

The background of the slide is dark blue and features abstract circuit-like patterns. These include white, yellow, and blue lines that form various geometric shapes, some resembling circuit traces and others resembling stylized trees or branching structures. Some lines end in small circles, while others are open. The patterns are distributed across the slide, with a higher density in the top and bottom areas.

# Quantum Approximate Optimization Algorithm for the Shortest Vector Problem

Miriam Kim





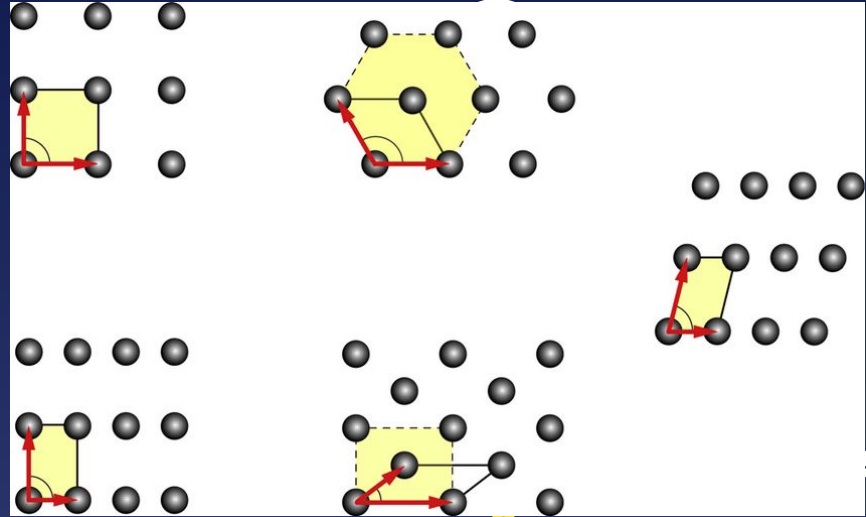
01

# Intro and Background

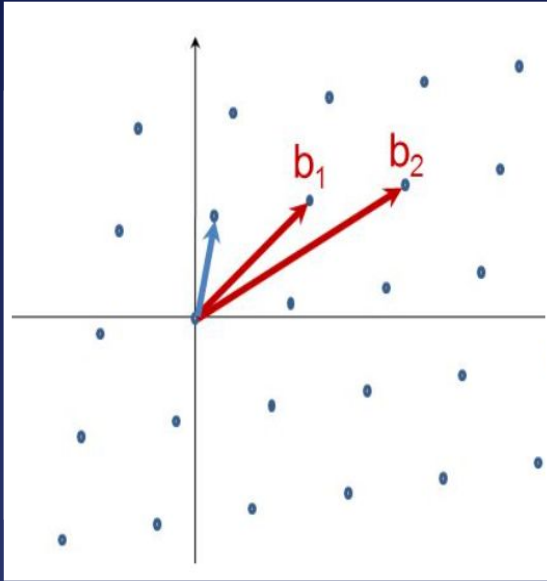


# Important Concepts

- Lattices
  - Lattice based cryptography
- Optimization Problems
  - How are they solved?
  - Applications
- Hamiltonian

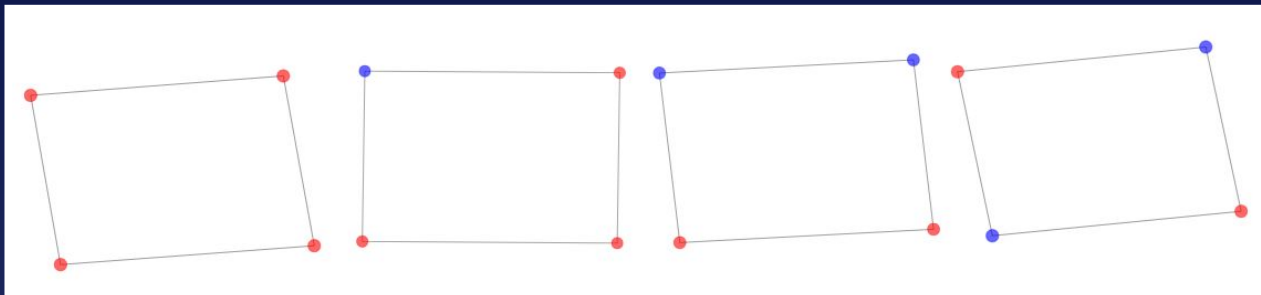


# Shortest Vector Problem



- A **Lattice Problem**
- Given a **basis** of vector space  $V$ , a norm  $N$ , for lattice  $L$ , **find the shortest non-zero vector in  $L$**
- Relevance

