Quantum Algorithms:
Variational Quantum Algorithms

Outline

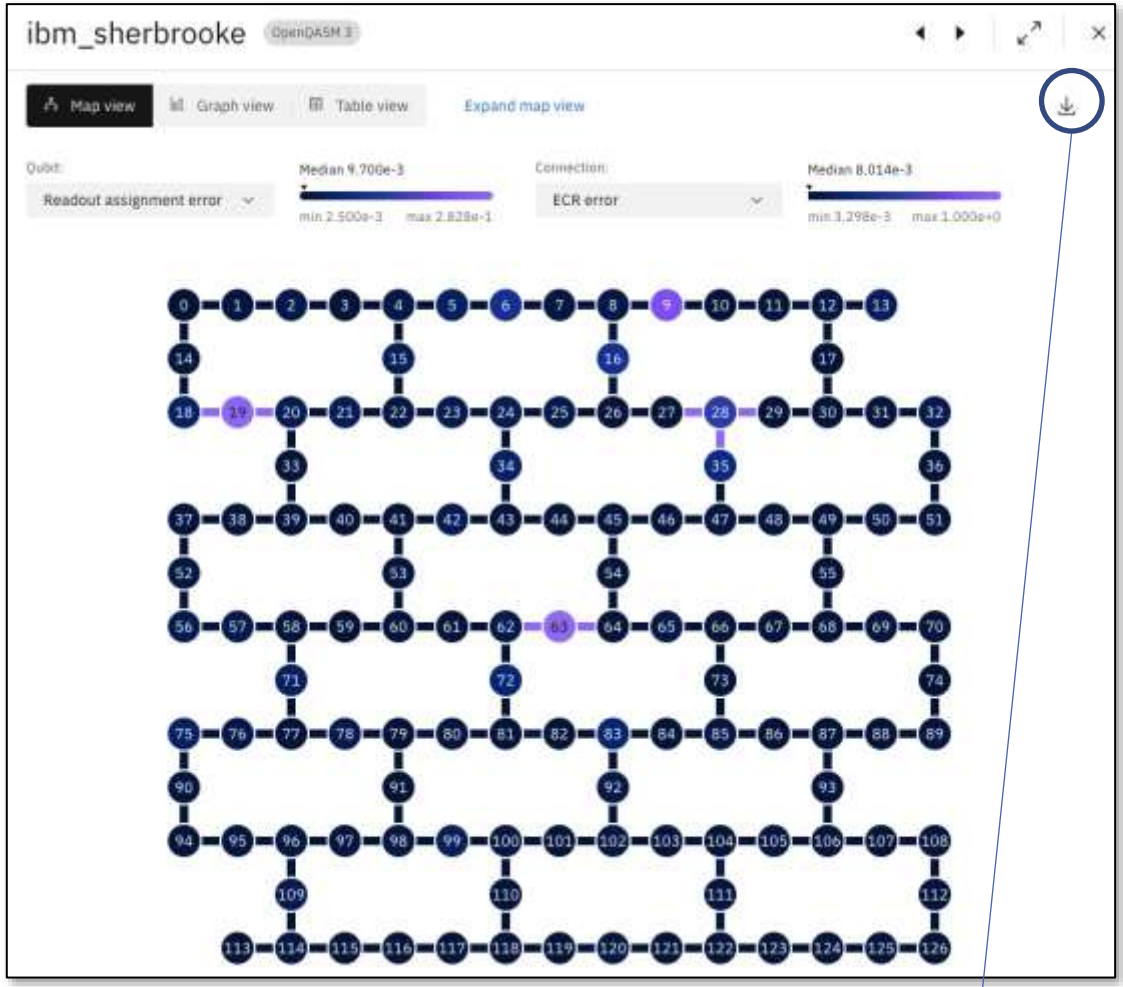
- Overview of hybrid quantum-classical approach
- Introduction of optimization
  - How to convert an optimizatioon problem into a Hamiltonian
- Variational quantum eigensolver (VQE)
- Quantum approximate optimization algorithm (QAOA)

- Hands-on
  - VQE
  - QAOA for Maxcut

Noisy quantum devices

- Various limitations of the current noisy quantum devices
  - Gate / Readout errors
  - Limited connectivity of qubits
    - Additional two-qubit gates are necessary to apply two-qubit gates to two distant qubits

- Hard to run deep quantum circuits and obtain accurate results due to noise
  - What if you execute k gates whose error rate is p?
  - Fidelity of the result will be $F = (1-p)^k$
  - E.g., $p = 0.01$, $k=100 \rightarrow F = 0.366$

