Quantum computing lecture 3:

Variational Algorithms

Wenyang Qian

12-1 PM, Apr 26, 2024

University of California, Los Angeles

What we will learn

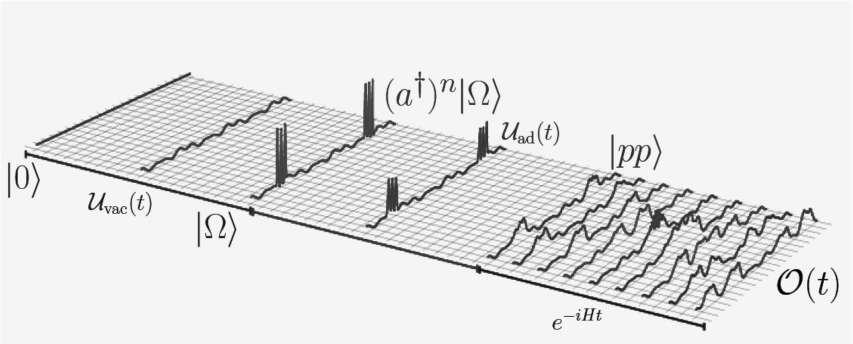
1. How to simulate exp(- i H t) on quantum circuit?

2. How to perform measurement to extract any observable <O>?

3. How to study real world problems using variational algorithms?

1. What does simulation exp(- i H t) mean?

It literally means evolving some quantum state by applying an Hamiltonian H for a period of time t.



Hamiltonian as in Pauli matrices

The Hamiltonian H is usually expressed in terms of Pauli matrices, I, X, Y, Z

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

$$\vec{\sigma} = (\sigma^0, \sigma^1, \sigma^2, \sigma^3) = (I, X, Y, Z)$$

such that the Hamiltonian is a linear combinations of products of these Pauli matrices

$$H = \sum_i c_i P_i$$

For example:

$$\begin{aligned} H_{ex} &= I_0 I_1 + X_0 X_1 + Y_0 Y_1 + Z_0 Z_1 \\ &= II + XX + YY + ZZ \end{aligned}$$

$$\exp(-iH_{ex}t) = \boxed{\exp(-i[II + XX + YY + ZZ]t)}$$

Trotter Formula

We learned that the Trotter formula could help us reduce the exponential of operator sums to individual exponential terms.

$$\exp(A+B) = \lim_{n\to\infty} \left(\exp(A/n)\exp(B/n)\right)^n$$

When A and B are commutative (AB = BA), the limit vanishes.

Our previous example becomes,

$$\begin{aligned}\exp(-iH_{ex}t) &= \exp(-i\left[II + XX + YY + ZZ\right]t)\\ &= \exp(-iIIt)\exp(-iXXt)\exp(-iYYt)\exp(-iZZt)\end{aligned}$$

But still, how do we time evolve any single Pauli term on the quantum circuit?

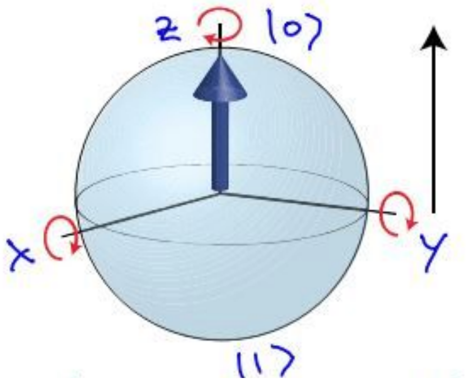
Single qubit simulation

Rotational operators take care single qubit Pauli terms (Taylor expansion)

$$e^{-iXt} = \cos(t)I - i\sin(t)X = \begin{pmatrix} \cos(t) & -i\sin(t) \\ -i\sin(t) & \cos(t) \end{pmatrix} \qquad R_x(2t)$$

$$e^{-iYt} = \cos(t)I - i\sin(t)Y = \begin{pmatrix} \cos(t) & -\sin(t) \\ \sin(t) & \cos(t) \end{pmatrix} \qquad R_y(2t)$$

$$e^{-iZt} = \cos(t)I - i\sin(t)Z = \begin{pmatrix} e^{-it/2} & 0 \\ 0 & e^{it/2} \end{pmatrix} \qquad R_z(2t)$$



Rotational gates on Bloch sphere

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \; |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Exponential of Pauli operators

For multiple qubits, we need the following identity for any operator:

$$\exp(iAt) = \cos(t)I + i\,\sin(t)A, \qquad \text{with } A^2 = I$$

Proof:

$$\begin{aligned}
\exp(iAt) &= I + (iAt) + \frac{(iAt)^2}{2!} + \frac{(iAt)^3}{3!} + \frac{(iAt)^4}{4!} + \frac{(iAt)^5}{5!} + \cdots \\
&= \underbrace{I + \frac{(iAt)^2}{2!} + \frac{(iAt)^4}{4!} + \cdots}_{\text{even powers}} + \underbrace{(iAt) + \frac{(iAt)^3}{3!} + \frac{(iAt)^5}{5!} + \cdots}_{\text{odd powers}} \\
&= I\left(1 - \frac{t^2}{2!} + \frac{t^4}{4!} + \cdots\right) + iA\left(1 - \frac{t^3}{3!} + \frac{t^5}{5!} \cdots\right) \\
&= I\cos(t) + iA\sin(t)
\end{aligned}$$

Not surprisingly, this formula can be applied to any Pauli or Pauli product, since $X^2 = Y^2 = Z^2 = I$

Representation in Z-basis

We are familiar with the Z-basis or computational basis $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

In fact any operations can be represented in this way:

$$\sigma_0 \equiv I \equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = |0\rangle\langle 0| + |1\rangle\langle 1|,$$

$$\sigma_1 \equiv X \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = |1\rangle\langle 0| + |0\rangle\langle 1|,$$

$$\sigma_2 \equiv Y \equiv \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = i\,|1\rangle\langle 0| - i\,|0\rangle\langle 1|,$$

$$\sigma_3 \equiv Z \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|$$

Including the Control-Not gate!

$$\mathrm{CNOT} = \big(|0\rangle\langle 0|\big)_0 \otimes I_1 + \big(|1\rangle\langle 1|\big)_0 \otimes X_1$$

Many qubits, Pauli-ZZ simulation

We start with exp(-i ZZ t), by claiming the following:

$$\exp(-iZ_0Z_1t) = \mathrm{CNOT}_{0,1}\exp(-iZ_1t)\mathrm{CNOT}_{0,1}$$

Proof:

$$RHS = \Big(\big(|0\rangle\langle 0|\big)_0 \otimes I_1 + \big(|1\rangle\langle 1|\big)_0 \otimes X_1\Big) \times \Big(I_0 \otimes \cos(t)I_1 - I_0 \otimes i\sin(t)Z_1\Big) \times \Big(\big(|0\rangle\langle 0|\big)_0 \otimes I_1 + \big(|1\rangle\langle 1|\big)_0 \otimes X_1\Big)$$

$$= \Big(\cos(t)\big(|0\rangle\langle 0|\big)_0 \otimes I_1 - i\sin(t)\big(|0\rangle\langle 0|\big)_0 \otimes Z_1 + \cos(t)\big(|1\rangle\langle 1|\big)_0 \otimes X_1 - i\sin(t)\big(|1\rangle\langle 1|\big)_0 \otimes X_1Z_1\Big)$$

$$\times \Big(\big(|0\rangle\langle 0|\big)_0 \otimes I_1 + \big(|1\rangle\langle 1|\big)_0 \otimes X_1\Big)$$

$$= \cos(t)\big(|0\rangle\langle 0|\big)_0 \otimes I_1 - i\sin(t)\big(|0\rangle\langle 0|\big)_0 \otimes Z_1 + \cos(t)\big(|1\rangle\langle 1|\big)_0 \otimes \underbrace{X_1X_1}_{I_1} - i\sin(t)\big(|1\rangle\langle 1|\big)_0 \otimes \underbrace{X_1Z_1X_1}_{-Z_1}$$

$$= \cos(t)\big(|0\rangle\langle 0| + |1\rangle\langle 1|\big)_0 \otimes I_1 - i\sin(t)\big(|0\rangle\langle 0| - |1\rangle\langle 1|\big)_0 \otimes Z_1$$

$$= \cos(t)I_1 \otimes I_0 - i\sin(t)Z_0 \otimes Z_1 = \exp(-iZ_0 \otimes Z_1t) = LHS$$

Short proof: Pauli matrix are anticommutative.

