Day 2

Quantum Optimization
Distance based classifier
Grover’s algorithm
Shor’s algorithm
Quantum phase estimation

Day 3

Applications in HEP
Data-reuploading
HHL algorithm
PennyLane implementation for combinatorial optimization

Quantum Optimization

- Introduction
- Quantum Approximation Optimization Algorithm (QAOA
- Adaptive Derivative Assembled Problem Tailored QAOA (ADAPT-QAOA)
- Feedback-based ALgorithm Quantum Optimization (FALQON)

Quantum Optimization

- Optimization problems are everywhere: math, science, business, finance etc
  - In general, time-consuming.
  - In many cases, can not be solved in polynomial time.
  - Need approximation algorithms: find approximation of the best solution rather than the best solution (time complexity is reduced).
- Two classes
  - Continuous optimization
  - Discrete optimization: combinatorial optimization
    - Quadratic Unconstrained Binary Optimization (QUBO)
- Apply quantum algorithms to solve optimization problem
  - (1) Gate model: use universal gates (Pauli's), problem-independent.
  - (2) Non-gate model (quantum annealer): relies on adiabatic theorem to find a minimum energy of Hamiltonian corresponding to the minimum value of some cost function.

Quadratic Unconstrained Binary Optimization (QUBO)

- QUBO: combinatorial optimization problem with a wide range of applications from finance to ML (partitioning, graph coloring, task allocation, max-sat, max-cut etc)

$$f : \mathbb{Z}_2^n \longrightarrow \mathbb{R}$$

Quadratic polynomial over binary variable

$$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{i} q_{ij}\, x_i\, x_j + \sum_{i=1}^{n} h_i\, x_i$$

$$x_i \in \mathbb{Z}_2 = \{0,1\}, \;\; h_i, q_{ij} \in \mathbb{R}$$

$$x = x_n x_{n-1} \cdots x_2 x_1$$

(binary strings of n-bits)

- Find a binary vector x^* which minimizes f

$$x^* = \operatorname*{argmin}_{x \in \mathbb{Z}_2^n} f(x)$$

- In matrix notation, $f(x) = x^T Q x,$ where $Q \in \mathbb{R}^{n \times n}$

Quadratic Unconstrained Binary Optimization (QUBO)

- In matrix notation, $f(x) = x^T Q x,$ where $Q \in \mathbb{R}^{n \times n}$

$$f(x) = -2x_1 - 3x_2 + 8x_3 + 4x_4 + 4x_1x_2 + 5x_1x_3 + 6x_2x_3 + 10x_3x_4$$

$$= (x_1\, x_2\, x_3\, x_4) \begin{pmatrix} -2 & 2 & 5/2 & 0 \\ 2 & -3 & 3 & 0 \\ 5/2 & 3 & 8 & 5 \\ 0 & 0 & 5 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = x^T\, Q\, x$$

$$x_i = x_i^2$$

$$x_i \in Z_2 = \{0, 1\}$$

- QUBO:
  - NP hard problem
  - Quadratic function might have several local minima
  - Close connection to Ising model

P vs NP

- In Theoretical Computer Science, the two most basic classes of problems are P and NP.
- P includes all problems that can be solved efficiently.
  - For example: add two numbers. The formal definition of "efficiently" is in time that's polynomial in the input's size.
- NP (nondeterministic polynomial (time)) includes all problems that given a solution, one can efficient verify that the solution is correct.
  - An example is the following problem: given a bunch of numbers, can they be split into two groups such that the sum of one group is the same as the other. Clearly, if one is given a solution (two groups of numbers), it's simple to verify that the sum is the same. (This is a partitioning problem).
- What's unknown is whether problems such as the one above have an efficient algorithm for finding the solution. This is the (in)famous (unsolved) P = NP problem, and the common conjecture is that no such algorithm exists.
- Now, NP hard problems are such problems that were shown that if they can be efficiently solved (which, as mentioned, is believed to not be the case), then each and every problem in NP (each and every problem whose results can be efficiently verified) can be efficiently solved. In other words, if you're up to showing that P=NP, you might want to take a stand at any of those NP-hard problems since they are "equivalent" in some way to all others.

Ising Model

- Mathematical model for ferromagnetism in statistical mechanics.
- The energy of spin configuration for a given lattice is given by the following classical Hamiltonian

$$E(s) = -\sum_{i,j} J_{ij}\, s_i\, s_j - \sum_i h_i\, s_i \quad s = \{s_i\}, \quad s_i \in \{-1,1\}$$

- J_{ij} is called an interaction, spin-spin coupling, and h_i is an external magnetic field, interacting with spin s_i.
- The configuration probability is given by the Boltzmann distribution

$$P(s) = \frac{e^{-\beta H(s)}}{\sum_s e^{-\beta H(s)}}, \qquad \beta = \frac{1}{k_B T}$$

- Quantum Ising model: $$H = -\sum_{i,j} J_{ij}\, \sigma_i^z\, \sigma_j^z - \sum_i h_i\, \sigma_i^z$$

Quadratic Unconstrained Binary Optimization (QUBO)

- QUBO: combinatorial optimization problem with a wide range of applications from finance to ML (partitioning, graph coloring, task allocation, max-sat, max-cut etc)

$$f : \mathbb{Z}_2^n \longrightarrow \mathbb{R}$$

Quadratic polynomial over binary variable

$$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{i} q_{ij}\, x_i\, x_j + \sum_{i=1}^{n} h_i\, x_i$$

$$x_i \in \mathbb{Z}_2 = \{0,1\}, \;\; h_i, q_{ij} \in \mathbb{R}$$

$$x = x_n x_{n-1} \cdots x_2 x_1$$

(binary strings of n-bits)

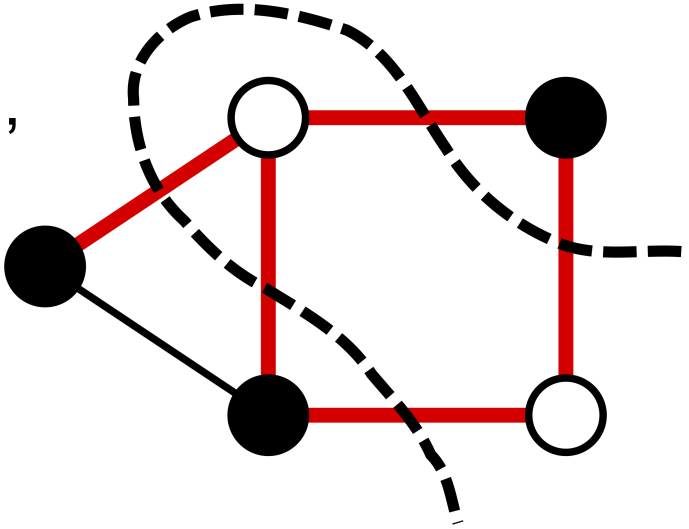
- Find a binary vector $x*$ which minimizes f

$$x* = \underset{x \in \mathbb{Z}_2^n}{\operatorname{argmin}}\, f(x)$$

- In matrix notation, $f(x) = x^T Q x,$ where $Q \in \mathbb{R}^{n \times n}$

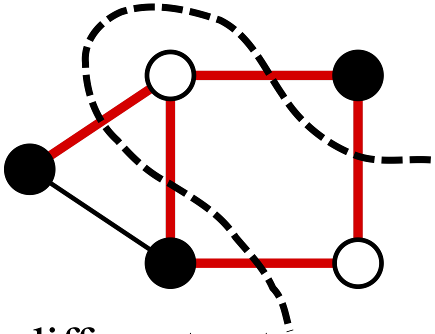
QUBO example: Max-cut Problem

- Max-Cut is the NP-hard problem of finding a partition of the graph's vertices into an two distinct sets that maximizes the number of edges between the two sets.
- Undirected Graph: G = (V, E)
  - V: set of nodes, and E: set of edges
- Partition vertices into two complementary sets such that the number of edges between the two sets is as large as possible.
- As the Max-Cut Problem is NP-hard, no polynomial-time algorithms for Max-Cut in general graphs are known.



QUBO example: Max-cut Problem

- The cost function to be maximized:



$$C(x) = \sum_{(i,j)\in E}\left(x_i + x_j - 2x_ix_j\right) \quad \text{where } x_i \in \{0,1\}$$

$$x_i + x_j - 2x_ix_j = 1, \text{ if } x_i \text{ and } x_j \text{ belong in different sets}.$$

$s_i \in Z_2 = \{-1,1\}$

$$x_i + x_j - 2x_ix_j = 0, \text{ if } x_i \text{ and } x_j \text{ belong in the same set}.$$

- Introducing $x_i = \dfrac{s_i + 1}{2}$, the cost function can be rewritten

$$C(s) = \frac{1}{2}\sum_{(i,j)\in E}\left(1 - s_is_j\right) \longrightarrow C(s) = \frac{1}{2}\sum_{(i,j)\in E}\left(1 - \sigma_i^z\sigma_j^z\right)$$

$(i,j):$ the edge index

$i:$ vertex index

$$\sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad \sigma^z|0\rangle = +1|0\rangle \quad \sigma^z|1\rangle = -1|1\rangle$$

σ_i^z : Pauli's Z matrix actingon the i^{th} vertex

σ_j^z : Pauli's Z matrix actingon the j^{th} vertex

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

Matrices = linear operators = observables

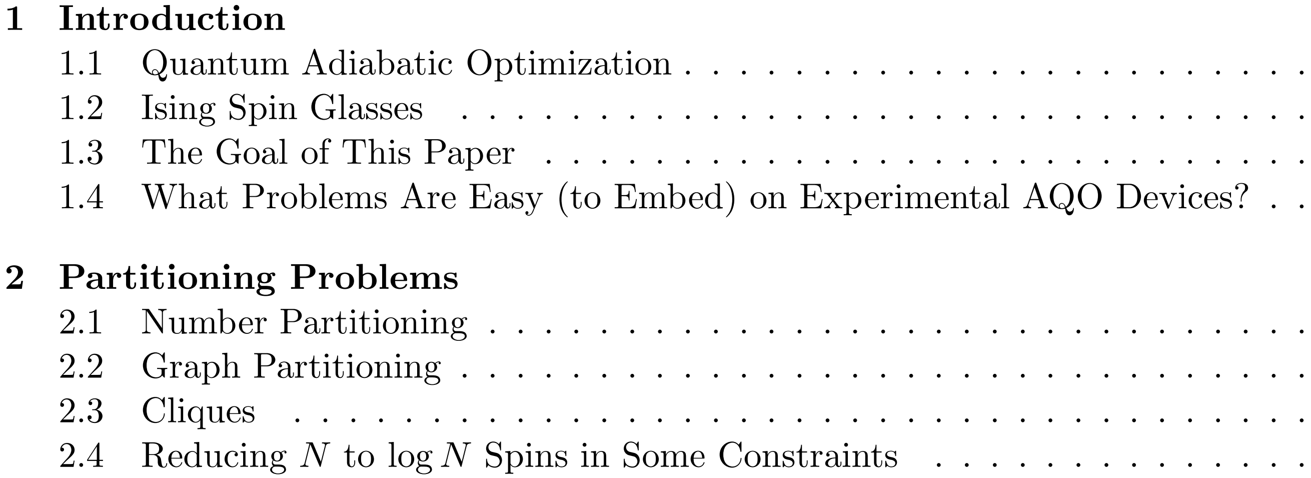
Eigenvalues = what are actually measured in experiments

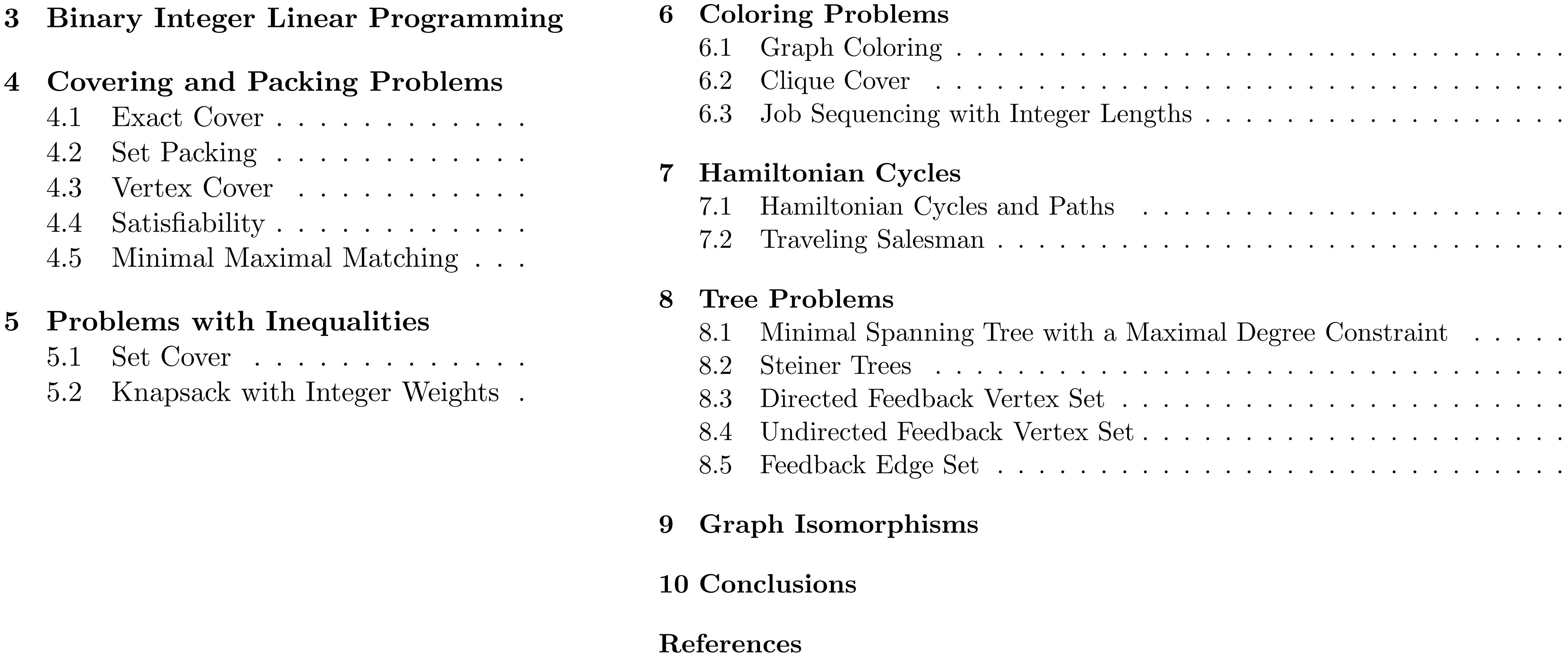
Ising formulations of many NP problems

Andrew Lucas

1302.5843

$$H = -\sum_{i,j} J_{ij}\,\sigma_i^z\,\sigma_j^z - \sum_i h_i\,\sigma_i^z$$





- It is possible to generalize the Ising model beyond QUBO.
  - Polynomial unconstrained binary optimization (PUBO)
  - Higher-order unconstrained binary optimization (HUBO)
  - Unconstrained binary quadratic problem (UBQP)
  - Constrained optimization problems

Adiabatic Theorem

- Schrodinger equation: $i\hbar \frac{d\psi(t)}{dt} = H(t)\,\psi(t)$
- Instantaneous eigenstate: $H(t)\,\psi_n(t) = E_n(t)\,\psi_n(t)$
- Initial condition: $\psi(t=0) = \psi_0$
- If evolution is slow enough, $\psi(t) \approx e^{i\theta(t)}\,\psi_0$

Born and Folk 1928

$$\psi(t) = U(t)\,\psi(0)$$

$$U_I(t) = \sum_{q=0}^{\infty} (-i)^q \int_0^t \mathrm{d}t_q \cdots \int_0^{t_2} \mathrm{d}t_1 H_I(t_q) \cdots H_I(t_1)$$

Adiabatic Theorem

- Adiabatic theorem: A physical system remains in its instantaneous eigenstate, if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum. (Max Born and Vladimir Folk 1928)
- Under a slowly changing Hamiltonian H(t) with instantaneous eigenstate $|n(t)\rangle$ and the corresponding energy $E(t)$, a quantum system evolves from initial state $|\psi(0)\rangle = \sum_n c_n(0)\,|n(0)\rangle$ to final state $|\psi(t)\rangle = \sum_n c_n(t)\,|n(t)\rangle$ where

  $c_n(t) = c_n(0)\,e^{i\theta_n(t)}\,e^{i\gamma_n(t)}$ with the dynamical phase $\theta_n(t) = -\dfrac{1}{\hbar}\displaystyle\int_0^t E_n(t')\,dt'$ and

  geometrical phase $\gamma(t) = i\displaystyle\int_0^t \langle n(t')\,|\,\dot{n}(t')\rangle\,dt'$
- Adiabatic approximation: the rate of change of Hamiltonian $\dot{H}(t)$ is small and there is finite gap $E_m(t) - E_n(t) \neq 0$ between energies for $m \neq n$ →

  $$\langle n(t')\,|\,\dot{n}(t')\rangle = -\frac{\langle m(t)\,|\,\dot{H}(t)\,|\,n(t)\rangle}{E_m(t) - E_n(t)} \to 0$$
- $|c_n(t)|^2 = |c_n(0)|^2$ so if the system begins in an eigenstate of H(0), it remains in an eigenstate of H(t) during the evolution with a change of phase only.

Adiabatic Theorem

$H(t)\,|\,n(t)\rangle = E_n(t)\,|\,n(t)\rangle$ $\qquad |\,n(t)\rangle$: is eigenstates of Hamiltonian, basis

$|\,\psi(t)\rangle = \sum_n c_n(t)\,|\,n(t)\rangle$ $\qquad$ satisfies time-dependent Schrödinger equation $\qquad i\hbar\frac{\partial}{\partial t}|\,\psi(t)\rangle = H(t)\,|\,\psi(t)\rangle$

$$\frac{d}{dt}: \quad \dot{H}(t)\,|\,n(t)\rangle + H(t)\,|\,\dot{n}(t)\rangle = \dot{E}_n(t)\,|\,n(t)\rangle + E_n(t)\,|\,\dot{n}(t)\rangle$$

Assume $m \neq n$ and perform inner product with $|\,m(t)\rangle$: $\qquad H(t)\,|\,m(t)\rangle = E_m(t)\,|\,m(t)\rangle$

$$\langle m(t)\,|\,n(t)\rangle = \delta_{mn},$$

$$\langle m(t)\,|\,\dot{H}(t)\,|\,n(t)\rangle + \langle m(t)\,|\,H(t)\,|\,\dot{n}(t)\rangle = \dot{E}_n(t)\,\langle m(t)\,|\,n(t)\rangle + E_n(t)\,\langle m(t)\,|\,\dot{n}(t)\rangle$$

$$\langle m(t)\,|\,\dot{H}(t)\,|\,n(t)\rangle + E_m(t)\,\langle m(t)\,|\,\dot{n}(t)\rangle = E_n(t)\,\langle m(t)\,|\,\dot{n}(t)\rangle \quad \rightarrow \quad \boxed{\langle m(t)\,|\,\dot{n}(t)\rangle = -\frac{\langle m(t)\,|\,\dot{H}(t)\,|\,n(t)\rangle}{E_m(t) - E_n(t)}}$$

Adiabatic approximation: the rate of change in Hamiltonian $\dot{H}(t)$ is small and there is finite gap $E_m(t) - E_n(t) \neq 0$ between energies $\rightarrow$ $\langle m(t)\,|\,\dot{n}(t)\rangle \approx 0$.

Adiabatic Theorem

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = H(t)|\psi(t)\rangle$$

$$|\psi(t)\rangle = \sum_n c_n(t)|n(t)\rangle$$

$$\rightarrow\ i\hbar\sum_n \dot{c}_n(t)|n(t)\rangle + c_n(t)|\dot{n}(t)\rangle = \sum_n E_n(t)\,c_n(t)|n(t)\rangle$$

Inner product with $|m(t)\rangle$: $\langle m(t)|\left[i\hbar\sum_n \dot{c}_n(t)|n(t)\rangle + c_n(t)|\dot{n}(t)\rangle = \sum_n E_n(t)\,c_n(t)|n(t)\rangle\right]$

Using $\langle m(t)|n(t)\rangle = \delta_{mn}$, we obtain

$$i\hbar\,\dot{c}_m(t) + i\hbar\sum_n c_n(t)\,\langle m(t)|\dot{n}(t)\rangle = c_m(t)\,E_m(t)$$

In the adiabatic limit,
$\langle m(t)|\dot{n}(t)\rangle \approx 0 \text{ for } m \neq n$

$$i\hbar\,\dot{c}_m(t) + i\hbar c_m(t)\,\langle m(t)|\dot{m}(t)\rangle = c_m(t)\,E_m(t)$$

$$i\,\dot{c}_m(t) = \left(\frac{E_m(t)}{\hbar} - i\,\langle m(t)|\dot{m}(t)\rangle\right)c_m(t) \quad\rightarrow\quad \dot{c}_m(t) = i\left(-\frac{E_m(t)}{\hbar} + i\,\langle m(t)|\dot{m}(t)\rangle\right)c_m(t)$$

$$\frac{d}{dt}\ln c_m(t) = \frac{1}{c_m(t)}\frac{dc_m(t)}{dt} = \frac{\dot{c}_m(t)}{c_m(t)} = -\frac{i}{\hbar}E_m(t) + i\,i\langle m(t)|\dot{m}(t)\rangle$$

$$c_m(t) = c_m(0)e^{i\theta_m(t)}e^{i\gamma_m(t)} \qquad \theta_m(t) = -\frac{1}{\hbar}\int_0^t E_m(t')\,dt' \qquad \gamma(t) = i\int_0^t \langle m(t')|\dot{m}(t')\rangle\,dt'$$

dynamical phase,
real, function of E

geometrical phase,
pure real

Adiabatic Theorem

$$c_m(t) = c_m(0)e^{i\theta_m(t)}e^{i\gamma_m(t)} \qquad \theta_m(t) = -\frac{1}{\hbar}\int_0^t E_m(t')\,dt' \qquad \gamma(t) = i\int_0^t \langle m(t')\,|\,\dot{m}(t')\rangle\,dt'$$

dynamical phase,
real, function of E

geometrical phase,
pure real,
Has something to do with
direction in the Hilbert space

$$0 = \frac{d}{dt}\langle m(t)\,|\,m(t)\rangle = \langle \dot{m}(t)\,|\,m(t)\rangle + \langle m(t)\,|\,\dot{m}(t)\rangle$$

$$= \langle m(t)\,|\,\dot{m}(t)\rangle^* + \langle m(t)\,|\,\dot{m}(t)\rangle$$

$$= 2\,\mathrm{Re}\,\langle m(t)\,|\,\dot{m}(t)\rangle \qquad \rightarrow \quad \gamma_m(t): \ \text{pure real}$$

$$\langle \phi\,|\,\psi\rangle^* = \langle \psi\,|\,\phi\rangle$$

Adiabatic Theorem

- Schrodinger equation: $$i\hbar \frac{d\psi(t)}{dt} = H(t)\,\psi(t)$$
- Instantaneous eigenstate: $H(t)\,\psi_n(t) = E_n(t)\,\psi_n(t)$
- Initial condition: $\psi(t = 0) = \psi_0$
- If evolution is slow enough, $\psi(t) \approx e^{i\theta(t)}\,\psi_0$

Born and Folk 1928

$$\psi(t) = U(t)\,\psi(0)$$

$$U_I(t) = \sum_{q=0}^{\infty} (-i)^q \int_0^t \mathrm{d}t_q \cdots \int_0^{t_2} \mathrm{d}t_1 H_I(t_q) \cdots H_I(t_1)$$

Quantum Annealing

- H_p is the problem Hamiltonian whose ground state encodes the solution to the optimization problem
- H_0 is the initial Hamiltonian whose ground state is easy to prepare.
- Prepare a quantum system to be in the ground state of H_0 and evolve the system using the following time-dependent Hamiltonian,

$$H(t) = \left(1 - \frac{t}{T}\right) H_0 + \frac{t}{T} H_p$$

- The system will remain to its ground state at all times, which means for t=T, the system will be in the ground state of H_p, our problem Hamiltonian.
- D-wave has built Quantum Annealing that solves optimization problem by transferring the original optimization to a hardware, that allows nearest neighbor interaction of qubits.
- If the energy gap b/w the ground state and 1st excited state is small, T must be very large → computationally difficult.