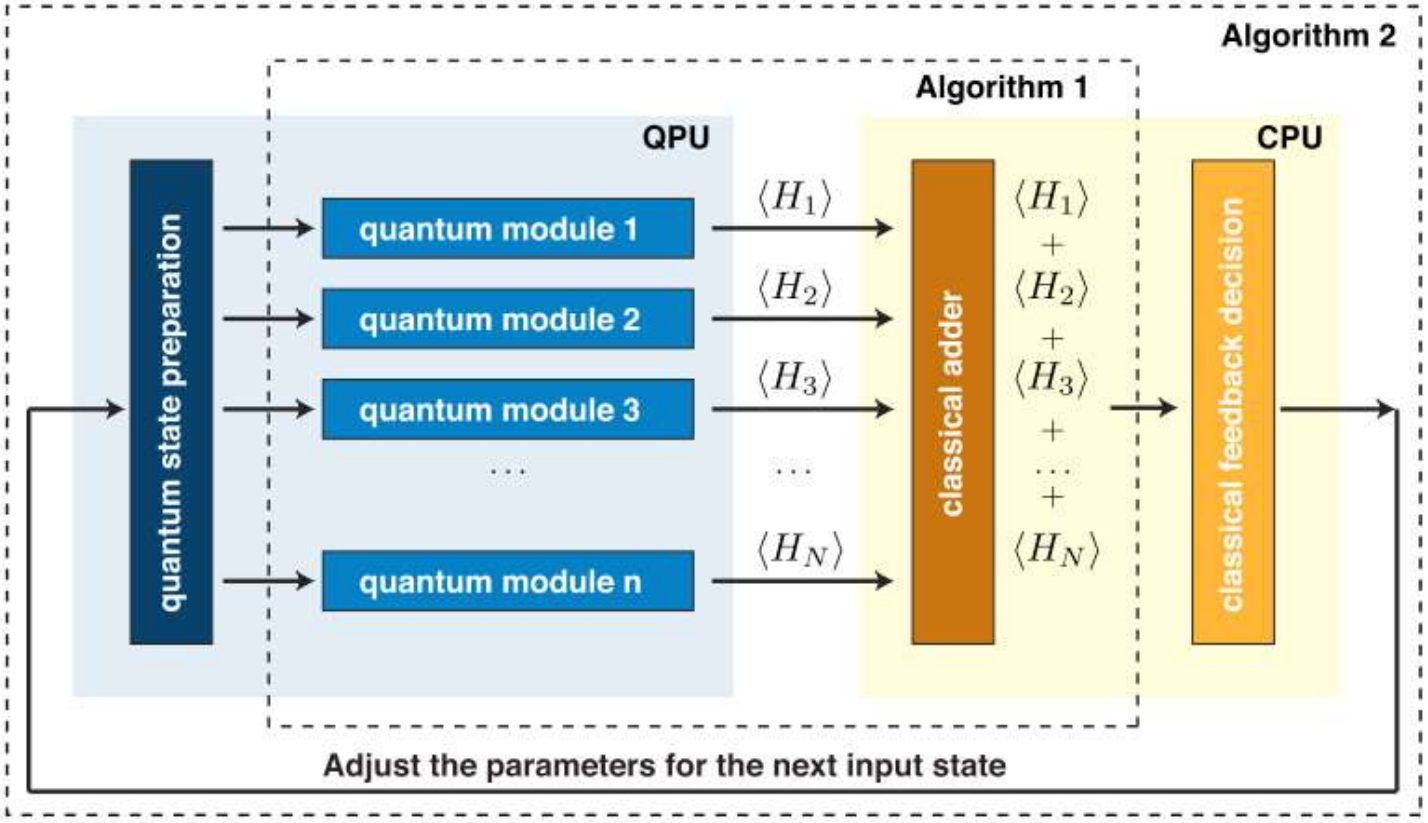


Qubit	T1 (us)	T2 (us)	Frequency (G	Anharmonici	Readout assi
0	429.466092	226.081228	4.63565818	-0.313276	0.0034
1	247.246073	147.812876	4.73628579	-0.3129181	0.0122
2	269.67534	173.177742	4.81918435	-0.3112952	0.0239
3	266.597805	156.894802	4.74720009	-0.3111535	0.0063
4	404.955751	244.408634	4.78786259	-0.3109454	0.0193
5	178.495111	144.195542	4.85085267	-0.3105661	0.0425
6	286.910529	82.3613097	4.89951164	-0.3090558	0.0948

https://quantum.ibm.com/services/resources?system=ibm_sherbrooke&tab=systems

Hybrid Classical and Quantum Algorithm Workflow

- [CPU] Prepare a set of *shallow* parametrized quantum circuits
- [QPU] Execute the quantum circuits
- [CPU] Aggregate the results of the executions and obtain the energy function value
- [CPU] Adjust parameters of the energy function



Source: Peruzzo et al., “A Variational Eigenvalue Solver on a Photonic Quantum Processor.” Nature Comm. 5 (2): 1–10, 2014.

Quantum Computer and Optimization

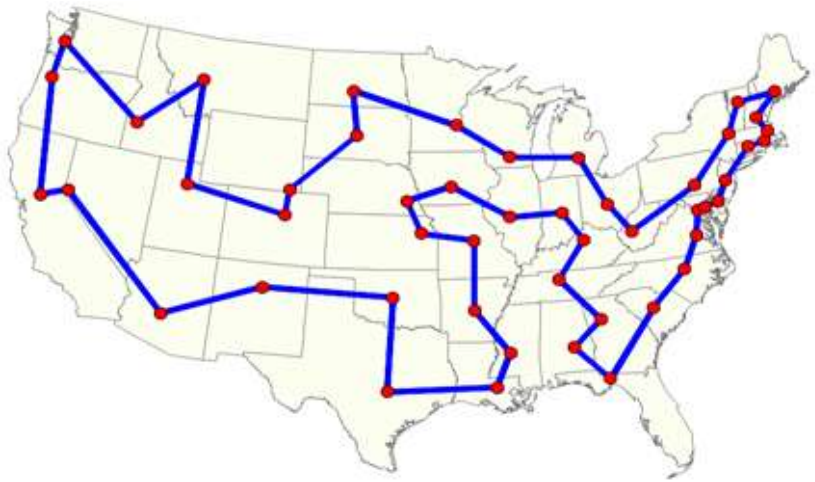
- You may have heard news that some optimization problems are solved by quantum computer more efficiently than classical computer
  - Traveling salesman problem (TSP)
  - Maximum cut problem (Maxcut)

- Some news also explain that superposition can deal with combinatorial optimization problem efficiently because it can compute f(x) of all possible x simultaneously
  - Is it true?

- It is true that you can computer superposition of f(x) of all possible x.
- But you cannot get the minimum/maximum easily…
  - Just applying measurement to f(x) results in f(x) of a random x
    - No guarantee of optimality
  - You may know some quantum algorithms such as Grover search and notice that you need some techniques to obtain relevant solutions.

Traveling Salesman Problem 1/2

- Definition: Given a set of n cities, you are asked to find the shortest route that visits each city and returns to the start.
  - TSP is known to be NP-hard
  - Naïve approach: Check all permutations of the cities. O(n!) time
  - Claim: If you have several tens of cities, it takes millions of years to solve it with conventional computers. Is it true?
    - E.g., 50! ≒ 3 x 10^{64}
- History
  - An instance with 49 cities was solved in 1954(!)
  - An instance with 85900 cities was solved in 2006
- Reference
  - http://www.math.uwaterloo.ca/tsp/index.html



Source: https://physics.aps.org/articles/v10/s32

Year	Research Team	Size of Instance	Name
1954	G. Dantzig, R. Fulkerson, and S. Johnson	49 cities	dantzig42
1971	M. Held and R.M. Karp	64 cities	64 random points
1975	P.M. Camerini, L. Fratta, and F. Maffioli	67 cities	67 random points
1977	M. Grötschel	120 cities	gr120
1980	H. Crowder and M.W. Padberg	318 cities	lin318
1987	M. Padberg and G. Rinaldi	532 cities	att532
1987	M. Grötschel and O. Holland	666 cities	gr666
1987	M. Padberg and G. Rinaldi	2,392 cities	pr2392
1994	D. Applegate, R. Bixby, V. Chvátal, and W. Cook	7,397 cities	pla7397
1998	D. Applegate, R. Bixby, V. Chvátal, and W. Cook	13,509 cities	usa13509
2001	D. Applegate, R. Bixby, V. Chvátal, and W. Cook	15,112 cities	d15112
2004	D. Applegate, R. Bixby, V. Chvátal, W. Cook, and K. Helsgaun	24,978 cities	sw24798

Milestones in the solutions of TSP instances

Traveling Salesman Problem 2/2

- Concorde is the state-of-the-art TSP solver
- I solved a random instance with 50 cities **exactly** within 0.1 second
  - Macbook Pro (13-inch, Early 2015, Core i5 2.7GHz)

- 50 cities 0.06 sec

```
Using random seed 1543820247
Random 50 point set
XSet initial upperbound to 301 (from tour)
  LP Value  1: 291.000000  (0.01 seconds)
  LP Value  2: 298.000000  (0.02 seconds)
  LP Value  3: 301.000000  (0.03 seconds)
New lower bound: 301.000000
Final lower bound 301.000000, upper bound 301.000000
Exact lower bound: 301.000000
DIFF: 0.000000
Final LP has 89 rows, 145 columns, 904 nonzeros
Optimal Solution: 301.00
Number of bbnodes: 1
Total Running Time: 0.06 (seconds)
```