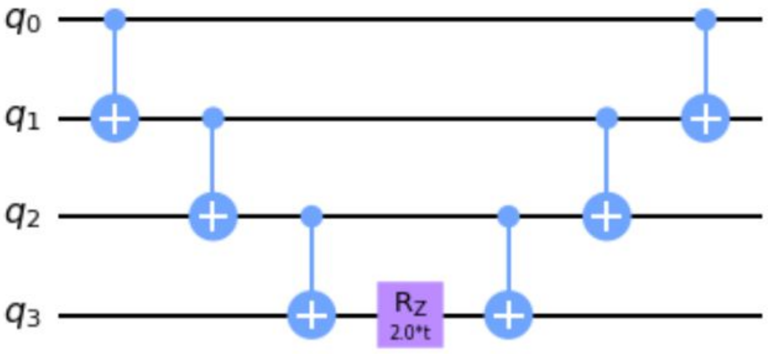
Many qubits, Pauli-ZZ simulation

$$\exp(-iZ_0Z_1t) = \mathrm{CNOT}_{0,1}\exp(-iZ_1t)\mathrm{CNOT}_{0,1}$$

Many qubits, Pauli-ZZ…Z simulation

The results can be generalized to exp(-i ZZ….Z t), where the circuit is a tower of control gates

For example, exp(-i ZZZZ t) becomes



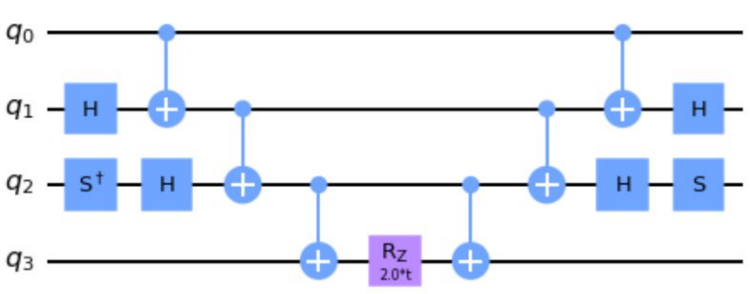
Many qubits, any Pauli simulation

The results can be generalized to exp(-i P t) for any Pauli string P by finding a unitary matrix $\sigma_k^j = U_k Z U_k^\dagger$

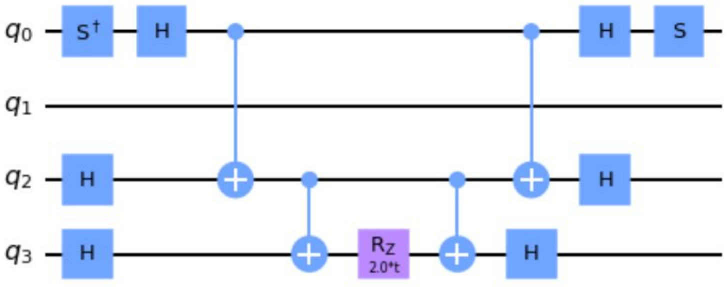
$$X = HZH^\dagger, \quad Y = (SH)Z(SH)^\dagger$$

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad S = \sqrt{Z} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$

because $U_k \exp(-iPt)U_k^\dagger = U_k\big(\cos(t)I - i\sin(t)P\big)U_k^\dagger = \cos(t)I - i\sin(t)(U_k P U_k^\dagger)$ for each qubit k



$\exp(-iZ_3X_2Y_1Z_0t)$



$\exp(-iY_3I_2X_1X_0t)$

Qiskit Lab

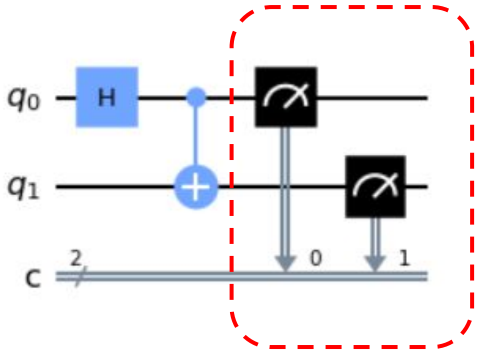
Today, you will have a chance to verify these basic simulation strategies in Qiskit.

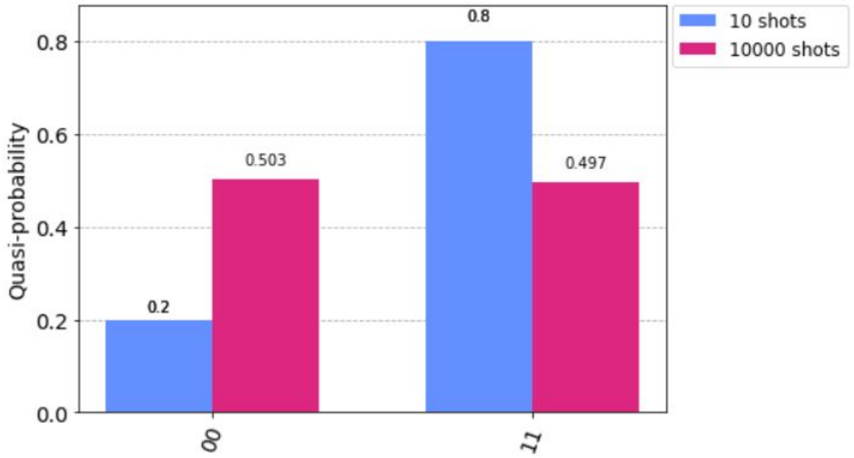
You will learn how to construct Hamiltonian using simple Python commands and build quantum circuit automatically.

2. How to extract observable <O> from measurement?

Firstly, what is measurement?

By quantum mechanics, measurement collapse the wavefunction (in a sense, we lost the full information of the state). Therefore, we must measure/sample the quantum state many times (shots) to partially reconstruct the state.





Observable

- What is observable?

Quantity of interests that can be measured. For example, energy of the state.

- How are observables constructed?

Formally, it is the expectation value of an operator $\langle O \rangle = \langle \psi | \hat{O} | \psi \rangle$

On the quantum circuit, it is computed as a certain weighted sum of probabilities in the computational basis.

