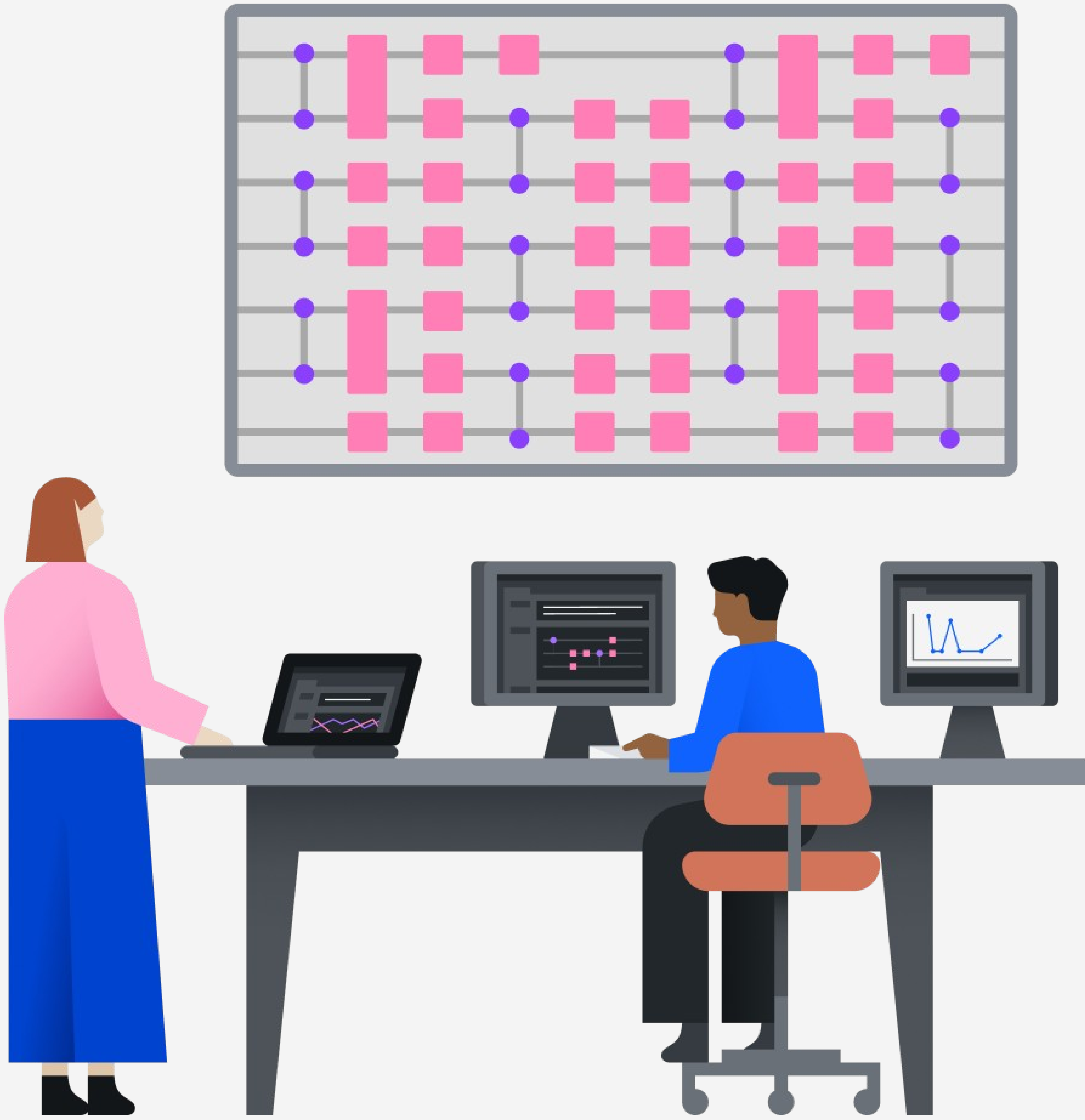
Qiskit Global Summer School 2024

# Mapping problems to qubits

**Kevin J. Sung**
Software developer
IBM



## Qiskit pattern workflow

| | | | |
|---|---|---|---|
| 1. Map problem to quantum circuits and operators | 2. Optimize circuits for target hardware | 3. Execute on target hardware | 4. Post-process results |

## Qiskit pattern workflow

| | | | |
|---|---|---|---|
| **1. Map problem to quantum circuits and operators** | 2. Optimize circuits for target hardware | 3. Execute on target hardware | 4. Post-process results |

## This lecture will cover two applications:

- Combinatorial optimization
- Quantum chemistry

# Combinatorial optimization

## Combinatorial optimization

In combinatorial optimization, the goal is to find an input that maximizes (or minimizes) the value of a cost function.

$$\max_x C(x)$$

Examples:

Knapsack problem
- *x*: Selection of items to include in the knapsack
- *C(x)*: Total value of items

Vehicle routing problem
- *x*: Routes for a fleet of vehicles
- *C(x)*: Total distance traveled by all vehicles

## Approximate solutions

Many combinatorial optimizations are too difficult to solve exactly.

Often, we settle for approximate solutions.

Approximation ratio achieved by solution *x*:

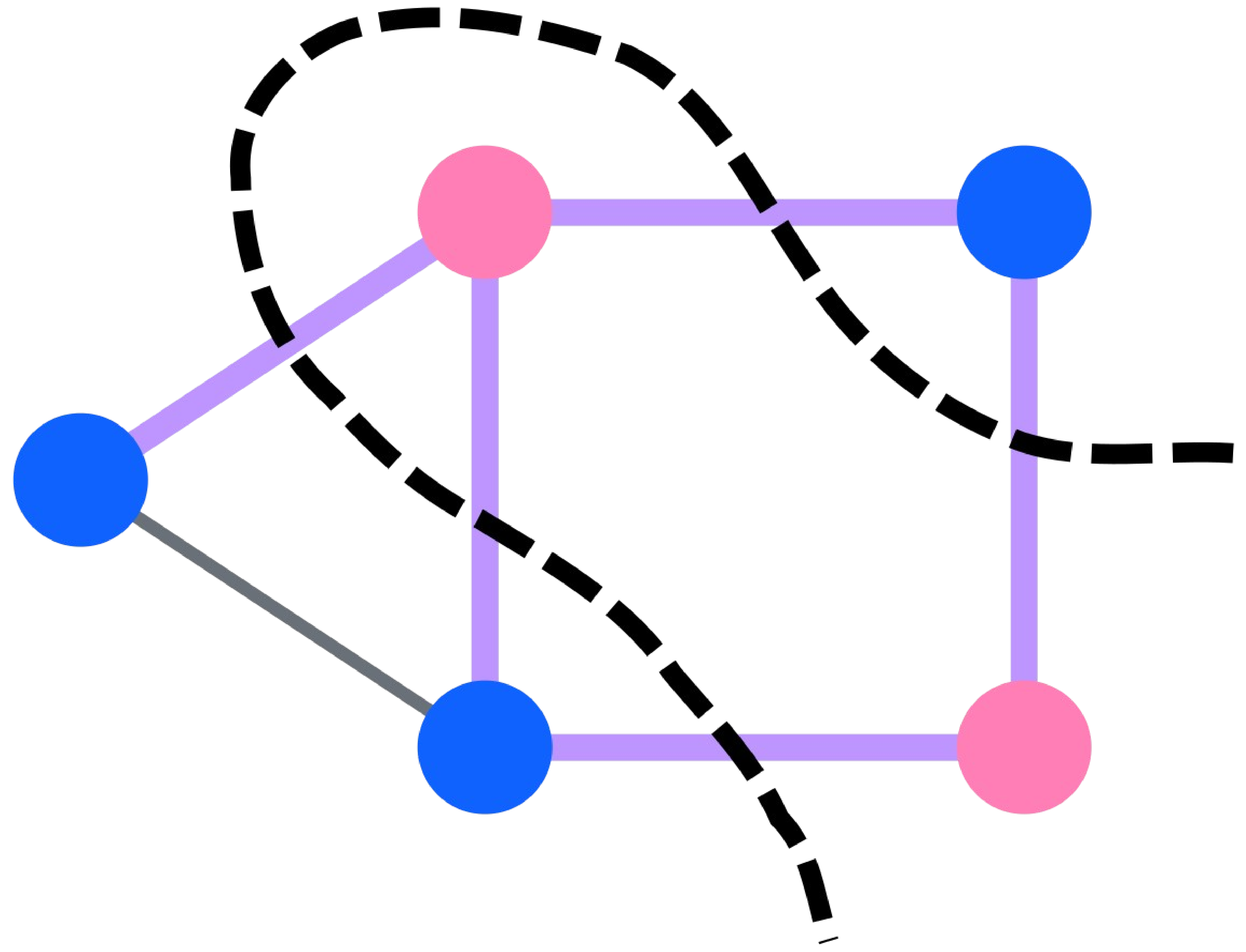
$$\frac{C(x)}{\mathrm{OPT}}$$

$$\mathrm{OPT} = \max_x C(x)$$

## The max-cut problem

**Given a graph, partition its vertices into two sets such that the number of edges between the two sets is maximized.**

The (decision version of the) max-cut problem is NP-complete.



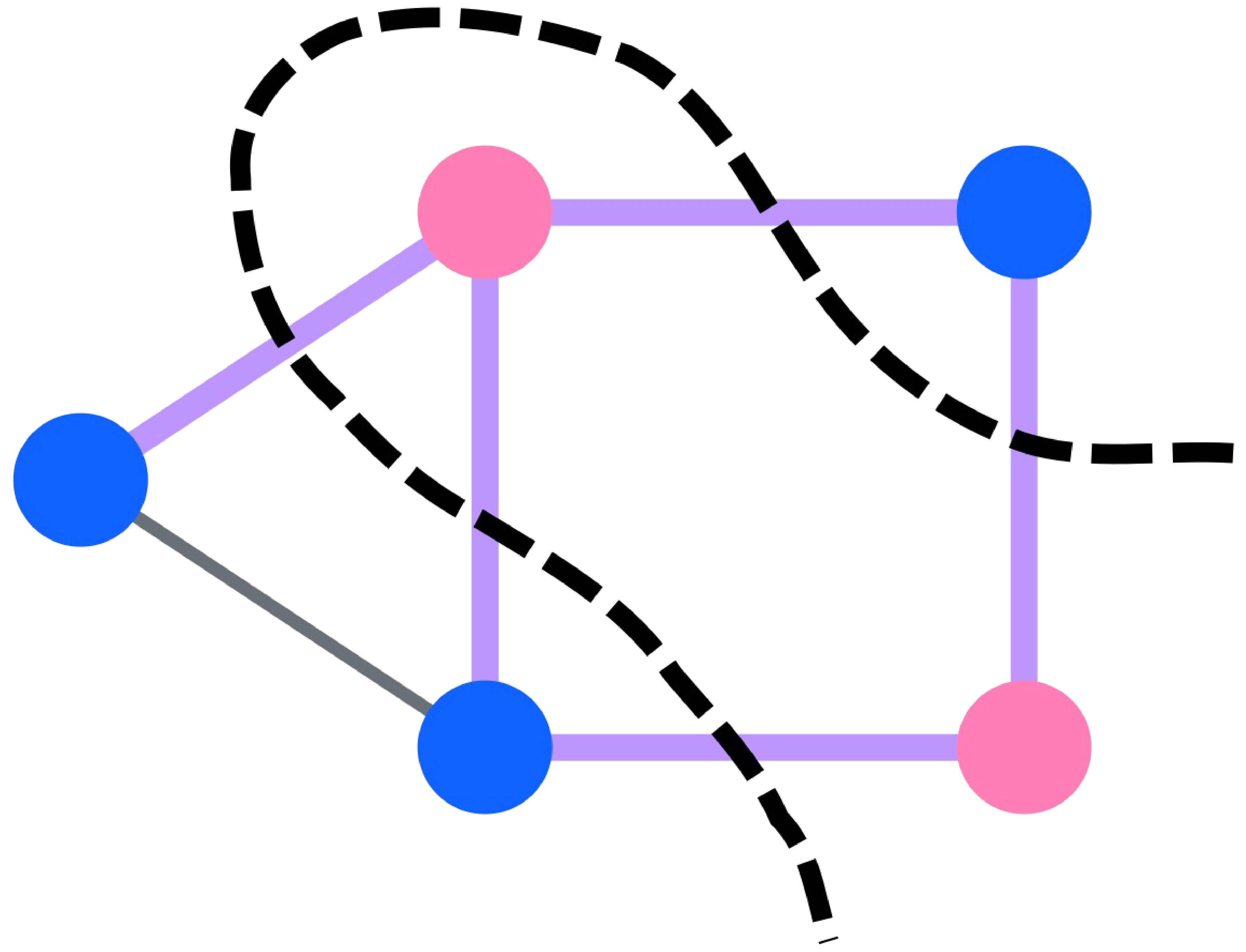
By Miym - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=6002348
Colors modified

## The max-cut problem

A partition of vertices into two sets is called a **cut**.

The **size** of a cut is the number of edges between the two sets of the partition.

Goal: Find a cut with the largest possible size.



By Miym - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=6002348
Colors modified

## The max-cut problem

A cut can be represented as a bitstring

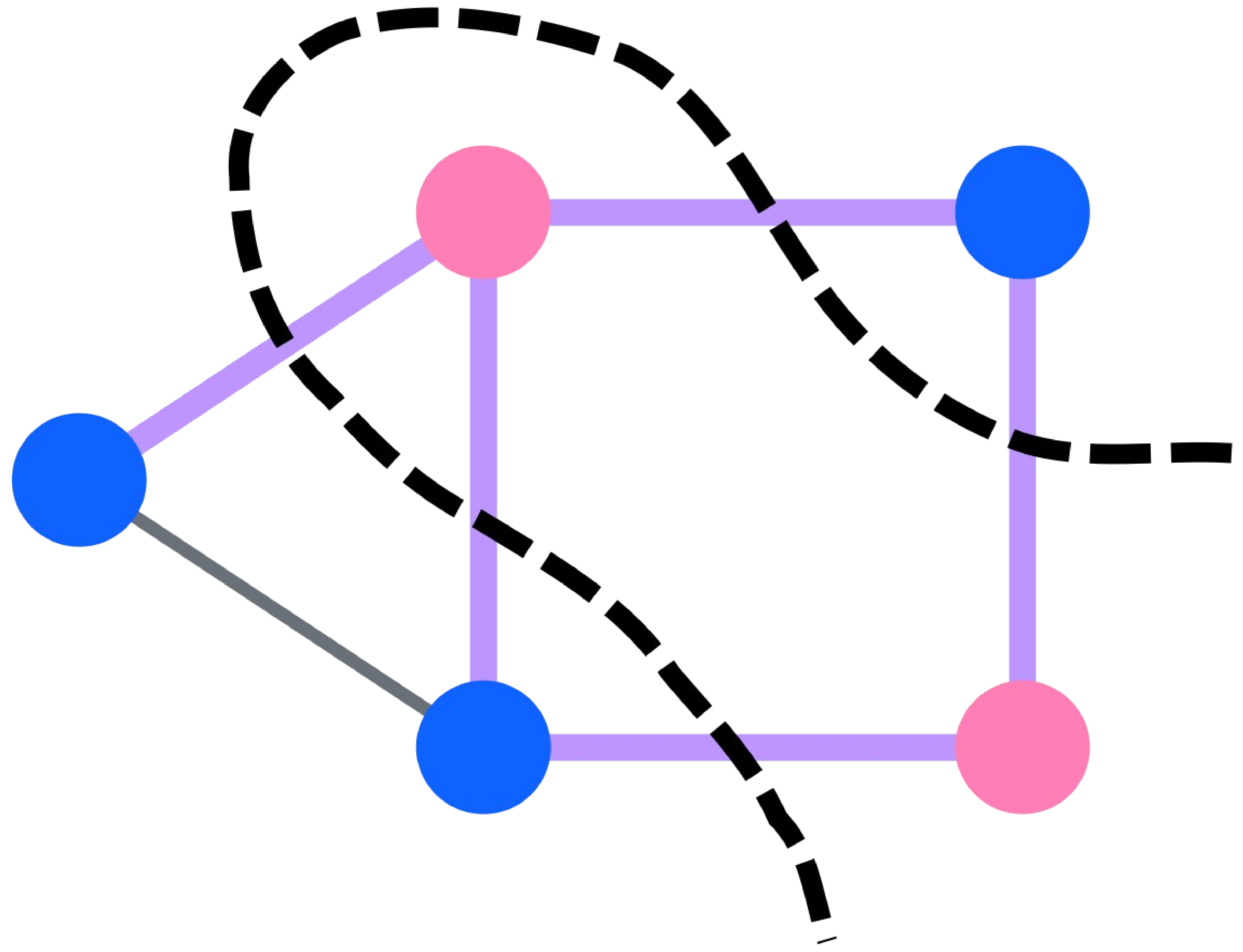
$$b = b_1 b_2 \cdots b_n$$

There's one bit for each vertex, and the value of the bit indicates which set the vertex belongs to.

Goal: Find a bitstring that maximizes the function

$$C(b) = \sum_{\langle jk \rangle} C_{\langle jk \rangle}(b)$$

$$C_{\langle jk \rangle}(b) = \begin{cases} 1 & \text{if } b_j \neq b_k \\ 0 & \text{otherwise} \end{cases}$$



## The max-cut problem

Goal: Find a bitstring that maximizes the function

$$C(b) = \sum_{\langle jk \rangle} C_{\langle jk \rangle}(b)$$

$$C_{\langle jk \rangle}(b) = \begin{cases} 1 & \text{if } b_j \neq b_k \\ 0 & \text{otherwise} \end{cases}$$

Change variables:

$$z_j = (-1)^{b_j}$$

Originally, bits were represented using 0 and 1.
Now, bits are represented using +1 and –1.

Using the new variables,

$$C_{\langle jk \rangle}(z) = \frac{1}{2}(1 - z_j z_k)$$

This is an example of the *Fourier expansion of a Boolean function*.

## The max-cut problem

Goal: Find a bitstring that maximizes the function

$$C(z) = \frac{1}{2}\sum_{\langle jk\rangle}(1 - z_j z_k)$$

Change variables:

$$z_j = (-1)^{b_j}$$

Originally, bits were represented using 0 and 1.
Now, bits are represented using +1 and –1.

Using the new variables,

$$C_{\langle jk\rangle}(z) = \frac{1}{2}(1 - z_j z_k)$$

This is an example of the *Fourier expansion of a Boolean function*.

## The max-cut problem

Goal: Find a bitstring that maximizes the function

$$C(z) = \frac{1}{2} \sum_{\langle jk \rangle} (1 - z_j z_k)$$

Define the quantum operator

$$C = \frac{1}{2} \sum_{\langle jk \rangle} (I - Z_j Z_k)$$

Here we abuse notation and write *C* for both the objective function and the quantum operator.

Remember, the bits $z_j$ take values in {-1, 1}.

Now, the final trick is to promote the $z_j$ to quantum variables.

Then,

$$C|z\rangle = C(z)|z\rangle$$

$$\langle z|C|z\rangle = C(z)$$

for a computational basis state. *C* is called the **max-cut Hamiltonian**.

## The max-cut problem

Max-cut Hamiltonian:

$$C = \frac{1}{2} \sum_{\langle jk \rangle} (I - Z_j Z_k)$$

Goal: Find a computational basis vector with the highest eigenvalue.

$$\max_z \langle z|C|z\rangle$$

$$\langle z|C|z\rangle = C(z)$$

$C$ is a $2^n \times 2^n$ diagonal matrix. Its diagonal entries are the values of the cost function.

$$C = \begin{pmatrix} C(00\cdots0) & & & \\ & C(00\cdots1) & & \\ & & \ddots & \\ & & & C(11\cdots1) \end{pmatrix}$$

Here we reverted to using 0 and 1 for bits to match quantum computing convention.

## The max-cut problem

Max-cut Hamiltonian:

$$C = \frac{1}{2} \sum_{\langle jk \rangle} (I - Z_j Z_k)$$

Goal: Find a computational basis vector with the highest eigenvalue.

$$\max_z \langle z | C | z \rangle$$

$$\langle z | C | z \rangle = C(z)$$

**Classical strategy:**
Sample *z* from a probability distribution generated by an efficient classical algorithm. Best known algorithm (Goemans-Williamson) achieves an approximation ratio of about 0.878.

**Quantum strategy:**
Prepare a quantum state on a quantum computer using an efficient quantum algorithm, then measure it in the computational basis.

**Conjecture (quantum advantage):**
The quantum strategy can sample from more (and better) probability distributions.

## QAOA for the max-cut problem

Max-cut Hamiltonian:

$$C = \frac{1}{2} \sum_{\langle jk \rangle} (I - Z_j Z_k)$$

Initial state:

$$|s\rangle = \frac{1}{\sqrt{2^n}} \sum_z |z\rangle$$

"Driver" Hamiltonian:

$$B = \sum_j X_j$$

QAOA ansatz:

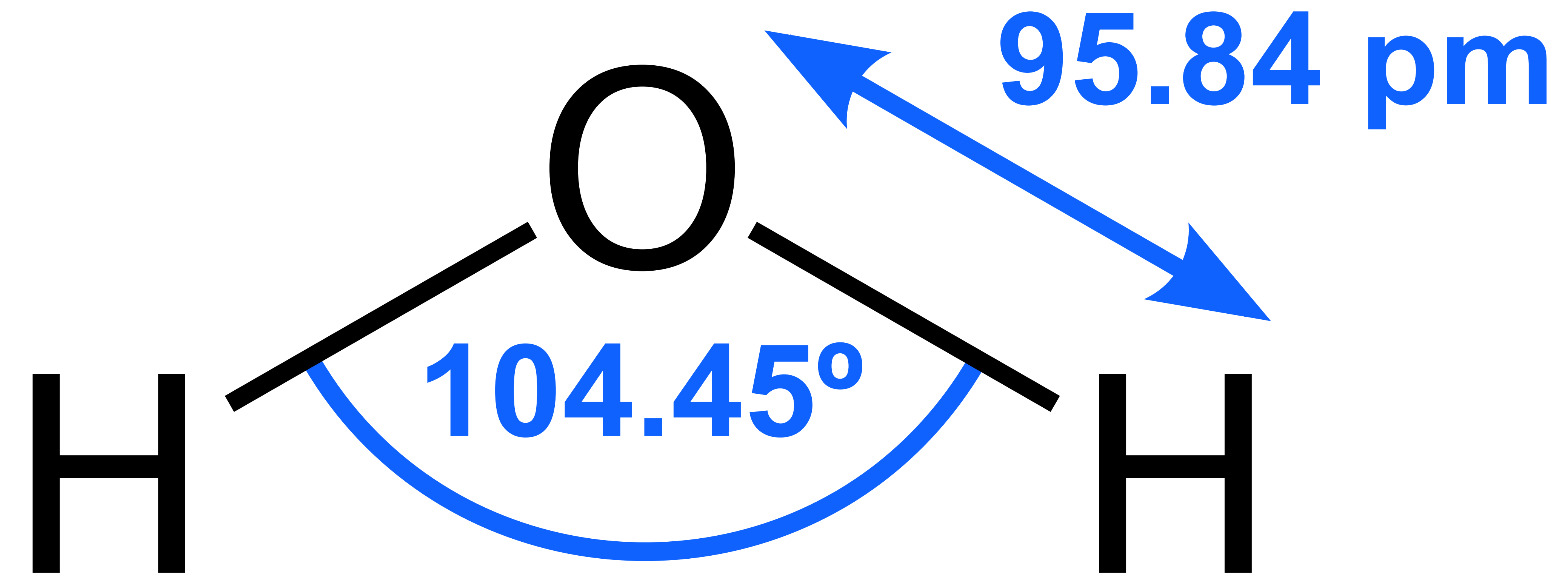
$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = e^{-i\beta_p B} e^{-i\gamma_p C} \cdots e^{-i\beta_1 B} e^{-i\gamma_1 C} |s\rangle$$

# Quantum chemistry

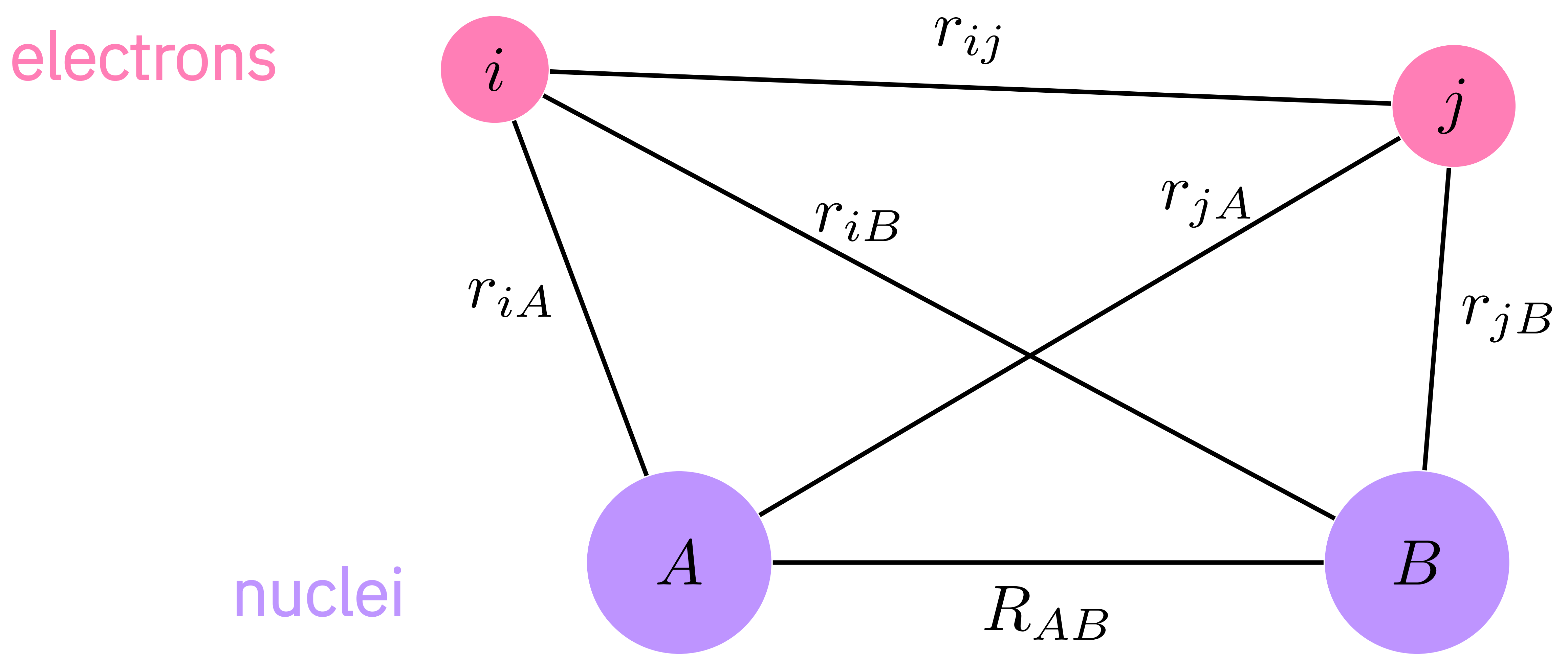
## Quantum chemistry and electronic structure

**Given a configuration of atoms, what are its properties?**

Predicting the properties of molecules and materials is one of the most anticipated applications of quantum computers.



## Quantum chemistry and electronic structure



Molecular Hamiltonian:

$$\begin{aligned} H &= T_\mathrm{n} + T_\mathrm{e} + V_\mathrm{ne} + V_\mathrm{ee} + V_\mathrm{nn} \\ &= -\sum_A \frac{\nabla_A^2}{2M_A} - \sum_i \frac{\nabla_i^2}{2} - \sum_{i,A} \frac{Z_A}{r_{iA}} + \sum_{i<j} \frac{1}{r_{ij}} + \sum_{A<B} \frac{Z_A Z_B}{r_{AB}} \end{aligned}$$

## The Born-Oppenheimer approximation

Molecular Hamiltonian:

$$H = \underbrace{\cancel{T_{\mathrm{n}}}}_{\text{zero}} + T_{\mathrm{e}} + V_{\mathrm{ne}} + V_{\mathrm{ee}} + \underbrace{\cancel{V_{\mathrm{nn}}}}_{\text{constant}}$$

Nuclei move much slower than electrons, let's treat them as fixed points.

Electronic Hamiltonian:

$$H(\mathbf{R}) = -\sum_i \frac{\nabla_i^2}{2} - \sum_{i,A} \frac{Z_A}{r_{iA}} + \sum_{i<j} \frac{1}{r_{ij}}$$

Goal: Solve the Schrödinger equation for this Hamiltonian, which means finding its (low-lying) eigenvalues and eigenvectors.

## Second quantization

Electronic Hamiltonian:

$$H(\mathbf{R}) = -\sum_i \frac{\nabla_i^2}{2} - \sum_{i,A} \frac{Z_A}{r_{iA}} + \sum_{i<j} \frac{1}{r_{ij}}$$

Electronic Hamiltonian in discretized form:

$$H = \sum_{pq} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{pqrs} h_{pqrs} a_p^\dagger a_r^\dagger a_s a_q$$

To represent this Hamiltonian on a computer, we need to discretize it.

The electronic Hamiltonian can be discretized using a formalism called second quantization.

## Second quantization

Electronic Hamiltonian:

$$H(\mathbf{R}) = -\sum_i \frac{\nabla_i^2}{2} - \sum_{i,A} \frac{Z_A}{r_{iA}} + \sum_{i<j} \frac{1}{r_{ij}}$$

Electronic Hamiltonian in discretized form:

$$H = \sum_{pq} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{pqrs} h_{pqrs} a_p^\dagger a_r^\dagger a_s a_q$$

A system of *N* **fermionic modes (orbitals)** is described using a set of *N* fermionic **annihilation operators** $\{a_1, \ldots, a_N\}$ that satisfy the **fermionic anticommutation relations**:

$$a_p a_q + a_q a_p = 0$$
$$a_p a_q^\dagger + a_q^\dagger a_p = \delta_{pq}$$

The adjoint $a_p^\dagger$ of an annihilation operator is called a **creation operator**.

Consequences of the fermionic anticommutation relations

$$a_p a_q + a_q a_p = 0$$
$$a_p a_q^\dagger + a_q^\dagger a_p = \delta_{pq}$$

The operators $\{a_p^\dagger a_p\}$ commute and have eigenvalues 0 and 1.

These are the **occupation number operators**.

There is a normalized vector $|\text{vac}\rangle$ which is a mutual 0-eigenvector of all the occupation number operators.

This is the **vacuum state**.

If $|\psi\rangle$ is a 0-eigenvector of $a_p^\dagger a_p$ , then $a_p^\dagger|\psi\rangle$ is a 1-eigenvector.

( $a_p^\dagger$ creates a fermion in mode *p*.)

If $|\psi\rangle$ is a 1-eigenvector of $a_p^\dagger a_p$ , then $a_p|\psi\rangle$ is a 0-eigenvector.

( $a_p$ destroys a fermion in mode *p*.)

The creation and annihilation operators square to zero:

$$a_p^2 = 0$$
$$\left(a_p^\dagger\right)^2 = 0$$

This is the **Pauli exclusion principle**.

We can construct an orthonormal basis of *2^N* vectors labeled by bitstrings *z*:

$$\begin{aligned}|z\rangle &= |z_N \cdots z_1\rangle \\ &\equiv \left(a_1^\dagger\right)^{z_1} \cdots \left(a_N^\dagger\right)^{z_N} |\text{vac}\rangle\end{aligned}$$

These basis vectors are called **Slater determinants**.

## Consequences of the fermionic anticommutation relations

$$a_p a_q + a_q a_p = 0$$
$$a_p a_q^\dagger + a_q^\dagger a_p = \delta_{pq}$$

An annihilation operator acts on basis vectors as

$$a_p |z_N \cdots 0_p \cdots z_1\rangle = 0$$

$$a_p |z_N \cdots 1_p \cdots z_1\rangle = (-1)^{\sum_{i=1}^{p-1} z_i} |z_N \cdots 0_p \cdots z_1\rangle$$

This behavior differs from qubits!

Operators applied to a subset of qubits are tensor product with identity on the rest of the qubits.

The notion of **locality** differs between qubits and fermionic modes.

## Mapping fermions to qubits

A system of fermionic modes is represented using fermionic creation and annihilation operators.

$$a_p a_q + a_q a_p = 0$$
$$a_p a_q^\dagger + a_q^\dagger a_p = \delta_{pq}$$

Qubit operators are represented using tensor products of the Pauli operators $I$, $X$, $Y$, and $Z$.

Question: How can we represent the fermionic operators using Pauli operators?

## Mapping fermions to qubits

We know:

$$a_p a_q + a_q a_p = 0$$
$$a_p a_q^\dagger + a_q^\dagger a_p = \delta_{pq}$$

$$a_p|z_N \cdots 0_p \cdots z_1\rangle = 0$$

$$a_p|z_N \cdots 1_p \cdots z_1\rangle = (-1)^{\sum_{i=1}^{p-1} z_i}|z_N \cdots 0_p \cdots z_1\rangle$$

First attempt:

$$a_p \mapsto |0\rangle\langle 1|_p = \frac{1}{2}(X_p + iY_p)$$

$$|0\rangle\langle 1| = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

Result:

$$a_p|z_N \cdots 0_p \cdots z_1\rangle = 0$$

$$a_p|z_N \cdots 1_p \cdots z_1\rangle = |z_N \cdots 0_p \cdots z_1\rangle$$

The phase factor is missing!

## Mapping fermions to qubits

We know:

$$a_p a_q + a_q a_p = 0$$

$$a_p a_q^\dagger + a_q^\dagger a_p = \delta_{pq}$$

$$a_p |z_N \cdots 0_p \cdots z_1\rangle = 0$$

$$a_p |z_N \cdots 1_p \cdots z_1\rangle = (-1)^{\sum_{i=1}^{p-1} z_i} |z_N \cdots 0_p \cdots z_1\rangle$$

Compute the phase factor by adding a "*Z* string":

$$a_p \mapsto \frac{1}{2}(X_p + iY_p) Z_1 \cdots Z_{p-1} = \underbrace{I \otimes \cdots \otimes I}_{N-p} \otimes |0\rangle\langle 1| \otimes \underbrace{Z \otimes \cdots \otimes Z}_{p-1}$$

This is the **Jordan-Wigner transformation**.

## Jordan-Wigner transformation

$$a_p a_q + a_q a_p = 0$$
$$a_p a_q^\dagger + a_q^\dagger a_p = \delta_{pq}$$

Jordan-Wigner transformation:

$$a_p \mapsto \frac{1}{2}(X_p + iY_p) Z_1 \cdots Z_{p-1} = \underbrace{I \otimes \cdots \otimes I}_{N-p} \otimes |0\rangle\langle 1| \otimes \underbrace{Z \otimes \cdots \otimes Z}_{p-1}$$

Operators that are local in terms of fermions are not local in terms of qubits!

The Jordan-Wigner transformation is just one of many possible fermion-to-qubit mappings.

Overhead (weight of *Z* strings): *O(N)*

Using the **Bravyi-Kitaev transformation**, the overhead can be reduced to *O(log N)*!

## Jordan-Wigner transformation

Occupation number operator

$$a_p^\dagger a_p \mapsto \frac{1}{2}(I - Z_p)$$

Tunneling interaction

$$a_p^\dagger a_q + a_q^\dagger a_p \mapsto \frac{1}{2}(X_p X_q + Y_p Y_q) Z_{p+1} \cdots Z_{q-1} \qquad (p < q)$$

Tunneling interaction for adjacent modes

$$a_p^\dagger a_{p+1} + a_{p+1}^\dagger a_p \mapsto \frac{1}{2}(X_p X_{p+1} + Y_p Y_{p+1})$$

## Orbital rotations

The **orbital rotation** is a fundamental operation in fermionic simulations.

$$a_p \mapsto \mathcal{U} a_p \mathcal{U}^\dagger = \sum_q U_{qp}^* a_q \equiv b_p$$

where *U* is a unitary matrix.

The $b_p$ operators also satisfy the fermionic anticommutation relations!

$$b_p b_q + b_q b_p = 0$$
$$b_p b_q^\dagger + b_q^\dagger b_p = \delta_{pq}$$

The $b_p$ are also fermionic annihilation operators. They destroy fermions in a different set of modes related to the original modes by a basis change.

Qubit analogy: Changing from *Z* basis to *X* basis.

## Orbital rotations

Example application: Time evolution by a quadratic Hamiltonian

$$H = \boxed{\sum_{pq} h_{pq} a_p^\dagger a_q} + \frac{1}{2} \sum_{pqrs} h_{pqrs} a_p^\dagger a_r^\dagger a_s a_q$$

The one-body part of the molecular Hamiltonian is a quadratic Hamiltonian.

The two-body part can also be simulated with orbital rotations via the "double-factorized" a.k.a. "low rank" representation.

A quadratic Hamiltonian

$$H = \sum_{pq} M_{pq} a_p^\dagger a_q$$

can always be rewritten as

$$H = \sum_{p} \varepsilon_p b_p^\dagger b_p$$

Where the $\varepsilon_p$ are real numbers and the $b_p$ are fermionic annihilation operators for modes in a rotated basis.

## Implementing orbital rotations

Given an *N* × *N* unitary matrix *U* representing an orbital rotation, how can we apply the orbital rotation on a quantum computer?

$$a_p \mapsto \mathcal{U} a_p \mathcal{U}^\dagger = \sum_q U_{qp}^* a_q \equiv b_p$$

The map

$$\underbrace{U}_{N \times N} \mapsto \underbrace{\mathcal{U}}_{2^N \times 2^N}$$

satisfies

$$UV \mapsto \mathcal{U}\mathcal{V}$$

## Implementing orbital rotations

We can use a Givens rotation decomposition of *U*

$$U = DG_L G_{L-1} \cdots G_1$$

This will yield a corresponding decomposition

$$\mathcal{U} = \mathcal{D}\mathcal{G}_L \mathcal{G}_{L-1} \cdots \mathcal{G}_1$$

The map

$$\underbrace{U}_{N \times N} \mapsto \underbrace{\mathcal{U}}_{2^N \times 2^N}$$

satisfies

$$UV \mapsto \mathcal{U}\mathcal{V}$$

## Implementing orbital rotations

We can use a Givens rotation decomposition of *U*

$$U = DG_L G_{L-1} \cdots G_1$$

This will yield a corresponding decomposition

$$\mathcal{U} = \mathcal{D}\mathcal{G}_L \mathcal{G}_{L-1} \cdots \mathcal{G}_1$$

Givens rotation:

$$G(p,q,\theta,\varphi) = \begin{array}{c} \begin{array}{ccccccc} \phantom{1} & \phantom{\cdots} & \overset{p}{\downarrow} & \phantom{\cdots} & \overset{q}{\downarrow} & \phantom{\cdots} & \phantom{0} \end{array} \\ \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s^* & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \end{array}$$

$$c = \cos(\theta)$$

$$s = e^{i\varphi}\sin(\theta)$$

## Implementing orbital rotations

A Givens rotation maps operators as

$$a_p \mapsto c a_p + s a_q$$
$$a_q \mapsto -s^* a_p + c a_q$$

This can be written in matrix form as

$$\begin{pmatrix} a_p \\ a_q \end{pmatrix} \mapsto \begin{pmatrix} c & s \\ -s^* & c \end{pmatrix} \begin{pmatrix} a_p \\ a_q \end{pmatrix}$$

Givens rotation:

$$G(p,q,\theta,\varphi) = \begin{array}{c} \begin{array}{ccccccc} & & p & & q & & \\ & & \downarrow & & \downarrow & & \end{array} \\ \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s^* & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \end{array}$$

$$c = \cos(\theta)$$
$$s = e^{i\varphi}\sin(\theta)$$

## Implementing orbital rotations

A Givens rotation maps operators as

$$\begin{aligned} a_p &\mapsto c a_p + s a_q \\ a_q &\mapsto -s^* a_p + c a_q \end{aligned}$$

This can be written in matrix form as

$$\begin{pmatrix} a_p \\ a_q \end{pmatrix} \mapsto \begin{pmatrix} c & s \\ -s^* & c \end{pmatrix} \begin{pmatrix} a_p \\ a_q \end{pmatrix}$$

A rotation of two modes is achieved by the unitary

$$\mathcal{G}(p, q, \theta, \varphi) = \exp\bigl(i\varphi a_p^\dagger a_p\bigr) \exp\bigl[\theta(a_p^\dagger a_q - a_q^\dagger a_p)\bigr] \exp\bigl(-i\varphi a_p^\dagger a_p\bigr)$$

