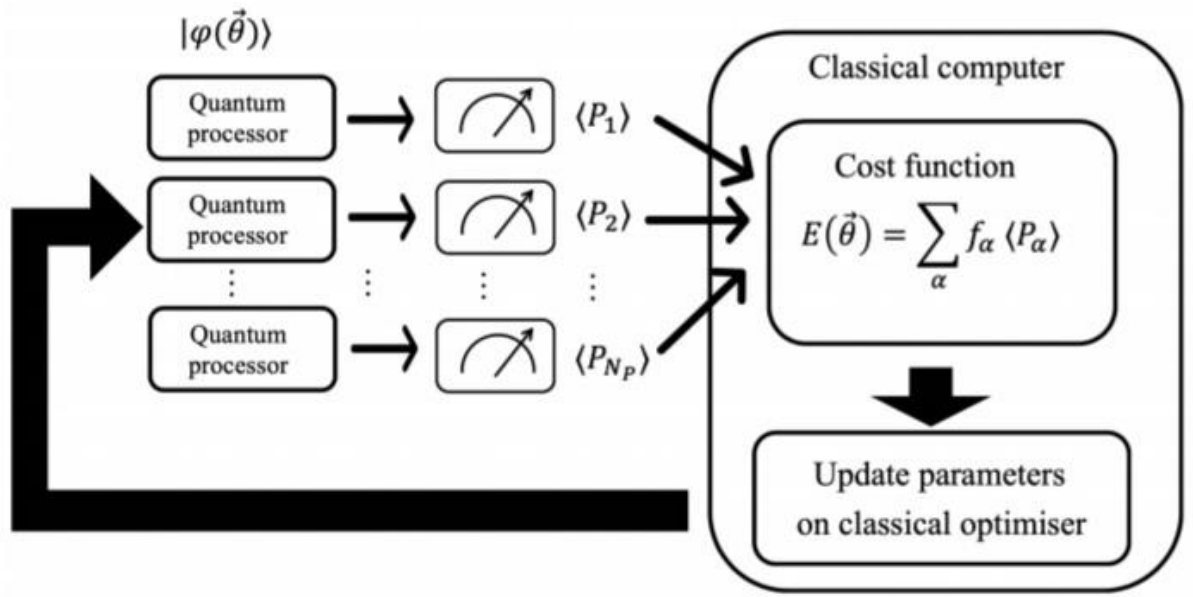
$$E(\vec{\theta}) = \sum_{\alpha} f_\alpha \langle \psi(\vec{\theta}) | P_\alpha | \psi(\vec{\theta}) \rangle$$

↳ measurement of P_α can be fully parallelised by using many quantum processors.

Later, update the parameters:

Gradient Descent: $\vec{\theta}^{(n+1)} = \vec{\theta}^{(n)} - \alpha \nabla E(\vec{\theta}^{(n)})$

α → step size parameter



QUBO (Quadratic Unconstrained Binary Optimization)