Pauli-Z based observable

For example, $\hat{O}_1 = \frac{1}{2} I_0 Z_1$ $\qquad |\psi\rangle = \sum_i c_i |i\rangle = c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle$

Then
$$\hat{O}_1 |\psi\rangle = \frac{1}{2}\Big(c_{00} IZ|00\rangle + c_{01} IZ|01\rangle + c_{10} IZ|10\rangle + c_{11} IZ|11\rangle\Big)$$
$$= \frac{1}{2}\Big(c_{00}|00\rangle - c_{01}|01\rangle - c_{10}|10\rangle + c_{11}|11\rangle\Big)$$

$$Z|0\rangle = |0\rangle$$
$$Z|1\rangle = -|1\rangle$$

$$\langle\psi|\hat{O}_1|\psi\rangle = \frac{1}{2}(|c_{00}|^2 - |c_{01}|^2 - |c_{10}|^2 + |c_{11}|^2)$$
$$= \frac{1}{2}(\mathrm{P}(00) - \mathrm{P}(01) - \mathrm{P}(10) + \mathrm{P}(11))$$

linear combo of probabilities

Obviously, this generalizes to any linear combinations of Pauli-Z observables

Any Pauli observable

What about Pauli-X, Pauli-Y based observables?

Just like the simulation problem, the trick is to append quantum gates that move the calculations to computational basis!

$$X = H^\dagger Z H, \quad Y = H_y^\dagger Z H_y \qquad\qquad H_y = HS^\dagger$$

For example,

$$\begin{aligned}\langle\psi|\, I_0 X_1 Y_2 Z_3\, |\psi\rangle &= \langle\psi|\, I_0 (H^\dagger Z H)_1 (H_y^\dagger Z H_y)_2 Z_3\, |\psi\rangle \\ &= \langle\psi| H_1 (H_y)_2 |\, I_0 Z_1 Z_2 Z_3\, | H_1 (H_y)_2 |\psi\rangle \\ &= \langle\psi'|\, I_0 Z_1 Z_2 Z_3\, |\psi'\rangle\end{aligned}$$

We see that any Pauli observables are just the same as Pauli-Z based observables by appending basis transformation gates.

Qiskit Lab

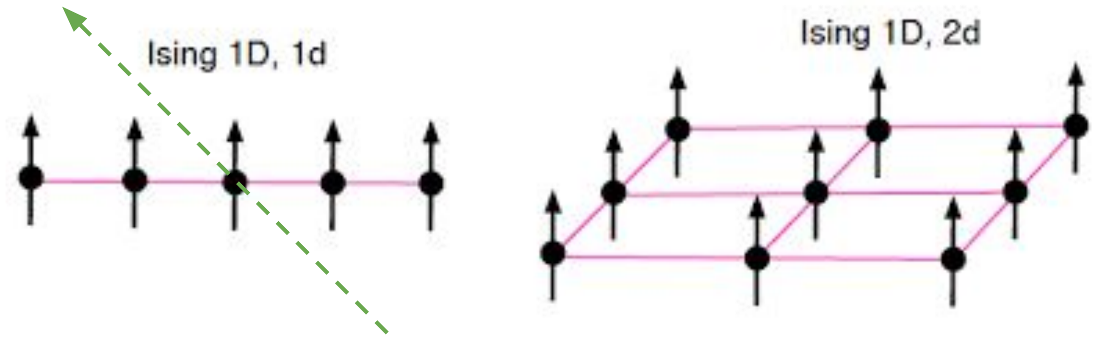
Today, you will have a chance to compute observables in Qiskit. In practice, all these probabilities arithmetics will be grouped and combined to minimize the number of operations.

Often observables are linear combination of Pauli's themselves, so one would need many different quantum circuits to compute the observable. Optimizing (minimizing) the number of circuits is itself an NP-hard problem…

Let us appreciate all the work done by Qiskit under the hood :)

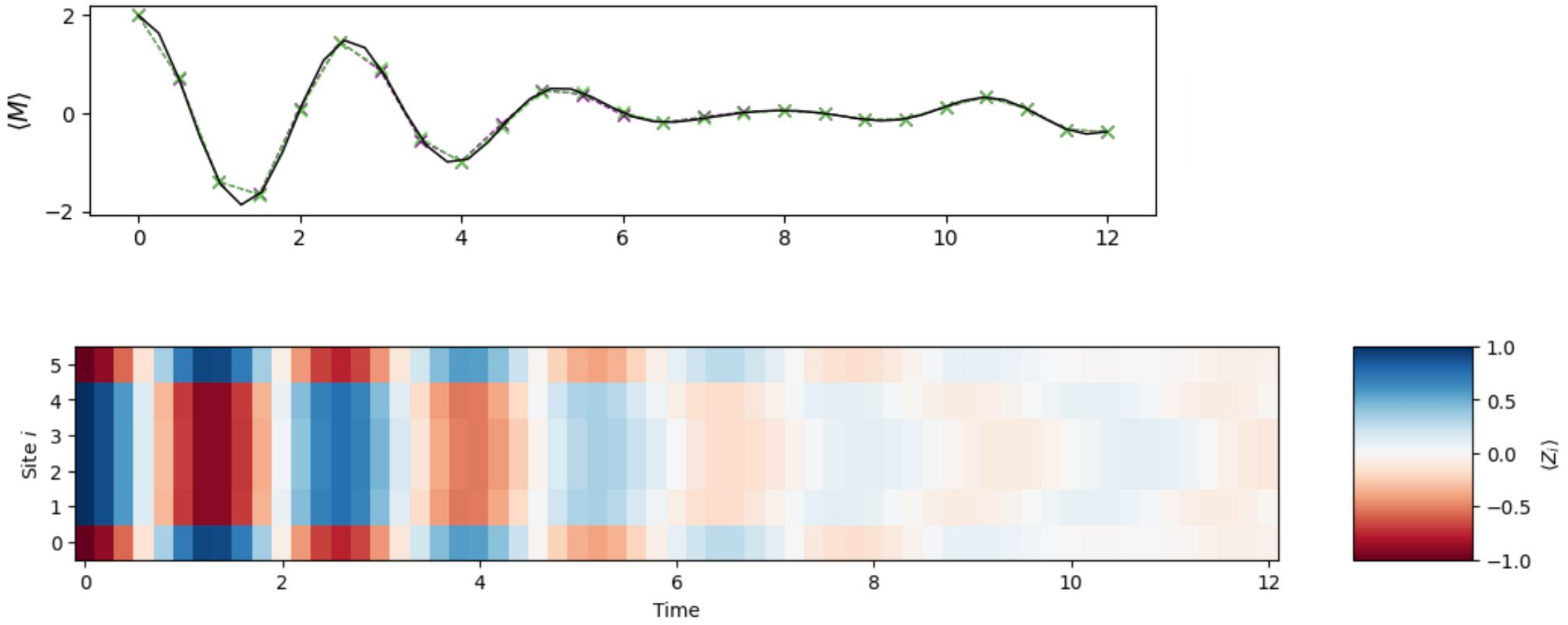
Qiskit Lab

Specifically today in the lab, we are will study the real-time evolution of the Ising model with an arbitrary external field.



Taking the 1D example, we analyze its global property (total magnetization) and local property (spin distribution) evolve over time.

Qiskit Lab



Quantum computing history

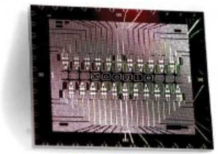
We have really come a long way in past 40 years!