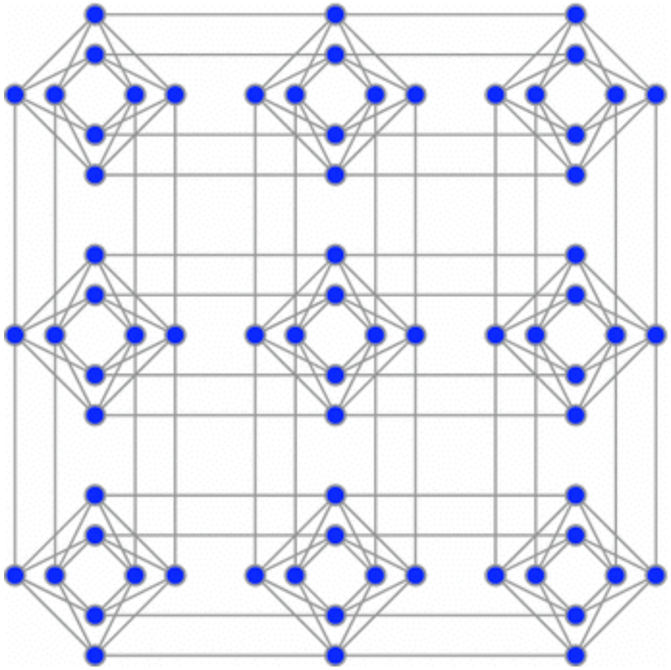
Apolloni, Bianchi, De Falco 1988

Breaking limitation of quantum annealer in solving optimization problems under constraints

Masayuki Ohzeki[1,2,3*]

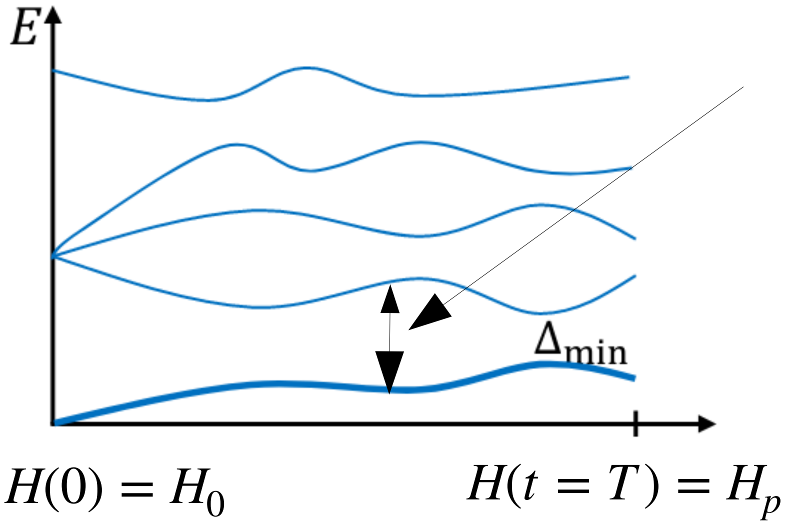
Quantum annealing is a generic solver for optimization problems that uses fictitious quantum fluctuation. The most groundbreaking progress in the research field of quantum annealing is its hardware implementation, i.e., the so-called quantum annealer, using artificial spins. However, the connectivity between the artificial spins is sparse and limited on a special network known as the chimera graph. Several embedding techniques have been proposed, but the number of logical spins, which represents the optimization problems to be solved, is drastically reduced. In particular, an optimization problem including fully or even partly connected spins suffers from low embeddable size on the chimera graph. In the present study, we propose an alternative approach to solve a large-scale optimization problem on the chimera graph via a well-known method in statistical mechanics called the Hubbard-Stratonovich transformation or its variants. The proposed method can be used to deal with a fully connected Ising model without embedding on the chimera graph and leads to nontrivial results of the optimization problem. We tested the proposed method with a number of partition problems involving solving linear equations and the traffic flow optimization problem in Sendai and Kyoto cities in Japan.

2002.05298



Limitation of Quantum Annealing

- Performance of quantum annealing are governed by the size of the gap.



$$\Delta t \gg \max_{0 \le \le 1} \left| \frac{\langle 1(s) | \frac{dH(s)}{ds} | 0(s) \rangle}{\Delta(s)^2} \right|$$

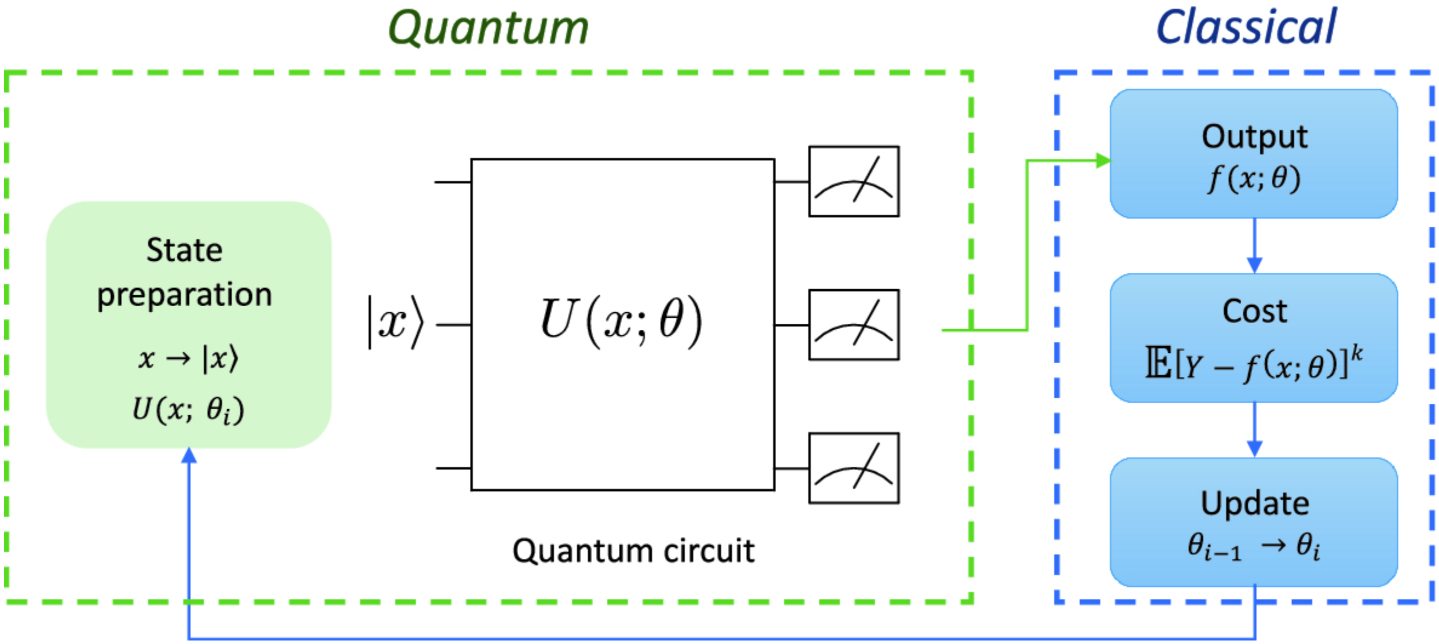
$$T = \Delta s\,, \quad 0 \le s \le 1$$

- Performance is poor, when eigenvalues are degenerate.

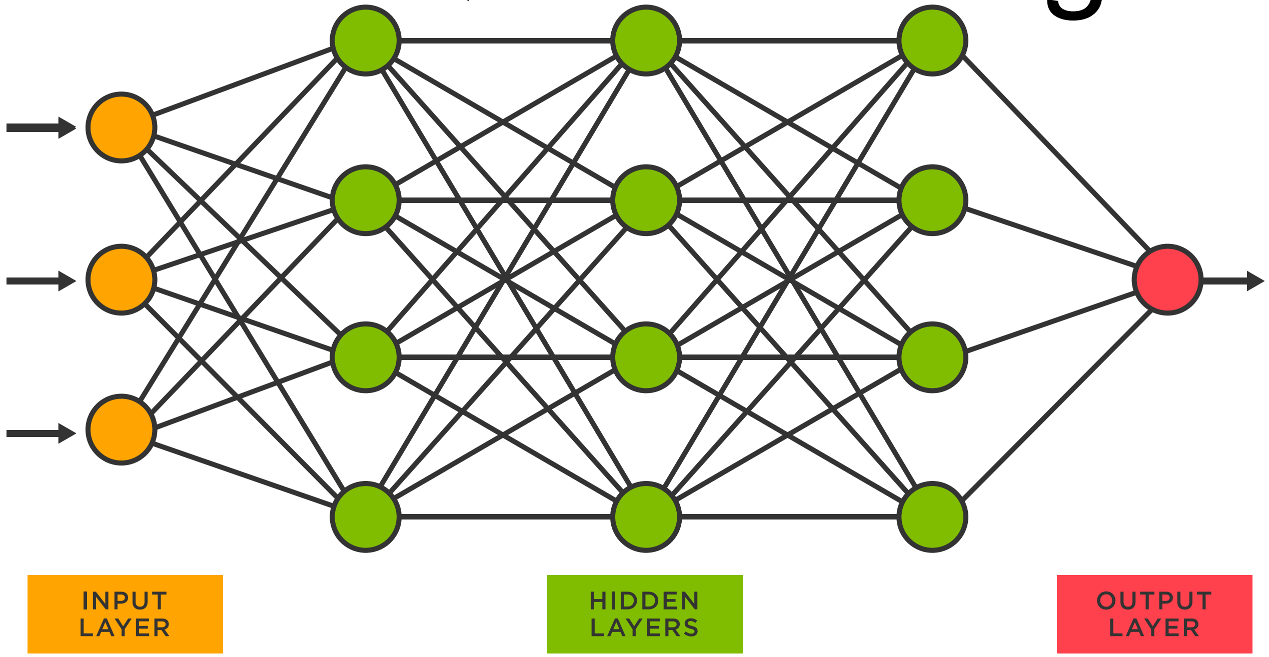
Variational Quantum Algorithms

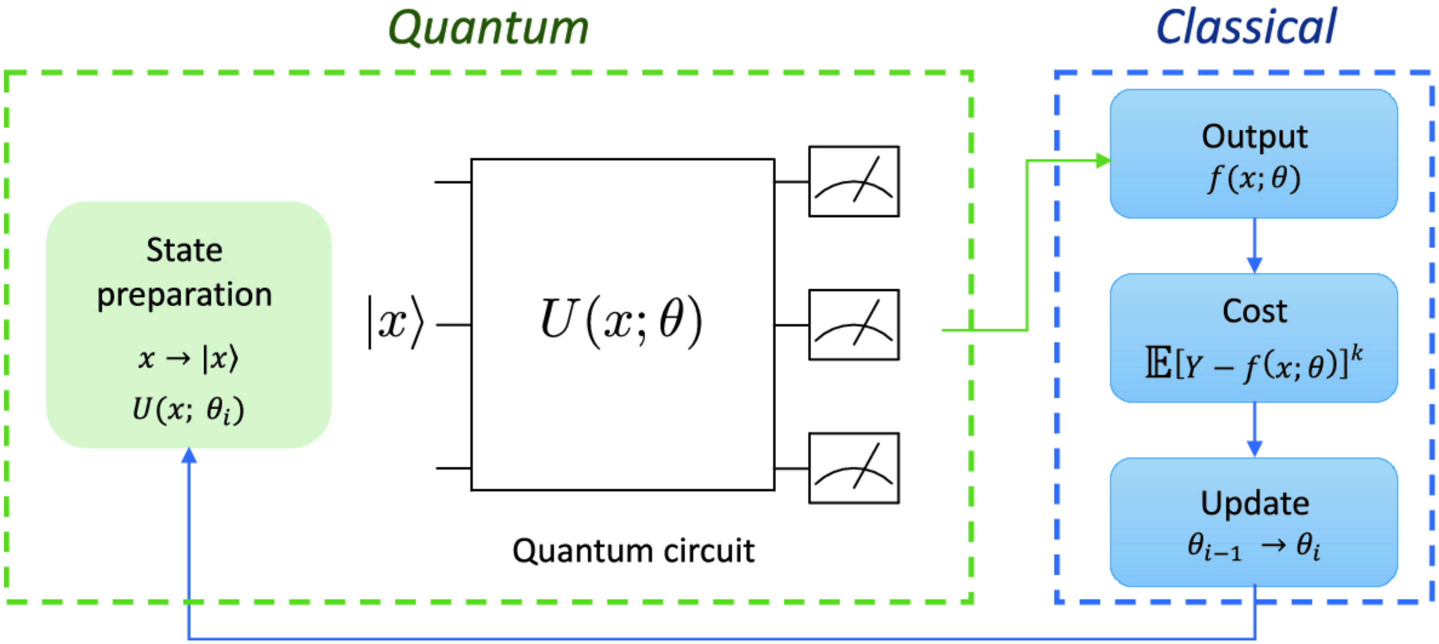
- Hybrid quantum-classical model is suggested to circumvent the issue of going slow with quantum annealer as well as implementing Hamiltonian in the available hardware.
- Quantum: parameterize wave function
- Classical: minimize/maximize the expectation value of H in the problem.

$$E(\vec{\theta}) = \langle\, \psi(\vec{\theta})\,|\, H \,|\, \psi(\vec{\theta}) \,\rangle$$



Variational Quantum Algorithms



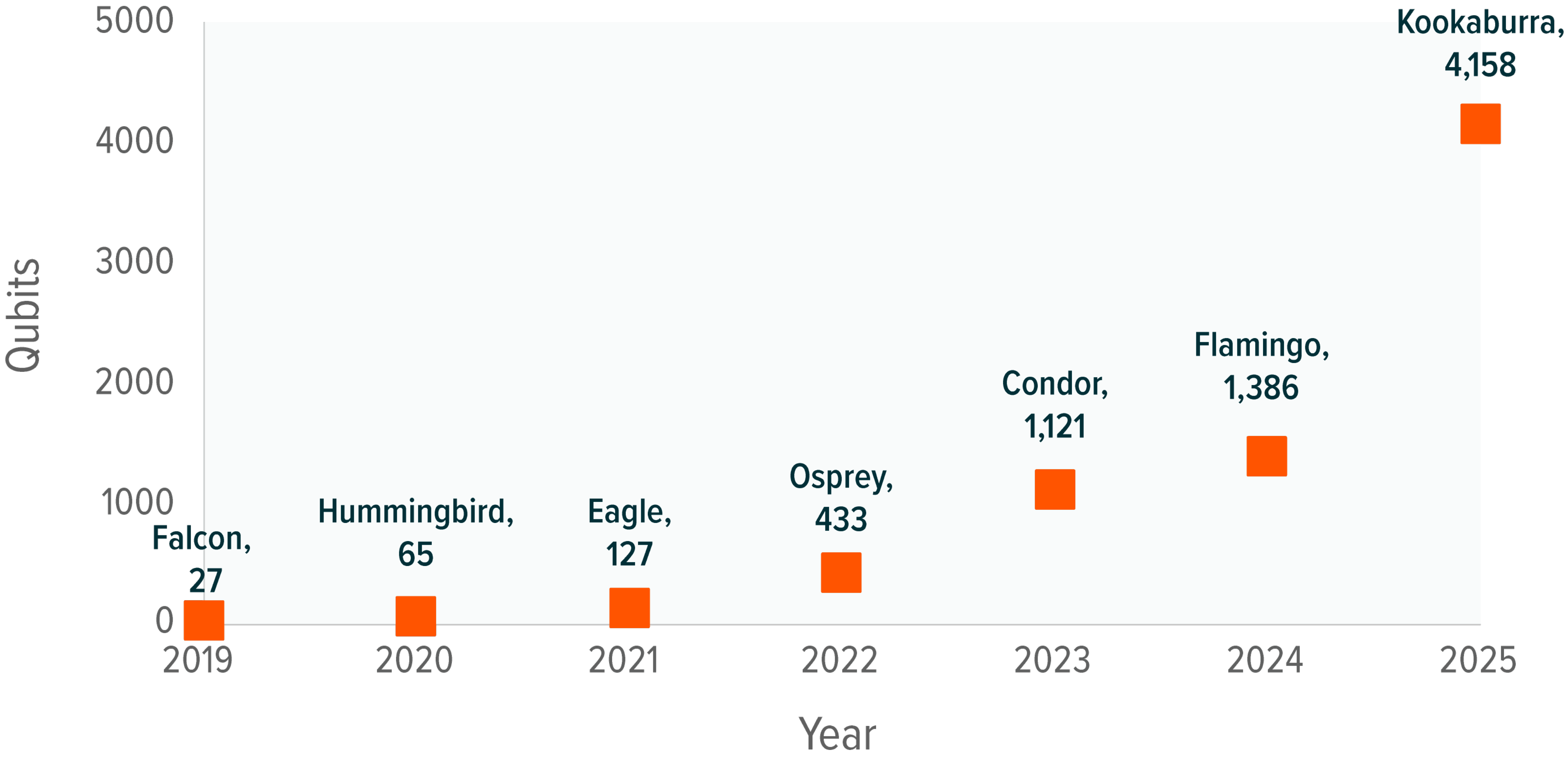


Variational Quantum Algorithms

- 2016: first cloud-based quantum computer became available.
- Current state-of-the-art device size ranges from 50 to 100 qubits which allows one to achieve ‘quantum supremacy’: outperforming the best classical supercomputer, for certain contrived mathematical tasks.
  - Sycamore (53 qubits, corresponding to a computational state-space of dimension $2^{53} \approx 10^{16}$): 200 seconds vs 10,000 years for sampling the output of a pseudo-random quantum circuit.
- Variational Quantum Algorithms (VQAs) have emerged as the leading strategy to obtain quantum advantage on NISQ (Noisy Intermediate-Scale Quantum) devices. Accounting for all of the constraints imposed by NISQ computers with a single strategy requires an optimization-based or learning- based approach, precisely what VQAs use.
- VQAs are arguably the quantum analog of highly successful machine-learning methods, such as neural networks.
- VQAs leverage the toolbox of classical optimization, since VQAs use parametrized quantum circuits to be run on the quantum computer, and then outsource the parameter optimization to a classical optimizer. This approach has the added advantage of keeping the quantum circuit depth shallow and hence mitigating noise, in contrast to quantum algorithms developed for the fault-tolerant era.

IBM QUANTUM PROCESSORS ROADMAP

Sources: Global X analysis of information derived from: IBM. (2022). Our new 2022 development roadmap.



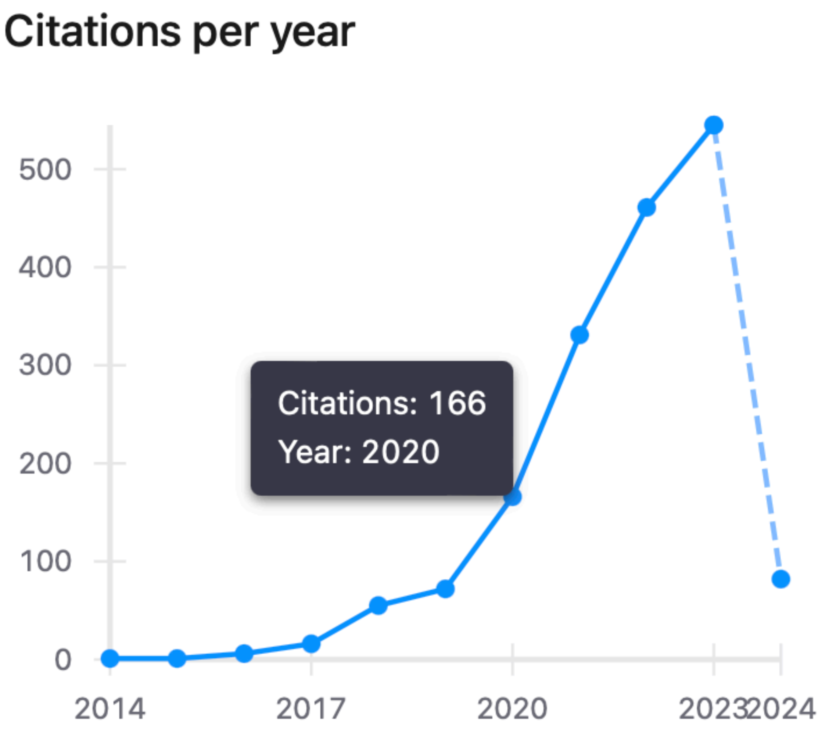
Note: 2022 onwards includes planned processor launches.

Quantum Approximate Optimization Algorithm (QAOA)

- Abstract: We introduce a quantum algorithm that produces approximate solutions for combinatorial optimization problems. The algorithm depends on a positive integer p and the quality of the approximation improves as p is increased. The quantum circuit that implements the algorithm consists of unitary gates whose locality is at most the locality of the objective function whose optimum is sought. The depth of the circuit grows linearly with p times (at worst) the number of constraints. If p is fixed, that is, independent of the input size, the algorithm makes use of efficient classical preprocessing. If p grows with the input size a different strategy is proposed. We study the algorithm as applied to MaxCut on regular graphs and analyze its performance on 2-regular and 3-regular graphs for fixed p. For p = 1, on 3-regular graphs the quantum algorithm always finds a cut tha is at least 0.6924 times the size of the optimal cut.

1411.4028

E. Farhi,
J. Goldstone,
S. Gutmann



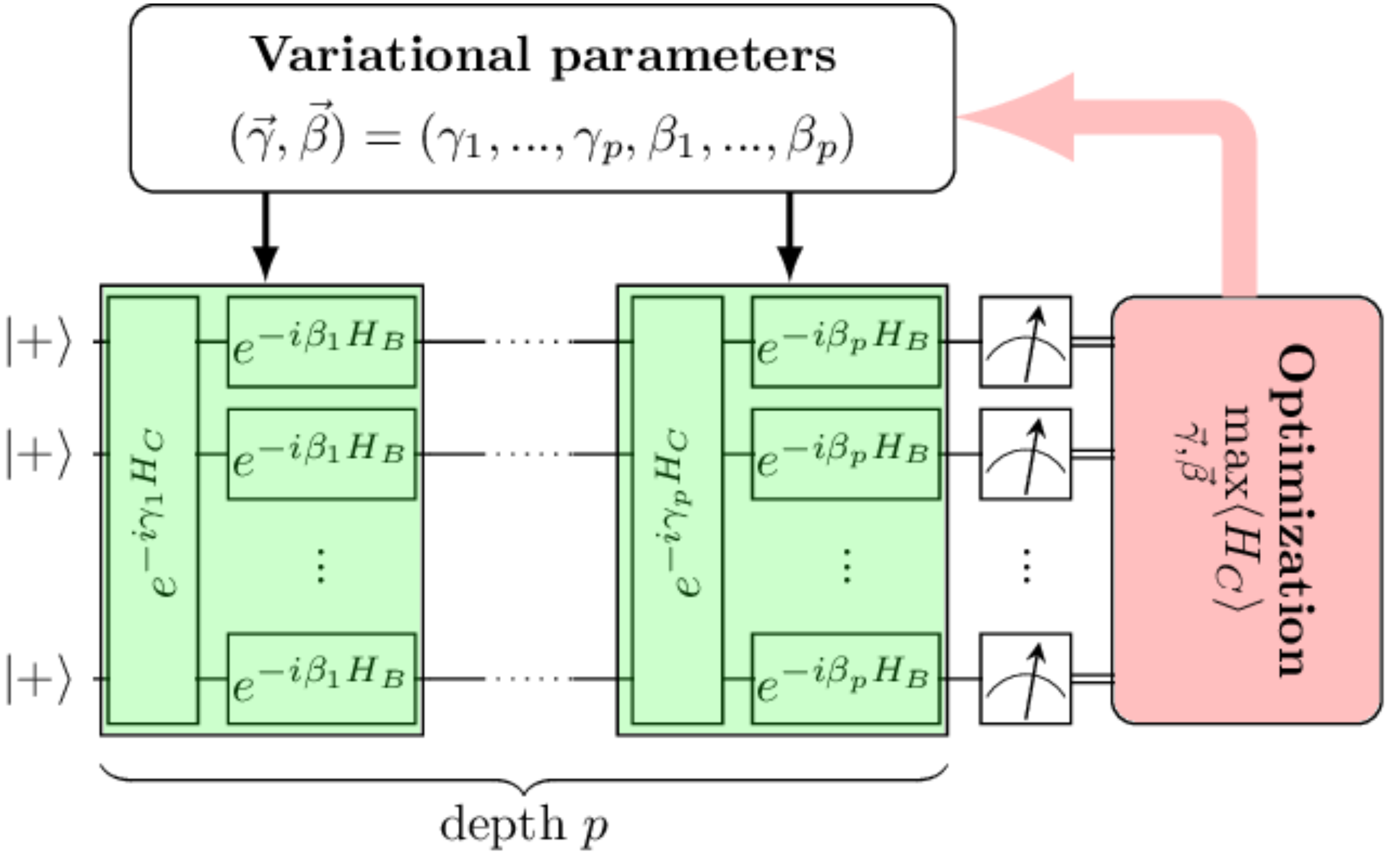
Why should we care about QAOA?

- Solve optimization problems
  - Solves quadratic unconstrained problems with binary variables
- Near-term algorithm
  - Algorithm runs on small quantum computers
  - Low depth, robust to errors
  - Requires relatively few physical qubits to get to interesting practical problem sizes
- Adaptable algorithm
  - In principle, we can easily model the objective function that we are trying to solve
- Expected to be faster than classical
  - Classical approaches move through the search space one solution at a time
  - In quantum, we can create a superposition of states and operate in all states in parallel.

Quantum Approximate Optimization Algorithm (QAOA)

Farhi et al 2014

- Hybrid quantum algorithm: contains a parameterized quantum circuit which depends on variational parameters.
- Use classical computer to optimize the output of the quantum circuit.
- Consider the Ising model for illustration.


Variational parameters
$(\vec{\gamma}, \vec{\beta}) = (\gamma_1, ..., \gamma_p, \beta_1, ..., \beta_p)$
$|+\rangle$
$|+\rangle$
$|+\rangle$
$e^{-i\gamma_1 H_C}$
$e^{-i\beta_1 H_B}$
$e^{-i\gamma_p H_C}$
$e^{-i\beta_p H_B}$
Optimization
$\max_{\vec{\gamma},\vec{\beta}} \langle H_C \rangle$
depth p

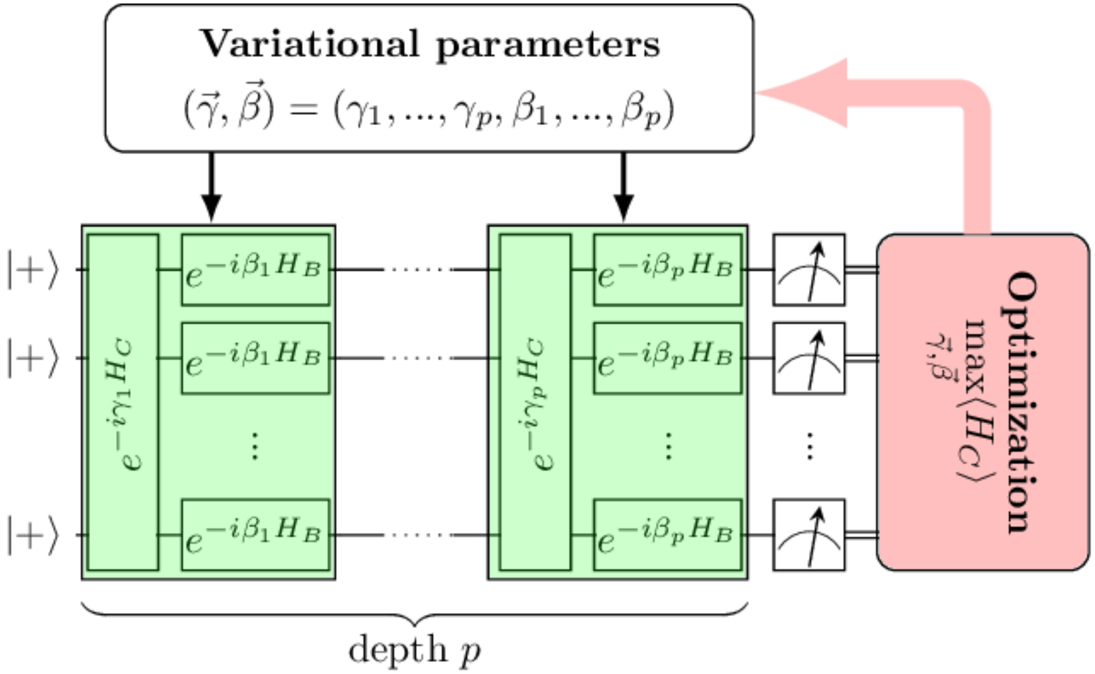
Quantum Approximate Optimization Algorithm (QAOA)

$$H(t) = \left(1 - \frac{t}{T}\right) H_M + \frac{t}{T} H_P$$

$H_M = H_B$: mixer Hamiltonian

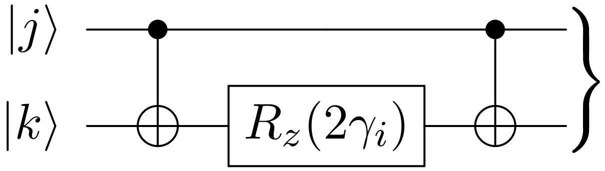
$H_P = H_C$: problem Hamiltonian

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = \prod_{j=1}^{p} U(H_M, \beta_j) U(H_P, \gamma_j) \, |+\rangle^{\otimes n}$$



$$U(H_P, \gamma_i) = \exp\left[-i\gamma_i \sum_{j,k \in E} \sigma_j^z \sigma_k^z\right] = \prod_{j,k \in E} \mathrm{CNOT}_{j,k} R_z^k(2\gamma_i) \mathrm{CNOT}_{j,k}$$

$$U(H_M, \beta_j) = \exp\left[-i\beta_j \sum_{i=1}^{n} \sigma_i^X\right]$$



$$\exp\left[-i(-1)^{j+k}\gamma_i\right] |jk\rangle$$

$$= \prod_{i=1}^{n} e^{-i\beta_j \sigma_i^X}$$

$$= \prod_{i=1}^{n} R_x^i(2\beta_j)$$

$$F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \boldsymbol{\gamma}, \boldsymbol{\beta} | H_P | \boldsymbol{\gamma}, \boldsymbol{\beta} \rangle$$

Quantum Approximate Optimization Algorithm (QAOA)

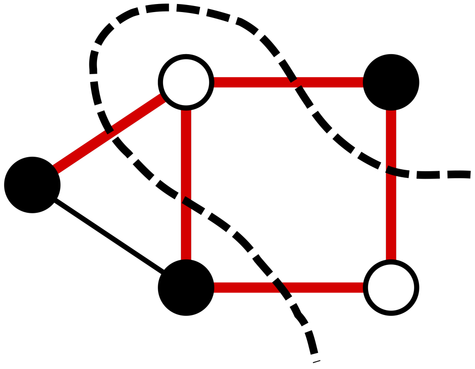
Farhi et al 2014

$$H_P = C(s) = H_C = \frac{1}{2}\sum_{(i,j)\in E}\left(1-\sigma_i^z\sigma_j^z\right)$$: Problem Hamiltonian

(i,j) : the edge index

i : vertex index

$$H_M = B = H_B = \sum_j \sigma_j^X$$: Mixer Hamiltonian



Full Hamiltonian: $$H(t) = \left(1-\frac{t}{T}\right)H_M + \frac{t}{T}H_P$$

$$|\psi\rangle = \exp\left[-i\int_0^t H(t')\,\mathrm{d}t'\right]|\psi_0\rangle \;=\exp\left[-i\sum_{j=1}^{p}H(j\Delta t)\Delta t\right]|\psi_0\rangle$$

Undirected Graph: G = (V, E)
V: set of nodes
E: set of edges

$$\approx\prod_{j=1}^{p}\exp\left[-i\Delta t\left[\left(1-\frac{j\Delta t}{T}\right)H_M+\frac{j\Delta t}{T}H_P\right]\right]|\psi_0\rangle$$

$$\approx\prod_{j=1}^{p}\exp\left[-i\Delta t\left(1-\frac{j\Delta t}{T}\right)H_M\right]\exp\left[-i\Delta t\frac{j\Delta t}{T}H_P\right]|\psi_0\rangle$$

Works in the adiabatic limit or $p\rightarrow\infty$

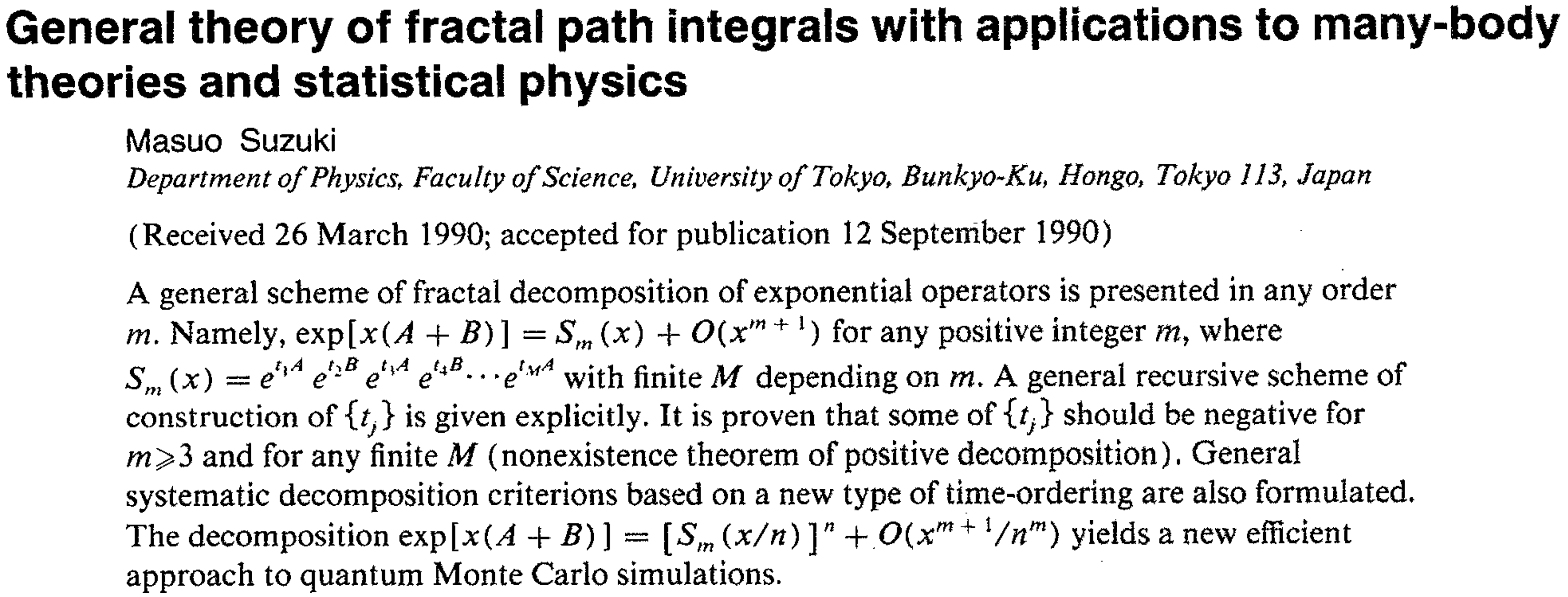
$$=\prod_{j=1}^{p}\underbrace{\exp\left[-i\beta_j H_M\right]}_{U(H_M,\beta_j)}\underbrace{\exp\left[-i\gamma_j H_P\right]}_{U(H_P,\gamma_j)}|\psi_0\rangle \quad\Rightarrow\quad |\boldsymbol{\gamma},\boldsymbol{\beta}\rangle=\prod_{j=1}^{p}U(H_M,\beta_j)U(H_P,\gamma_j)\,|+\rangle^{\otimes n}$$

Trotter formulas or Trotter–Suzuki decompositions

- Product formulas simulate the sum of terms of a Hamiltonian by simulating each one separately for a small time slice.

For $H = A + B + C$, $\quad U = e^{-i(A+B+C)t} = \left(e^{-iA\frac{t}{r}}\, e^{-iB\frac{t}{r}}\, e^{-iC\frac{t}{r}}\right)^r$, for a large r

r = the number of time steps to simulate for.



General theory of fractal path integrals with applications to many-body theories and statistical physics

Masuo Suzuki
Department of Physics, Faculty of Science, University of Tokyo, Bunkyo-Ku, Hongo, Tokyo 113, Japan

(Received 26 March 1990; accepted for publication 12 September 1990)

A general scheme of fractal decomposition of exponential operators is presented in any order m. Namely, $\exp[x(A+B)] = S_m(x) + O(x^{m+1})$ for any positive integer m, where $S_m(x) = e^{t_1A}\, e^{t_2B}\, e^{t_3A}\, e^{t_4B} \cdots e^{t_MA}$ with finite M depending on m. A general recursive scheme of construction of $\{t_j\}$ is given explicitly. It is proven that some of $\{t_j\}$ should be negative for $m \geqslant 3$ and for any finite M (nonexistence theorem of positive decomposition). General systematic decomposition criterions based on a new type of time-ordering are also formulated. The decomposition $\exp[x(A+B)] = [S_m(x/n)]^n + O(x^{m+1}/n^m)$ yields a new efficient approach to quantum Monte Carlo simulations.

Quantum Approximate Optimization Algorithm (QAOA)

Farhi et al 2014

$$C(s) = \frac{1}{2} \sum_{(i,j)\in E} \left(1 - \sigma_i^z \sigma_j^z\right), \quad B = \sum_j \sigma_j^X$$

(i, j) : the edge index

i : vertex index

$$\sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

σ_i^z : Pauli's Z matrix actingon the i^{th} vertex

σ_j^z : Pauli's Z matrix actingon the j^{th} vertex

$$\sigma^z |0\rangle = +1\,|0\rangle \qquad \sigma^z |1\rangle = -1\,|1\rangle \qquad |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$U(C,\gamma) = e^{-i\gamma C} = \prod_{(i,j)\in E} e^{-i\gamma C_{ij}}, \quad U(B,\beta) = e^{-i\beta B} = \prod_{j=1}^{n} e^{-i\beta B_j}$$

$$|\psi(\vec{\gamma},\vec{\beta})\rangle = \left[\prod_{i=1}^{p} U(B,\beta_i) U(C,\gamma_i)\right] H^{\otimes n} |0\rangle$$

$$= U(B,\beta_p)\, U(C,\gamma_p) \cdots U(B,\beta_1)\, U(C,\gamma_1) \frac{1}{\sqrt{2^n}} \sum_{i=1}^{2^n-1} |i\rangle$$

$$e^{-i\beta\sigma_j^X} = \cos\beta - i\sigma_j^X \sin\beta$$

2p angles (parameters): $\vec{\gamma} = (\gamma_1, \gamma_2, \cdots, \gamma_p)$, and $\vec{\beta} = (\beta_1, \beta_2, \cdots, \beta_p)$

Goal is to find minimum/maximum over angles: $M_p = \max_{\vec{\gamma},\vec{\beta}} \langle \psi(\vec{\gamma},\vec{\beta}) | C | \psi(\vec{\gamma},\vec{\beta}) \rangle$

Quantum Approximate Optimization Algorithm (QAOA)

- How do $U(C,\gamma)$ and $U(B,\beta)$ operate on $|\psi\rangle$?

$$U(C,\gamma)\,H^{\otimes n}|0\,\cdots\,0\rangle = e^{-i\gamma C}\,H^{\otimes n}|0\,\cdots\,0\rangle = \exp\Big[-i\gamma\frac{1}{2}\sum_{(i,j)\in E}\Big(1-\sigma_i^Z\sigma_j^Z\Big)\Big]H^{\otimes n}|0\,\cdots\,0\rangle$$

$$= \prod_{(i,j)\in E}\exp\Big[-i\gamma\frac{1}{2}\Big(1-\sigma_i^Z\sigma_j^Z\Big)\Big]H^{\otimes n}|0\,\cdots\,0\rangle$$

$$\exp\Big[-i\gamma\frac{1}{2}\Big(1-\sigma_i^Z\sigma_j^Z\Big)\Big]H^{\otimes n}|0\,\cdots\,0\rangle = \exp\Big(-i\frac{\gamma}{2}\Big)\exp\Big(+i\frac{\gamma}{2}\sigma_i^Z\sigma_j^Z\Big)H^{\otimes n}|0\,\cdots\,0\rangle$$

Four different possibilities:

$$\exp\Big(+i\frac{\gamma}{2}\sigma_i^Z\sigma_j^Z\Big)|\cdots\overset{i}{0}\cdots\overset{j}{0}\cdots\rangle = \exp\Big(+i\frac{\gamma}{2}1\cdot 1\Big)|\cdots 0\cdots 0\cdots\rangle$$

$$\exp\Big(+i\frac{\gamma}{2}\sigma_i^Z\sigma_j^Z\Big)|\cdots 0\cdots 1\cdots\rangle = \exp\Big(-i\frac{\gamma}{2}1\cdot 1\Big)|\cdots 0\cdots 1\cdots\rangle$$

$$\exp\Big(+i\frac{\gamma}{2}\sigma_i^Z\sigma_j^Z\Big)|\cdots 1\cdots 0\cdots\rangle = \exp\Big(-i\frac{\gamma}{2}1\cdot 1\Big)|\cdots 1\cdots 0\cdots\rangle$$

$$\exp\Big(+i\frac{\gamma}{2}\sigma_i^Z\sigma_j^Z\Big)|\cdots 1\cdots 1\cdots\rangle = \exp\Big(+i\frac{\gamma}{2}1\cdot 1\Big)|\cdots 1\cdots 1\cdots\rangle$$

Quantum Approximate Optimization Algorithm (QAOA)

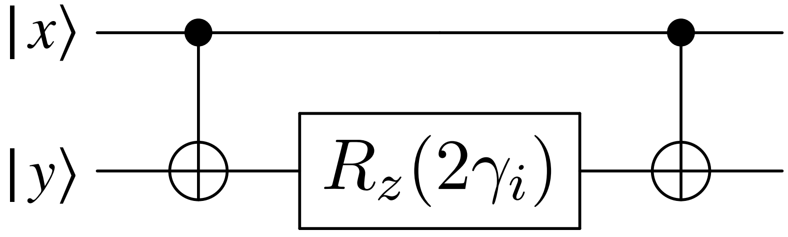
- If bits i and j are the same, $\exp\left[-i\gamma\frac{1}{2}\left(1-\sigma_i^Z\sigma_j^Z\right)\right]|\cdots\rangle = +1|\cdots\rangle$
- If bits i and j are different, $\exp\left[-i\gamma\frac{1}{2}\left(1-\sigma_i^Z\sigma_j^Z\right)\right]|\cdots\rangle = e^{-i\gamma}|\cdots\rangle$

- If bits i and j are different, rotate the output state around z axis by an angle γ.

$$\sigma_Z|a\rangle = (-1)^a|a\rangle \qquad R_Z(\theta) = \exp\left(-i\frac{\theta}{2}\sigma_Z\right) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{+i\theta/2} \end{pmatrix}$$

In circuit: $\mathrm{CNOT}\,|x\,y\rangle| = |x\,x\oplus y\rangle$

$$I\otimes R_z(2\gamma)\,|x\,x\oplus y\rangle = \exp\left(-i\gamma(-1)^{x\oplus y}\right)|x\,x\oplus y\rangle$$



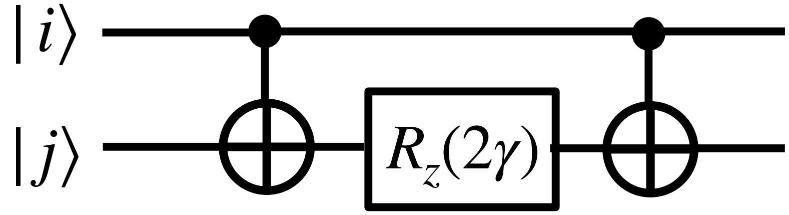
For $U(B,\beta) = e^{-i\beta B} = \prod_{j=1}^{n} e^{-i\beta\sigma_j^X} = \prod_{j=1}^{n} R_x^j(2\beta)$

- Rotation of all n-qubits about x-axis with angle 2β

Quantum Approximate Optimization Algorithm (QAOA)

$$|\psi(\vec{\gamma},\vec{\beta})\rangle = U(B,\beta_p)\,U(C,\gamma_p)\;\cdots\;U(B,\beta_1)\,U(C,\gamma_1)\frac{1}{\sqrt{2^n}}\sum_{i=1}^{2^n-1}|i\rangle$$

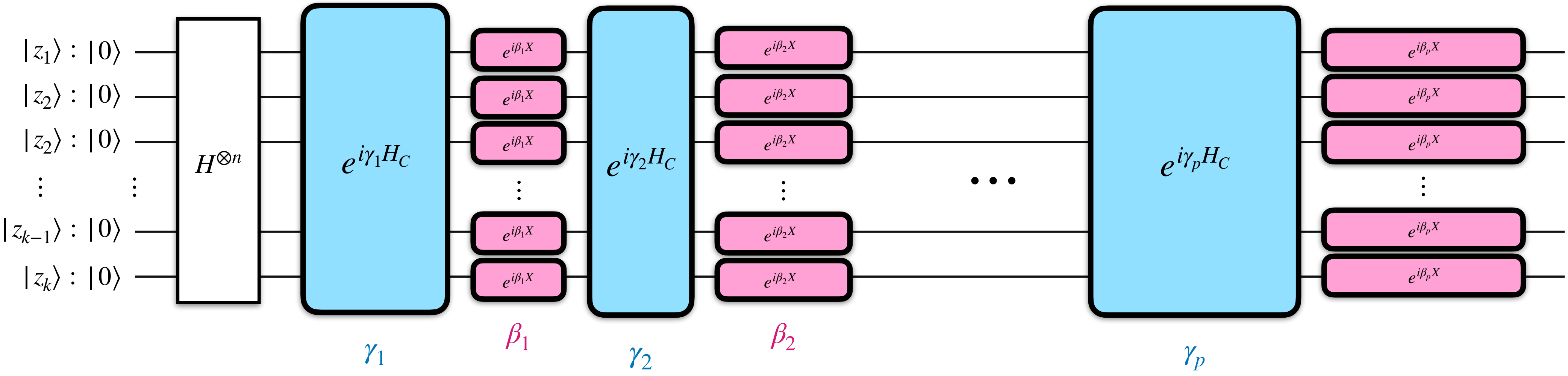
$$|\psi(\vec{\gamma},\vec{\beta})\rangle = \left[\prod_{i=1}^{p}U(B,\beta_i)U(C,\gamma_i)\right]H^{\otimes n}|0\rangle$$



$$e^{-i\beta\sigma_j^X} = \cos\beta - i\,\sin\beta\,\sigma_j^x = R_x^j(2\beta)$$

Rotate qubit j around x-axis by 2β

$$C(s) = \frac{1}{2}\sum_{(i,j)\in E}\left(1-\sigma_i^z\sigma_j^z\right),\quad B = \sum_j \sigma_j^X$$



Quantum Approximate Optimization Algorithm (QAOA)

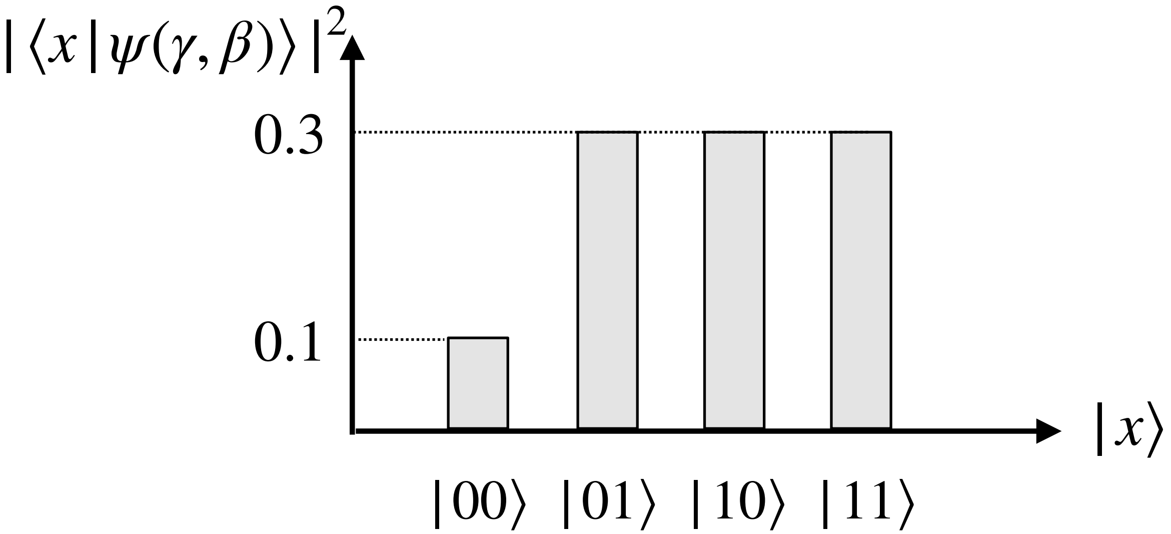
$$|\psi(\vec{\gamma},\vec{\beta})\rangle = U(B,\beta_p)\,U(C,\gamma_p)\,\cdots\,U(B,\beta_1)\,U(C,\gamma_1)\frac{1}{\sqrt{2}^n}\sum_{i=1}^{2^n-1}|i\rangle$$