# Max-Cut Problem



- Given a graph, partition nodes into two sets such that the edges between the sets is **maximum**
- Problem Hamiltonian up to a constant:

$$H_P = \frac{1}{2} (Z_0 \otimes Z_1 \otimes I_2 \otimes I_3) + \frac{1}{2} (I_0 \otimes Z_1 \otimes Z_2 \otimes I_3) + \frac{1}{2} (Z_0 \otimes I_1 \otimes I_2 \otimes Z_3) + \frac{1}{2} (I_0 \otimes I_1 \otimes Z_2 \otimes Z_3)$$

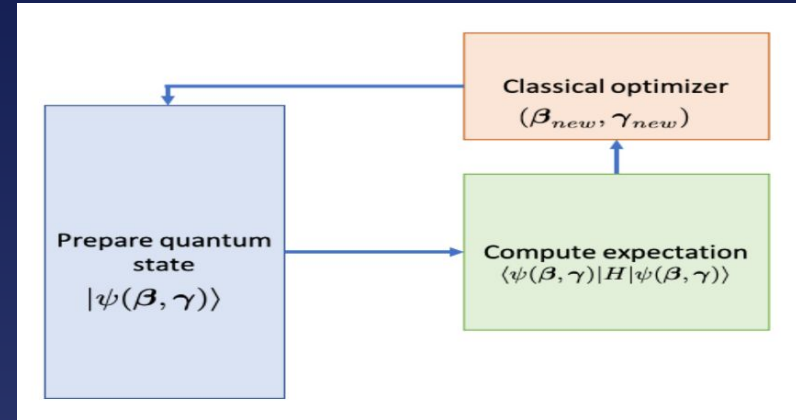
# About QAOA

Farhi et al.

## Quantum Approximate Optimization Algorithm

A quantum algorithm that attempts to solve combinatorial optimization problems

- Applies Hamiltonians to find optimal parameters  $(\beta, \gamma)$  to find the minimum eigenvalue of  $H$
- $U(\beta, \gamma) \rightarrow |\psi(\beta, \gamma)\rangle$
- Classical bit string expected to have good approximation ratio



# QAOA

01

## Initial State

- Apply Hadamard gate to each qubit

03

Pick a  $p$  and initialize  $\beta, \gamma$

05

Measure  $|\beta, \gamma\rangle$

02

## Construct Unitaries from Hamiltonians

- $U(H_B) = e^{-i\beta H_B}$
- $U(H_P) = e^{-i\gamma H_P}$

04

Apply Unitaries for  $p$  times to form state

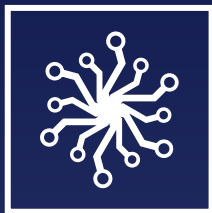
$$|\psi(\beta, \gamma)\rangle = \underbrace{U(\beta)U(\gamma) \cdots U(\beta)U(\gamma)}_{p \text{ times}} |\psi_0\rangle$$

06

Evaluate expectation value classically



# Classical and Quantum



## Classical Solution

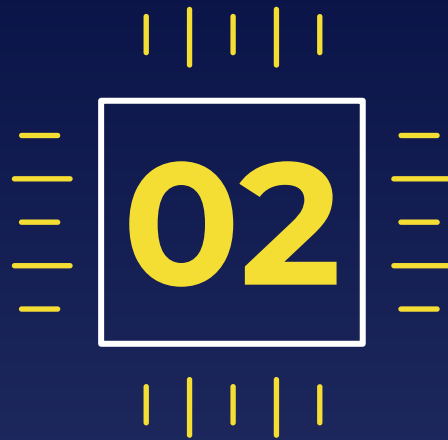
- Gram-Smith Orthogonalization
- LLL Algorithm



## Quantum Solution



- Superposition states → function can be applied to many possible inputs simultaneously





# Implementation

# Construction


$$H_P = \sum_{i,j}^N \hat{Q}^{(i)} \hat{Q}^{(j)} \mathbf{G}_{ij},$$

**Problem  
Hamiltonian**

$$H_M = \sum_{j=1}^n X_j,$$

**Mixing  
Hamiltonian**

$$|\psi_0\rangle = \left( \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \right)^{\otimes n}$$

**Initial State**

# Problem Hamiltonian

$$H_P = \sum_{i,j}^N \hat{Q}^{(i)} \hat{Q}^{(j)} G_{ij},$$

(Joseph et al.)