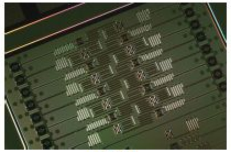
Feynman, "Simulating Physics with Computers" (1981)

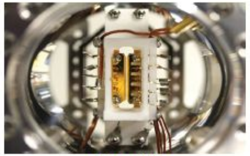
Toffoli Gate (1980) | Shor's Algo (1994) | Error Correction (1995) | Variational Eigensolver (2014) | Quantum Machine Learning (2017)

Currently, we are in the **Noisy Intermediate-Scale Quantum (NISQ)** era…

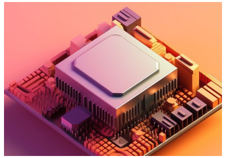
Preskill, Quantum, 2 (2018)


Google


IBM


Rigetti


Intel

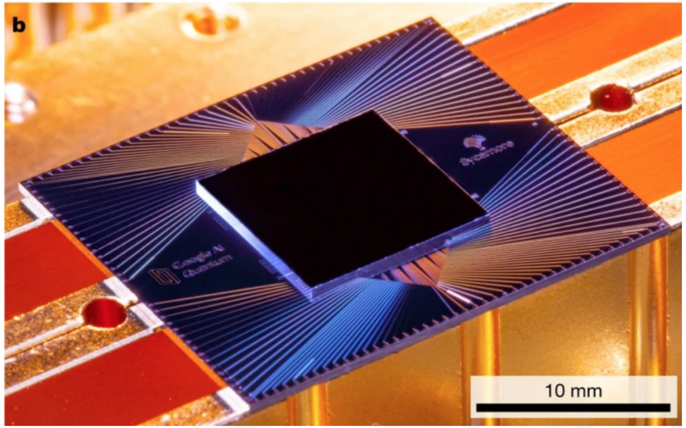

USTC

Qiskit

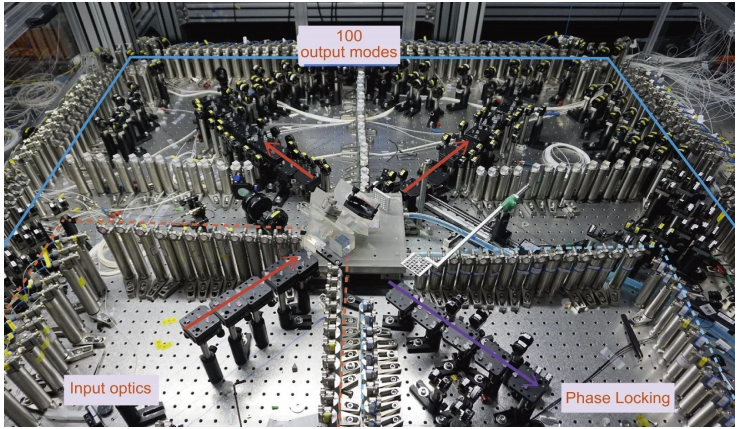
Cirq

XANADU

Towards Quantum Advantage



Google AI Quantum 1910.11333 (2019)



UTSC, Jian-Wei Pan’s group, Science 370, 1460 (2020)

Natural science
Optimization
Finance

Telecommunication
Quantum machine learning
…

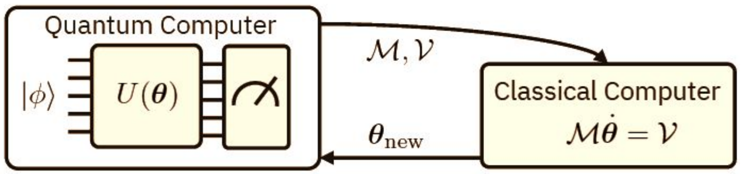
3. Variational algorithms

Perhaps the most successful applications of quantum computing is the variational algorithm.

Based on the Variational Principle $\langle\psi_{\text{guess}}|\, H\, |\psi_{\text{guess}}\rangle \geq \langle E_0\rangle = \langle\psi_0|\, H\, |\psi_0\rangle$

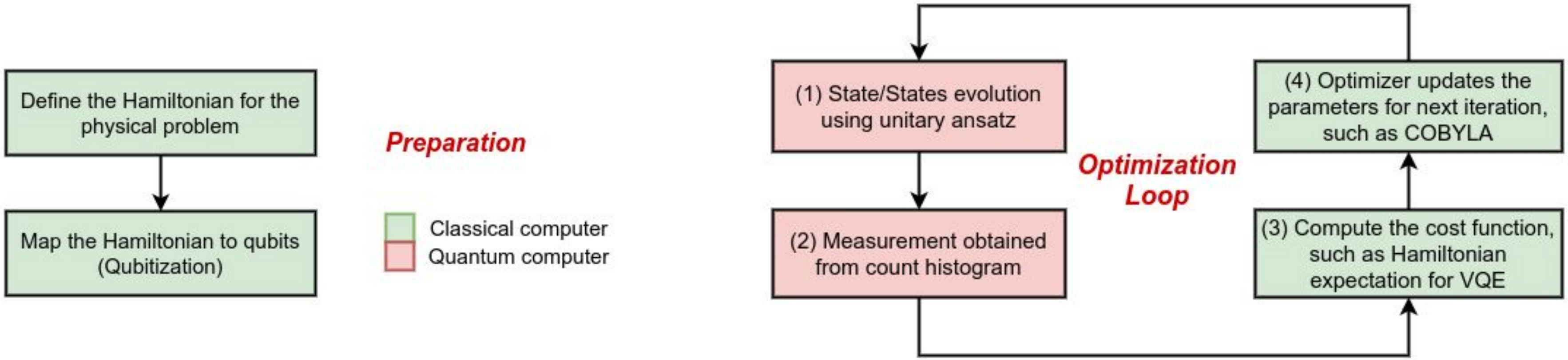
By tweaking the ansatz (guess wavefunction), we can obtain the ground state energy of the Hamiltonian.

Optimal circuit (i.e. solution) to the chemistry or physics problem is obtained by collaboration between quantum device and CPU.



VQE (Variational Quantum Eigensolver)

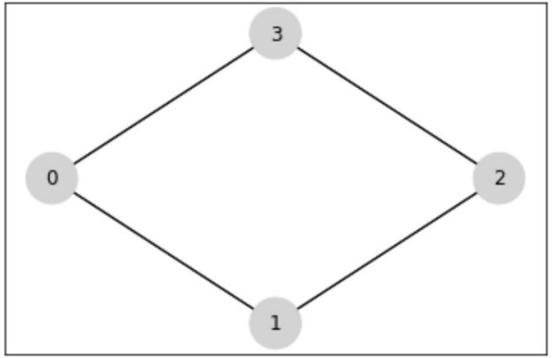
The simplest variational algorithm is VQE



Now, we look at an interesting problem that we can solve with VQE

Maxcut problem (maximum cut)

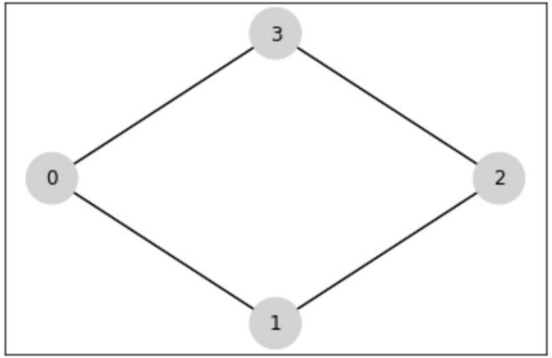
Given a graph of n nodes, how do we cut the graph into two partitions such that the number of edges between the two partitions are maximized?