For $n = 2$ and $p = 1$, $|\psi(\vec{\gamma},\vec{\beta})\rangle = \delta_0(\gamma,\beta)\,|00\rangle + \delta_1(\gamma,\beta)\,|01\rangle + \delta_2(\gamma,\beta)\,|10\rangle + \delta_3(\gamma,\beta)\,|11\rangle$

$$C = \frac{1}{2}\sum_{(i,j)\in E}\left(1 - \sigma_i^z\sigma_j^z\right)$$

$$C = \sum_{x\in\{0,1\}^{\otimes n}} C(x)\,|x\rangle\langle x|$$

$$|x\rangle = \frac{1}{\sqrt{2}^n}\sum_{i=1}^{2^n-1}|i\rangle$$



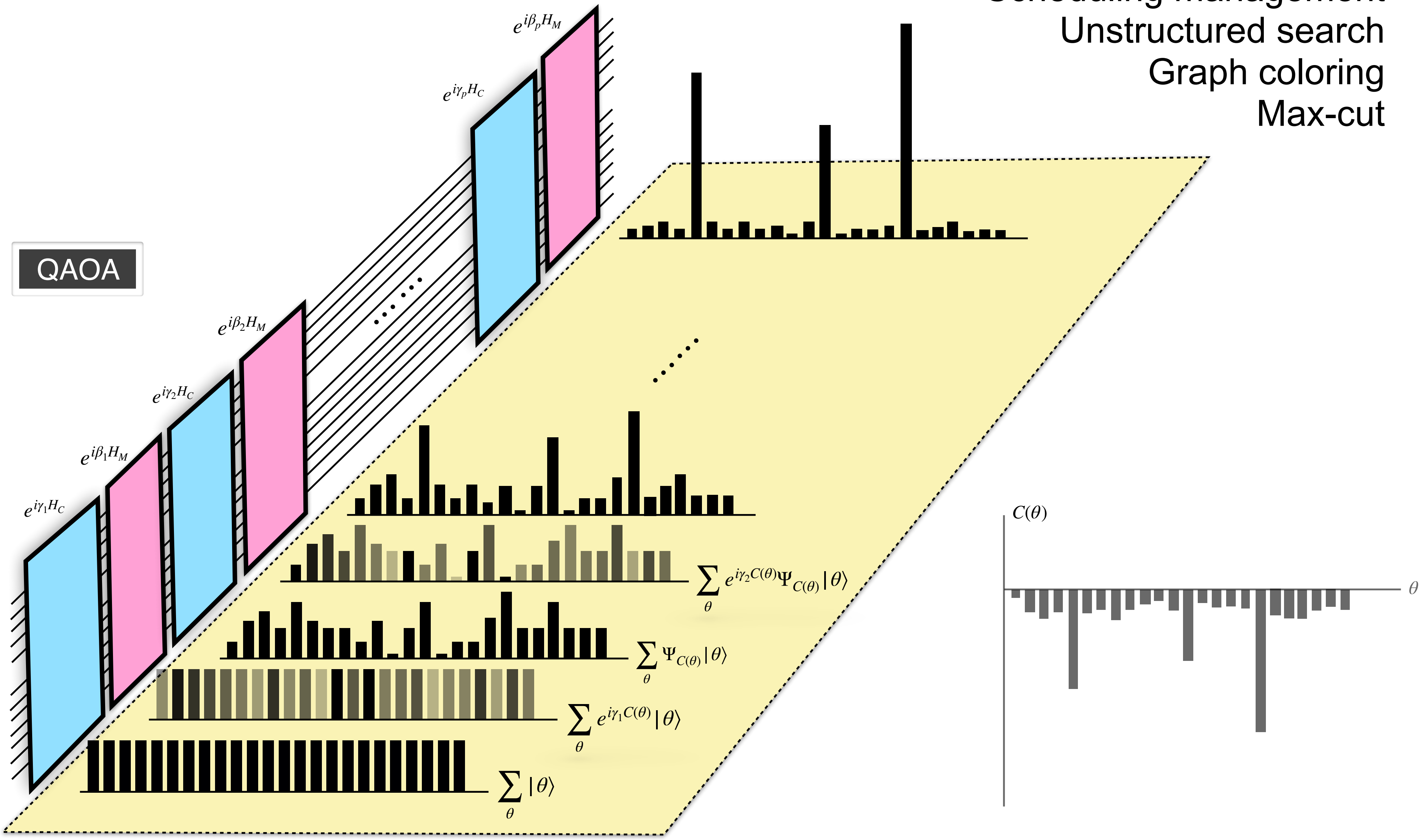
$$F_p(\gamma,\beta) = \max_{\gamma,\beta}\,\langle\psi(\gamma\,,\beta)\,|\,C\,|\,\psi(\gamma\,,\beta)\rangle = \sum_{x\in\{0,1\}^{\otimes n}} C(x)\left|\langle x|\psi(\gamma,\beta)\rangle\right|^2$$

- Measure of how good the approximation is to actual best value of the cost function

$$\alpha = \frac{F_p(\vec{\gamma},\vec{\beta})}{C_{max}}$$

QAOA

Maximum Likelihood detection
Traveling salesman problem
Scheduling management
Unstructured search
Graph coloring
Max-cut


QAOA
$e^{i\gamma_1 H_C}$
$e^{i\beta_1 H_M}$
$e^{i\gamma_2 H_C}$
$e^{i\beta_2 H_M}$
$e^{i\gamma_p H_C}$
$e^{i\beta_p H_M}$
$\sum_\theta e^{i\gamma_2 C(\theta)} \Psi_{C(\theta)} |\theta\rangle$
$\sum_\theta \Psi_{C(\theta)} |\theta\rangle$
$\sum_\theta e^{i\gamma_1 C(\theta)} |\theta\rangle$
$\sum_\theta |\theta\rangle$
$C(\theta)$
θ

Example: Max Cut

Ansatz	Main Idea	Enhancement & Applications
ma-QAOA [77] 2021	Multi-angle ansatz with a unique parameter for each element of cost and mixer Hamiltonians	Improves approximation ratio for MaxCut while reducing circuit depth
QAOA+ [78]	Augments traditional QAOA with an additional multi-parameter problem-independent layer	Higher approximation ratios for MaxCut on random regular graphs
DC-QAOA [79, 80]	Adds a problem-dependent counterdiabatic driving term to the QAOA ansatz	Improves the convergence rate of the approximation ratio while reducing circuit depth
ab-QAOA [81]	Incorporates local fields into the operators to reduce computation time	Computation time reduction for combinatorial optimization
ADAPT-QAOA [82] 2020	Iterative version of QAOA with systematic selection of mixers based on gradient criterion	Can be problem-specific and addresses hardware constraints
Recursive QAOA [83]	Non-local variant of QAOA that iteratively reduces problem size by eliminating qubits	Overcomes locality constraints and achieves better performance
QAOAnsatz [84]	Extends the original formulation with broader families of operators and allows for encoding of constraints	Adaptable to a wider range of optimization problems with hard and soft constraints
GM-QAOA [85]	Uses Grover-like selective phase shift mixing operators	Solves k-Vertex Cover, Traveling Salesperson Problem, Discrete Portfolio Rebalancing

2306.09198

Th-QAOA [86]	Replaces standard phase separator with a threshold function	Solves MaxCut, Max k-Vertex Cover, Max Bisection
Constraint Preserving Mixers [87]	Constructs mixers that enforce hard constraints	Solves optimization problems with hard constraints
WS-QAOA [88]	Modifies the initial state and mixer Hamiltonian based on the optimal solution to the relaxed QUBO problem	Solutions guaranteed to retain the GW bound for the MaxCut problem
FALQON [66] 2021	Uses qubit measurements for feedback-based quantum optimization, avoiding classical optimizers	Produces monotonically improving approximate solutions as circuit depth grows while bypassing classical optimization loops
FALQON+ [89]	Combines FALQON's initialization with QAOA for better parameter initialization	Improves initialization of standard QAOA for non-isomorphic graphs with 8 to 14 vertices
FQAOA [90]	Utilizes fermion particle number preservation to intrinsically impose constraints in QAOA process	Improves performance in portfolio optimization, applicable to Grover adaptive search and quantum phase estimation
Quantum Dropout [91]	Selectively drops out clauses defining the quantum circuit while keeping the cost function intact	Improves QAOA performance on hard cases of combinatorial optimization problems
ST-QAOA [92]	Uses an approximate classical solution to construct a problem instance-specific circuit	Achieves same performance guarantee as the classical algorithm, outperforms QAOA at low depths for MaxCut problem
Modified QAOA [31]	Modifies cost Hamiltonian with conditional rotations	Improves approximation ratio for MaxCut at $p = 1$

QAOA summary

- One can solve the optimization problems on a quantum computer by initializing the quantum device in the ground state of a hamiltonian that is easy to prepare and adiabatically tuning H into the problem Hamiltonian.
- In a digital quantum computer, this translates into a Trotterized version of the adiabatic evolution operator. In the limit of an infinite product, this Trotterized form becomes exact.
- QAOA is a hybrid quantum-classical variational algorithm with a finite order version of the evolution operator.
- Many experimental and theoretical studies, suggesting QAOA may provide a significant quantum advantage over classical algorithms, and that it is computationally universal.

Limitations and potential issues with QAOA

- The performance improves with the number of alternating layers in the Ansatz, which is limited by coherences times in exiting and near-term quantum processors.
- More layers implies more variational parameters (challenging for classical optimizers).
- Short-depth ansatz is not really the digitized version of the adiabatic problem but rather an adhoc ansatz, which does not guarantee to perform optimally.
- Fixed form of standard QAOA is not optimal but no systematic approach for finding a better ansatz.
- ADAPT-QAOA converges faster, reducing the required number of CNOT gates and optimization parameters.
- Connection to concept of shortcuts to adiabaticity.
- Inspired by ADAPT-VQE (Refs in 2005.10258).

Adaptive Derivative Assembled Problem Tailored - Quantum Approximate Optimization Algorithm (ADAPT-QAOA)

- https://arxiv.org/pdf/2005.10258.pdf

2005.10258

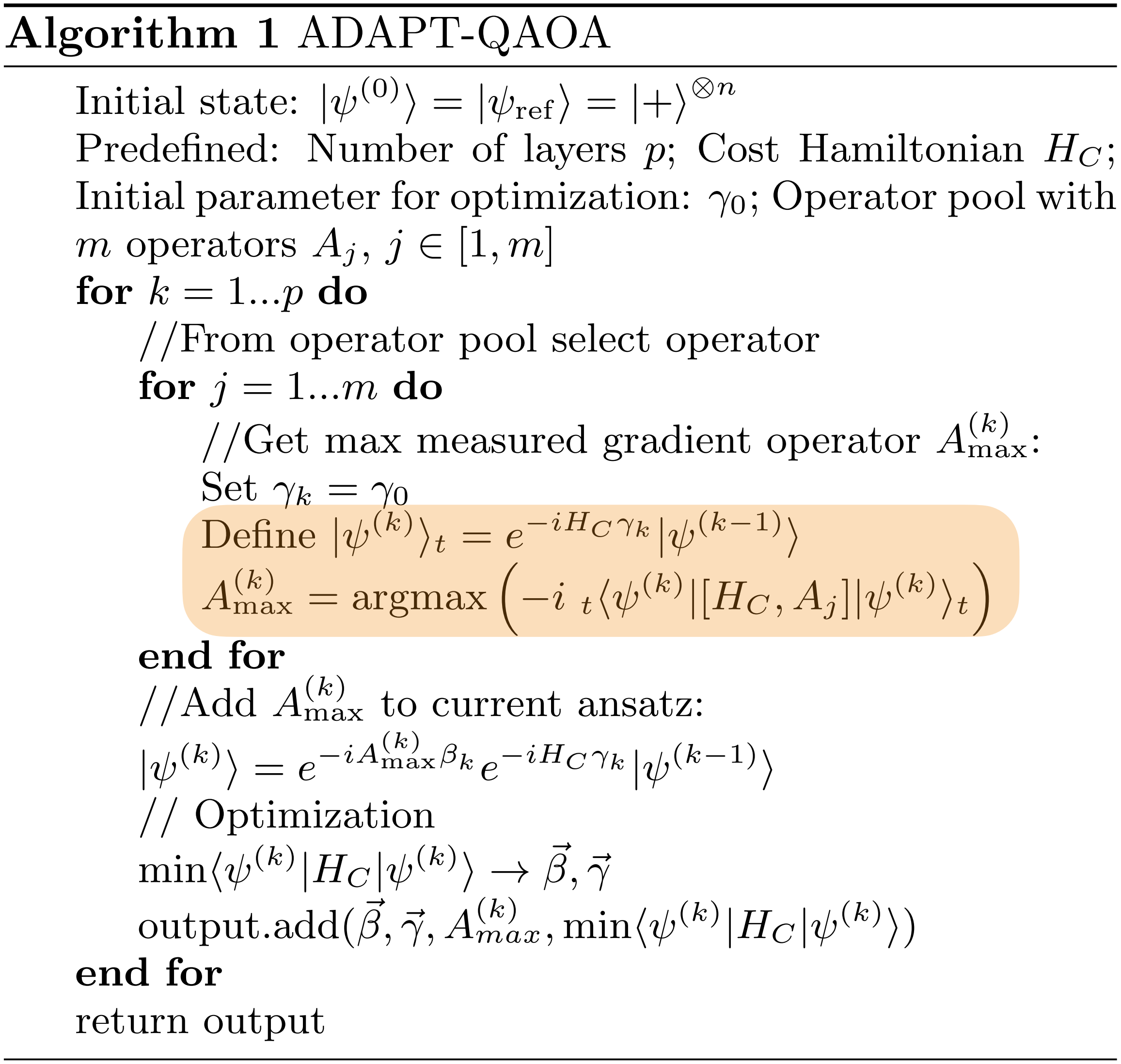
ADAPT-QAOA

$$\left|\psi_p(\vec{\gamma},\vec{\beta})\right\rangle = \left(\prod_{k=1}^{p}\left[e^{-iH_M\beta_k}e^{-iH_C\gamma_k}\right]\right)|\psi_{\text{ref}}\rangle \Rightarrow \left|\psi_p(\vec{\gamma},\vec{\beta})\right\rangle = \left(\prod_{k=1}^{p}\left[e^{-iA_k\beta_k}e^{-iH_C\gamma_k}\right]\right)|\psi_{\text{ref}}\rangle$$

$$\left|\psi^{(0)}\right\rangle = |\psi_{\text{ref}}\rangle = |+\rangle^{\otimes n}$$

n=number of qubits

mixer pool = set of A_j : $\{A_j\}$



```
Algorithm 1 ADAPT-QAOA
Initial state: $|\psi^{(0)}\rangle = |\psi_{\text{ref}}\rangle = |+\rangle^{\otimes n}$
Predefined: Number of layers $p$; Cost Hamiltonian $H_C$;
Initial parameter for optimization: $\gamma_0$; Operator pool with
$m$ operators $A_j$, $j \in [1, m]$
for $k = 1...p$ do
    //From operator pool select operator
    for $j = 1...m$ do
        //Get max measured gradient operator $A^{(k)}_{\max}$:
        Set $\gamma_k = \gamma_0$
        Define $|\psi^{(k)}\rangle_t = e^{-iH_C\gamma_k}|\psi^{(k-1)}\rangle$
        $A^{(k)}_{\max} = \text{argmax}\left(-i\ {}_t\langle\psi^{(k)}|[H_C, A_j]|\psi^{(k)}\rangle_t\right)$
    end for
    //Add $A^{(k)}_{\max}$ to current ansatz:
    $|\psi^{(k)}\rangle = e^{-iA^{(k)}_{\max}\beta_k}e^{-iH_C\gamma_k}|\psi^{(k-1)}\rangle$
    // Optimization
    $\min\langle\psi^{(k)}|H_C|\psi^{(k)}\rangle \rightarrow \vec{\beta}, \vec{\gamma}$
    output.add($\vec{\beta}, \vec{\gamma}, A^{(k)}_{max}, \min\langle\psi^{(k)}|H_C|\psi^{(k)}\rangle$)
end for
return output
```

2005.10258

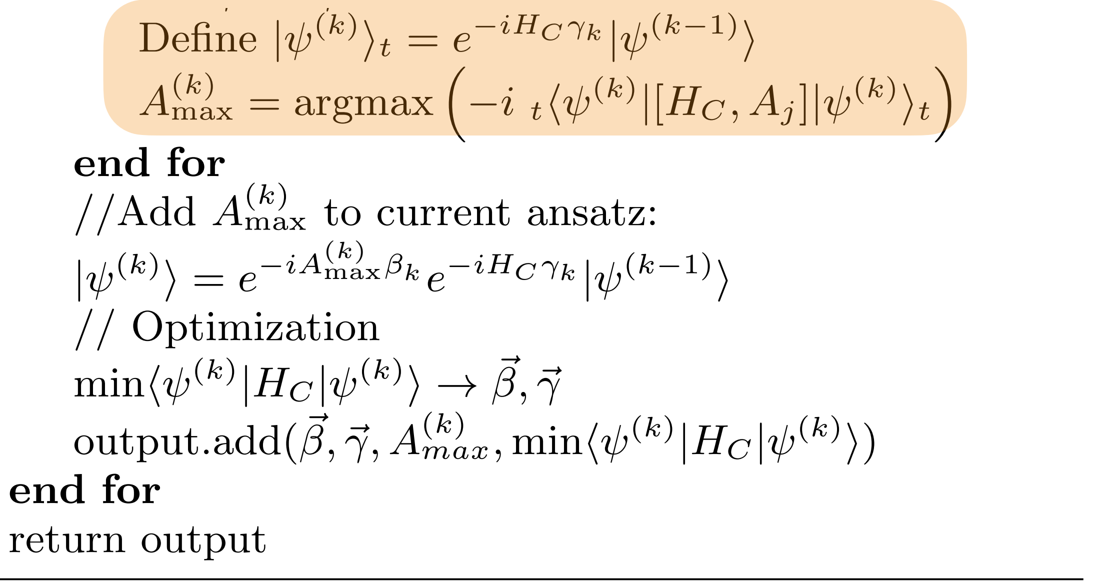
ADAPT-QAOA

$$\left|\psi_p(\vec{\gamma},\vec{\beta})\right\rangle = \left(\prod_{k=1}^{p}\left[e^{-iH_M\beta_k}e^{-iH_C\gamma_k}\right]\right)|\psi_{\text{ref}}\rangle \Rightarrow \left|\psi_p(\vec{\gamma},\vec{\beta})\right\rangle = \left(\prod_{k=1}^{p}\left[e^{-iA_k\beta_k}e^{-iH_C\gamma_k}\right]\right)|\psi_{\text{ref}}\rangle$$

$$\left|\psi^{(0)}\right\rangle = |\psi_{\text{ref}}\rangle = |+\rangle^{\otimes n}$$

n=number of qubits

mixer pool = set of A_j : $\{A_j\}$



```
        Define $|\psi^{(k)}\rangle_t = e^{-iH_C\gamma_k}|\psi^{(k-1)}\rangle$
        $A^{(k)}_{\text{max}} = \text{argmax}\left(-i\ {}_t\langle\psi^{(k)}|[H_C, A_j]|\psi^{(k)}\rangle_t\right)$
    end for
    //Add $A^{(k)}_{\text{max}}$ to current ansatz:
    $|\psi^{(k)}\rangle = e^{-iA^{(k)}_{\text{max}}\beta_k}e^{-iH_C\gamma_k}|\psi^{(k-1)}\rangle$
    // Optimization
    $\min\langle\psi^{(k)}|H_C|\psi^{(k)}\rangle \to \vec{\beta}, \vec{\gamma}$
    output.add($\vec{\beta}, \vec{\gamma}, A^{(k)}_{max}, \min\langle\psi^{(k)}|H_C|\psi^{(k)}\rangle$)
end for
return output
```

$$\Delta E_k^j \equiv \frac{\partial}{\partial\beta_k}\langle\psi_k|H_P|\psi_k\rangle\Big|_{\beta_k=0} = \frac{\partial}{\partial\beta_k}\langle\psi_{k-1}|e^{i\gamma_k H_P}e^{i\beta_k A_j}H_P e^{-i\beta_k A_j}e^{-i\gamma_k H_P}|\psi_{k-1}\rangle\Big|_{\beta_k=0}$$

$$= \langle\psi_{k-1}|e^{i\gamma_k H_P}e^{i\beta_k A_j}\left(iA_jH_P - iH_PA_j\right)e^{-i\beta_k A_j}e^{-i\gamma_k H_P}|\psi_{k-1}\rangle\Big|_{\beta_k=0}$$

$$H_P = H_C$$

$$= -i\,\langle\psi_{k-1}|e^{i\gamma_k H_P}[H_P, A_j]e^{-i\gamma_k H_P}|\psi_{k-1}\rangle$$

2005.10258

ADAPT-QAOA

$$\left|\psi_p(\vec{\gamma},\vec{\beta})\right\rangle = \left(\prod_{k=1}^{p}\left[e^{-iA_k\beta_k}e^{-iH_C\gamma_k}\right]\right)|\psi_{\mathrm{ref}}\rangle$$

$$\left|\psi^{(0)}\right\rangle = |\psi_{\mathrm{ref}}\rangle = |+\rangle^{\otimes n}$$

mixer pool = set of A_j : $\{A_j\}$

$\mathcal{O}(1)$ elements $\quad P_{\mathrm{QAOA}} = \left\{\sum_{i\in Q} X_i\right\}$

$\mathcal{O}(n)$ elements $\quad P_{\mathrm{single}} = \cup_{i\in Q}\{X_i, Y_i\} \cup \left\{\sum_{i\in Q} Y_i\right\} \cup P_{\mathrm{QAOA}}.$

$\mathcal{O}(n^2)$ elements $\quad P_{\mathrm{multi}} = \cup_{i,j\in Q\times Q}\{B_iC_j | B_i, C_j \in \{X, Y, Z\}\} \cup P_{\mathrm{single}}$

$$P_{\mathrm{QAOA}} \subset P_{\mathrm{single}} \subset P_{\mathrm{multi}}$$

→ Best performance

$$H_C = -\frac{1}{2}\sum_{i,j} w_{i,j}(I - Z_iZ_j)$$

- $H_C = H_P$ has a Z_2 symmetry associated with the operator $F = \otimes_i X_i$. Since $[F, H_C] = 0$, one can show that the gradient is only nonzero for $[F, A_j] = 0$. The A_j that commutes with F are Pauli strings that have an even number of Y or Z operators.

ADAPT-QAOA

2005.10258

Zhu et al 2020

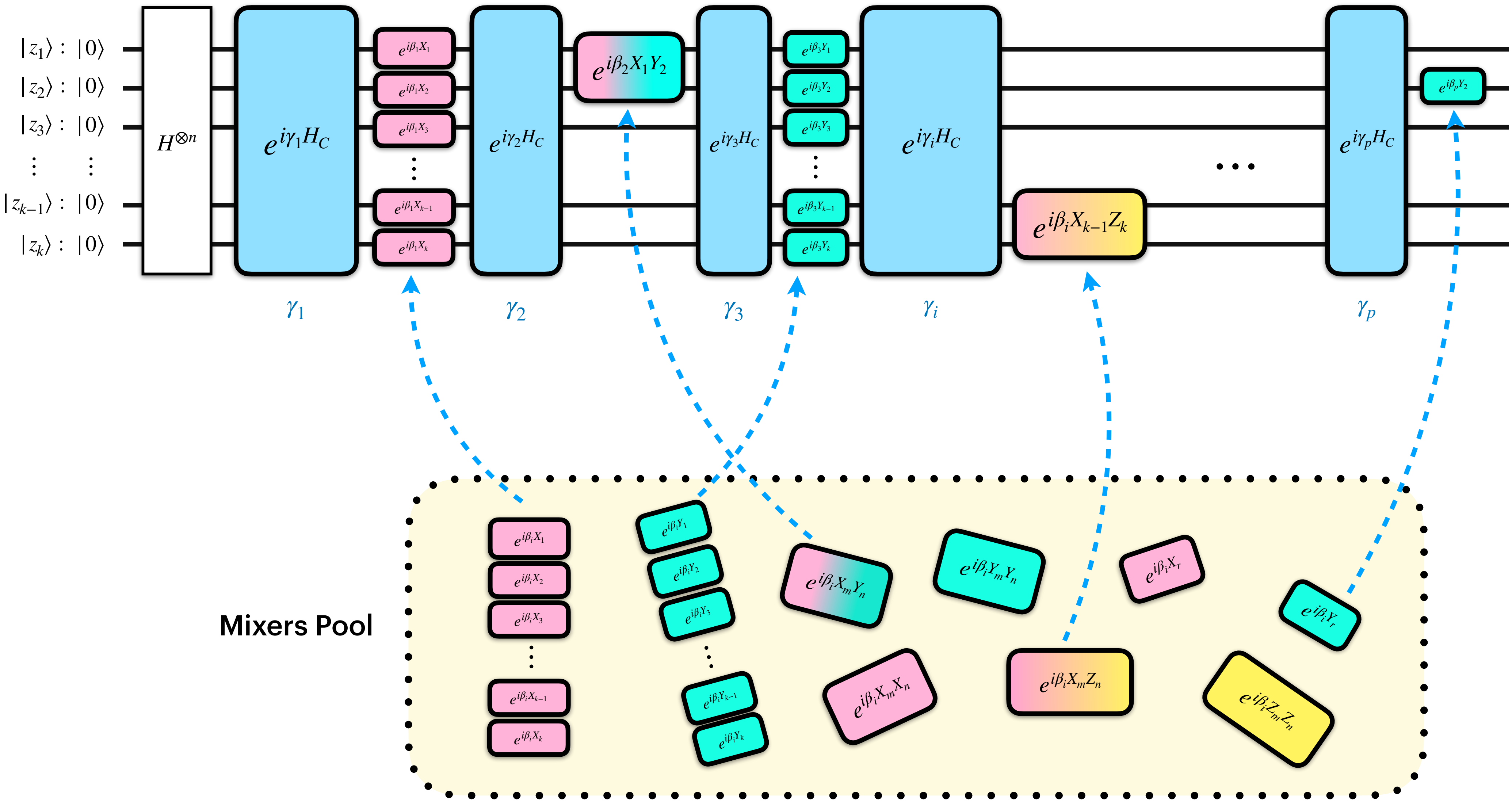


Figure taken from 2103.17047

2005.10258

ADAPT-QAOA applied to Mascot

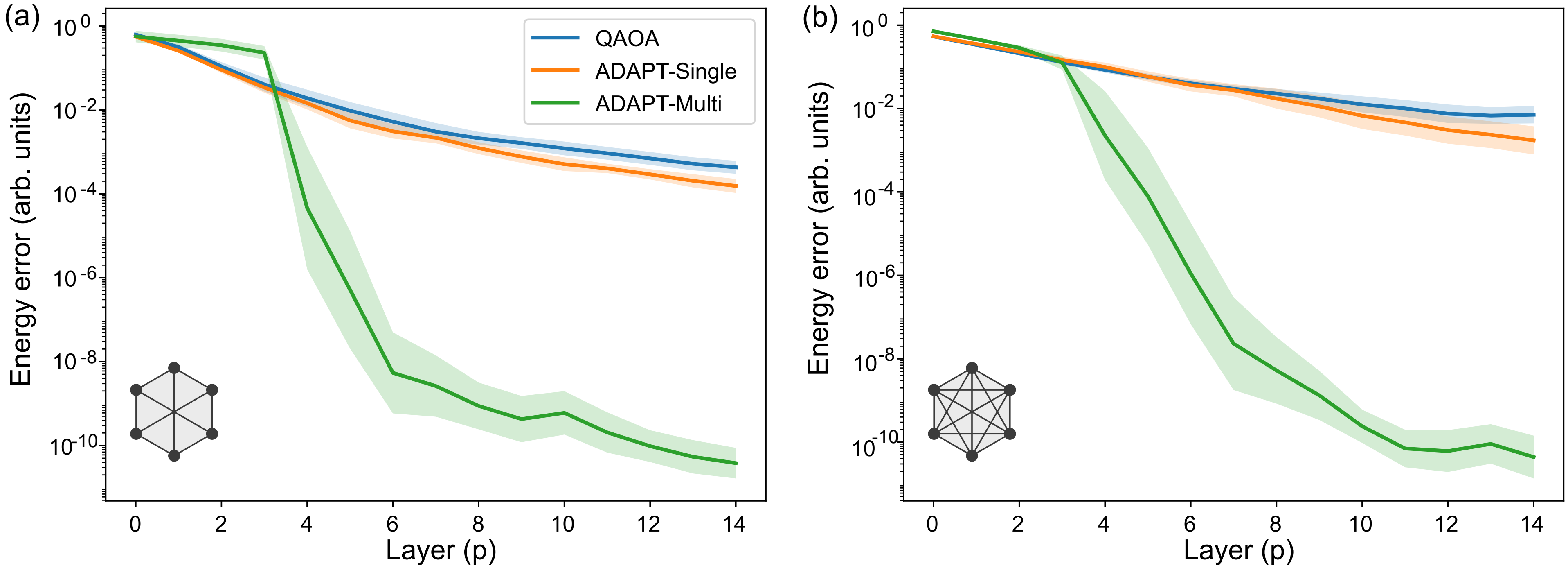


FIG. 1. Comparison of the performance of standard QAOA (blue) with ADAPT-QAOA for the single-qubit (orange) and multi-qubit (green) pools. The algorithms are run on the Max-Cut problem for the regular graphs shown in the figure, which have n=6 vertices and are of degree D=3 (a) and D=5 (b). The energy error (the difference between the energy estimate obtained by the algorithm and the exact ground state energy of H_C) is shown as a function of the number of layers in the ansatz. Results are shown for 20 different instances of edge weights, which are randomly sampled from the uniform distribution $U(0,1)$. The shaded regions indicate 95% confidence intervals.

Nelder-Mead for optimization
= downhill simplex method
= amoeba method
= polytope method

$\gamma_0 = 0.01$

- How much does the ADAPT-QAOA ansatz differ from the standard QAOA ansatz?
- When the single-qubit mixer pool is used, the single- qubit operators X_i are chosen instead of the standard mixer approximately 36.6% of the time for n=6,D=3 graphs and 25% of the time for n=6,D=5 graphs.
- For the multi-qubit mixer pool, the algorithm chooses operators other than the standard mixer approximately 75% of the time for n=6, D=3 graphs and 80% of the time for n=6, D=5 graphs.
- This trend supports the intuitive idea that a more connected graph requires more entanglement for a rapid convergence to the solution.

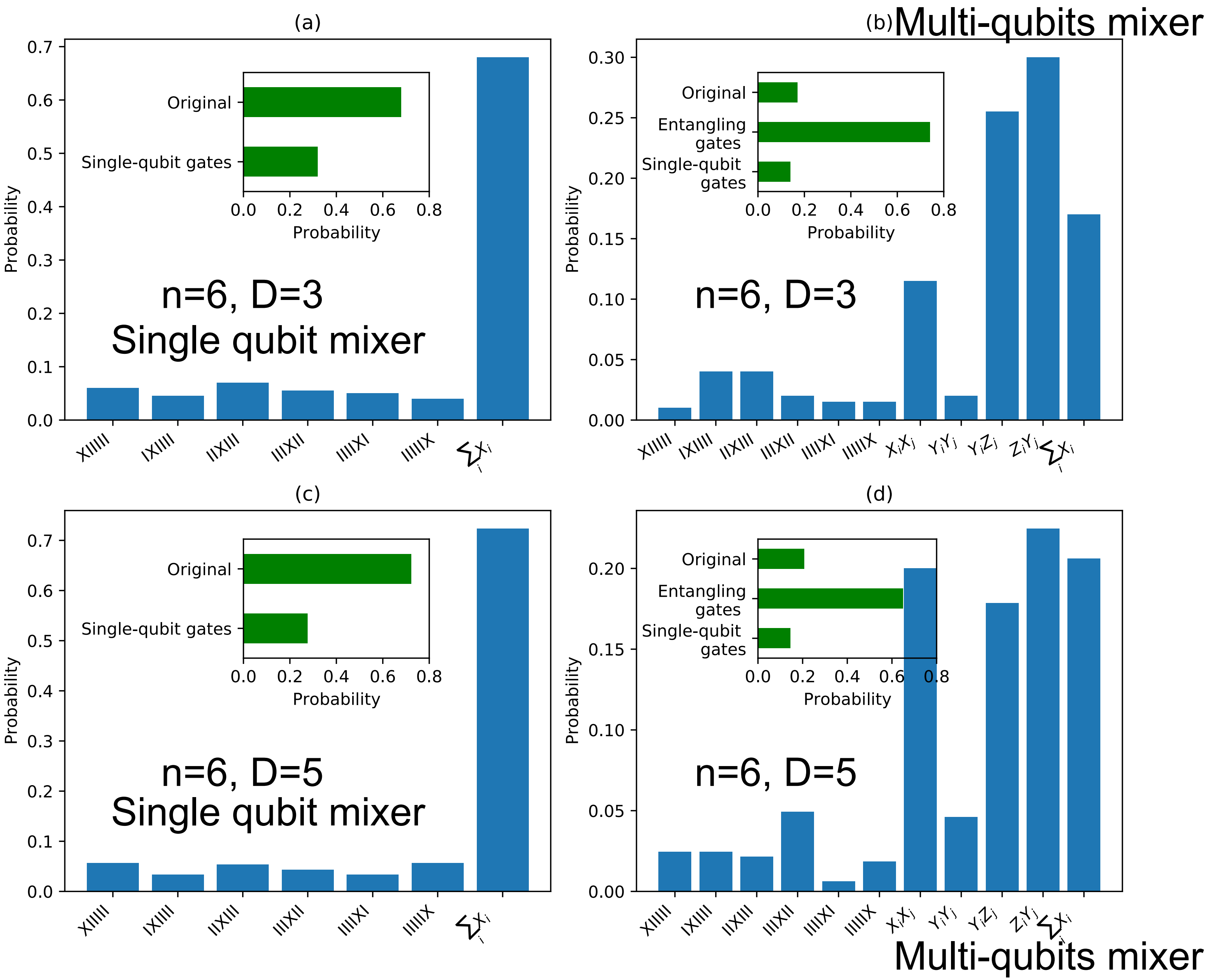
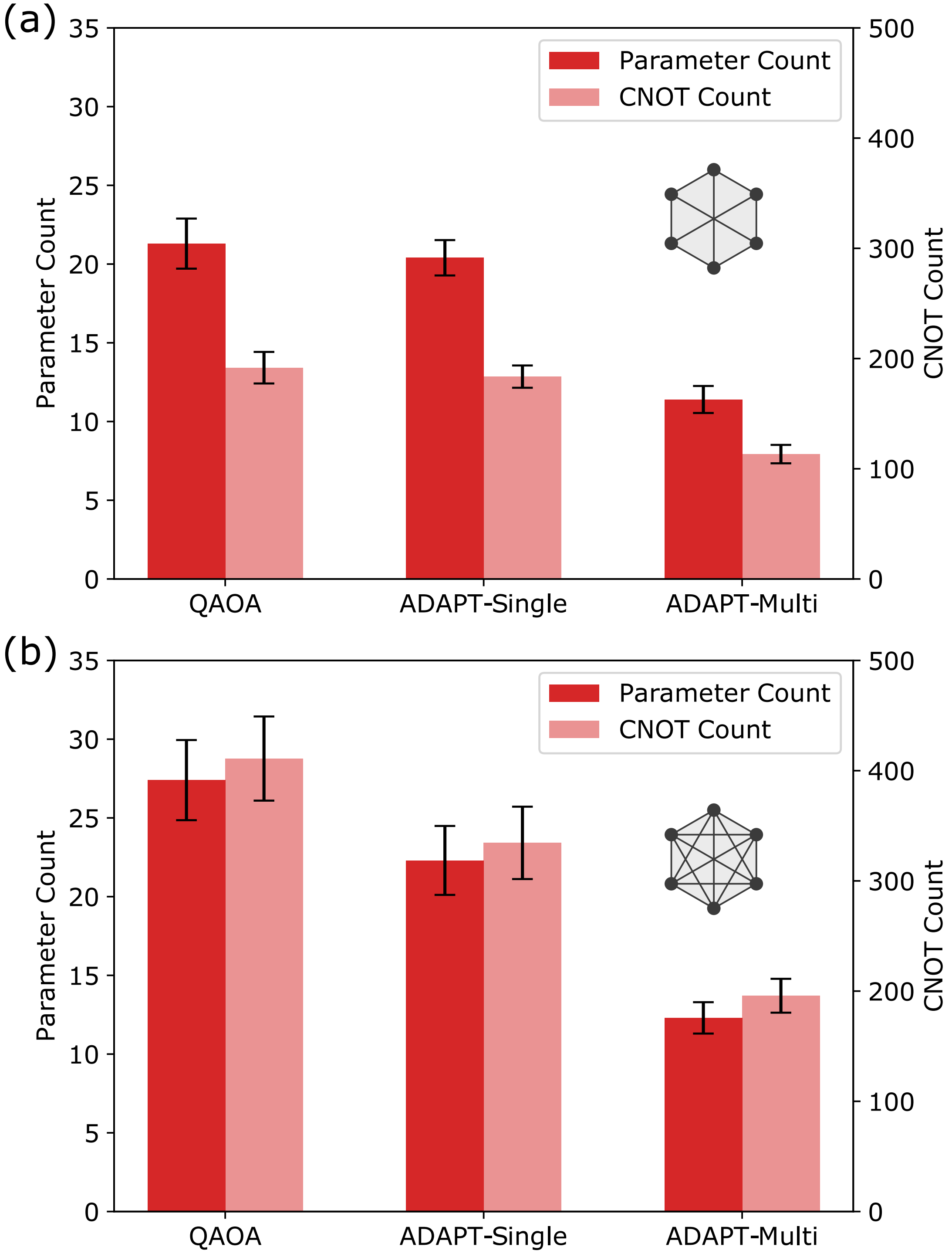


FIG. 6. Probability of operators picked by the original QAOA, ADAPT-QAOA with the single-qubit mixer and ADAPT-QAOA with multi-qubit pool for the Max-Cut problem on regular graphs with n=6 vertices with degree D=3 (a)(b) and D=5 (c)(d) with random edge weights sampled from a uniform distribution $U(0,1)$. The blue bars show the probability of each particular operator used for ansatz, and green bars show the probability of the original mixer, sum over all single-qubit gates and sum over all entangling gates used in ansatz. The results from 20 instances of random edge weights.

2005.10258

2005.10258

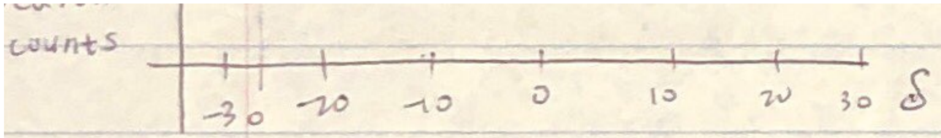


- ADAPT-QAOA provides a systematic way to both improve performance and reduce the number of parameters and CNOTs.

FIG. 2. Resource comparison of the standard QAOA, ADAPT-QAOA with the single-qubit mixer pool, and ADAPT-QAOA with the multi-qubit mixer pool for the Max-Cut problem on regular graphs with n=6 vertices and random edge weights. Panels (a) and (b) show the comparison for graphs of degree D=3 and D=5, respectively. For all cases except the standard QAOA applied to $D = 5$ graphs, we count the number of parameters and CNOTs needed to reach an energy error of $\delta E = 10^{-3}$. As standard QAOA for $D = 5$ graphs never reaches this error threshold, we instead count the CNOT gates and parameters at the end of the simulation (15 layers). The dark (light) red bars show variational parameter (CNOT gate) counts. The error bars show variances obtained by sampling over 20 different instances of edge weights.

Why ADAPT-QAOA performs better?

- Considering that the standard QAOA ansatz has a structure dictated by the adiabatic theorem, a possible explanation is related to Shortcuts to adiabaticity (STA).
- STA (counter-diabatic or transitionless driving) was introduced by Demirplak and Rice and later, independently, by Berry.
- If we want to drive a system such that it remains in the instantaneous ground state at all times, then by adding a certain term H_{CD} to the Hamiltonian, we can achieve this without paying the price of a slow evolution.
- Although the instantaneous eigenstates of the original Hamiltonian only solve the time-dependent Schrodinger equation in the adiabatic limit, they become exact solutions when the Hamiltonian is updated to include H_{CD}.
- The advantage of STA is that the evolution can be achieved non-adiabatically.



$\nabla\cdot\vec E = \rho$ Gauss's law

$\nabla\cdot\vec B = 0$ "

$\nabla\times\vec E + \frac{\partial\vec B}{\partial t} = 0$ Faraday's law

$\nabla\times\vec B - \frac{\partial\vec E}{\partial t} = \vec J$ Ampere-Maxwell's law

Gauge transformation in electromagnetism.

* consider time-independent scalar and vector potential $\phi(\vec x)$, and $\vec A(\vec x)$

$$\vec E = -\nabla\phi + \frac{1}{c}\frac{\partial\vec A}{\partial t} \qquad \vec B = \nabla\times\vec A$$

$$\vec E = -\nabla\phi$$

$(e<0)$

"minimal subtraction"

$$\begin{tikzcd} H=\frac{\vec p^{2}}{2m} \arrow[r] & H=\frac{1}{2m}\left(\vec p-\frac{e}{c}\vec A\right)^{2}+e\phi \\ L=\frac{1}{2}mv^{2} \arrow[u, "\text{Legendre tr}"] \arrow[r] & L=\frac{1}{2}m\vec v^{2}-q\phi+\frac{e}{c}\vec v\cdot\vec A \arrow[u, "\text{Legendre tr}"] \end{tikzcd}$$

$\vec p \rightarrow \vec p - \frac{e}{c}\vec A$

$H \rightarrow H - e\phi$

how to include EM interacti

* Gauge transformation : $\vec E = -\nabla\phi - \frac{1}{c}\frac{\partial\vec A}{\partial t}$, $\vec B = \nabla\times\vec A$ are invariant

under $\phi \rightarrow \phi - \frac{1}{c}\frac{\partial\Lambda(\vec x,t)}{\partial t}$ $\qquad A^{\mu} = (\phi, \vec A)$

$\vec A \rightarrow \vec A + \nabla\underset{\text{gauge function}}{\underset{\downarrow}{\Lambda(\vec x,t)}}$ $\qquad A^{\mu} \rightarrow A^{\mu} - \partial^{\mu}\Lambda$

$$\boxed{\frac{\partial \rho}{\partial t} + \nabla \cdot \vec{J} = 0},$$

$$\boxed{\begin{aligned} &\rho = |\psi|^2, \quad \vec{J} = \frac{\hbar}{2mi}\left(\psi^*\nabla\psi - \psi\nabla\psi^*\right) - \frac{e}{mc}\vec{A}\psi^*\psi \\ &\qquad\qquad\qquad\;\; = \left(\frac{\hbar}{m}\right) \mathrm{Im}\left(\psi^*\nabla\psi\right) - \left(\frac{e}{mc}\right)\vec{A}\psi^*\psi \\ &\vec{p} = -i\hbar\nabla \\ &\vec{p} \longrightarrow \vec{p} - \frac{e}{c}\vec{A} \qquad \nabla \longrightarrow \nabla - \left(\frac{ie}{\hbar c}\right)\vec{A} \end{aligned}}$$

Shortcuts to Adiabaticity (transitionless driving protocols)

$$i\,\partial_t\,|\psi\rangle = H(\theta(t))\,|\psi\rangle \qquad \Longrightarrow \qquad i\,\partial_t\,|\tilde{\psi}\rangle = (\tilde{H} - \dot{\theta}\tilde{A}_\theta)\,|\tilde{\psi}\rangle$$

$$\begin{array}{rcl} |\psi\rangle & \longrightarrow & |\tilde{\psi}\rangle = U^\dagger\,|\psi\rangle \\ H & \longrightarrow & \tilde{H} = U^\dagger\,H\,U \\ i\partial_t & \longrightarrow & i\partial_t - \dot{\theta}\tilde{A}_\theta \end{array} \qquad \begin{array}{l} H_{CD} = \dot{\theta}A_\theta \\ \tilde{A}_\theta = iU^\dagger\partial_\theta\,U \\ A_\theta = U\,\tilde{A}_\theta\,U^\dagger \end{array}$$

- Suppose that we consider a unitary transformation $U(\theta(t))$ to move the Hamiltonian $H(\theta(t))$ from the initial basis to its instantaneous eigenbasis, where $\tilde{H}(\theta) = U^\dagger(\theta)\,H(\theta)\,U(\theta)$ is diagonal at all times.
- The Schrodinger equation in the instantaneous eigenbasis is $i\,\partial_t\,|\tilde{\psi}\rangle = (\tilde{H} - \dot{\theta}\tilde{A}_\theta)\,|\tilde{\psi}\rangle$, where $\tilde{A}_\theta = iU^\dagger\,A_\theta\,U$ is the adiabatic gauge potential in the rotated frame. It is evident that the term $-\dot{\theta}\tilde{A}_\theta$ drives transitions between the energy levels of the original Hamiltonian H. Therefore, one can add the counterdiabatic term $H_{CD} = \dot{\theta}A_\theta$ to $H(\theta)$, with $A_\theta = U\,\tilde{A}_\theta\,U^\dagger$, to eliminate such transitions in the rotated frame. This is the core of transitionless driving protocols.
- Ref. [40] proposes an approximate gauge potential:

$$\mathcal{A}_\theta^{(p)} = i\sum_{k=1}^{p} a_k[\mathcal{H}, \partial_\theta\mathcal{H}]_{2k-1} \qquad\qquad [X,Y]_{k+1} \;=\; [X,[X,Y]]_k$$

Connection between ADAPT-QAOA and STA

- Apply the above formalism using the Hamiltonian $H = \frac{t}{T} H_C + \left(1 - \frac{t}{T}\right) \sum_{i=1}^{n} X_i$, and set $\theta = t$. T is the duration of the evolution from the initial state $|\psi_{\text{ref}}\rangle = |+\rangle^{\otimes n}$ to the ground state of the cost Hamiltonian H_C.
- In all cases, the mixer operator at the first layer is also an element of the set $\mathcal{O}_{CD}$.
- Going to higher order in the H_{CD} approximation increases the probability of finding the mixers in the set $\mathcal{O}_{CD}$.
- ADAPT-QAOA finds the appropriate rotation axes in Hilbert space for faster convergence to the solution, and that these axes may in some sense be universal across all possible choices of H(t) that interpolate between the initial and target states. This suggests that STA can be used as a tool to construct operator pools for ADAPT-QAOA.

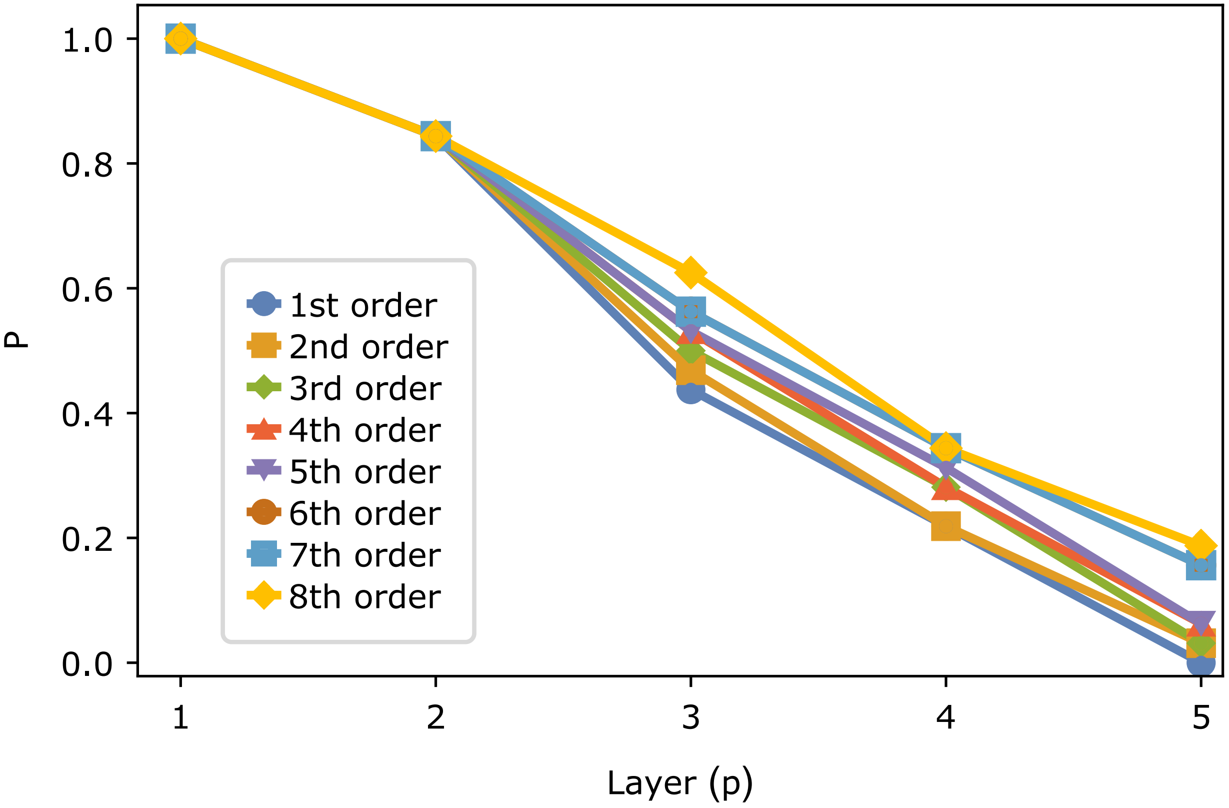


FIG. 3. Probability P of the operator at layer p of the ADAPT-QAOA ansatz to be among the Pauli strings with the largest coefficient in $\mathcal{H}_{CD}$ averaged over 32 graphs with $n = 6$, $D = 3$. The different curves correspond to different orders of the approximation.

Adaptive Derivative Assembled Problem Tailored QAOA (ADAPT-QAOA)

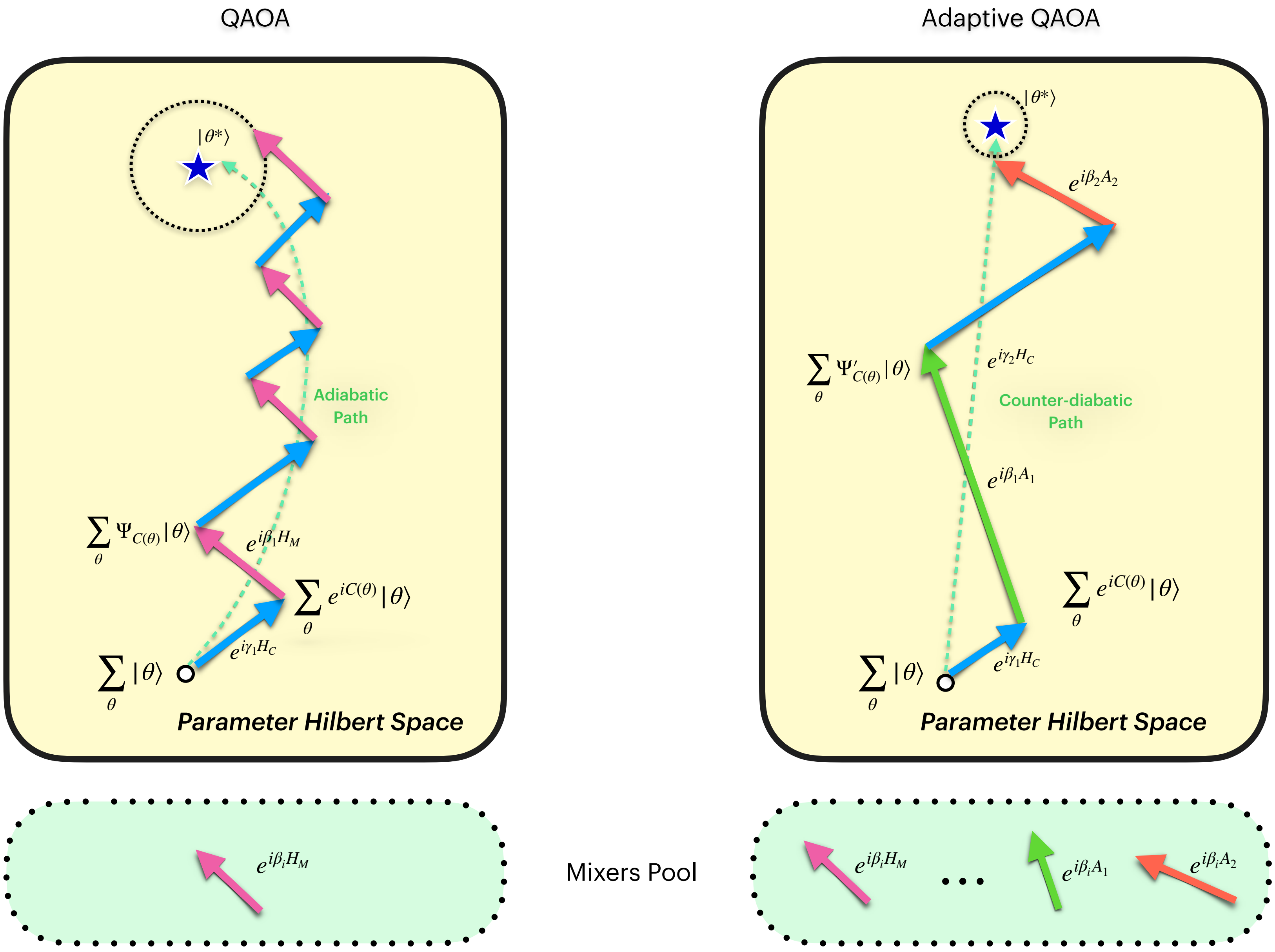


Figure taken from 2103.17047

Questions?