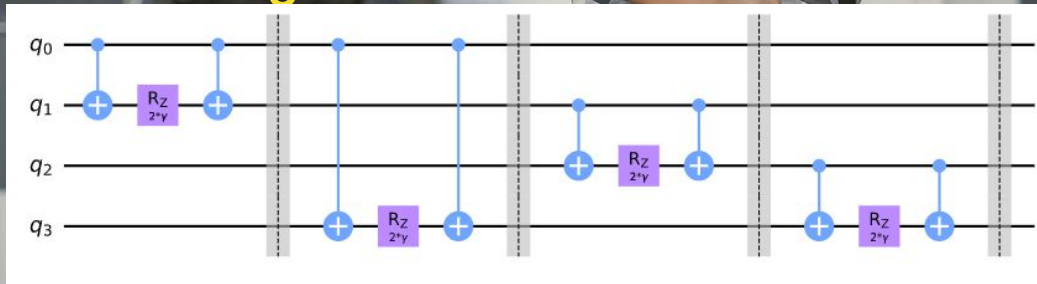
$$\begin{aligned} H_P &= \sum_{i,j}^{N=2} \hat{Q}^i \hat{Q}^j G_{ij} \\ &= \hat{Q}^1 \hat{Q}^1 G_{11} \\ &\quad + \hat{Q}^1 \hat{Q}^2 G_{12} \\ &\quad + \hat{Q}^2 \hat{Q}^2 G_{22} \\ &\quad + \hat{Q}^2 \hat{Q}^1 G_{21} \\ G_{11} &= \vec{b}_1 \cdot \vec{b}_1 = a^2 + b^2 \\ G_{12} &= \vec{b}_1 \cdot \vec{b}_2 = ac + bd \\ G_{22} &= \vec{b}_2 \cdot \vec{b}_2 = c^2 + d^2 \\ G_{21} &= \vec{b}_2 \cdot \vec{b}_1 = ca + db \\ Q_1 &= (z_i^0 + z_i^1 + z_i^2 \dots z_i^7) \\ Q_2 &= (z_j^0 + z_j^1 + z_j^2 \dots z_j^7) \\ U(\gamma) &= e^{-i\gamma H_P} \end{aligned}$$

$$U(\gamma) = e^{-i\gamma H_P}$$

$$U(H_B) = e^{-i\beta H_B}$$

```
gamma = Parameter("$\\gamma$")
qc_p = QuantumCircuit(nqubits)
for pair in list(G.edges()): # pairs of nodes
    qc_p.rzz(2 * gamma, pair[0], pair[1])
    qc_p.barrier()

qc_p.decompose().draw()
```



## Problem Unitary

```
nqubits = 4
```

```
beta = Parameter("$\\beta$")  
qc_mix = QuantumCircuit(nqubits)  
for i in range(0, nqubits):  
    qc_mix.rx(2 * beta, i)
```

```
qc_mix.draw()
```

$q_0$  —  $R_X$   
 $2*\beta$  —

$q_1$  —  $R_X$   
 $2*\beta$  —

$q_2$  —  $R_X$   
 $2*\beta$  —

$q_3$  —  $R_X$   
 $2*\beta$  —

## Mixing Unitary



# Classical Optimization

```
from scipy.optimize import minimize

expectation = get_expectation(G, p=1)

res = minimize(expectation,
               [1.0, 1.0],
               method='COBYLA')

res
```

After the initial state is prepared apply  $U(\beta, \gamma)$  and use classical techniques to optimize the parameters

- Prepare  $|\psi(\beta, \gamma)\rangle$
- Measure the state
- Compute  $\langle \psi(\beta, \gamma) | H_p | \psi(\beta, \gamma) \rangle$
- Find the new set of parameters
- Set the current parameters equal to the new and repeat

# Analyzing Results

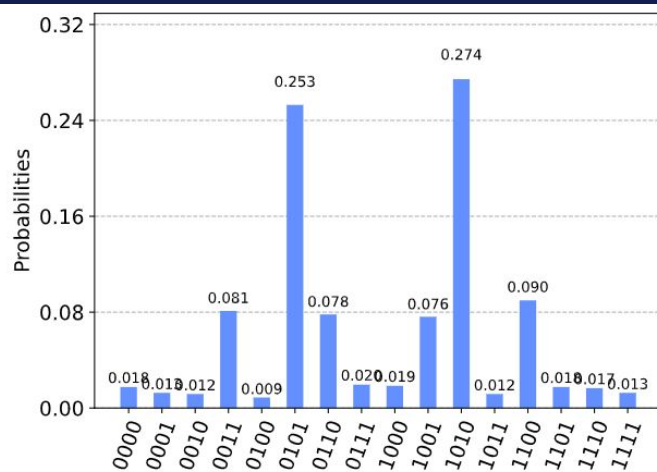
```
from qiskit.visualization import plot_histogram

backend = Aer.get_backend('aer_simulator')
backend.shots = 512

qc_res = create_qaoa_circ(G, res.x)

counts = backend.run(qc_res, seed_simulator=10).result().get_counts()

plot_histogram(counts)
```



# THANKS!

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik