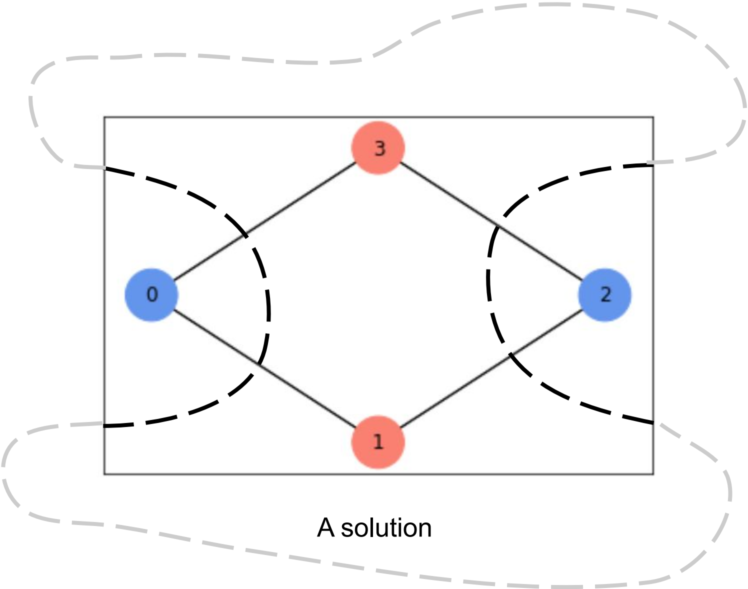
Initial Problem

Maxcut problem (maximum cut)

Given a graph of n nodes, how do we cut the graph into two partitions such that the number of edges between the two partitions are maximized?



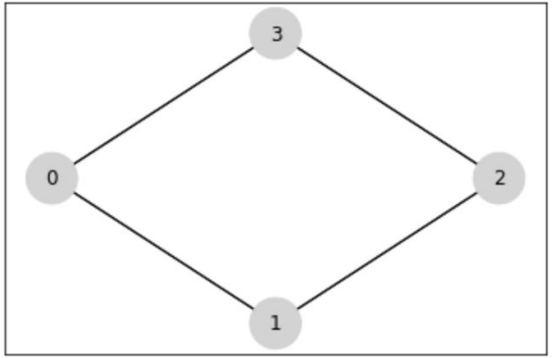
Initial Problem



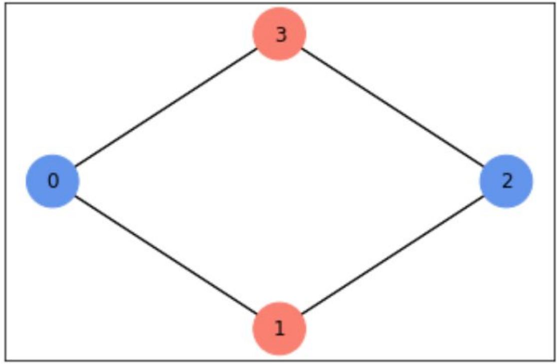
A solution

Maxcut problem (maximum cut)

Given a graph of n nodes, how do we cut the graph into two partitions such that the number of edges between the two partitions are maximized?



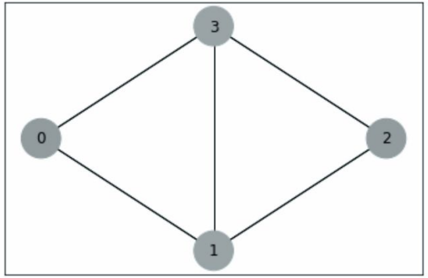
Initial Problem

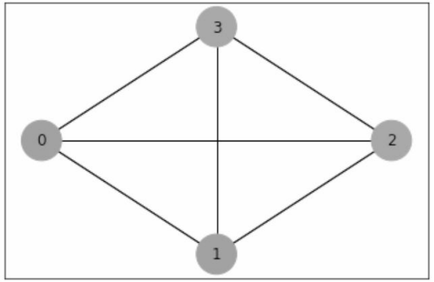


A solution

Why Maxcut? It represents an universal problem; involving lots of active research

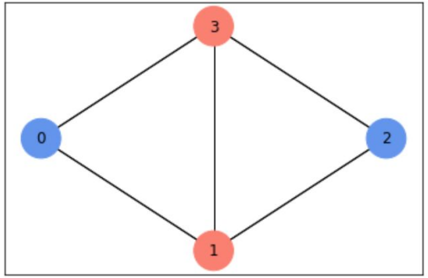
Maxcut problem is NP-Hard!

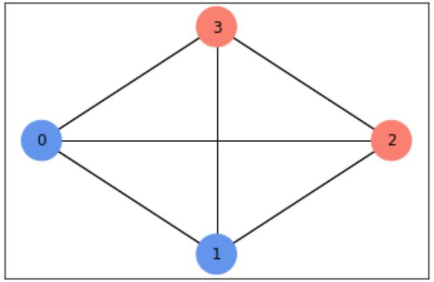




…

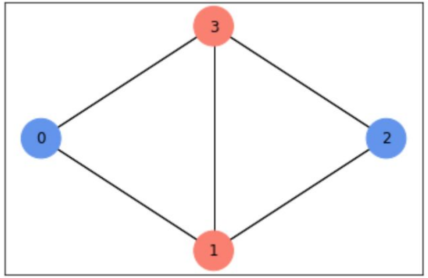
Maxcut problem is NP-Hard!

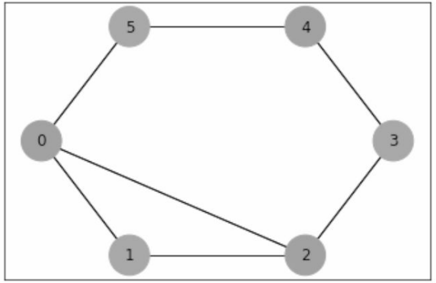


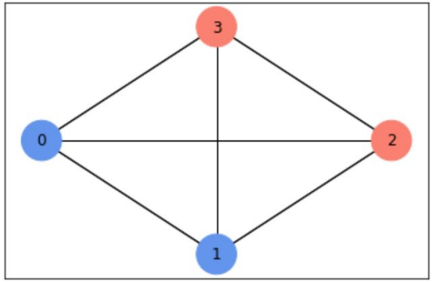


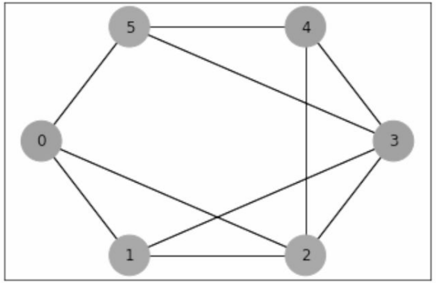
…

Maxcut problem is NP-Hard!



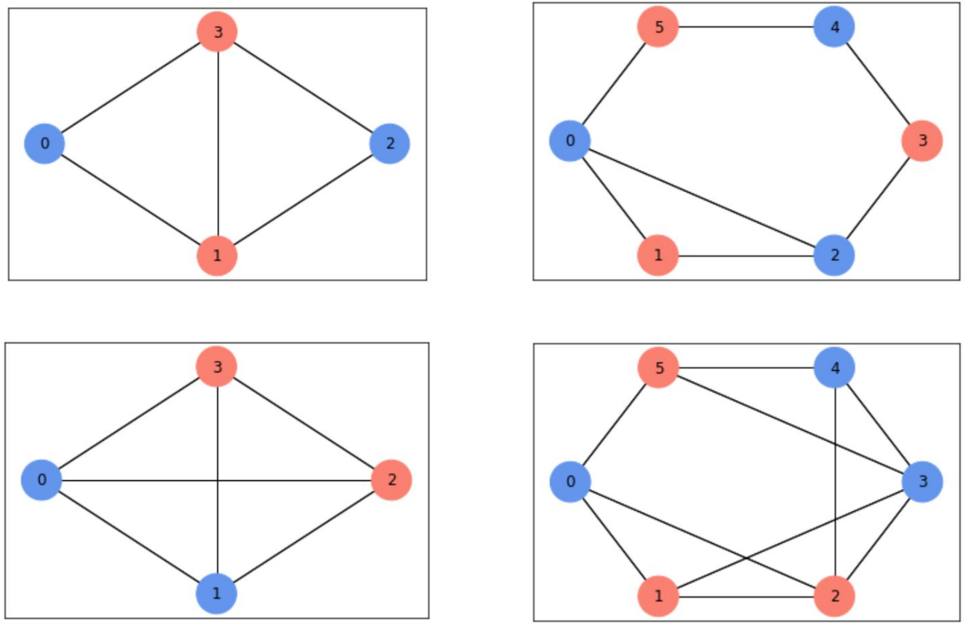






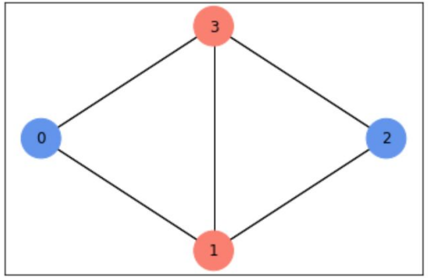
...