- Paper contains an interesting discussion on how to exploit non-adiabatic path.
- Paper contains evidence that ADAPT-QAOA is related to STA but not rigorous proof.
- Paper uses mixers with two Pauli matrices. What about 4, 6 or more?
- Non-Abelian shortcuts to adiabaticity on quantum simulation?
- There is an example code implemented in TensorFlowQuantum.

Quantum Neural Networks

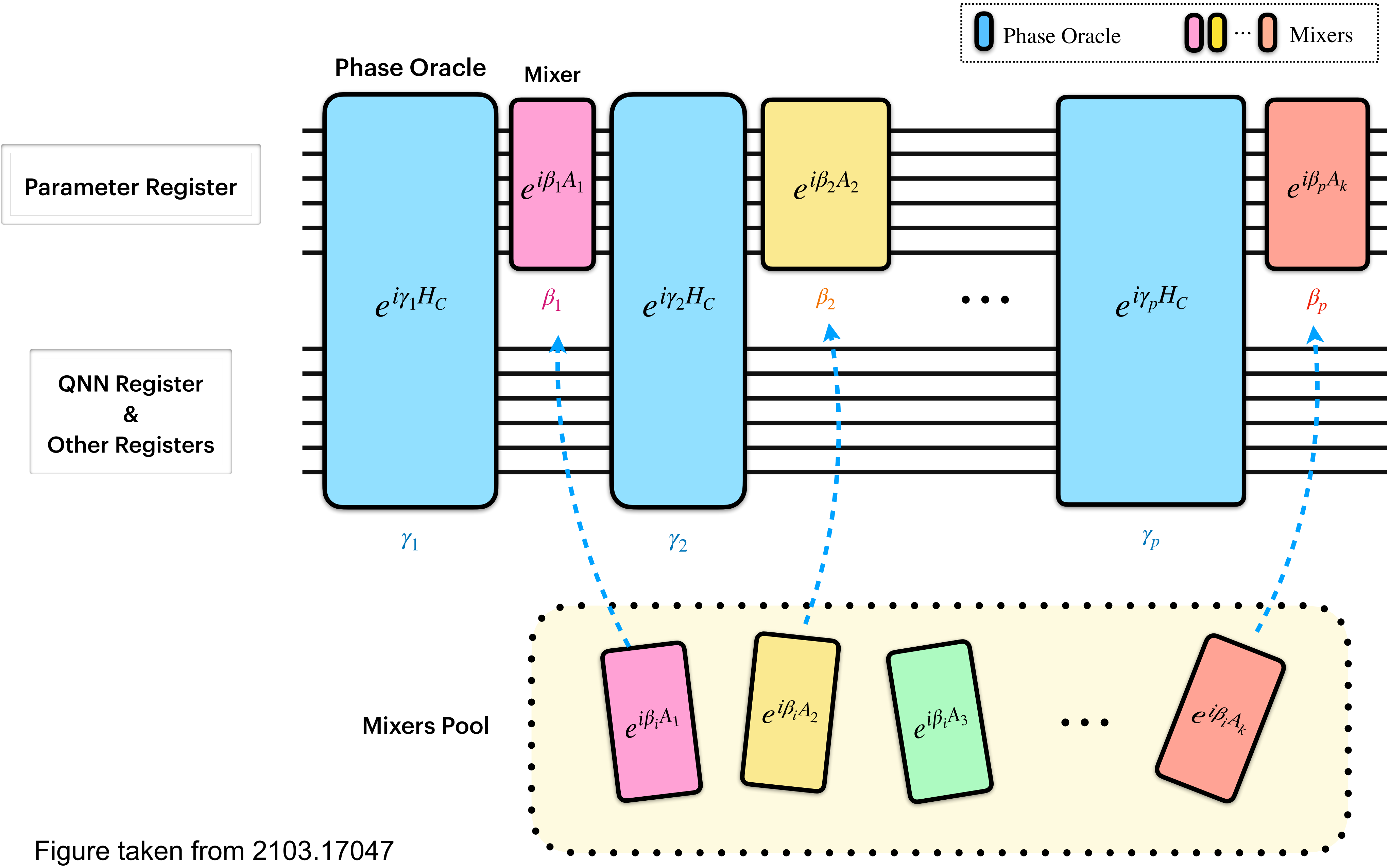
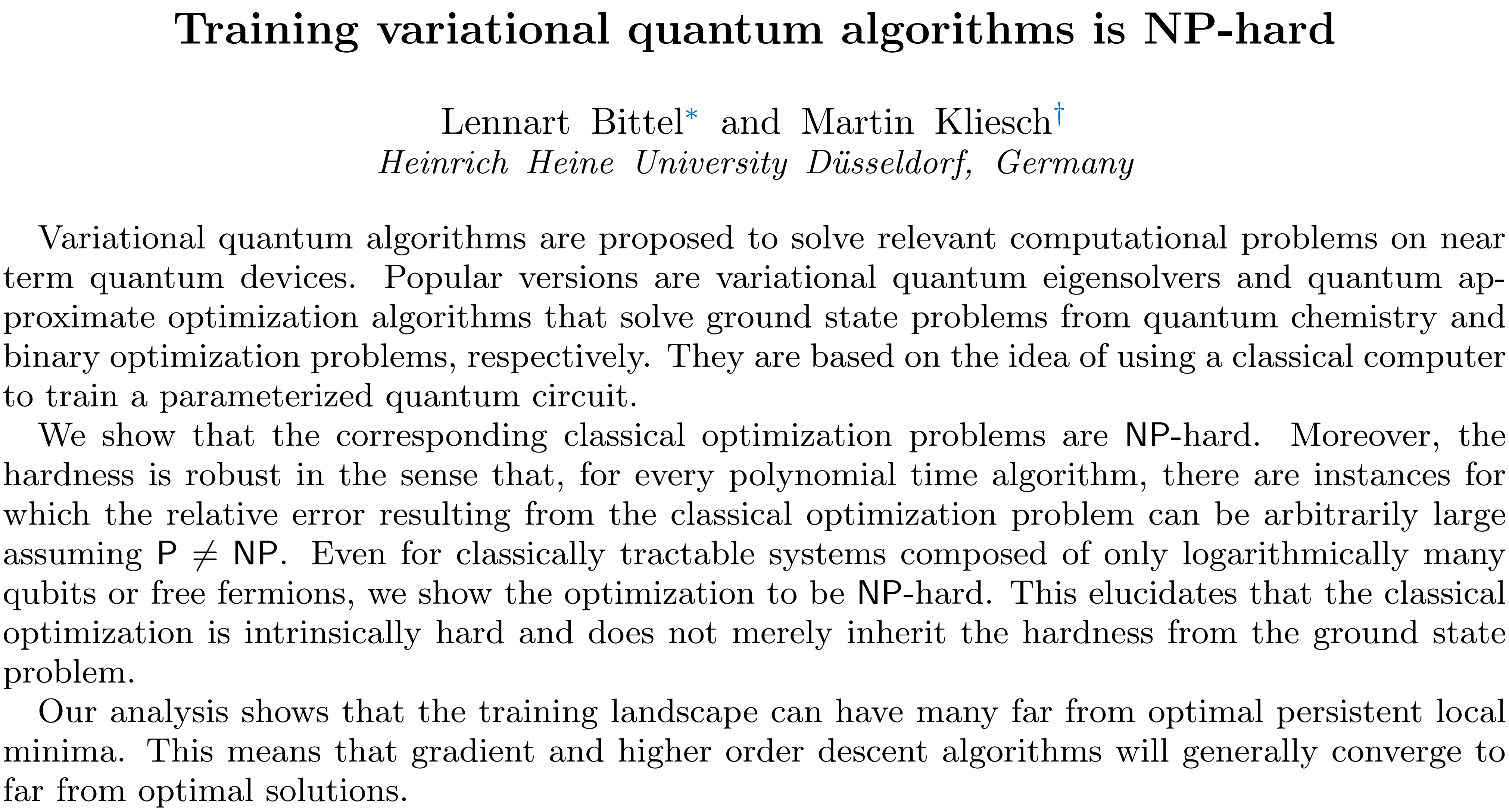


Figure taken from 2103.17047

- Training variational quantum algorithms is NP-hard
- https://arxiv.org/pdf/2101.07267.pdf



Training variational quantum algorithms is NP-hard

Lennart Bittel[*] and Martin Kliesch[†]
Heinrich Heine University Düsseldorf, Germany

Variational quantum algorithms are proposed to solve relevant computational problems on near term quantum devices. Popular versions are variational quantum eigensolvers and quantum approximate optimization algorithms that solve ground state problems from quantum chemistry and binary optimization problems, respectively. They are based on the idea of using a classical computer to train a parameterized quantum circuit.

We show that the corresponding classical optimization problems are NP-hard. Moreover, the hardness is robust in the sense that, for every polynomial time algorithm, there are instances for which the relative error resulting from the classical optimization problem can be arbitrarily large assuming $\mathsf{P} \neq \mathsf{NP}$. Even for classically tractable systems composed of only logarithmically many qubits or free fermions, we show the optimization to be NP-hard. This elucidates that the classical optimization is intrinsically hard and does not merely inherit the hardness from the ground state problem.

Our analysis shows that the training landscape can have many far from optimal persistent local minima. This means that gradient and higher order descent algorithms will generally converge to far from optimal solutions.

Feedback-based ALgorithm Quantum Optimization (FALQON)

Feedback-based ALgorithm Quantum Optimization (FALQON)

- 2103.08619, https://pennylane.ai/qml/demos/tutorial_falqon.html
- Consider a quantum system whose dynamics is governed by

$$i\frac{d}{dt}|\psi(t)\rangle = (H_{\rm p} + H_{\rm d}\beta(t))|\psi(t)\rangle$$

- Goal is to minimize: $\langle H_{\rm p}\rangle = \langle\psi(t)|H_{\rm p}|\psi(t)\rangle$

H_p : drift Hamiltonian (Problem H) $\beta(t)$: time – dependent control function

H_d : control Hamiltonian

- One can minimize $\langle H_p\rangle$ by designing $\beta(t)$ such that

$$\frac{d}{dt}\langle\psi(t)|H_{\rm p}|\psi(t)\rangle(t) \leq 0, \quad \forall t \geq 0$$

$$\frac{d}{dt}\langle\psi(t)\,|\,H_p\,|\,\psi(t)\rangle = i\langle\psi(t)\,|\,(H_p + H_d\beta(t))H_p\,|\,\psi(t)\rangle - i\langle\psi(t)\,|\,H_p(H_p + H_d\beta(t))\,|\,\psi(t)\rangle$$

$$= \langle\psi(t)\,|\,i\,[H_d, H_p]\,|\,\psi(t)\rangle\,\beta(t) \equiv A(t)\,\beta(t)$$

Feedback-based ALgorithm Quantum Optimization (FALQON)

$$\frac{d}{dt}\langle\psi(t)\,|\,H_p\,|\,\psi(t)\rangle = \langle\psi(t)\,|\,i\,[H_d, H_p]\,|\,\psi(t)\rangle\,\beta(t) \equiv A(t)\,\beta(t)$$

- We can choose any $\beta(t)$.
- Consider $\beta(t) = -\,w\,f(t, A(t))$ for $w > 0$, where $f(t, A(t))$ is any continuous function with $f(t,0) = 0$ and $A(t)f(t, A(t)) > 0$ for all $A(t) \neq 0$.
- Take $w = 1$ and $f(t, A(t)) = A(t)$ such that $\beta(t) = -\,A(t)$ for simplicity.
- Consider alternating (rather than concurrent) applications of H_p and H_d, leading to a time evolution:

$$U = U_d\,(\beta_\ell)\,U_p\,\cdots\,U_d\,(\beta_1)\,U_p$$

$$U_p = e^{-iH_p\Delta t} \qquad k = 1, 2, \cdots, \ell \qquad \beta_k = \beta(k\tau - \Delta t)$$

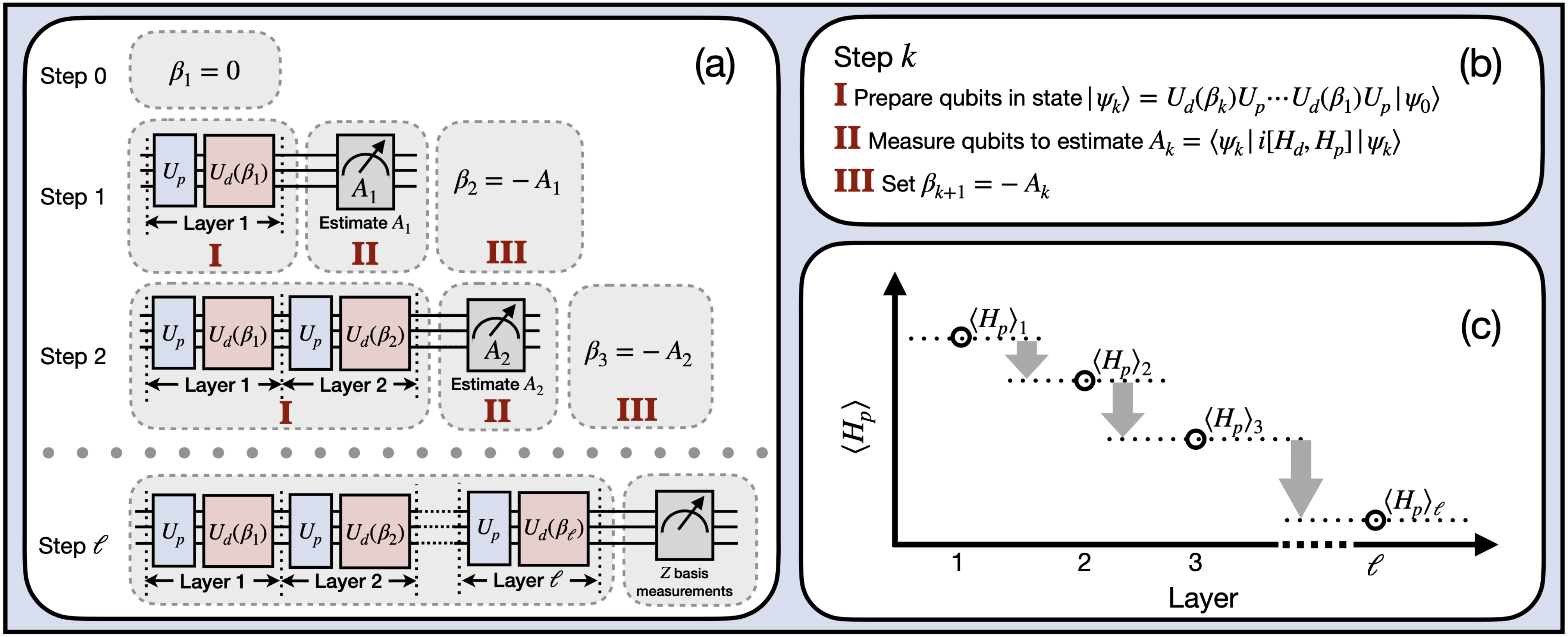
$$U_d\,(\beta_k) = e^{-i\beta_k H_d \Delta t} \qquad \tau = 2\Delta t \qquad = \beta((k-1)\Delta t)$$

- For small Δt, this unitary evolution yields Trotterized approximation to the continuous time evolution of the system.

Feedback-based ALgorithm Quantum Optimization (FALQON)

- During the time evolution when H_p is applied, $\frac{d}{dt}\langle H_p \rangle = 0$, but eigenstate of H_p accumulates phase changes. (H_p is time-independent.)
- For the time evolution when H_d is applied, we recover $\frac{d}{dt}\langle H_p \rangle = A(t)\beta(t)$
- Set $\beta_{k+1} = -A_k$, where $A_k = \langle \psi_k | \, i\,[H_d, H_p] \, | \psi_k \rangle$
- In this setting, it is always possible to choose Δt small enough such that $\frac{d}{dt}\langle \psi(t) | H_p | \psi(t) \rangle \ \leq 0$. If Δt is chosen to be too large, the inequality will be violated.
- FALQON is a constructive, optimization free procedure for assigning values to each β_k according to a feedback law.
- By design, the quality of the solution to the combinatorial optimization problem improves monotonically with respect to depth of the circuit, k.

Feedback-based ALgorithm Quantum Optimization (FALQON)



Step k

I Prepare qubits in state $|\psi_k\rangle = U_d(\beta_k)U_p \cdots U_d(\beta_1)U_p|\psi_0\rangle$

II Measure qubits to estimate $A_k = \langle \psi_k | i[H_d, H_p] | \psi_k \rangle$

III Set $\beta_{k+1} = -A_k$

Figure 1. (a) The procedure for implementing FALQON. The initial step is to seed the procedure by setting $\beta_1 = 0$. The qubits are then initialized in the state $|\psi_0\rangle$, and a single FALQON layer is implemented to prepare $|\psi_1\rangle = U_\mathrm{d}(\beta_1)U_\mathrm{p}|\psi_0\rangle$. The qubits are then measured to estimate A_1, whose result is fed back to set $\beta_2 = -A_1$, up to sampling error. For subsequent steps $k = 2, \cdots, \ell$, the same procedure is repeated, as shown in (b): the qubits are initialized as $|\psi_0\rangle$, after which k layers are applied to obtain $|\psi_k\rangle = U_\mathrm{d}(\beta_k)U_\mathrm{p} \cdots U_\mathrm{d}(\beta_1)U_\mathrm{p}|\psi_0\rangle$, and then the qubits are measured to estimate A_k, and the result is fed back to set the value of β_{k+1}. This procedure causes $\langle H_\mathrm{p} \rangle$ to decrease layer-by-layer as per $\langle \psi_1|H_\mathrm{p}|\psi_1\rangle \geq \langle \psi_2|H_\mathrm{p}|\psi_2\rangle \geq \cdots \geq \langle \psi_\ell|H_\mathrm{p}|\psi_\ell\rangle$, as shown in (c), such that the quality of the solution to the combinatorial optimization problem monotonically improves with circuit depth. The protocol can be terminated when the value of $\langle H_\mathrm{p} \rangle$ converges or a threshold number of layers ℓ is reached. Then, after the final step, Z basis measurements on $|\psi_\ell\rangle$ can be used to determine a best candidate solution to the combinatorial optimization problem of interest, by repeatedly sampling from the probability distribution over bit strings induced by $|\psi_\ell\rangle$ and selecting the outcome associated with the best solution.

FALQON vs QAOA

- Circuits used in QAOA has the same alternative structure as those in FALQON with additional parameters $\vec{\gamma} = (\gamma_1, \cdots, \gamma_\ell)$ that enter into U_p such that $U_{QAOA} = U_d(\beta_\ell)U_p(\gamma_\ell)\cdots U_d(\beta_1)U_p(\gamma_1)$.
- Solution to the original combinatorial optimization is found by minimizing $\langle \psi(\vec{\gamma}, \vec{\beta}) \,|\, H_p \,|\, \psi(\vec{\gamma}, \vec{\beta}) \rangle$ over 2ℓ parameters, using classical optimization. $(\,|\,\psi(\vec{\gamma}, \vec{\beta})\rangle = U_{QAOA}\,|\,\psi_0\rangle)$
- FALQON minimizes $\langle H_p \rangle$ over a sequence of quantum circuit layers, guided by qubit measurement-based feed back without classical optimization.

FALQON for MaxCut problem

- MaxCut: $H_p = -\sum_{(i,j)\in E} \frac{1}{2}\left(1 - Z_i Z_j\right)$ and $H_d = \sum_{j=1}^{n} X_j$
- $i\,[H_d, H_p] = \sum_{(i,j)\in E} Y_i Z_j + Z_i Y_j$ where X_j, Y_j and Z_j are Pauli's matrices.

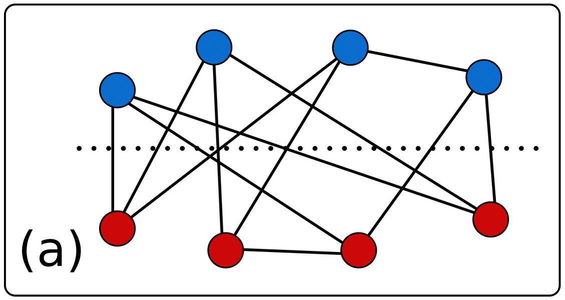
FALQON for MaxCut problem

Approximation ration:

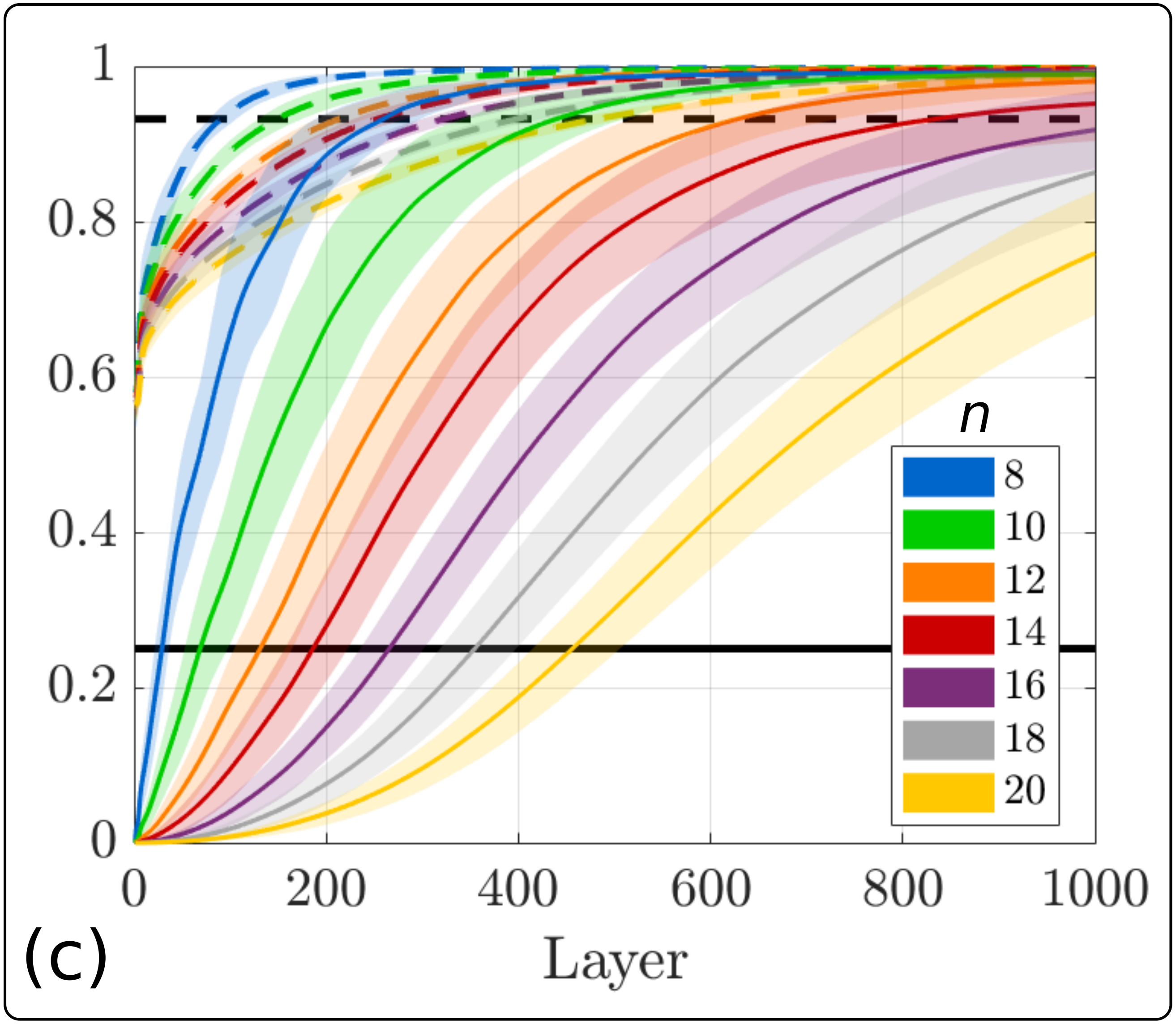
$$r_{\mathrm{A}} = \langle H_{\mathrm{p}} \rangle / \langle H_{\mathrm{p}} \rangle_{\min}$$