100 cities 0.10 sec

```
Using random seed 1543820289
Random 100 point set
Set initial upperbound to 752 (from tour)
  LP Value  1: 724.000000  (0.01 seconds)
  LP Value  2: 748.666667  (0.03 seconds)
  LP Value  3: 752.000000  (0.04 seconds)
New lower bound: 752.000000
Final lower bound 752.000000, upper bound 752.000000
Exact lower bound: 752.000000
DIFF: 0.000000
Final LP has 143 rows, 258 columns, 1039 nonzeros
Optimal Solution: 752.00
Number of bbnodes: 1
Total Running Time: 0.10 (seconds)
```

200 cities 0.59 sec

```
Using random seed 1543820334
Random 200 point set
Set initial upperbound to 2177 (from tour)
  LP Value  1: 2079.071429  (0.03 seconds)
  LP Value  2: 2150.000000  (0.06 seconds)
  LP Value  3: 2166.691534  (0.13 seconds)
  LP Value  4: 2170.278860  (0.19 seconds)
  LP Value  5: 2171.166667  (0.24 seconds)
  LP Value  6: 2172.261364  (0.28 seconds)
  LP Value  7: 2173.177475  (0.37 seconds)
  LP Value  8: 2174.543860  (0.42 seconds)
  LP Value  9: 2175.000000  (0.49 seconds)
  LP Value 10: 2175.000000  (0.49 seconds)
New lower bound: 2175.000000
New upperbound from x-heuristic: 2175.00
Final lower bound 2175.000000, upper bound 2175.000000
Exact lower bound: 2175.000000
DIFF: 0.000000
Final LP has 315 rows, 544 columns, 3174 nonzeros
Optimal Solution: 2175.00
Number of bbnodes: 1
Total Running Time: 0.59 (seconds)
```

- How about quantum computers?

- Reference: http://www.math.uwaterloo.ca/tsp/concorde/index.html

Brief Introduction of Optimization

$$
\begin{aligned}
&\text{Minimize} && f(x) \\
&\text{Subject to} && g_i(x) = b_i \\
& && h_i(x) \leq c_i \\
& && x = (x_1, \ldots, x_n) \\
& && l_i \leq x_i \leq u_i \\
&x_i\text{: binary / integer / continuous}
\end{aligned}
$$

- Optimization problem consists of three components
  - Decision variables
    - Discrete: binary / integer
    - Continuous
  - Objective function: to be minimized or maximized
  - Constraints: None / Equality / Inequality
- Task: You are asked to find the best solution among all candidates
  - A candidate that satisfied the constraints is called a *feasible solution*

- Various types of optimization problems
  - Linear programming (LP)
  - Mixed integer linear programming (MILP)
  - Quadratic programming (QP)
  - Quadratic unconstrained binary optimization (QUBO)
  - Semidefinite programming (SDP)
  - etc.

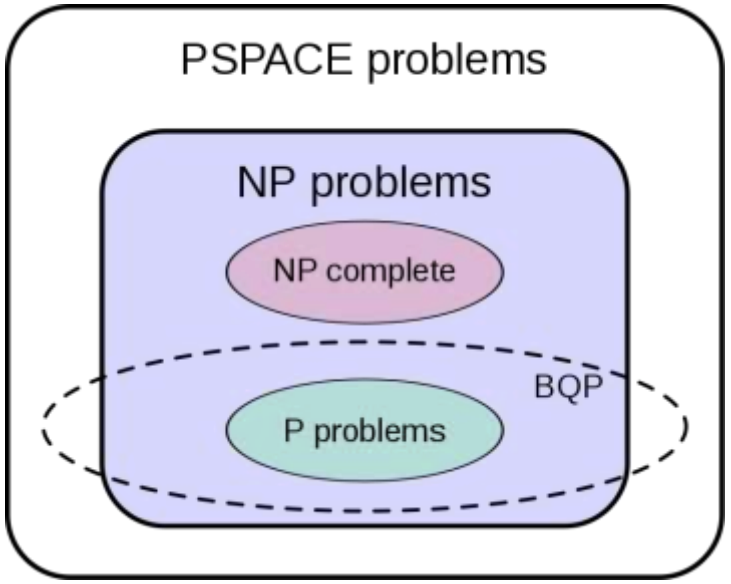
Algorithms to Optimization Problems

- What does “solve” mean?

- Exact algorithm
  - Guarantee of the optimality of the solution
  - E.g., Exhaustive search, Dynamic programming, Branch-and-bound
- Approximation algorithm
  - Solution is not always optimal, but there is a guarantee of distance from the optimal solution
  - E.g., Christofides algorithm (1.5 approximation algorithm to TSP, the solution is 1.5 times longer than the optimal solution at most)
- Heuristics
  - No theoretical guarantee of the quality of solutions, but works well practically
  - E.g., Greedy search, Local search
  - Meta-heuristics
    - Higher level of design of heuristics
    - Often inspired from nature
    - E.g., Genetic algorithm, Simulated annealing, Ant colony optimization

Can Quantum Computers Solve Optimization Problems Efficiently?

- "Efficiently" usually means that a problem is solvable exactly in polynomial time
  - We assume decision problems, which is a yes-no type problem
  - E.g., whether the objective function value ≤ a threshold or not
- Classes of decision problems
  - P: solvable by conventional computers (in polynomial time)
  - BQP: solvable by quantum computers
  - NP complete: very hard to solve with conventional computers
- Does BQP include NP complete?
  - Likely to be No
- Some problems are in BQP \ P
  - integer factorization (Shor's algorithm)
- Quantum computers can be still useful to solve optimization problem by studying quantum-based heuristics



Source: https://en.wikipedia.org/wiki/BQP

Convert Optimization Problem to Hamiltonian

- Input: Quadratic programming problem
  - Only binary and integer variables (no continuous variables)
    - $x = (x_1, \dots, x_n)$
    - Integer variables have lower bounds and upper bounds
  - Objective function: minimize $f(x) = x^T Ax + Bx$
    - Maximization problem can be converted into a minimization problem: $\text{minimize } -f(x)$
- Output: Hamiltonian $H = \Sigma_i w_i P_i$
  - Map $\psi(x) = x'$ from variable values x to a bitstring x'
  - The objective function value is equal to the expectation value: $f(x) = \langle \psi(x)|H|\psi(x) \rangle$

Quadratic programming problem

$$\begin{aligned} &\text{Minimize} && x^T Ax + Bx \\ &\text{Subject to} && c_i^T x \le d_i \\ & && x = (x_1, \dots, x_n) \\ & && l_i \le x_i \le u_i \\ & && x_i\text{: binary / integer} \end{aligned}$$

Quadratic unconstrained binary optimization (QUBO)

$$\begin{aligned} &\text{Minimize} && x'^T A' x' + B' x' \\ & && + \lambda \Sigma_i \left(c'^T_i x' + s_i - d'_i \right)^2 \\ &\text{Subject to} && x' = (x'_1, \dots, x'_m) \\ & && x'_i\text{: binary} \end{aligned}$$

Hamiltonian

$H = \Sigma_i w_i P_i$

P_i: Pauli string (e.g., $ZI = Z \otimes I$)

= tensor product of Pauli matrices

Quadratic Program to QUBO

- Convert inequality constraints into equality constraints
  - $x \leq b \ \rightarrow x + s = b, \quad 0 \leq s \leq u$ slack variable s
- Convert equality constraints into penalties and add them to the objective function
  - $x = b \ \rightarrow \lambda(x - b)^2, \quad \lambda$: positive constant
- Encode integer variables with binary variables
  - $x = l + \sum_{i=0} 2^i \cdot y_i + k \cdot y\,, \quad y_i, y$: binary variables

Quadratic programming problem

$$\begin{aligned} &\text{Minimize} && x^T A x + Bx \\ &\text{Subject to} && c_i^T x \leq d_i \\ & && x = (x_1, \dots, x_n) \\ & && l_i \leq x_i \leq u_i \\ & && x_i\text{: binary / integer} \end{aligned}$$

Quadratic unconstrained binary optimization (QUBO)

$$\begin{aligned} &\text{Minimize} && x'^T A' x' + B' x' \\ & && + \lambda \Sigma_i \left(c'^T_i x' + s_i - d'_i\right)^2 \\ &\text{Subject to} && x' = (x'_1, \dots, x'_m) \\ & && x'_i\text{: binary} \end{aligned}$$

Hamiltonian

$H = \Sigma_i w_i P_i$

P_i: Pauli string (e.g., $ZI = Z \otimes I$)

= tensor product of Pauli matrices

QUBO to Hamiltonian

- Replace binary variables x' with integer variables z
  - $x'_i \to (1 - z_i)/2$ where $z_i \in \{-1, 1\}$
  - $x'_i = 0 \leftrightarrow z_i = 1,\ x'_i = 1 \leftrightarrow z_i = -1$
    - $\langle 0|Z|0\rangle = 1, \langle 1|Z|1\rangle = -1$
- Replace integer variables z with Pauli Z
  - $z_i \to Z_i\ (= I \otimes I \otimes \cdots \overset{i}{Z} \cdots \otimes \overset{2}{I} \otimes \overset{1}{I})$
  - Constant value $\to I^{\otimes n}$
  - $|\psi(x)\rangle = |x'_n\rangle \otimes \cdots |x'_i\rangle \cdots \otimes |x'_1\rangle$
    - $x'_i = 0 \leftrightarrow |0\rangle,\ x'_i = 1 \leftrightarrow |1\rangle$ on i-th qubit

Quadratic programming problem

$$\begin{aligned}\text{Minimize}\quad & x^T A x + Bx \\ \text{Subject to}\quad & c_i^T x \le d_i \\ & x = (x_1, \ldots, x_n) \\ & l_i \le x_i \le u_i \\ & x_i\text{: binary / integer}\end{aligned}$$

Quadratic unconstrained binary optimization (QUBO)

$$\begin{aligned}\text{Minimize}\quad & x'^T A' x' + B' x' \\ & + \lambda \Sigma_i \left(c'^T_i x' + s_i - d'_i\right)^2 \\ \text{Subject to}\quad & x' = (x'_1, \ldots, x'_m) \\ & x'_i\text{: binary}\end{aligned}$$

Hamiltonian

$H = \Sigma_i w_i P_i$

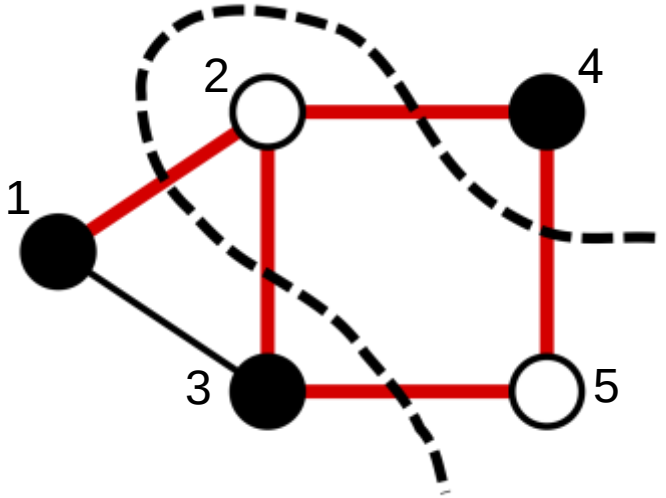
P_i: Pauli string (e.g., $ZI = Z \otimes I$)
= tensor product of Pauli matrices

Example: Maxcut Problem 1/2

- Given a graph G = (V, E) and weights w_{ij} of edges (i, j)
  - $w_{ij} > 0$ and $w_{ij}=w_{ji}$
- Separate nodes into two groups such that you maximize the sum of weights between the groups
- Formulation

$$\text{Maximize } \sum_{(i,j)\in E}[w_{ij}x_i(1-x_j)+w_{ij}(1-x_i)x_j]$$
$$\text{Subject to } x_i \in \{0,1\}, i \in V$$

Take a sum of the cases $(x_i, x_j) = (0,1), (1,0)$



Source: https://en.wikipedia.org/wiki/Maximum_cut

- Conversion

$$\text{Minimize } \sum_{(i,j)\in E}[-\frac{w_{ij}}{4}(1-z_i)(1+z_j)-\frac{w_{ij}}{4}(1+z_i)(1-z_j)] = \sum_{(i,j)\in E}[-\frac{w_{ij}}{2}(1-z_iz_j)]$$
$$\text{Subject to } z_i \in \{-1,1\}, i \in V$$

Note: multiply -1 to change maximization into minimization

$$H = \sum_{(i,j)\in E}\frac{w_{ij}}{2}Z_iZ_j - \sum_{(i,j)\in E}\frac{w_{ij}}{2}I^{\otimes n}$$