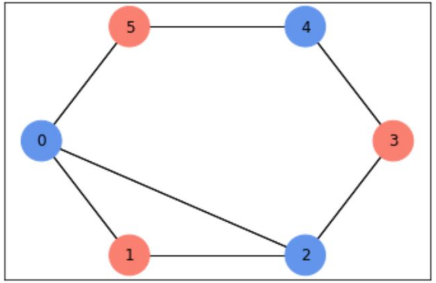
Maxcut problem is NP-Hard!

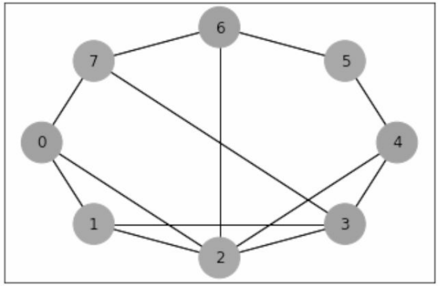


…

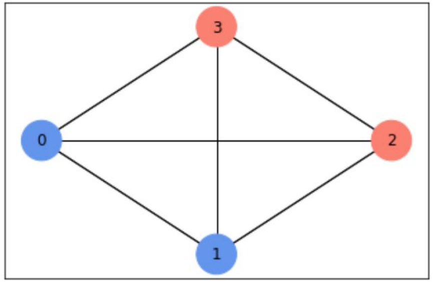
Maxcut problem is NP-Hard!

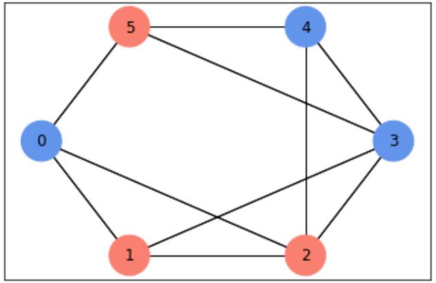


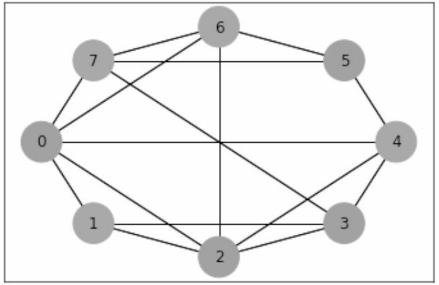




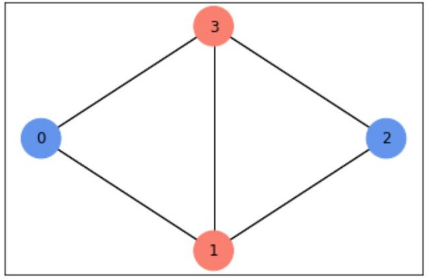
…

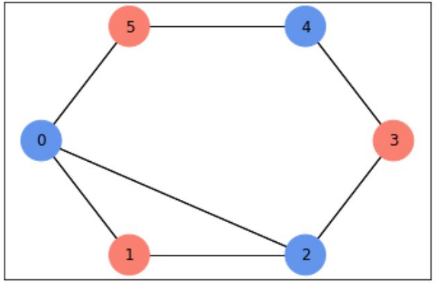


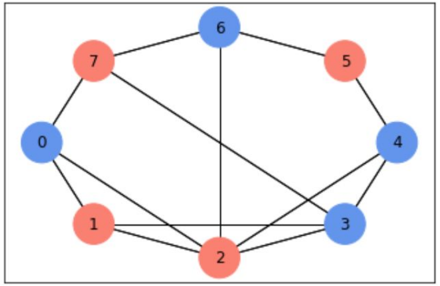




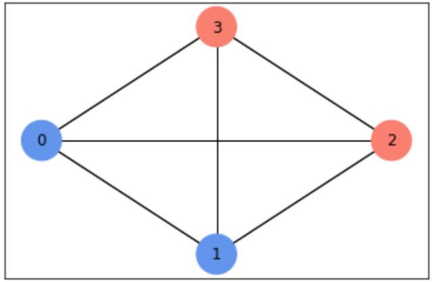
Maxcut problem is NP-Hard!