The largest known approximation ratio $r_A = 0.932$ by algorithm of Goemans and Williamson.

approximation ratio (dashed curves) and the success probability of measuring the degenerate ground state (solid curves)

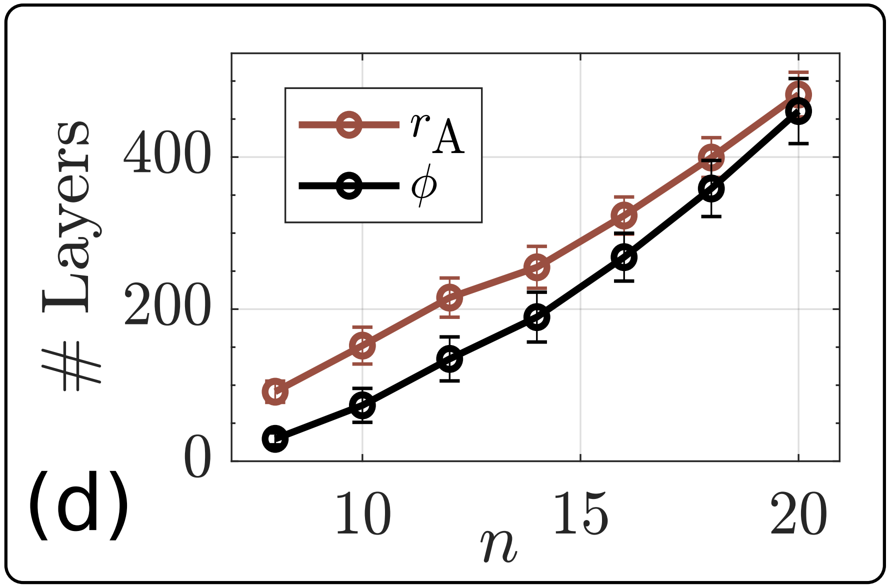


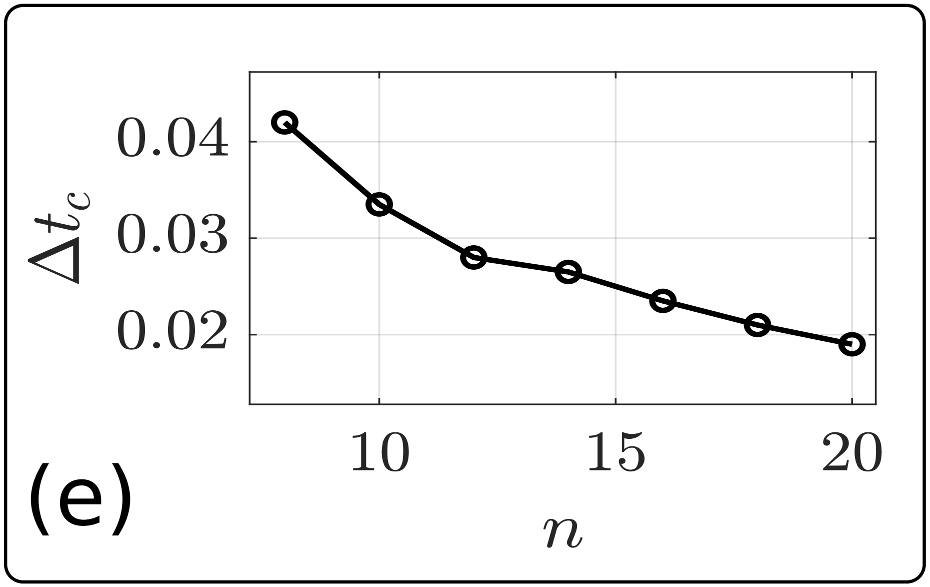
Pictorial representation of MaxCut on a 3-regular graph with 8 vertices.



n = the number of vertices

FALQON for MaxCut problem





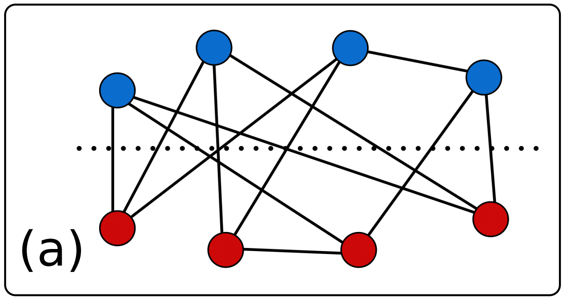
The mean number of layers needed to achieve the reference values of $r_A = 0.932$ (dashed curve) and $\phi = 0.25$ (solid curve) is shown; error bars report the associated standard deviation.

The critical Δt values for different problem sizes are plotted.

The only free parameter is time step Δt, which is tuned to be as large as possible.

$r_A = \langle H_p \rangle / \langle H_p \rangle_{\min}$ = approximation ratio

$\phi = \sum_i |\langle \psi | q_{0,i} \rangle|^2$ = the success probability of measuring the (potentially degenerate) ground state(s) $\{|q_{0,i}\rangle\}$,



Pictorial representation of MaxCut on a 3-regular graph with 8 vertices.

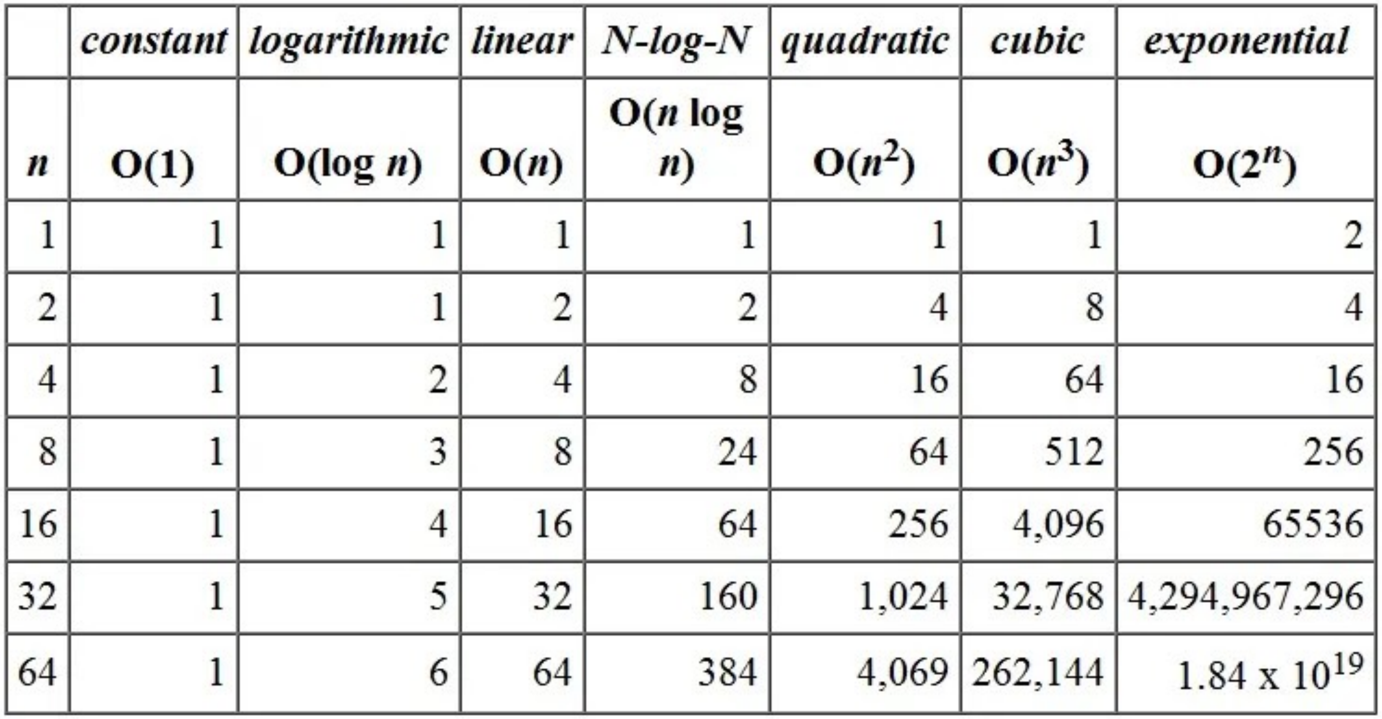
Ising formulations of many NP problems

Andrew Lucas

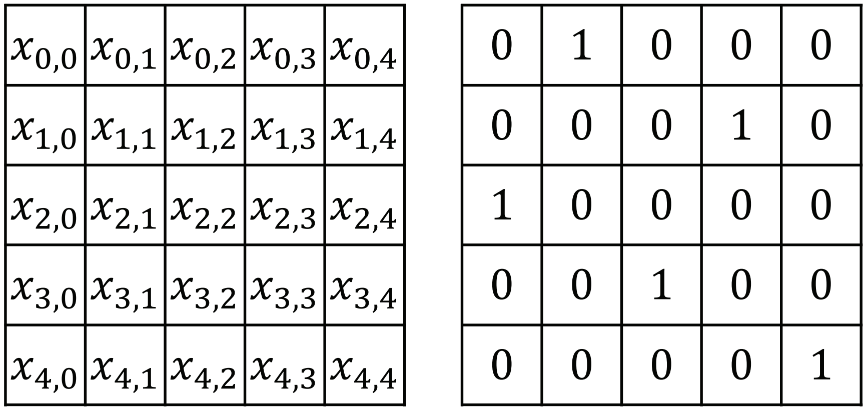
1302.5843

$$H = -\sum_{i,j} J_{ij}\,\sigma_i^z\,\sigma_j^z - \sum_i h_i\,\sigma_i^z$$

N-Queens problem



	constant	*logarithmic*	*linear*	*N-log-N*	*quadratic*	*cubic*	*exponential*
n	**O(1)**	**O(log *n*)**	**O(*n*)**	**O(*n* log *n*)**	**O(n^2)**	**O(n^3)**	**O(2^n)**
1	1	1	1	1	1	1	2
2	1	1	2	2	4	8	4
4	1	2	4	8	16	64	16
8	1	3	8	24	64	512	256
16	1	4	16	64	256	4,096	65536
32	1	5	32	160	1,024	32,768	4,294,967,296
64	1	6	64	384	4,069	262,144	1.84×10^{19}

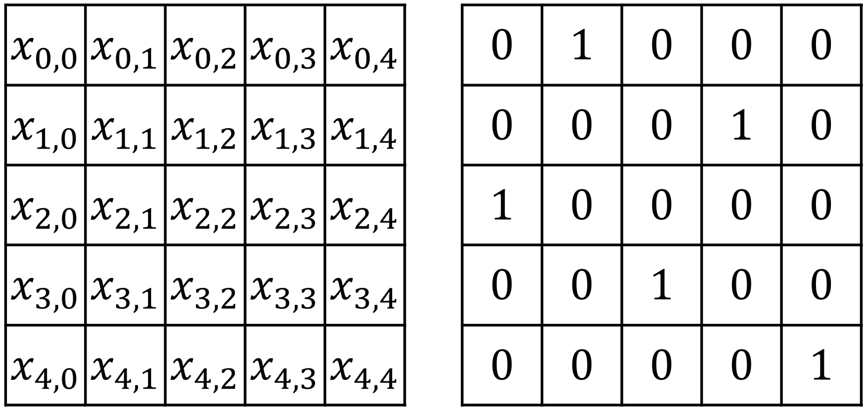


$$\begin{array}{|c|c|c|c|c|}\hline x_{0,0} & x_{0,1} & x_{0,2} & x_{0,3} & x_{0,4} \\ \hline x_{1,0} & x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ \hline x_{2,0} & x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} \\ \hline x_{3,0} & x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} \\ \hline x_{4,0} & x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \\ \hline \end{array} \qquad \begin{array}{|c|c|c|c|c|}\hline 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 \\ \hline \end{array}$$

Fig. 4. An 5 × 5-matrix for the 5-Queen problem and an example of the solution.

N-Queens problem

$$E_1(X) = (1 - \sum_{i=0}^{n-1} x_i)^2 \quad = -\sum_{i=0}^{n-1} x_i + 2\sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} x_i x_j + 1$$



$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$	$x_{0,4}$
$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$
$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$
$x_{3,0}$	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$
$x_{4,0}$	$x_{4,1}$	$x_{4,2}$	$x_{4,3}$	$x_{4,4}$

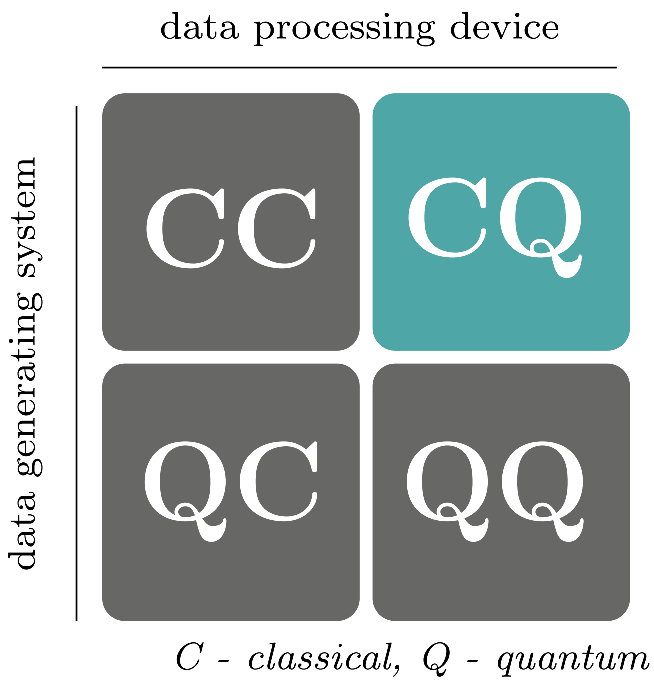
0	1	0	0	0
0	0	0	1	0
1	0	0	0	0
0	0	1	0	0
0	0	0	0	1

Fig. 4. An 5 × 5-matrix for the 5-Queen problem and an example of the solution.

Distance-based classifier

Quantum Machine Learning

- Artificial Intelligence: Statistical prediction
- Machine Learning: Learn from data
- Quantum Machine Learning: Learn from data with quantum algorithms
  - Subdiscipline of quantum computing and quantum information science

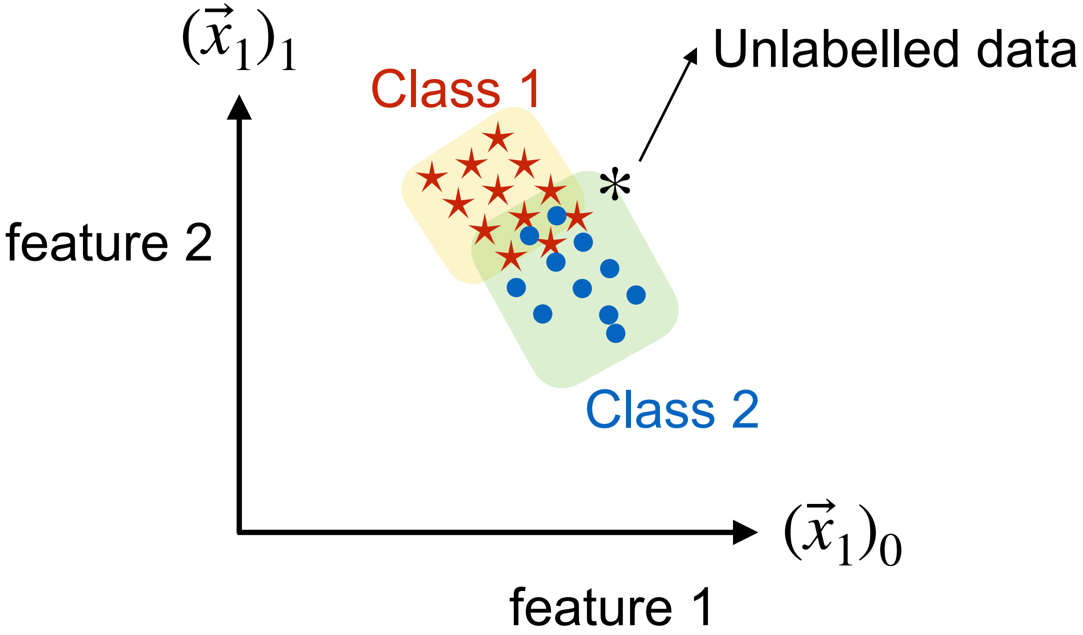


C - classical, Q - quantum

- CC: classical data being processed classically
- QC: how machine learning can help with quantum computing
- CQ: classical data fed into quantum computer for analysis (quantum machine learning)
- QQ: quantum data being processed by quantum computer (ex: Quantum simulation)

Distance-based classifier

- A distance-based classifier with a quantum interference circuit: arXiv:1703:10793 (supervised binary classification)



training data set

$$D = \left\{(\vec{x}_1, y_1),\ (\vec{x}_2, y_2),\ \cdots,\ (\vec{x}_M, y_M)\right\}$$

$$\vec{x}_m \in \mathbb{R}^N \quad y_m \in \{-1,\ +1\}$$

$$m = 1, 2,\ \cdots, M$$

$$M = \text{the number of data}$$

$$N = \text{the number of features}$$

$$\vec{\tilde{x}}_m \in \mathbb{R}^N: \ \text{unlabelled data}$$

→ Find the label $\tilde{y} \in \{-1,1\}$

Classical Kernel Method

- Kernel methods: kNN (k-nearest neighborhood), KDE (kernel density estimation), SVM (support vector machine), Gaussian processes
  - Nearest neighborhood method: a new input data is given the same label as the data point closest to it → k-nearest neighborhood (kNN)
  - Closeness = distance measure
  - (ex) Euclidean distance $|\vec{\tilde{x}} - \vec{x}_m|^2$

$$\tilde{y} = \text{sign}\left[\sum_{m=1}^{M} y_m\left(1 - \frac{1}{4M}|\vec{\tilde{x}} - \vec{x}_m|^2\right)\right]$$

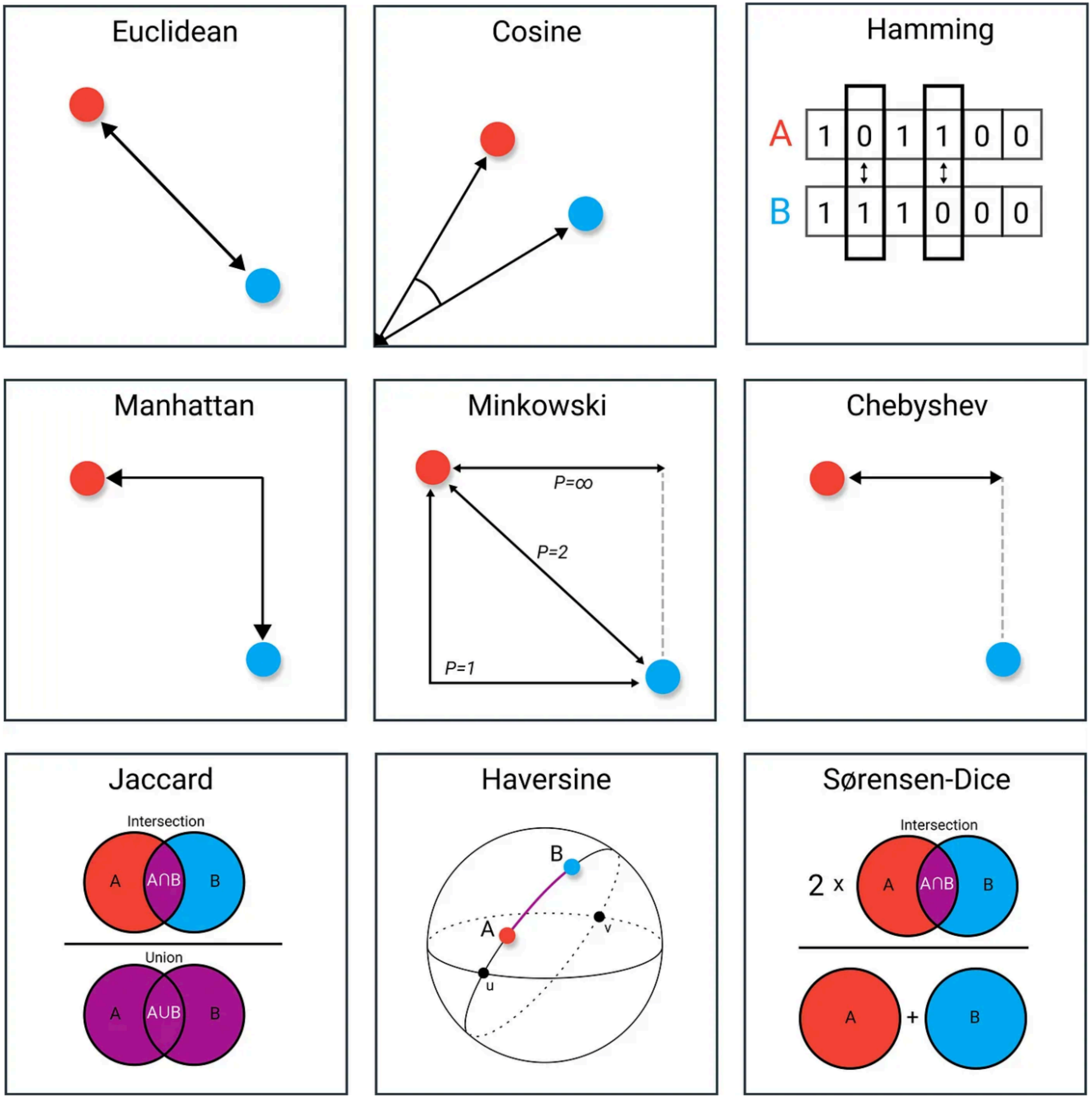
- include all data but weigh influence of each data toward the decision by the weight $\kappa(\vec{\tilde{x}}, \vec{x}_m)$

$$\tilde{y} = \text{sign}\left[\sum_{m=1}^{M} w_m\, y_m\, \kappa(\vec{\tilde{x}}, \vec{x}_m)\right]$$

Kernel

weight

Label ±1 for $\vec{x}_m$


Euclidean
Cosine
Hamming
A 1 0 1 1 0 0
B 1 1 1 0 0 0
Manhattan
Minkowski
P=∞
P=2
P=1
Chebyshev
Jaccard
Intersection
A
A∩B
B
Union
A
AUB
B
Haversine
B
A
v
u
Sørensen-Dice
Intersection
2 x
A
A∩B
B
A
+
B

Wasserstein distance (Kantorovich–Rubinstein metric)

- A distance function defined between probability distributions on a given metric space M (named after “Vaseršteĭn" (Russian: Васерштейн))
- If P is an empirical measure with samples $X_1, \cdots, X_n$ and Q is an empirical measure with samples $Y_1, \cdots, Y_n$ the p-Wasserstein distance is a simple function of the order statistics:

$$W_p(P,Q) = \left(\frac{1}{n} \sum_{i=1}^{n} \|X_{(i)} - Y_{(i)}\|^p \right)^{1/p}.$$

Classical Kernel Method

- Kernel methods: kNN (k-nearest neighborhood), KDE (kernel density estimation), SVM (support vector machine), Gaussian processes
  - Nearest neighborhood method: a new input data is given the same label as the data point closest to it → k-nearest neighborhood (kNN)
  - Closeness = distance measure
  - (ex) Euclidean distance $|\vec{\tilde{x}} - \vec{x}_m|^2$

$$\tilde{y} = \text{sign}\left[\sum_{m=1}^{M} y_m\left(1 - \frac{1}{4M}|\vec{\tilde{x}} - \vec{x}_m|^2\right)\right]$$

- include all data but weigh influence of each data toward the decision by the weight $\kappa(\vec{\tilde{x}}, \vec{x}_m)$

$$\tilde{y} = \text{sign}\left[\sum_{m=1}^{M} w_m\, y_m\, \kappa(\vec{\tilde{x}}, \vec{x}_m)\right]$$

Kernel

weight

Label ±1 for $\vec{x}_m$

Distance-based classifier

- Choose $w_m = 1$ for all equally important data

$$\kappa(\vec{\tilde{x}}, \vec{x}_m) = 1 - \frac{1}{4M} |\vec{\tilde{x}} - \vec{x}_m|^2$$

Close data (small distance) are weighted more importantly.