Quadratic Program

Quadratically constrained optimization problem

minimize/maximize $x^T Q x + c^T x$

subject to $Ax \leq b$

$x^T Q_i x + a_i^T x \leq r_i$

$l_j \leq x_j \leq u_j$

QUBO →

Min/Max $x^T Q x + c^T x$

$x \in \{0,1\}^n$

$Q \in R^{n \times n}$

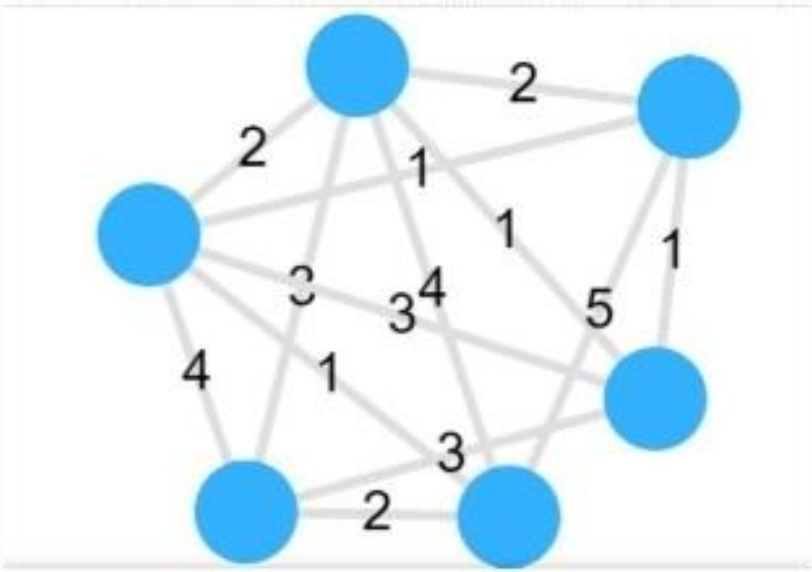
$c \in R^n$

MaxCut as a Quadratic Program

To find a partition of the graph vertices into two disjoint sets such that cumulative weight of edges from different cuts is maximized.

$$C(x) = \sum_{i,j=1}^{n} w_{ij}\, x_i (1 - x_j)$$

$C(x)$: Cost Function; w_{ij}: weight matrix



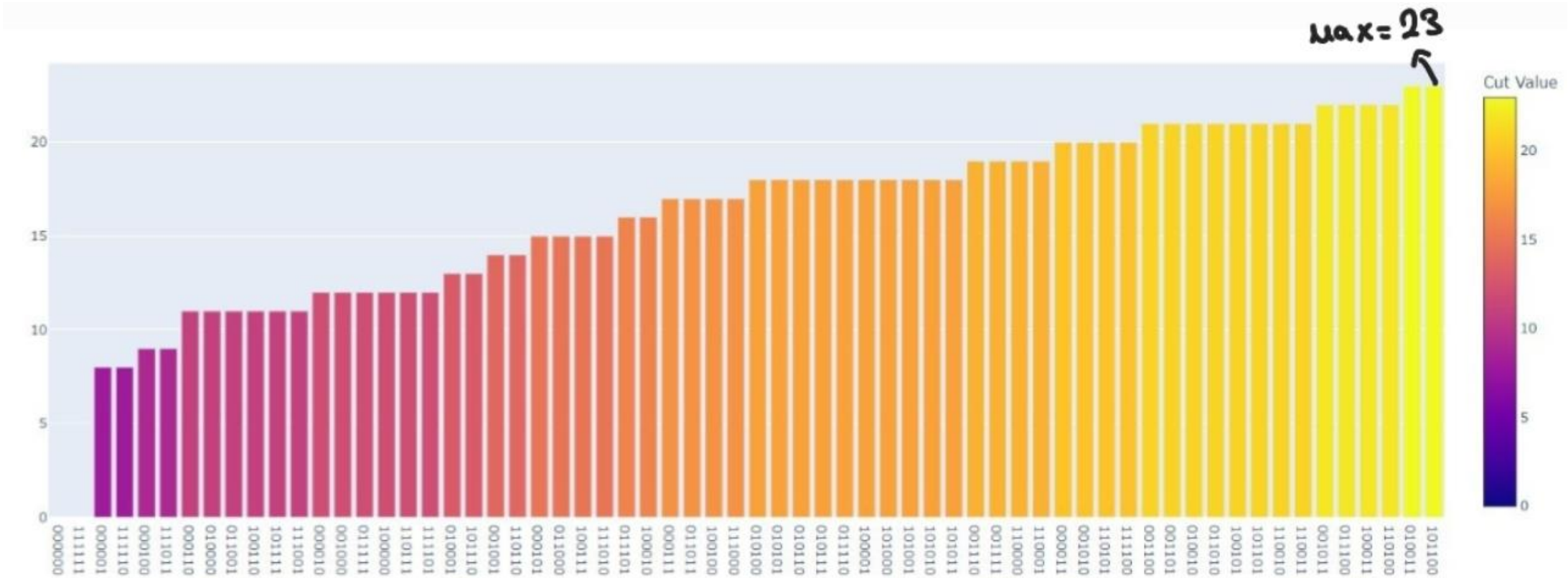
QGSS'21-Lab1

$$\sum_{i,j=1}^{n} w_{ij}\, x_i (1 - x_j) = \sum_{i,j=1}^{n} w_{ij}\, x_i - w_{ij}\, x_i x_j$$

$$= \sum_{i=1}^{n} \left(\sum_{j=1}^{n} w_{ij} \right) x_i - \sum_{i,j=1}^{n} w_{ij}\, x_i x_j$$