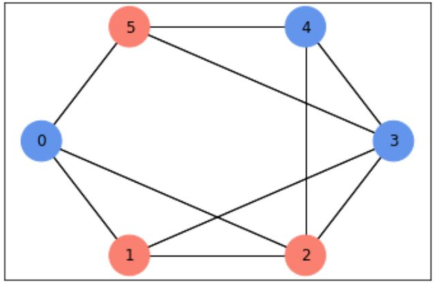


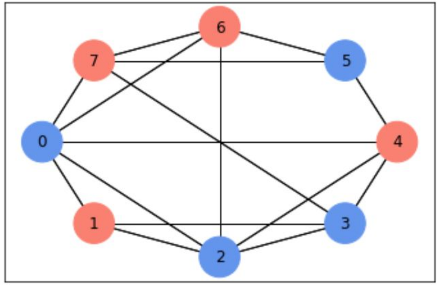




…

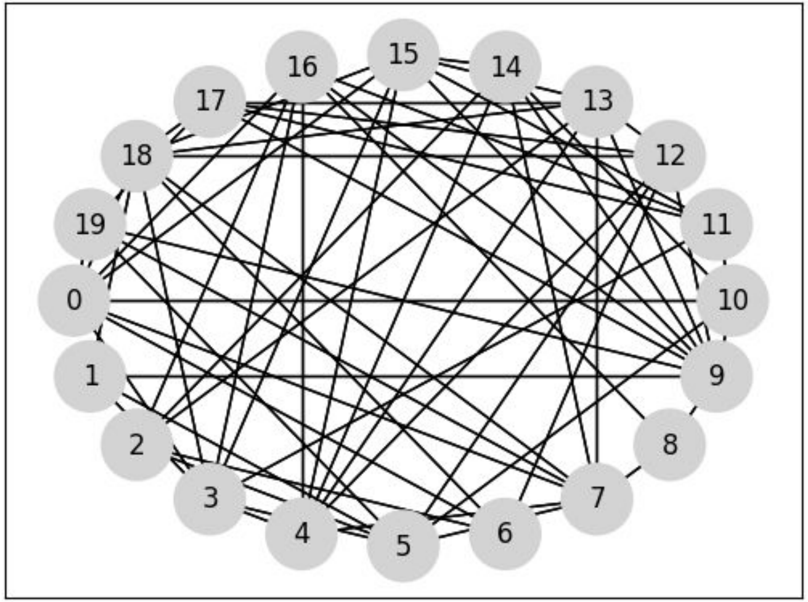






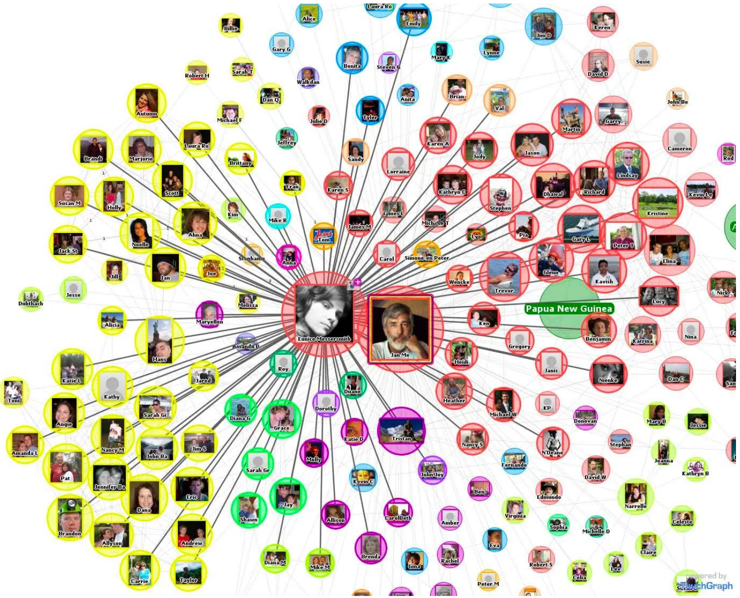
Maxcut problem is NP-Hard!

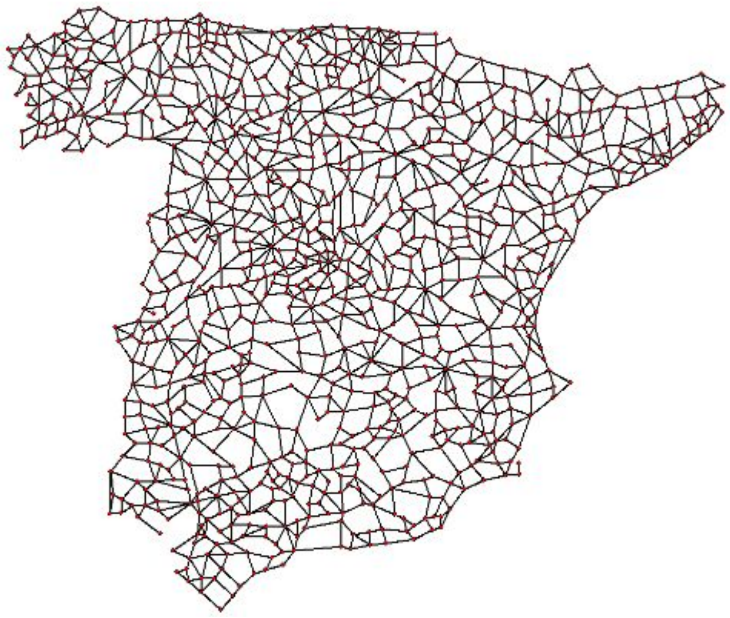
Eventually, bruteforce solutions $\mathcal{O}(n!)$ for large graphs becomes impractical…



n = 20 is already so slow to compute…

Yet, Maxcut problem is so universal & important





Images are from Google Search

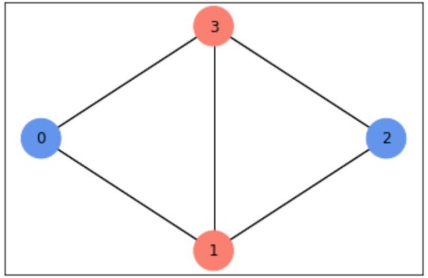
Maxcut problem, binary encoding

How to leverage quantum computing to solve Maxcut problems?

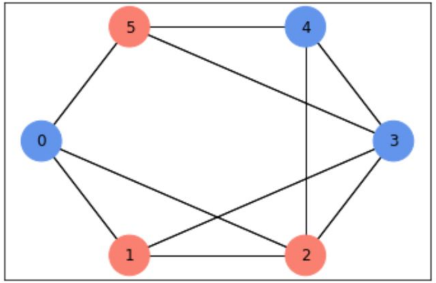
We represent a solution to the n-node Maxcut using binary sequence,

$$\boldsymbol{x} = x_0 x_1 x_2 \cdots x_{n-1} = |x_0\rangle |x_1\rangle |x_2\rangle \cdots |x_{n-1}\rangle$$

where $x_i = 0$ indicates node-i in Partition 0 (red), and $x_i = 1$ indicates node-i in Partition 1 (blue).



$$\boldsymbol{x} = 1010 = |1\rangle_0 |0\rangle_1 |1\rangle_2 |0\rangle_3$$



$$\boldsymbol{x} = 100110 = |100110\rangle$$

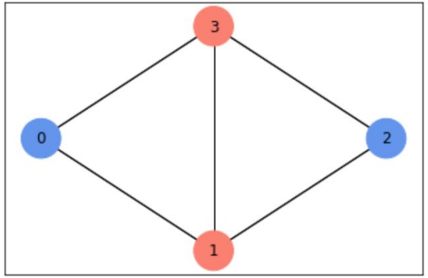
Maxcut problem, as optimization problem

With these binary variables, we can build a problem Hamiltonian for any Maxcut problem

$$H(\boldsymbol{x}) = \sum_{i,j} w_{ij} x_i (1 - x_j) \qquad w_{ij} = \begin{cases} 1, & \text{if edge}(i,j) \in G \\ 0, & \text{if edge}(i,j) \notin G \end{cases}$$

where w_{ij} is connectivity (adjacency matrix).

We claim a solution to the maxcut problem must maximize this Hamiltonian!



$$\boldsymbol{x} = 1010 = |1\rangle_0 |0\rangle_1 |1\rangle_2 |0\rangle_3$$

$$w = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$$w_{01} x_0 (1 - x_1) = w_{01} \cdot 1 \cdot (1 - 0) = 1$$
$$w_{03} x_0 (1 - x_3) = w_{03} \cdot 1 \cdot (1 - 0) = 1$$
$$w_{10} x_1 (1 - x_0) = w_{10} \cdot 0 \cdot (1 - 1) = 0$$

$$w_{12} x_1 (1 - x_2) = 0 \qquad w_{30} x_3 (1 - x_0) = 0$$
$$w_{13} x_1 (1 - x_3) = 0 \qquad w_{31} x_3 (1 - x_1) = 0$$
$$w_{21} x_2 (1 - x_1) = 1 \qquad w_{32} x_3 (1 - x_2) = 0$$
$$w_{23} x_2 (1 - x_3) = 1$$

$$\boxed{H(\boldsymbol{x}) = 4}$$

Maxcut problem, as optimization problem

To map the Hamiltonian to quantum circuit, we use

$$x_i \to \frac{I - Z}{2} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

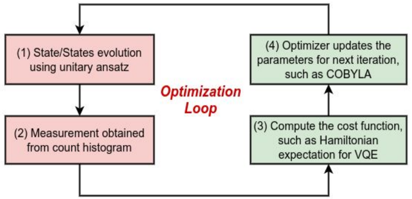
Why?
Check its eigenvalue and eigenvector

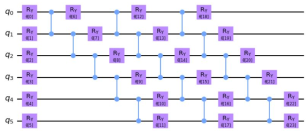
Then, the Maxcut Hamiltonian becomes a sum of Pauli-Z and Pauli-ZZ operators (i.e., Ising Hamiltonian)

$$H \to \sum_{ij} w_{ij} \frac{I_i - Z_i}{2}\left(I - \frac{I_j - Z_j}{2}\right) = \sum_i c_i Z_i + \sum_{ij} d_{ij} Z_i Z_j$$

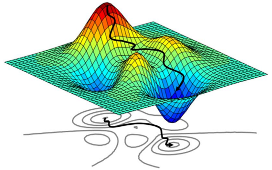
Our task today is to find optimal solution to this Maxcut Hamiltonian using VQE.