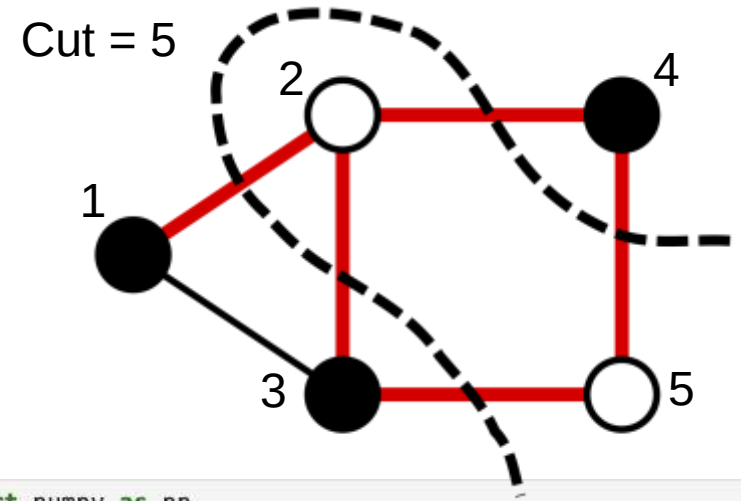
Note: replace z_i with Z_i

where $Z_i = I^{\otimes n-i-1} \otimes Z \otimes I^{\otimes i-1}, i \in V$ (*i*-th position is Z, otherwise I)

Example: Maxcut Problem 2/2

Black: 0, White: 1

$x = (x_1, x_2, x_3, x_4, x_5) = (0, 1, 0, 0, 1)$



- Formulation
  - Node i corresponds to binary variable x_i
  - Edge weight = 1 for all 6 edges
    - [(1,2), (1,3), (2,3), (2,4), (3,5), (4,5)]

Maximize $cut(x) = \sum_{(i,j)\in E}[x_i(1-x_j) + (1-x_i)x_j]$

Subject to $x_i \in \{0,1\}, i \in [1, \ldots, 5]$

- Convert the original problem into a minimization problem

Minimize $f(x) = -cut(x) = 2(x_1x_2 + x_1x_3 + x_2x_3 + x_2x_4 + x_3x_5 + x_4x_5)$
$-2x_1 - 3x_2 - 3x_3 - 2x_4 - 2x_5$

Subject to $x_i \in \{0,1\}, i \in [1, \ldots, 5]$

$f\big((0,1,0,0,1)\big) = -3x_2 - 2x_5 = -5$

- Hamiltonian

$H = \frac{1}{2}(Z_1Z_2 + Z_1Z_3 + Z_2Z_3 + Z_2Z_4 + Z_3Z_5 + Z_4Z_5) - 3I^{\otimes 5}$

Minimum eigenvalue = -5

$\langle 10010|H|10010\rangle = -5 = f(x)$

```
import numpy as np
from qiskit.quantum_info import SparsePauliOp, Statevector
```

```
op = SparsePauliOp.from_list([
    ("IIIZZ", 0.5),
    ("IIZIZ", 0.5),
    ("IIZZI", 0.5),
    ("IZIZI", 0.5),
    ("ZIZII", 0.5),
    ("ZZIII", 0.5),
    ("IIIII", -3),
])
```

```
print("Minimum eigenvalue",
      np.linalg.eigvalsh(op.to_matrix()).min())
```

```
Minimum eigenvalue -5.0
```

```
vec = Statevector.from_label("10010")
print("Expectation value",
      np.real_if_close(vec.expectation_value(op)))
```

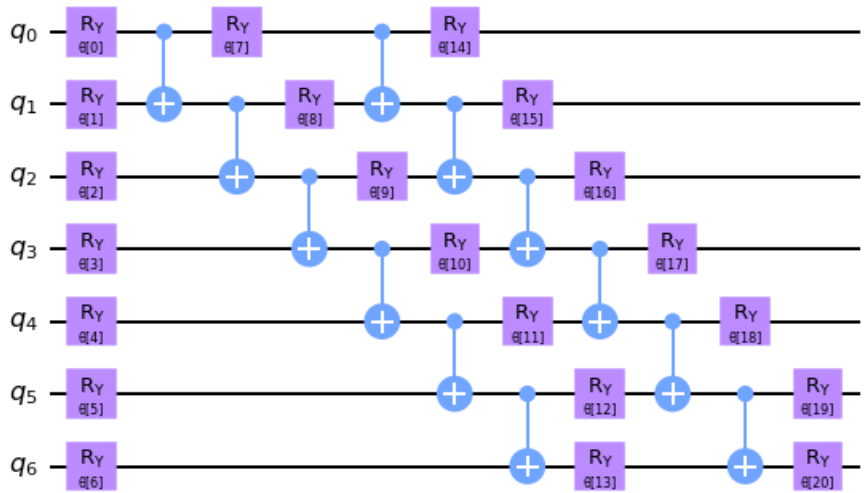
```
Expectation value -5.0
```

Hamiltonian and Variational Principle

- Basic strategy: represent a problem with a matrix (Hamiltonian) and find the minimum eigenvalue and eigenvector
  - Minimum eigenvector (eigenvalue) is often called ground state (energy)
- Hermitian matrix
  - Self-adjoint matrix, i.e., $H^\dagger = H$
  - Eigenvalues are real
    - Suppose $|\psi_i\rangle$ and λ_i are an eigenvector (eigenvalue) of H, it holds $\langle\psi_i|H|\psi_i\rangle=\lambda_i\langle\psi_i|\psi_i\rangle=\lambda_i$
    - $\langle\psi_i|H^\dagger|\psi_i\rangle$ $(=\langle H\psi_i|\psi_i\rangle = \lambda_i^*\langle\psi_i|\psi_i\rangle = \lambda_i^*) = \langle\psi_i|H|\psi_i\rangle = \lambda_i \rightarrow \lambda_i = \lambda_i^*$
  - H can be expressed using eigenvalues and eigenvectors as follows
    - $H=\sum_{i=1}^{N}\lambda_i|\psi_i\rangle\langle\psi_i|$
- Variational principle
  - Let $\langle H\rangle_\phi \equiv \langle\phi|H|\phi\rangle$ denote the expectation value of H on a quantum state $|\phi\rangle$
  - $\langle H\rangle_\phi=\langle\phi|H|\phi\rangle=\sum_{i=1}^{N}\lambda_i|\langle\psi_i|\phi\rangle|^2$
  - $|\langle\psi_i|\phi\rangle|^2 \geq 0 \rightarrow \langle H\rangle_\phi \geq \lambda_{\text{min}}$, $\langle H\rangle_{\psi_{\mathbf{min}}} = \lambda_{\mathbf{min}}$ ($|\psi_{\text{min}}\rangle$ is the ground state)
  - We can get an approximate ground state by trying various quantum states
    - Pick up the quantum state with the smallest expectation value

Variational Quantum Eigensolver

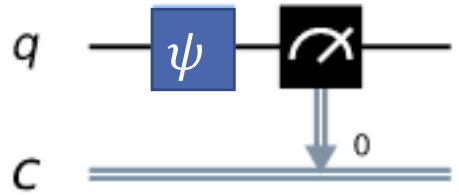
- Task: Given a Hamiltonian H, guess a quantum state $|\psi\rangle$ that minimizes the expectation value $\langle H \rangle_\psi$
- Variational forms
  - Parametrized quantum circuit (PQC) representing a unitary transformation $U(\theta)$
- Ansatz: guessed quantum state
  - $|\psi(\theta)\rangle \equiv U(\theta)|\rho\rangle$
    - $|\rho\rangle$: reference state, e.g., $|0\rangle^{\otimes n}$ and $|+\rangle^{\otimes n}$
- Variational quantum eigensolver (VQE)
  - Quantum part
    - Execute the quantum circuits (ansatz) and get probabilities or counts of bitstrings
  - Classical part
    - Calculate the expectation value $\langle H \rangle_{\psi(\theta)}$ by aggregating the probabilities
    - Adjust (optimize) θ of the variational form
  - Return $\text{argmin}_{\psi(\theta)} \langle H \rangle_{\psi(\theta)}$ as an approximate ground state