## Implementing orbital rotations

Strategy: Decompose U as

$$U = DG_L G_{L-1} \cdots G_1$$

Apply the sequence of Givens rotation gates

$$\mathcal{G}(p, q, \theta, \varphi) = \exp\bigl(i\varphi a_p^\dagger a_p\bigr) \exp\bigl[\theta(a_p^\dagger a_q - a_q^\dagger a_p)\bigr] \exp\bigl(-i\varphi a_p^\dagger a_p\bigr)$$

The diagonal factor *D* can be implemented using the gates $\exp\bigl(i\varphi a_p^\dagger a_p\bigr)$

## Implementing orbital rotations

$$U = D G_L G_{L-1} \cdots G_1$$

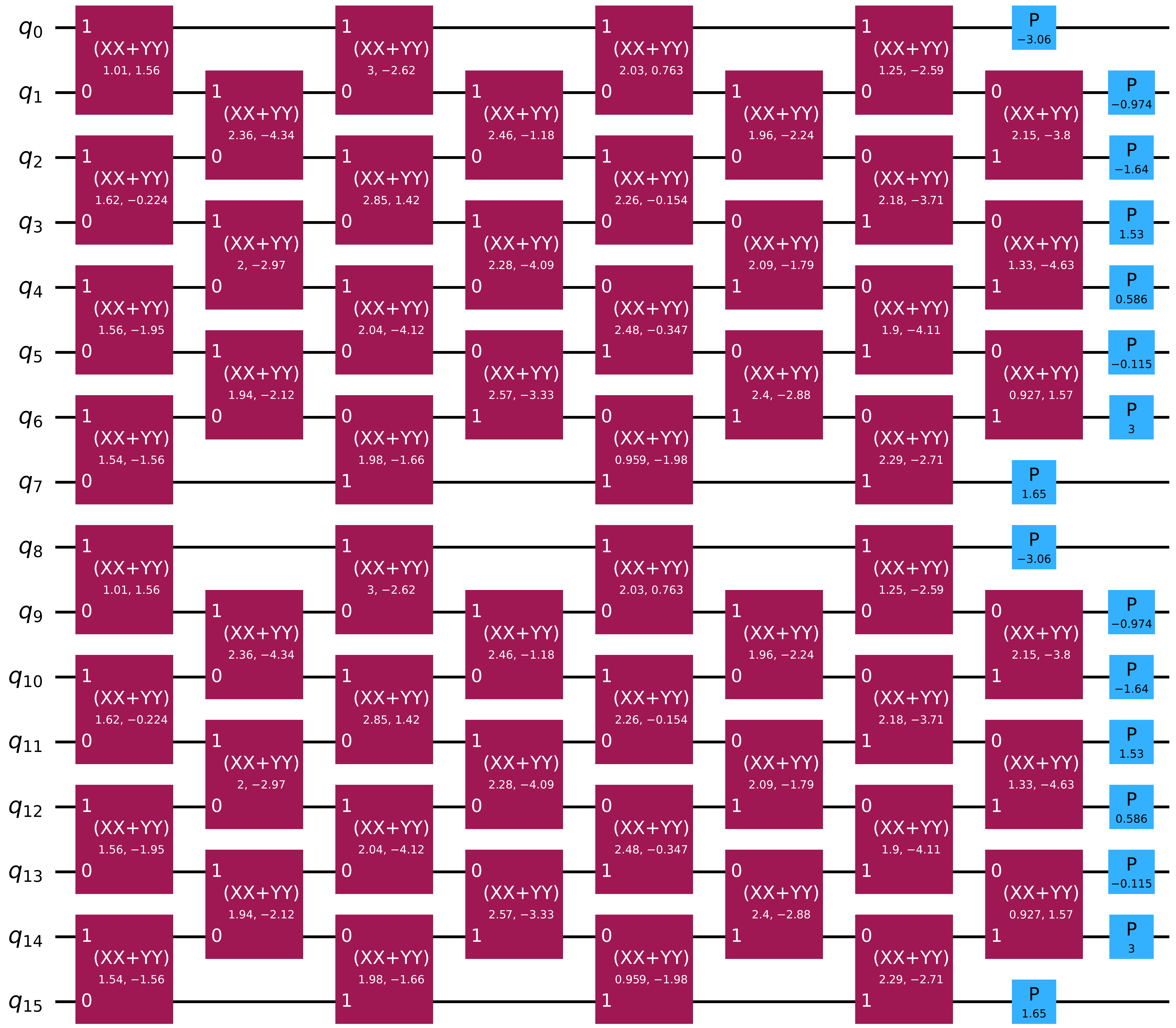
Two-qubit gate, if *p* and *q* are adjacent

$$\mathcal{G}(p, q, \theta, \varphi) = \boxed{\exp\left(i\varphi a_p^\dagger a_p\right)} \boxed{\exp\left[\theta(a_p^\dagger a_q - a_q^\dagger a_p)\right]} \exp\left(-i\varphi a_p^\dagger a_p\right)$$

Single-qubit *Z* rotation

We want to find a sequence of Givens rotations that each act on adjacent indices.

## Orbital rotation circuit



## Deriving the second-quantized Hamiltonian

Electronic Hamiltonian:

$$H(\mathbf{R}) = -\sum_i \frac{\nabla_i^2}{2} - \sum_{i,A} \frac{Z_A}{r_{iA}} + \sum_{i<j} \frac{1}{r_{ij}}$$

Electronic Hamiltonian in discretized form:

$$H = \sum_{pq} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{pqrs} h_{pqrs} a_p^\dagger a_r^\dagger a_s a_q$$

How does the discretization work?

Idea: Choose a finite set of functions to represent the vector space. Project the Hamiltonian into the space spanned by these functions.

## Single-electron wave functions

Electronic Hamiltonian:

$$H(\mathbf{R}) = -\sum_i \frac{\nabla_i^2}{2} - \sum_{i,A} \frac{Z_A}{r_{iA}} + \sum_{i<j} \frac{1}{r_{ij}}$$

Start by choosing a finite set of **spatial orbitals** to represent a function of a single electron's position

$$f(\mathbf{r}) = \sum_{i=1}^{N} c_k \phi_k(\mathbf{r})$$

Describing an electron requires specifying its **spin** in addition to position.

Spin is described by two orthonormal functions $\alpha(\omega)$ (spin up) and $\beta(\omega)$ (spin down). A **spin orbital** $\chi$ describes both position and spin.

$$\chi(\mathbf{x}) = \begin{cases} \phi(\mathbf{r})\alpha(\omega) \\ \quad \text{or} \\ \phi(\mathbf{r})\beta(\omega) \end{cases}$$

$$\mathbf{x} = (\mathbf{r}, \omega)$$

## Many-electron wave functions

How can we build a many-electron wave function from spin orbitals?

A many-electron wavefunction must be antisymmetric with respect to exchanging the coordinate **x** of any two electrons.

$$\psi(\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_j, \ldots, \mathbf{x}_n) = -\psi(\mathbf{x}_1, \ldots, \mathbf{x}_j, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_n)$$

First attempt (two-electron example):

$$\psi(\mathbf{x}_1, \mathbf{x}_2) = \chi_a(\mathbf{x}_1)\chi_b(\mathbf{x}_2)$$

This function is not antisymmetric. Let's fix it:

$$\psi(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{\sqrt{2}}[\chi_a(\mathbf{x}_1)\chi_b(\mathbf{x}_2) - \chi_b(\mathbf{x}_1)\chi_a(\mathbf{x}_2)]$$

This function is antisymmetric.

$$\psi(\mathbf{x}_1, \mathbf{x}_2) = -\psi(\mathbf{x}_2, \mathbf{x}_1)$$

## Slater determinants

Two electrons:

$$\psi(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{\sqrt{2}}[\chi_a(\mathbf{x}_1)\chi_b(\mathbf{x}_2) - \chi_b(\mathbf{x}_1)\chi_a(\mathbf{x}_2)]$$

## Slater determinants

Two electrons:

$$\psi(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{\sqrt{2}}[\chi_a(\mathbf{x}_1)\chi_b(\mathbf{x}_2) - \chi_b(\mathbf{x}_1)\chi_a(\mathbf{x}_2)] = \frac{1}{\sqrt{2}} \det \begin{bmatrix} \chi_a(\mathbf{x}_1) & \chi_b(\mathbf{x}_1) \\ \chi_a(\mathbf{x}_2) & \chi_b(\mathbf{x}_2) \end{bmatrix}$$

## Slater determinants

Two electrons:

$$\psi(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{\sqrt{2}}[\chi_a(\mathbf{x}_1)\chi_b(\mathbf{x}_2) - \chi_b(\mathbf{x}_1)\chi_a(\mathbf{x}_2)] = \frac{1}{\sqrt{2}} \det \begin{bmatrix} \chi_a(\mathbf{x}_1) & \chi_b(\mathbf{x}_1) \\ \chi_a(\mathbf{x}_2) & \chi_b(\mathbf{x}_2) \end{bmatrix}$$

*n* electrons:

$$\psi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = \frac{1}{\sqrt{n!}} \det \begin{bmatrix} \chi_a(\mathbf{x}_1) & \chi_b(\mathbf{x}_1) & \cdots & \chi_c(\mathbf{x}_1) \\ \chi_a(\mathbf{x}_2) & \chi_b(\mathbf{x}_2) & \cdots & \chi_c(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \chi_a(\mathbf{x}_n) & \chi_b(\mathbf{x}_n) & \cdots & \chi_c(\mathbf{x}_n) \end{bmatrix}$$

## Slater determinants

Slater determinant:

$$\psi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = \frac{1}{\sqrt{n!}} \det \begin{bmatrix} \chi_a(\mathbf{x}_1) & \chi_b(\mathbf{x}_1) & \cdots & \chi_c(\mathbf{x}_1) \\ \chi_a(\mathbf{x}_2) & \chi_b(\mathbf{x}_2) & \cdots & \chi_c(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \chi_a(\mathbf{x}_n) & \chi_b(\mathbf{x}_n) & \cdots & \chi_c(\mathbf{x}_n) \end{bmatrix}$$

Simplified notation:

$$|\psi\rangle = |\chi_a \chi_b \cdots \chi_c\rangle$$

Antisymmetry:

$$|\cdots \chi_a \cdots \chi_b \cdots\rangle = -|\cdots \chi_b \cdots \chi_a \cdots\rangle$$

In this Slater determinant, the orbitals $\{\chi_a, \chi_b, \ldots, \chi_c\}$ are occupied.

## Slater determinants

Connection to creation and annihilation operators and bitstrings:

$$a_p^\dagger|\text{vac}\rangle = |\chi_p\rangle$$

Example with 8 orbitals:

$$|\chi_1\chi_2\chi_4\rangle = a_1^\dagger a_2^\dagger a_4^\dagger|\text{vac}\rangle = |00001011\rangle$$

Simplified notation:

$$|\psi\rangle = |\chi_a\chi_b\cdots\chi_c\rangle$$

In this Slater determinant, the orbitals $\{\chi_a, \chi_b, \ldots, \chi_c\}$ are occupied.

Antisymmetry:

$$|\cdots\chi_a\cdots\chi_b\cdots\rangle = -|\cdots\chi_b\cdots\chi_a\cdots\rangle$$

## The Hartree-Fock method

The **Hartree-Fock method** approximates the ground state of the electronic Hamiltonian as a Slater determinant.

$$|\psi\rangle = |\chi_1 \chi_2 \cdots \chi_n\rangle$$

The best Slater determinant is found by optimizing over the choice of orbitals $\{\chi_1, \chi_2, \ldots, \chi_n\}$.

To implement the Hartree-Fock method on a computer, we choose a finite set of spatial basis functions $\{\phi_1, \ldots, \phi_N\}$ and use them to represent the spatial part of the spin orbitals.

$$f(\mathbf{r}) = \sum_{i=1}^{N} c_k \phi_k(\mathbf{r})$$

The computed energy depends on the choice of basis functions, which forms the **basis set**. Often, they are constructed from Gaussians. The accuracy increases as more functions are included.

## One- and two-body integrals

Electronic Hamiltonian:

$$H(\mathbf{R}) = -\sum_i \frac{\nabla_i^2}{2} - \sum_{i,A} \frac{Z_A}{r_{iA}} + \sum_{i<j} \frac{1}{r_{ij}}$$

Electronic Hamiltonian in discretized form:

$$H = \sum_{pq} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{pqrs} h_{pqrs} a_p^\dagger a_r^\dagger a_s a_q$$

The coefficients are integrals of the spin orbitals.

$$h_{pq} = \int d\mathbf{x} \chi_p^*(\mathbf{x}) \left( \frac{\nabla^2}{2} - \sum_A \frac{Z_A}{|\mathbf{R}_A - \mathbf{r}|} \right) \chi_q(\mathbf{x})$$

$$h_{pqrs} = \int \int d\mathbf{x}_1 d\mathbf{x}_2 \frac{\chi_p^*(\mathbf{x}_1)\chi_q^*(\mathbf{x}_2)\chi_s(\mathbf{x}_1)\chi_r(\mathbf{x}_2)}{|\mathbf{r}_1 - \mathbf{r}_2|}$$

# Review

## Review

### Problem mapping for combinatorial optimization

- Write the objective function in terms of binary ±1 variables $z_j$.
- Convert the objective function to a diagonal Hamiltonian by substituting the binary variable $z_j$ with the Pauli operator $Z_j$.
- Design a quantum circuit and measure its output state in the computational basis to obtain a solution to the problem.

### Problem mapping for quantum chemistry

- Discretize the electronic Hamiltonian using second quantization.
- Map fermionic creation and annihilation operators to qubit operators using e.g. the Jordan-Wigner transformation.
- Compile your circuits efficiently by taking advantage of $Z$ string cancellation.

IBM