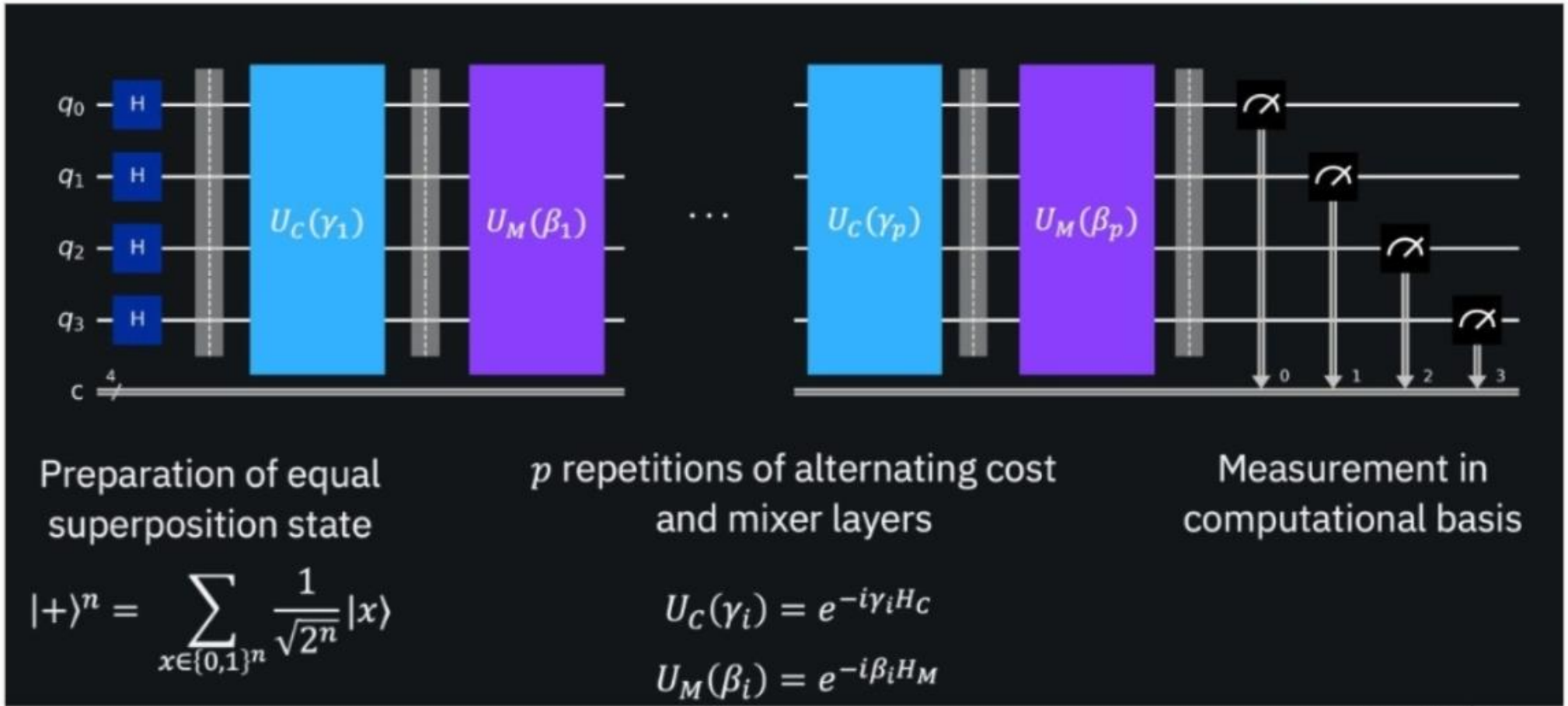
$$C(x) = c^T x + x^T Q x$$

$$Q_{ij} = -w_{ij} \qquad c_i = \sum_{j=1}^{n} w_{ij}$$



All possible cuts of a created weighted graph from QGSS'21-Lab2

QAOA (Quantum Approximate Optimization Algorithm)



QGSS'21-Lab2

Let's derive the QAOA Circuit!

Mixer Hamiltonian: $H_M = \sum_{i=1}^{n} X_i \longrightarrow U_M(\beta) = e^{-i\beta H_M} = \prod_{i=1}^{n} R_{x_i}(2\beta)$

Cost Hamiltonian: From QUBO to Hamiltonian

$$H_C|x\rangle = (x^T Q x + c^T x)|x\rangle = \left(\sum_{i,j=1}^{n} x_i Q_{ij} x_j + \sum_{i=1}^{n} c_i x_i\right)|x\rangle$$

Convert to computational basis:

$$Z_i|x\rangle = (-1)^{x_i}|x\rangle = (1-2x_i)|x\rangle \rightsquigarrow x_i|x\rangle = \frac{I - Z_i}{2}|x\rangle$$

Thus $$H_C|x\rangle = \sum_{i,j=1}^{n} \frac{1}{4} Q_{ij} Z_i Z_j - \sum_{i=1}^{n} \frac{1}{2}\left(c_i + \sum_{j=1}^{n} Q_{ij}\right) Z_i + \left(\sum_{i,j=1}^{n} \frac{Q_{ij}}{4} + \sum_{i=1}^{n} \frac{c_i}{2}\right)$$

$$U_C(\gamma) = e^{-i\gamma H_C} = \prod_{\substack{i,j=1 \\ i \neq j}}^{n} R_{Z_i Z_j}\left(\frac{1}{2} Q_{ij}\gamma\right) \prod_{i=1}^{n} R_{Z_i}\left(\left(c_i + \sum_{j=1}^{n} Q_{ij}\right)\gamma\right)$$

@ $p = 1^{st}$ layer

Note: I implemented the QAOA circuit on the QGSS'21 Lab2 Module. Normally, this circuit is tested by Qiskit Grader. However, I could not manage to set the Qiskit Grader, I guess I don't have authorization to do it :(

I moved to QAOA which is Qiskit own implementation version.

```
from qiskit.algorithms import QAOA
from qiskit_optimization.algorithms import MinimumEigenOptimizer
backend = Aer.get_backend('statevector_simulator')
qaoa = QAOA(optimizer = ADAM(), quantum_instance = backend, reps=1, initial_point = [0.1,0.1])
eigen_optimizer = MinimumEigenOptimizer(min_eigen_solver = qaoa)
quadratic_program = quadratic_program_from_graph(graphs['custom'])
result = eigen_optimizer.solve(quadratic_program)
print(result)
```

→ the created weight-graph

→ written before

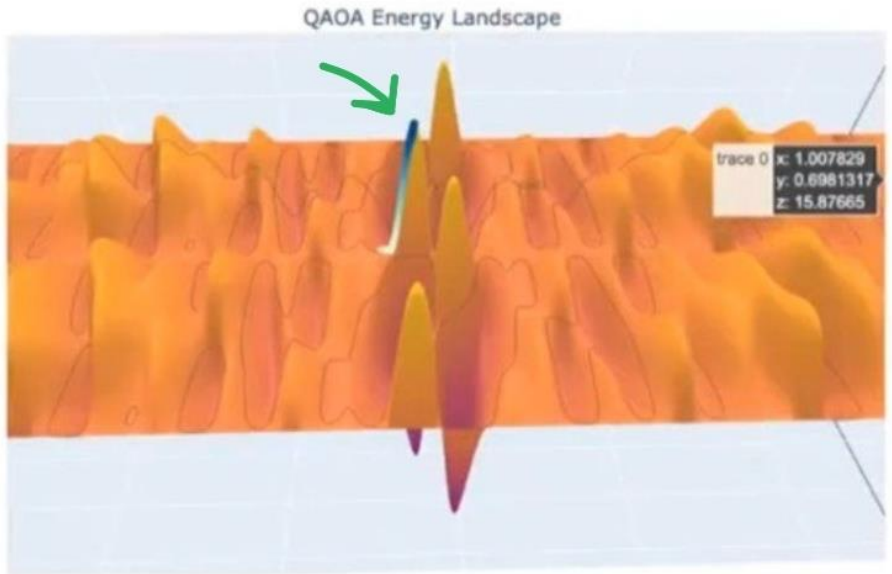
```
fval=23.0, x_0=1.0, x_1=0.0, x_2=1.0, x_3=1.0, x_4=0.0, x_5=0.0, status=SUCCESS
```