Example of variational form (`RealAmplitudes` of Qiskit)

Compute Expectation Values from Probabilities

- Hamiltonian is usually expressed as a sum of Pauli strings
  - n-qubit Pauli string is a tensor product of n Pauli matrices (including I)
    - E.g., $H = 2 \times I \otimes X \otimes Y + 3 \times X \otimes Z \otimes Z$ (2 Pauli strings with 3 qubits)
  - We usually use only Z and I for optimization problems
  - Recall that Pauli matrices and I are Hermitian (and unitary too)
    - $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
- Example: $H = I$ or Z and $|\psi\rangle = a|0\rangle + b|1\rangle$
  - If we execute the following circuit, we will get probabilities of bitstrings {'0': $|a|^2$, '1': $|b|^2$}
    - We need to convert counts to probabilities
  - $\langle I \rangle_\psi = \langle\psi|\psi\rangle = |a|^2 + |b|^2 = 1$
  - $\langle Z \rangle_\psi = \langle\psi|Z|\psi\rangle = \langle\psi|(a|0\rangle - b|1\rangle) = |a|^2 - |b|^2$
  - What if $\langle Z \otimes Z \rangle$?
    - Hint: there are 4 bitstrings, i.e., '00', '01', '10', and '11'
  - What if $\langle X \rangle$?
    - Hint: you need a trick to diagonalize a matrix



Optimize Parameters

- You need to solve an optimization problem
  - $\min_\theta g(\theta)$, where $g(\theta) = \langle H \rangle_{\psi(\theta)}$
  - Concern: Is it as difficult as the original optimization problem?

- If you optimize $g(\theta)$ directly, it is a black-box optimization
  - It is hard to solve a large-scale problem
  - Nelder-Mead method, Powell method, COBYLA

- By taking advantage of gradients $\nabla g(\theta)$, you can deal with a larger-scale problem
  - Two types of gradients: Numerical gradient, Analytical gradient
  - Gradient-based algorithm: Simultaneous Perturbation Stochastic Approximation optimizer (SPSA), quasi-Newton method, trust region method
    - Mari, et al. 2021. “Estimating the gradient and higher-order derivatives on quantum hardware,” Physical Review A, 103(1), 012405.

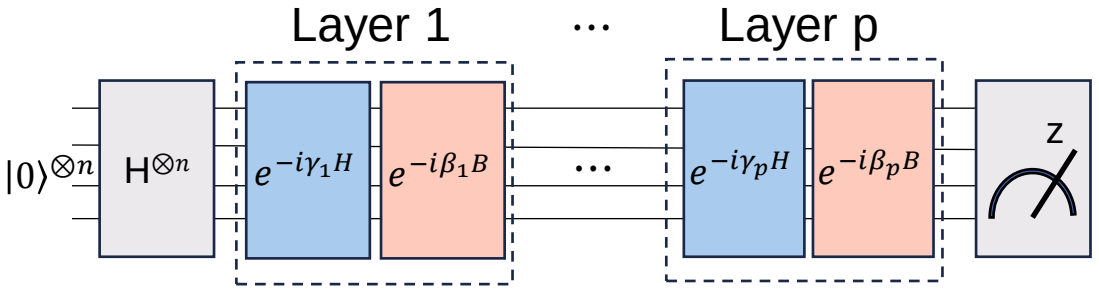
- By taking advantage of some properties of $g(\theta)$ (unitary), you can directly find out local minima w.r.t. specific variables
  - “NFT”: Nakanishi, et al. 2020. “Sequential Minimal Optimization for Quantum-Classical Hybrid Algorithms,” Physical Review Research, 2(4), 043158.
  - “Fraxis”: Watanabe, et al. 2023. “Optimizing Parameterized Quantum Circuits With Free-Axis Single-Qubit Gates,” IEEE Transactions on Quantum Engineering, 4, 1–16.
  - “Rotoselect”: Ostaszewski, et al. 2021. “Structure optimization for parameterized quantum circuits”, Quantum 5, p. 391.

Quantum Approximate Optimization Algorithm (QAOA)

- Focus on finding a good approximation solution to combinatorial problems (page 10 [1])
  - Approximation algorithm to max-cut of 3-regular graph (0.6924, with p=1)
  - C.f., SDP-based 0.87856-approximation algorithm [2]
- Ansatz (parameter $p \geq 1$; *n* qubits)
  - $|\psi_p(\gamma,\beta)\rangle = e^{-i\beta_p B}e^{-i\gamma_p H} \ldots e^{-i\beta_1 B}e^{-i\gamma_1 H}|+\rangle^{\otimes n}$
  - H: Hamiltonian, $B = \sum_{i=1}{}^{n} X_i$: Mixer
  - $\gamma = (\gamma_1, \ldots, \gamma_p), \beta = (\beta_1, \ldots, \beta_p)$: parameters
  - (Can be seen as discretization (Trotter decomposition) of quantum adiabatic algorithm)
- Expectation value and minimization
  - $F_p(\gamma,\beta) = \langle\psi_p(\gamma,\beta)|H|\psi_p(\gamma,\beta)\rangle$
  - $\gamma^*, \beta^* = \mathrm{argmin}_{\gamma,\beta} F_p(\gamma,\beta)$ gives an approximate ground state $|\psi_p(\gamma^*,\beta^*)\rangle$
    - Apply the same optimizers as VQE does
  - If *p* is large, F_p will be better, but it will be harder to optimize due to more parameters. It will be harder to execute the corresponding circuits with higher *p*
- References

  - [1] Farhi et al. 2014. “A Quantum Approximate Optimization Algorithm,” arxiv:1411.4028.
  - [2] Goemans and Williamson. 1995. “Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming.” JACM 42 (6): 1115–45.