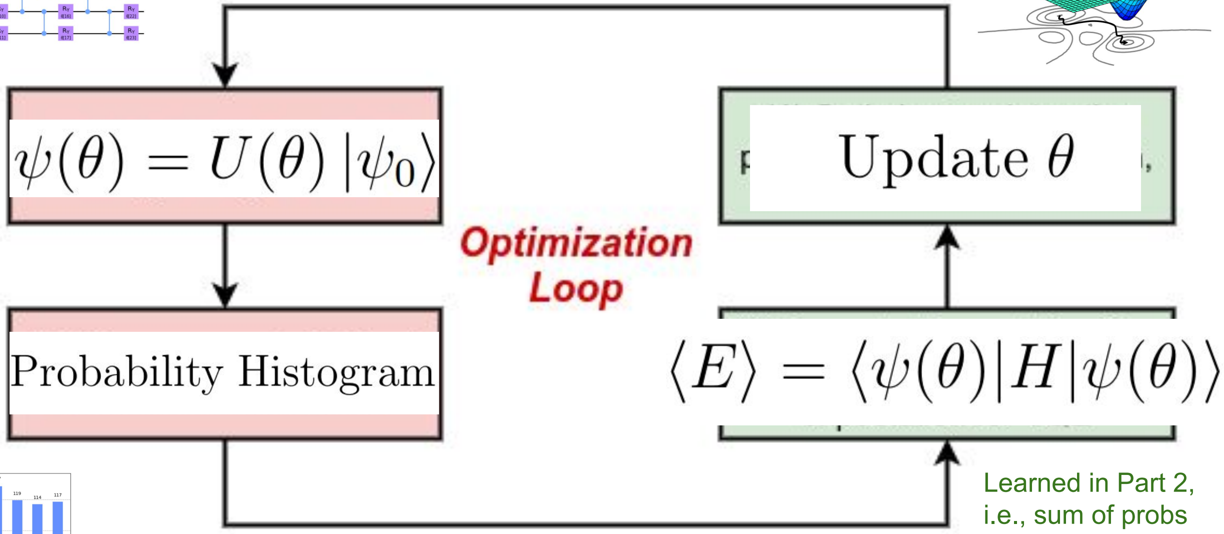
VQE to Maxcut, overview



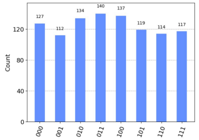


Learned in Part 1,
will use a simpler ansatz





Learned in Part 2,
i.e., sum of probs



Goal: $\text{maxcut} \rightarrow \langle E\rangle_{\text{max}}$ $\quad$ $\text{how} \rightarrow \psi(\theta_{\text{best}})$

Qiskit Lab

Today, you will build a VQE algorithm in Qiskit from scratch, and then we will test our algorithm on the Maxcut problem. I hope you have a taste of quantum computing techniques to real life problems.

The game does not simply stop here…

1. We can use problem-inspired ansatz, such as QAOA (quantum approximate optimization algorithm), where quantum speedup over classical algorithm is proven in some cases.

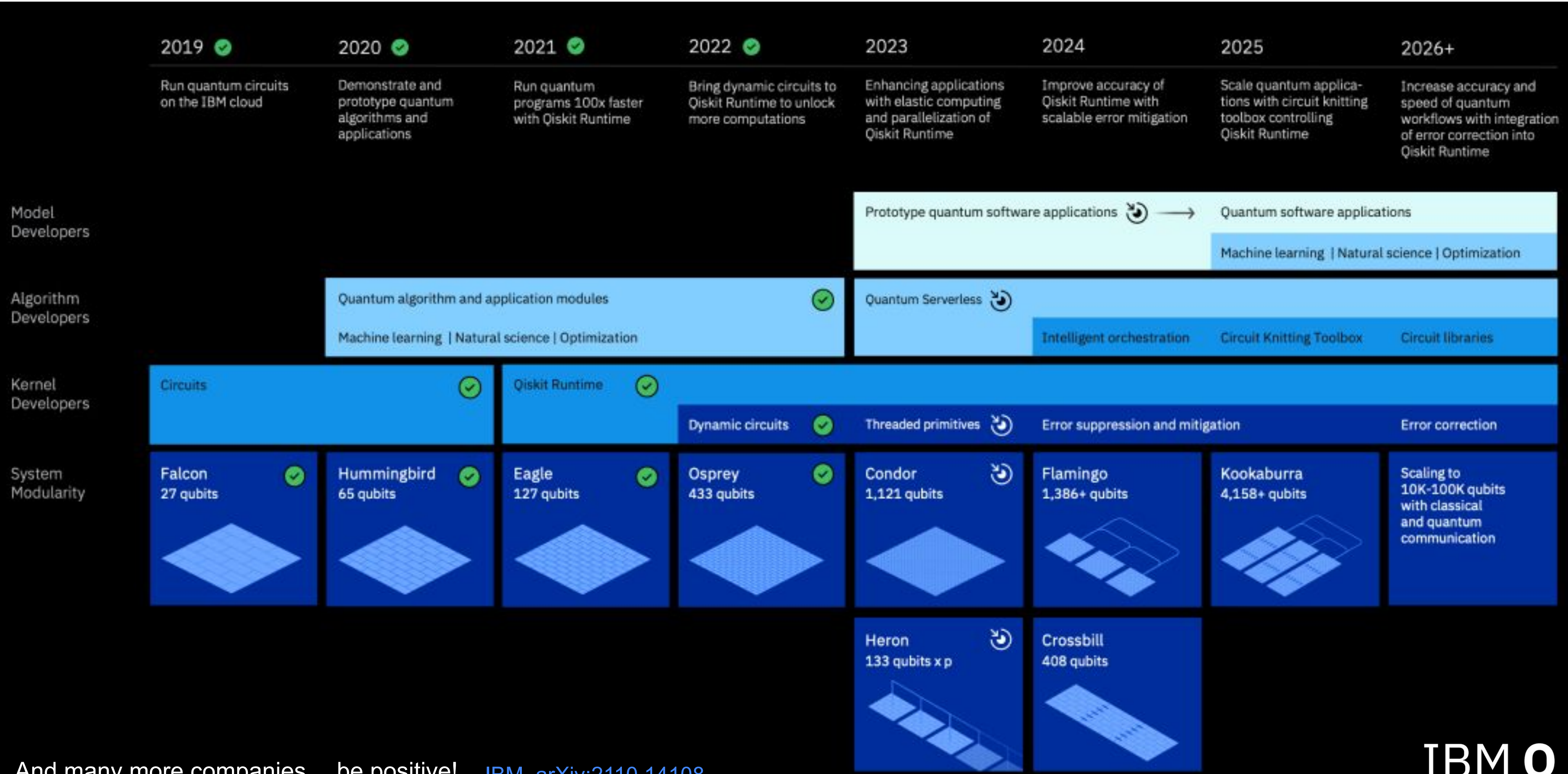
2. We can add weights on edges, constraints on the graph, and essentially describe many problems using this Ising model, such as Traveling Salesman, Knapsack, Satisfiability problem, Social networks…

…

Why quantum computing?

"Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical"

Richard Feynman



And many more companies… be positive! IBM, arXiv:2110.14108