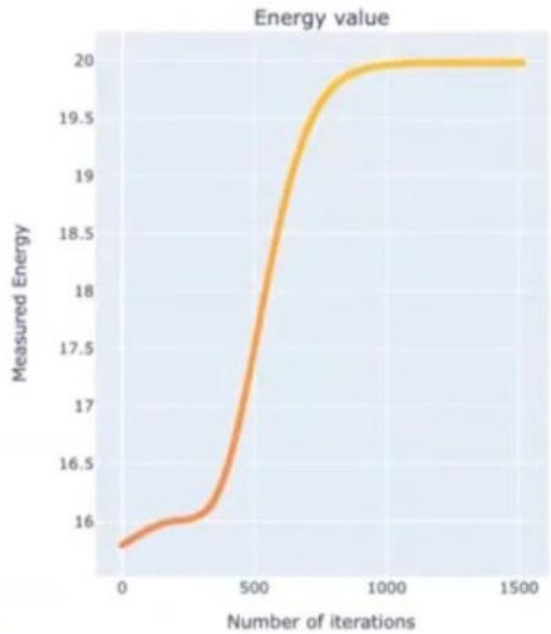
✓

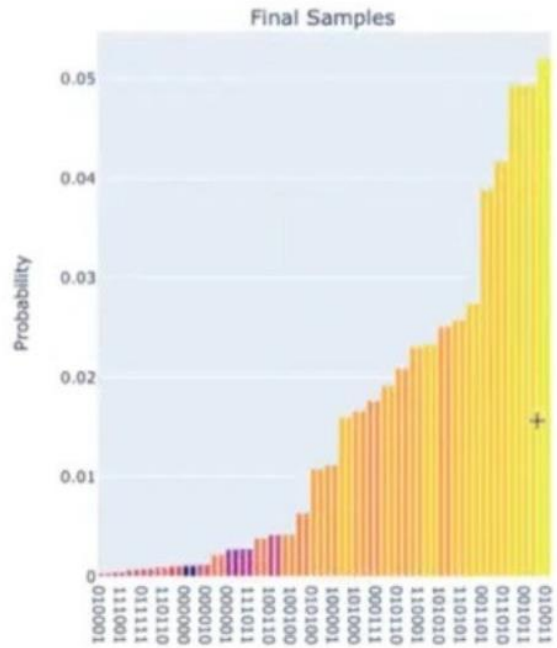
Optimizer : Classical Optimizer → COBYLA, SLSQP, ADAM (gradient based)

MinimumEigenOptimizer : handles conversion from a quadratic program to a qubit operator.

DISCUSSIONS:

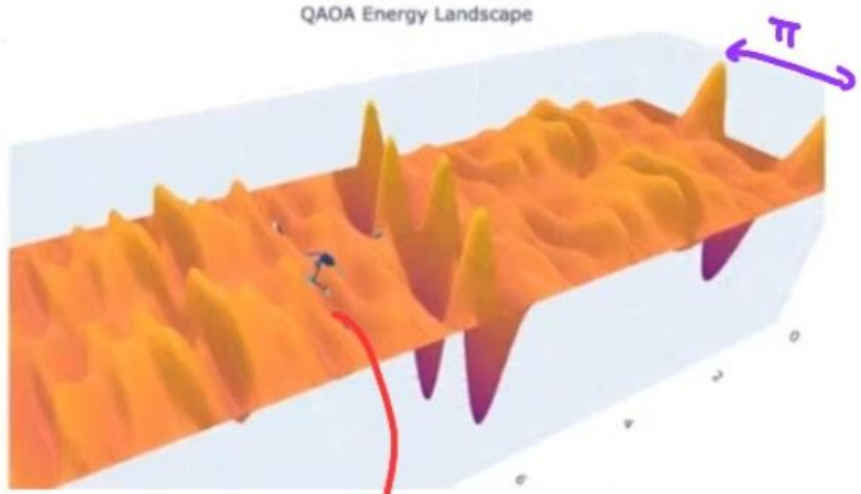


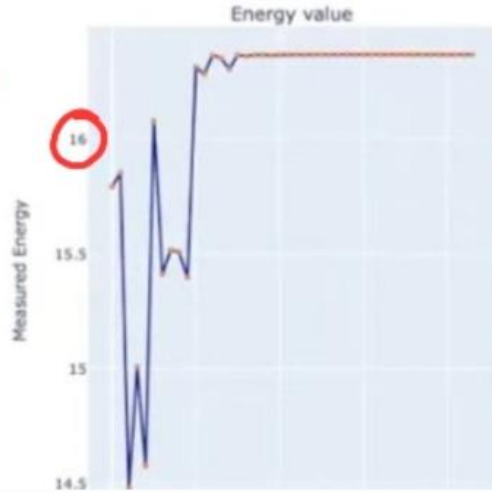


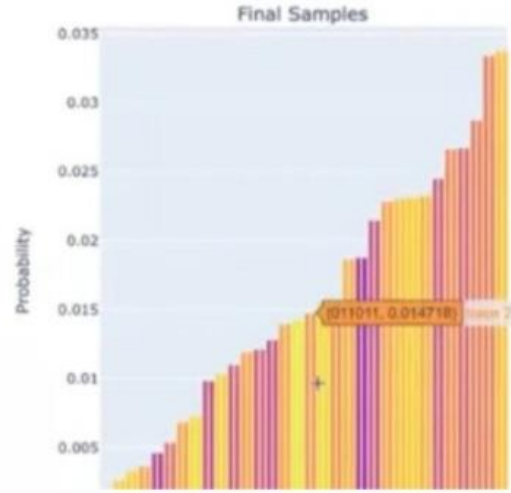


QAOA Energy Landscape Simulations from QGSS'21 Lab2 Video

✓ Very good solution







Different initial point!

Colors are not in order as well

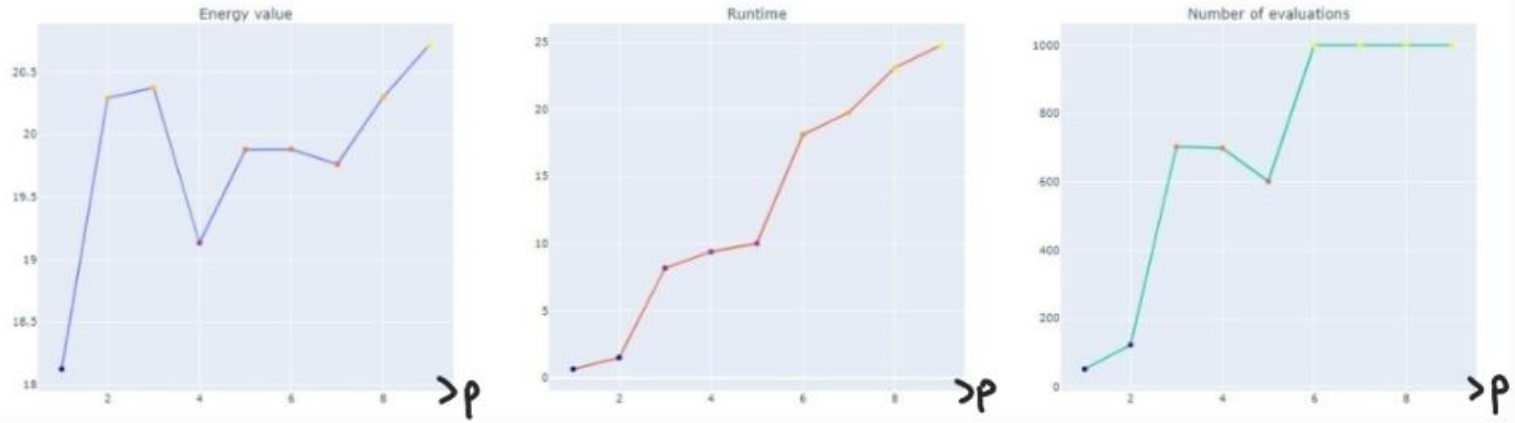
It stucked at the local maxima :(

Also it is important to note that β parameters are periodic with $\tau = \pi$ (due to R_x) "Fixed period"

γ parameters' periodicity is not fixed.

Higher values of p (depth)

Using the quadratic_program created with the weighted graph:



$$M_p = \max_{\beta\gamma \in \mathbb{R}^n} \langle \Psi_p(\beta,\gamma) | H_c | \Psi_p(\beta,\gamma) \rangle$$

$$M_{p+1} \geq M_p$$

Using adiabatic theorem → $\lim_{p\to\inf} M_p = C_{max}$

! With increasing # of parameters, finding the global optimum becomes increasingly harder

References

- Lecture 5.2 - Introduction to the Quantum Approximate Optimization Algorithm and Applications
- Lab 2: Variational Algorithms
- https://qiskit.org/textbook/ch-applications/qaoa.html
- https://learn.qiskit.org/summer-school/2021/lab2-variational-algorithms
- Modified Code of Lab 2 Variational Algorithms
- Hybrid quantum-classical algorithms and quantum error mitigation