Quantum Circuits of QAOA



- Ansatz (parameter $p \geq 1$; n qubits)
  - $|\psi_p(\gamma,\beta)\rangle = e^{-i\beta_p B}e^{-i\gamma_p H} \ldots e^{-i\beta_1 B}e^{-i\gamma_1 H}|+\rangle^{\otimes n}$
  - H: Hamiltonian, $B = \sum_{i=1}^{n} X_i$: Mixer
- Matrix exponential
  - $e^{A+B} = e^A e^B$ if matrices A and B commute, i.e., $AB = BA$
    - (Otherwise, Trotter decomposition is often used)
  - $H = \sum w_i P_i$: P_i, P_j commute because they are tensor products of Pauli Z and I
  - $B = \sum_{i=1}^{n} X_i$: X_i, X_j commute because they are tensor products of Pauli X and I
  - $e^{-i\frac{w}{2}Z_j} \rightarrow$ RZ gate
  - $e^{-i\frac{w}{2}Z_j Z_k} \rightarrow$ RZZ gate
  - $e^{-i\frac{w}{2}X_j} \rightarrow$ RX gate

$$RZ(\lambda) = exp(-i\frac{\lambda}{2}Z) = \begin{pmatrix} e^{-i\frac{\lambda}{2}} & 0 \\ 0 & e^{i\frac{\lambda}{2}} \end{pmatrix} \qquad RX(\theta) = exp(-i\frac{\theta}{2}X) = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

$$R_{ZZ}(\theta) = exp(-i\frac{\theta}{2}Z\otimes Z) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 & 0 & 0 \\ 0 & e^{i\frac{\theta}{2}} & 0 & 0 \\ 0 & 0 & e^{i\frac{\theta}{2}} & 0 \\ 0 & 0 & 0 & e^{-i\frac{\theta}{2}} \end{pmatrix}$$

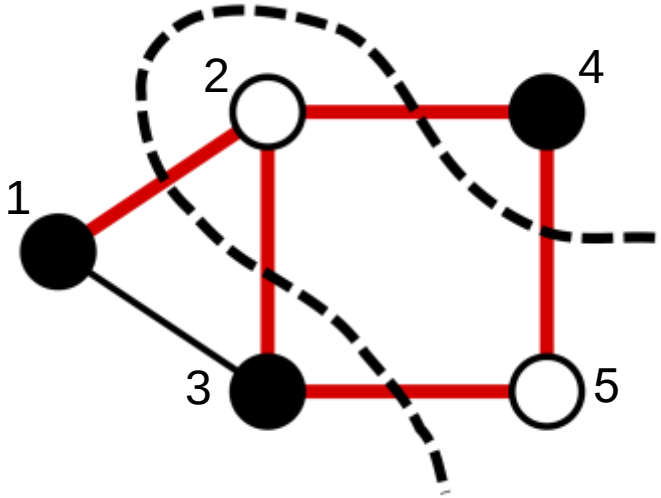
- Reference
  - Qiskit API reference: https://docs.quantum.ibm.com/api/qiskit/circuit_library

Example: Maxcut

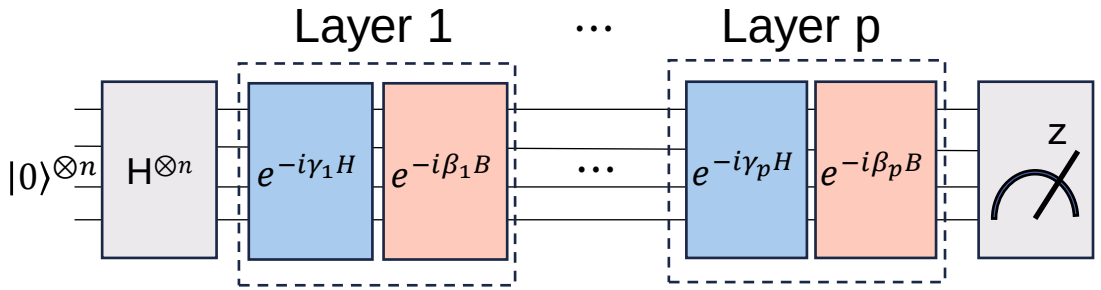
- Hamiltonian

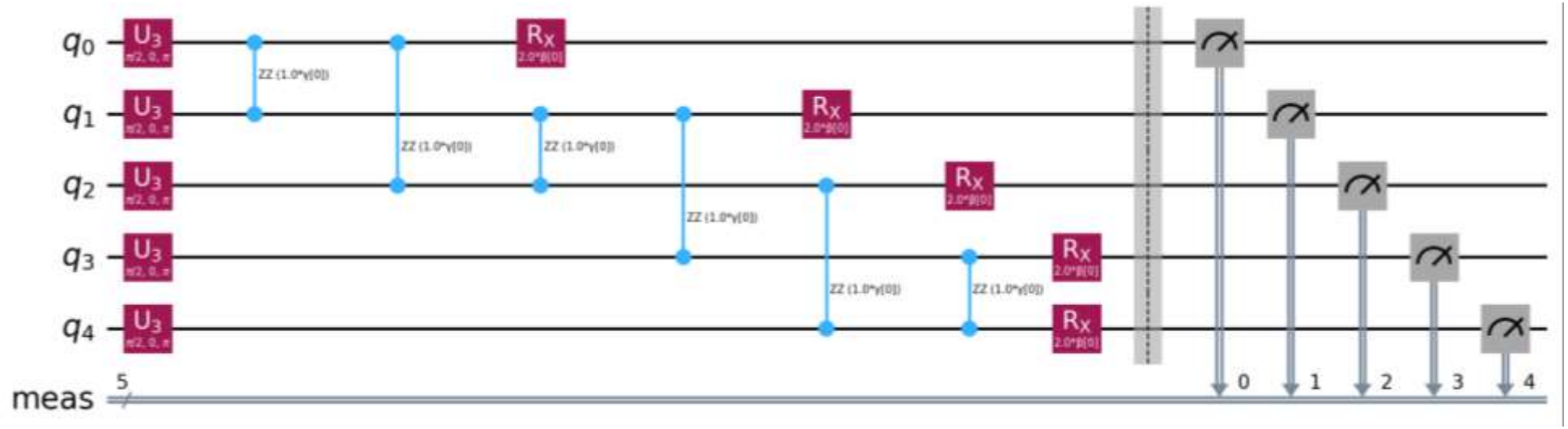
$$H = \frac{1}{2}(Z_1Z_2 + Z_1Z_3 + Z_2Z_3 + Z_2Z_4 + Z_3Z_5 + Z_4Z_5) - 3I^{\otimes 5}$$

Minimum eigenvalue = -5



- QAOA ansatz



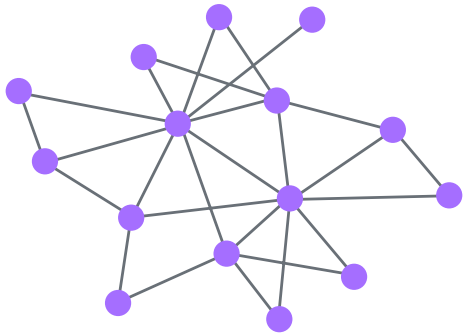


Qiskit Patterns

The anatomy of a quantum algorithm

Step 1

Map classical inputs to a quantum problem



Step 2

Optimize problem for quantum execution.

```
PassManager([UnitarySynthesis(),
             BasisTranslator(),
             EnlargeWithAncilla(),
             AISwap(),
             Collect1qRuns(),
             Optimize1qGates(),
             Collect2qBlocks(),
             ConsolidateBlocks()])
```

Step 3

Execute using Qiskit Runtime Primitives.

Sampler

circuit($\vec{\theta}$)

000101...,
110110...

bit-strings

Estimator

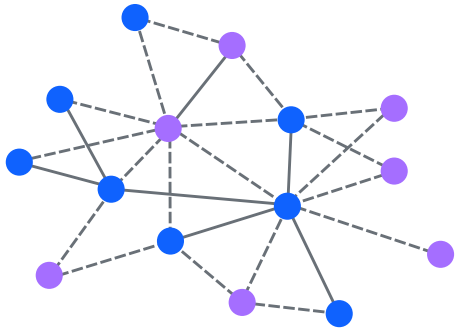
circuit($\vec{\theta}$) +
observable $\hat{O}$

$\langle O \rangle$

expectation value

Step 4

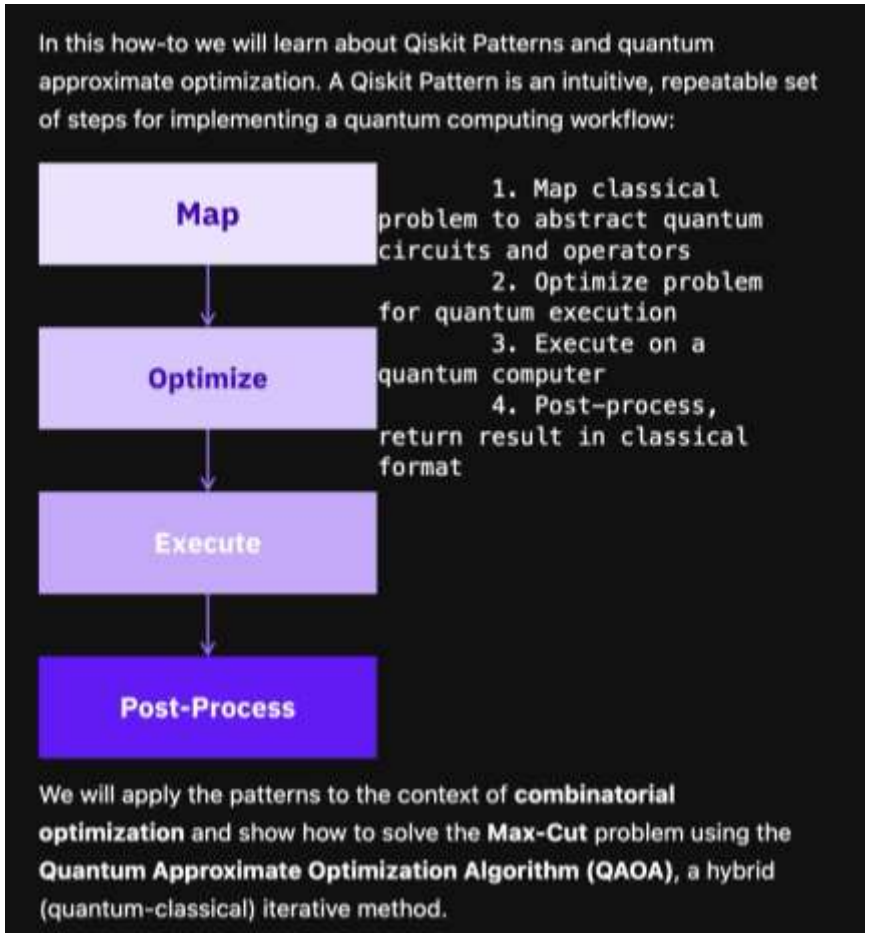
Post-process, return result in classical format.



Hands-on: Quantum Optimization with Qiskit Patterns

- Find the minimum eigenvalue of a simple Hamiltonian with VQE
  - Learn how VQE works
- Solve Maxcut problem with QAOA
  - Learn how Qiskit patterns work for an optimization application

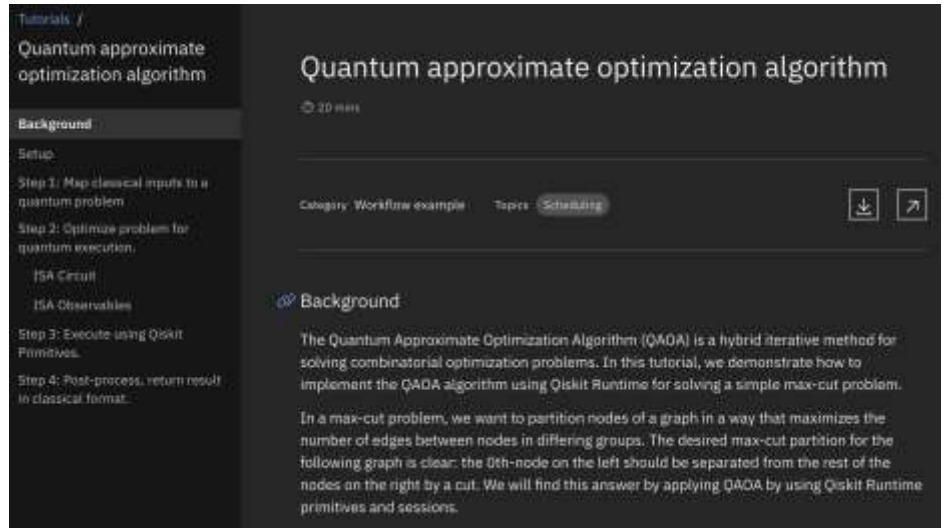
- Prerequisites
  - Download the notebook
  - Setup an environment following the instruction in the notebook
    - You need to install qiskit-optimization



Resources

- IBM Quantum Learning
  - Quantum approximate optimization algorithm
    - https://learning.quantum.ibm.com/tutorial/quantum-approximate-optimization-algorithm
  - Variational Algorithms
    - https://learning.quantum.ibm.com/course/variational-algorithm-design/variational-algorithms
  - Variational quantum eigensolver
    - https://learning.quantum.ibm.com/tutorial/variational-quantum-eigensolver
- Qiskit Optimization
  - Documentation
    - https://qiskit-community.github.io/qiskit-optimization/#
  - Tutorials
    - https://qiskit-community.github.io/qiskit-optimization/tutorials/index.html
  - GitHub
    - https://github.com/qiskit-community/qiskit-optimization
  - Best practices
    - https://github.com/qiskit-community/qopt-best-practices

IBM Quantum Learning



Summary

- Due to noise of the current quantum computers, it’s hard to run deep quantum circuits

- Hybrid quantum-classical appraoch is useful to deal with the noise
  - Converts original problems into Hamiltonians
  - Finds the minimum eigenvalue and eigenvector of the Hamiltonian

- Introduced the overview of optimization and how to convert optimization problems into Hamiltonians
  - Quadratic program → QUBO → Hamiltonian

- Introduced how VQE computes the minimum eigenvalue
  - QAOA for optimization problems