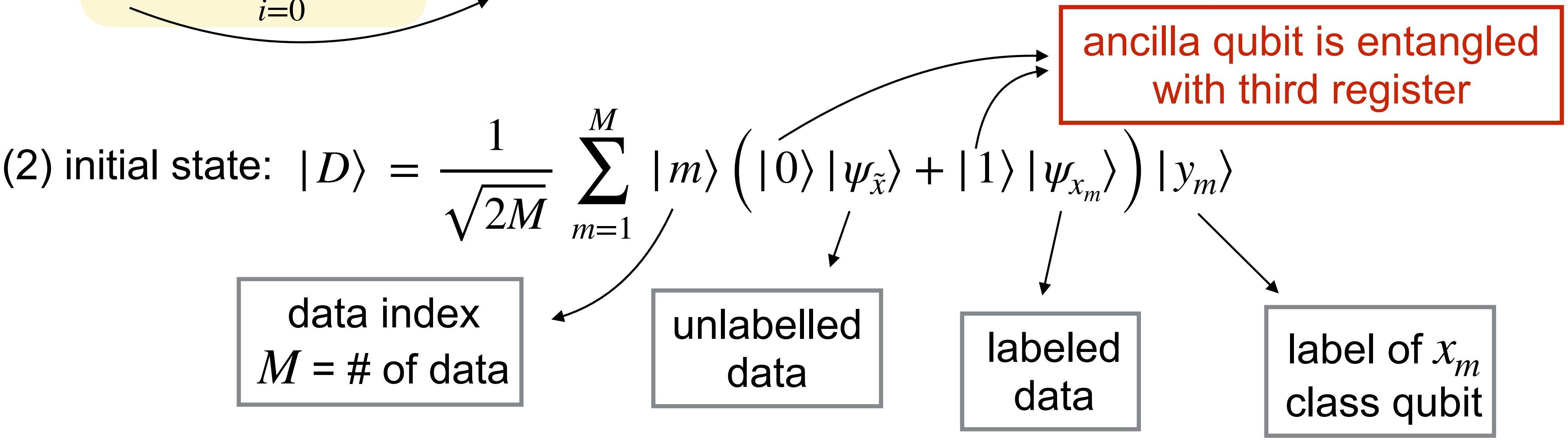
(1) Encode input data (features) into the amplitude of a quantum system (amplitude encoding). For classical vector $\vec{x} \in \mathbb{R}^N$, $(N = 2^n)$ Assume $x^T x = \vec{x} \cdot \vec{x} = 1$ (normalized to 1)

$N = 2^n$: number of features

$$|\psi_X\rangle = \sum_{i=0}^{N-1} x_i |i\rangle$$

i : index in the computational basis

Dimension of Hilbert space $\approx O(\log N)$



(2) initial state: $|D\rangle = \frac{1}{\sqrt{2M}} \sum_{m=1}^{M} |m\rangle \left(|0\rangle |\psi_{\tilde{x}}\rangle + |1\rangle |\psi_{x_m}\rangle \right) |y_m\rangle$

ancilla qubit is entangled with third register

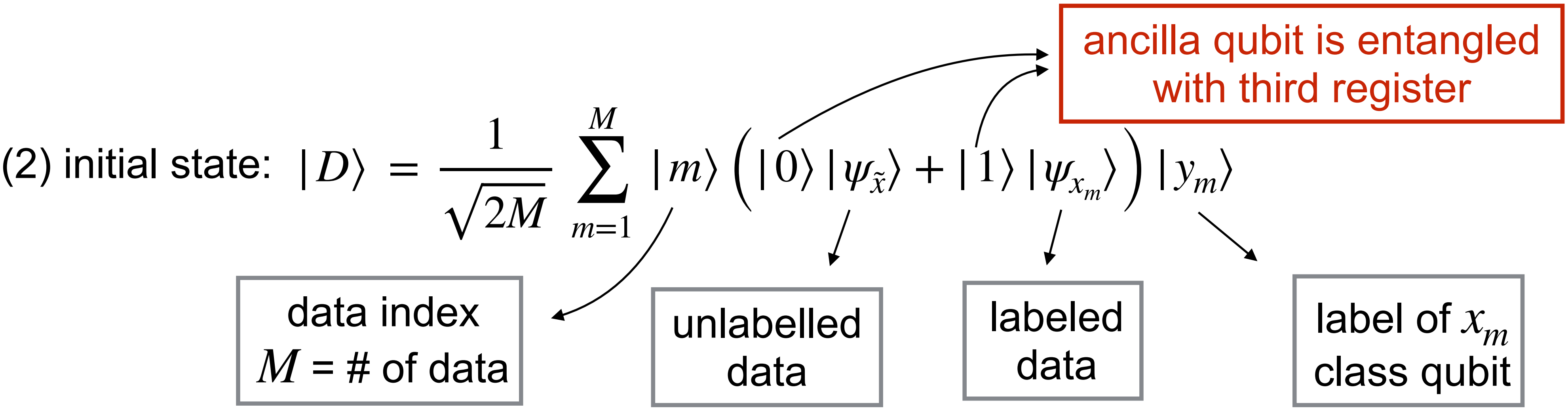
data index M = # of data

unlabelled data

labeled data

label of x_m class qubit

Distance-based classifier



(2) initial state: $|D\rangle = \frac{1}{\sqrt{2M}} \sum_{m=1}^{M} |m\rangle \Big(|0\rangle\, |\psi_{\tilde{x}}\rangle + |1\rangle\, |\psi_{x_m}\rangle \Big)\, |y_m\rangle$

ancilla qubit is entangled with third register

data index
M = # of data

unlabelled data

labeled data

label of x_m
class qubit

$$|\psi_{x_m}\rangle = \sum_{i=0}^{N-1} x_m^i\, |i\rangle$$ encoding of m-th training data (labeled)

$$|\psi_{\tilde{x}}\rangle = \sum_{i=0}^{N-1} \tilde{x}^i\, |i\rangle$$ encoding of new data (unlabeled)

$$|y_m\rangle = \begin{cases} |0\rangle\,, & \text{if } y_m = -1 \\ |1\rangle\,, & \text{if } y_m = +1 \end{cases}$$

$|D\rangle$ contains all training data as well as M copies of new inputs.

Distance-based classifier

(3) Apply Hadamard gate on the ancilla (second) qubit.

$$|0\rangle \;\rightarrow\; \frac{1}{\sqrt{2}}\Big(\,|0\rangle + |1\rangle\Big)$$

$$|1\rangle \;\rightarrow\; \frac{1}{\sqrt{2}}\Big(\,|0\rangle - |1\rangle\Big)$$

$$|D\rangle \;=\; \frac{1}{\sqrt{2M}} \sum_{m=1}^{M} |m\rangle \Big(|0\rangle\,|\psi_{\tilde{x}}\rangle + |1\rangle\,|\psi_{x_m}\rangle\Big)\,|y_m\rangle$$

$$\downarrow$$

$$|D'\rangle \;=\; \frac{1}{2\sqrt{M}} \sum_{m=1}^{M} |m\rangle \Big(|0\rangle\,|\psi_{\tilde{x}+x_m}\rangle + |1\rangle\,|\psi_{\tilde{x}-x_m}\rangle\Big)\,|y_m\rangle$$

$$|\psi_{\tilde{x}\pm x_m}\rangle \;=\; |\psi_{\tilde{x}}\rangle \pm |\psi_{x_m}\rangle \;=\; \sum_{i=0}^{M-1} \left(\,\tilde{x}^i \pm x_m^i\,\right)|i\rangle$$

(4) Conditional measurement selecting the branch with ancilla state $|0\rangle$.

Likely to succeed if the collective Euclidean distance b/w $\tilde{x}$ and training data set is small. For standard data, $p \geq 0.5$.

Probability is $\quad p \;=\; \dfrac{1}{4M} \displaystyle\sum_{m} |\vec{\tilde{x}} + \vec{x}_m|^2$

$$|D''\rangle \;=\; \frac{1}{2\sqrt{Mp}} \sum_{m=1}^{M} \sum_{i=0}^{N-1} |m\rangle \left(\tilde{x}^i + x_m^i\right)|i\rangle\,|y_m\rangle$$

Distance-based classifier

(5) Probability of measuring the class qubit $|y_m\rangle = |0\rangle$

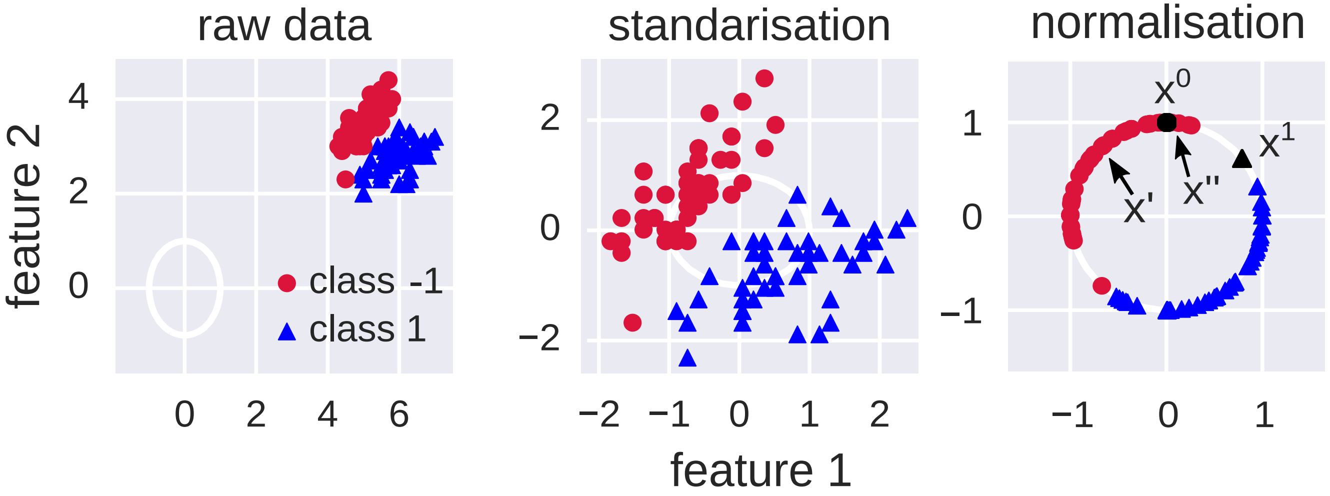
$$|D''\rangle = \frac{1}{2\sqrt{Mp}} \sum_{m=1}^{M} \sum_{i=0}^{N-1} |m\rangle \left(\tilde{x}^i + x_m^i\right) |i\rangle |y_m\rangle$$

$$P(\tilde{y} = 0) = \frac{1}{4Mp} \sum_{y_m=0,\, m=1}^{M} |\vec{\tilde{x}} + \vec{x}_m|^2 = 1 - \frac{1}{4Mp} \sum_{y_m=0,\, m=1}^{M} |\vec{\tilde{x}} - \vec{x}_m|^2$$

Class 1

using normalization condition

→ choosing the class with the higher probability gives result of kernel method.
The # of measurement needed to estimate $P(\tilde{y} = 0)$ to error ϵ with a reasonably high confidence interval grows with $O(\epsilon^{-1})$.

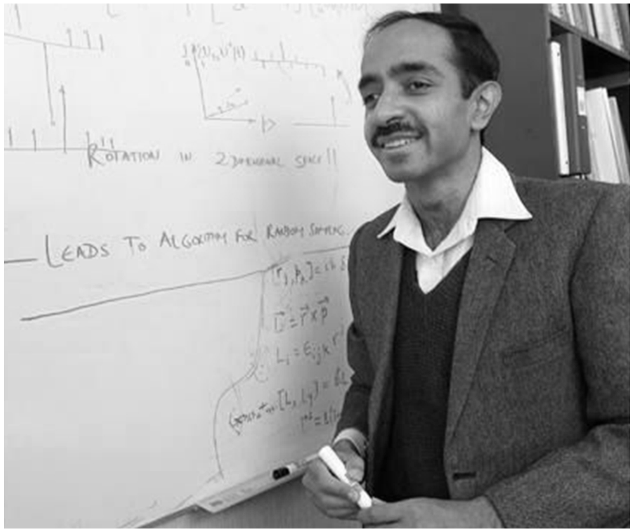


- arXiv:1703:10793 used Iris data

https://www.quantum-inspire.com/kbase/jupyter-classifier-part1/

Grover’s search algorithm

Grover’s search algorithm


ROTATION IN 2-DIMENSIONAL SPACE
LEADS TO ALGORITHM FOR RANDOM SAMPLING

- Grover’s algorithm involves “amplitude amplification”
  - G. Brassard, P. Hoyer 1997, Lov Grover 1998
  - QFT is used for Shor’s and Simon’s algorithms
- Example: Find a name in a phone directory (ordered list)
  - Go to the midpoint of the list, see which half contains the name. Repeat the same → bisection method takes $\log_2 N$ operations until one of left.
- If we are given an unordered list, we will have to check all entries one a time. On average, this would take $N/2$ operations
  - For $N = 10^6$, $\log_2 N \approx 20$ and $N/2 \approx 5 \times 10^5$.
- Grover’s algorithm (unstructured search): determines the special value with $p \approx 1$ (close to 1) by calling subroutine only $\frac{\pi}{4}\sqrt{N}$ times. → quadratic speed up compared with a classical computer.
  - (cf) exponential speed up is expected in Shor’s algorithm.

Grover’s algorithm: Black Box (Oracle)

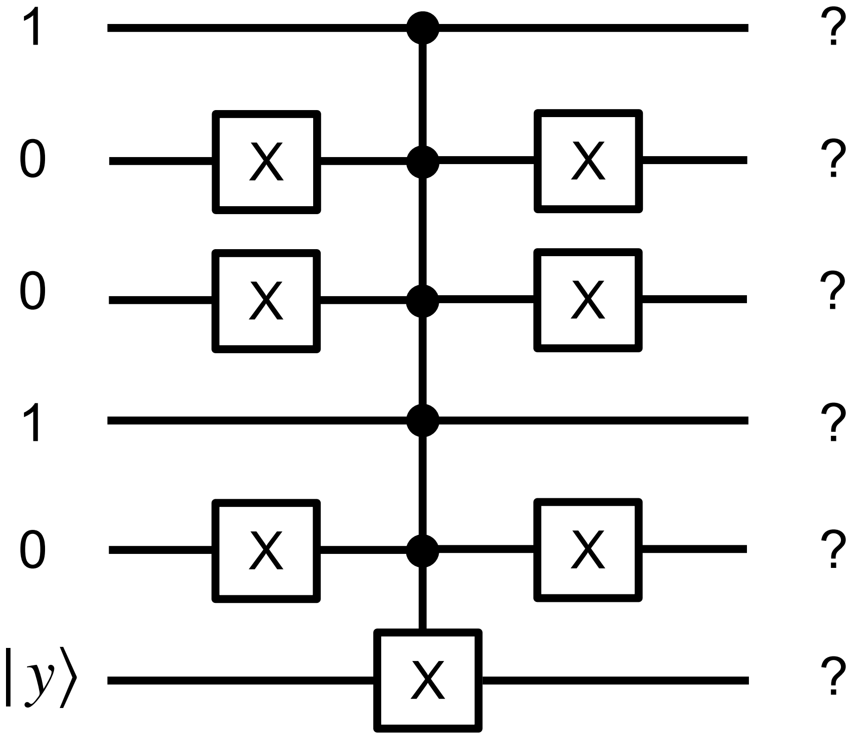
- Consider n-bit integers.
- “a” is a special number, and the goal is to find “a”.
- Define a subroutine which output 1 if input value x is equal to a, and output 0 otherwise.

$$f(a) = 1, \quad f(x) = 0 \ \text{ for } x \neq a$$

Example: a=01001

$$\boxed{U\,|x\rangle_n \otimes |y\rangle_1 = |x\rangle_n \otimes |y \oplus f(x)\rangle_1}$$

n qubits one qubit



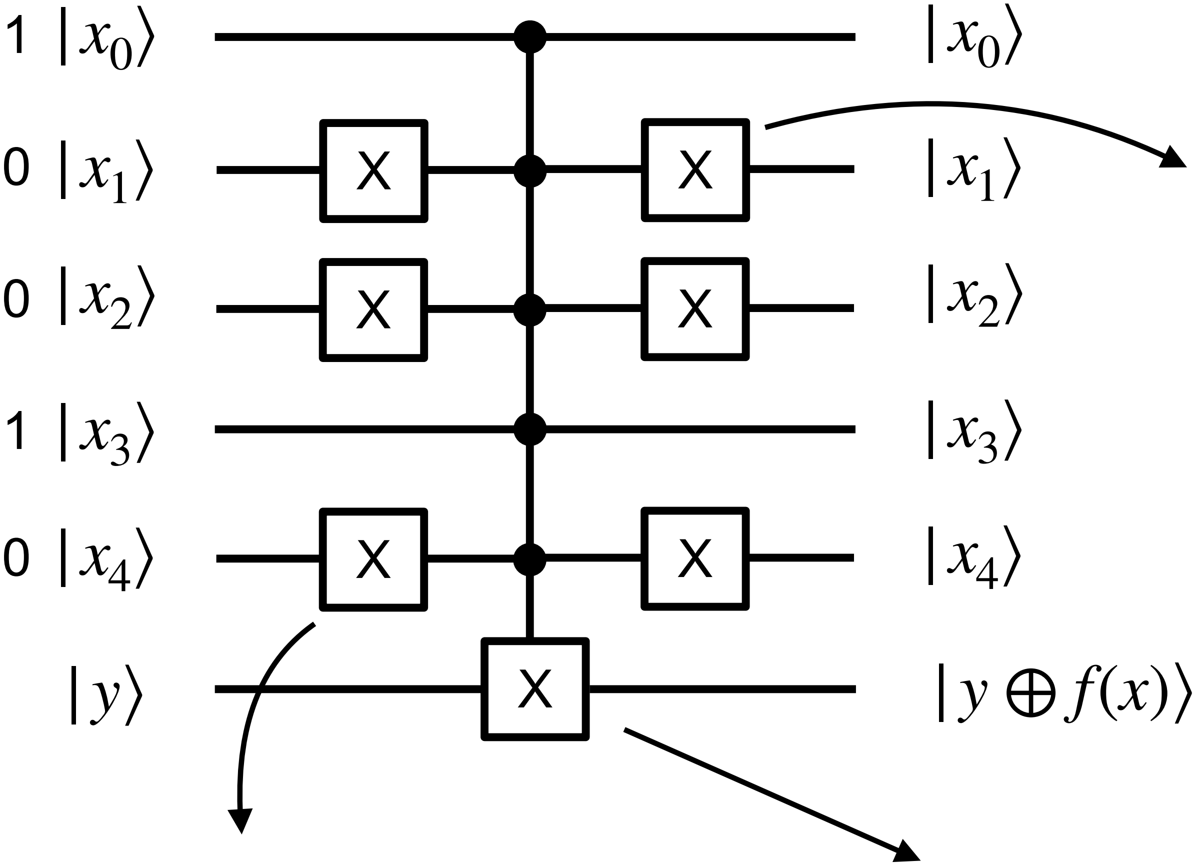
$$a = x_4x_3x_2x_1x_0 = 01001$$

Grover’s algorithm: Black Box (Oracle)

$a = x_4 x_3 x_2 x_1 x_0 = 01001$

$f(a) = 1$

$f(x) = 0, \text{ if } x \neq a$



(3) X-gates on the right flip back to the original input.

(1) X-gates on the left flip qubits x_1, x_2 and x_4 → target qubit is flipped only if $x_4 x_3 x_2 x_1 x_0 = 01001$

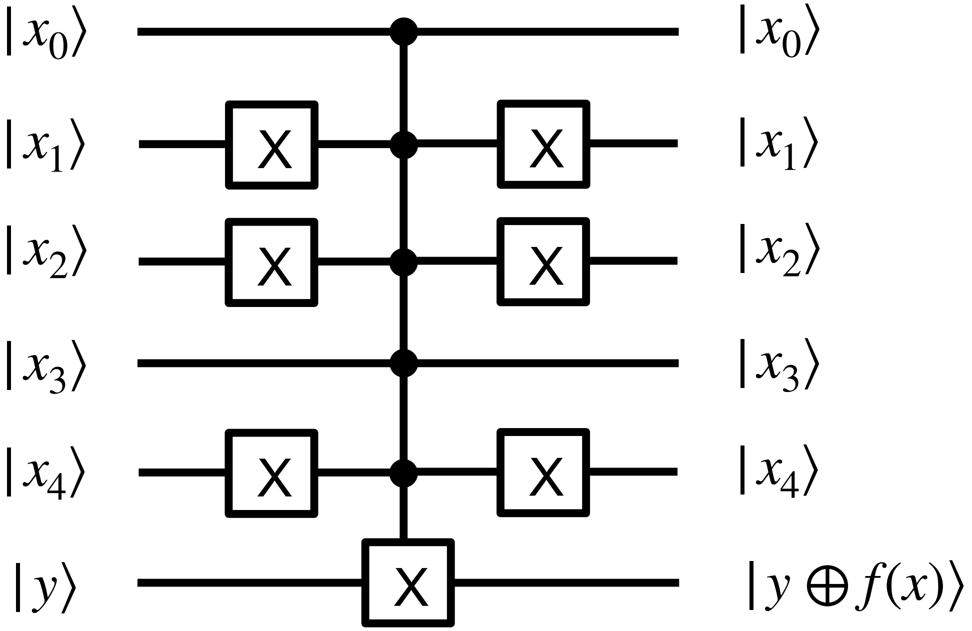
(2) Five-fold-controlled NOT acts to flip the target qubit y, only if all control bits are 1.

Grover's algorithm: Black Box (Oracle)

$a = x_4x_3x_2x_1x_0 = 01001$

$f(a) = 1$

$f(x) = 0, \text{ if } x \neq a$



- Useful to initialize $|y\rangle = |1\rangle$ and apply H before U.
- The output qubit is

$$H|1\rangle = \frac{1}{\sqrt{2}}\Big(|0\rangle - |1\rangle\Big)$$

$$\text{if } f(x) = 0, \qquad |0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle = |0\rangle - |1\rangle$$

$$f(x) = 1, \qquad |0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle = |1\rangle - |0\rangle = -\big(|0\rangle - |1\rangle\big)$$

Target qubit changes the sign, depending on the function value.

$$U\Big(|x\rangle \otimes H|1\rangle\Big) = (-1)^{f(x)}|x\rangle \underbrace{\otimes\ H|1\rangle}_{}$$

Output remains the same.

Grover’s search algorithm

$$U\Big(|x\rangle \otimes H|1\rangle\Big) = (-1)^{f(x)}|x\rangle \otimes H|1\rangle$$

U and Q are linear operators.

$$\text{Define}: \;\; Q|x\rangle = (-1)^{f(x)}|x\rangle = \begin{cases} |x\rangle, & \text{for} \;\; x \neq a \\ -|a\rangle, & \text{for} \;\; x = a \end{cases}$$

For a general state, $|\psi\rangle = \sum_x C_x|x\rangle,$

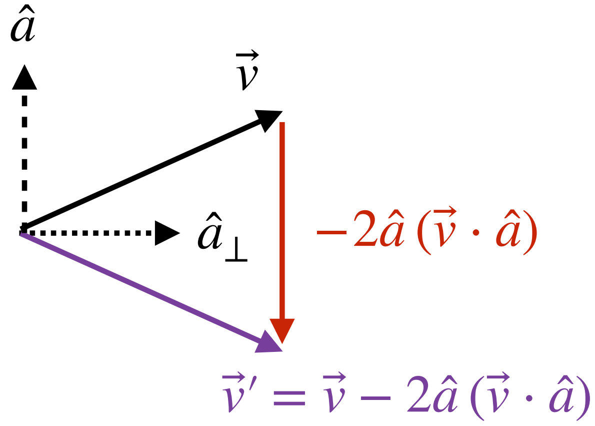
$$C_a \equiv \langle a|\psi\rangle$$

$$|\psi'\rangle = Q|\psi\rangle = \sum_{x\neq a} C_x|x\rangle - C_a|a\rangle = \sum_x C_x|x\rangle - 2C_a|a\rangle = |\psi\rangle - 2|a\rangle\langle a|\psi\rangle$$

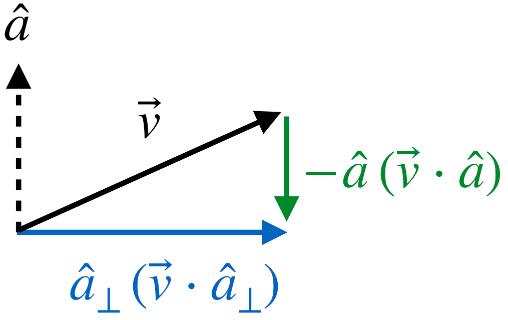
$\langle a|\psi'\rangle = \langle a|\psi\rangle - 2\langle a|\psi\rangle = -\langle a|\psi\rangle$ → Suppose $|x\rangle$ satisfies $\langle x|a\rangle = 0$ for $x \neq a$

$\langle a_\perp|\psi'\rangle = \langle a_\perp|\psi\rangle$

Define such $|x\rangle$ as $|a_\perp\rangle$ with $\langle a|a_\perp\rangle = 0$



$\hat{a}$
$\vec{v}$
$\hat{a}_\perp$
$-2\hat{a}\,(\vec{v}\cdot\hat{a})$
$\vec{v}\,' = \vec{v} - 2\hat{a}\,(\vec{v}\cdot\hat{a})$



$\hat{a}$
$\vec{v}$
$-\hat{a}\,(\vec{v}\cdot\hat{a})$
$\hat{a}_\perp(\vec{v}\cdot\hat{a}_\perp)$

$\hat{a}\cdot\hat{a}_\perp = 0$

$\vec{v}\cdot\hat{a} = -\vec{v}\,'\cdot\hat{a}$

$\vec{v}\cdot\hat{a}_\perp = \vec{v}\,'\cdot\hat{a}_\perp$

reflection around the direction perpendicular to $\hat{a}$

Grover's search algorithm

- Consider uniform superposition of all possible basis states.

$$|\psi_0\rangle = H^{\otimes n}|0\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle \qquad N = 2^n \qquad \langle a|a_\perp\rangle = 0$$

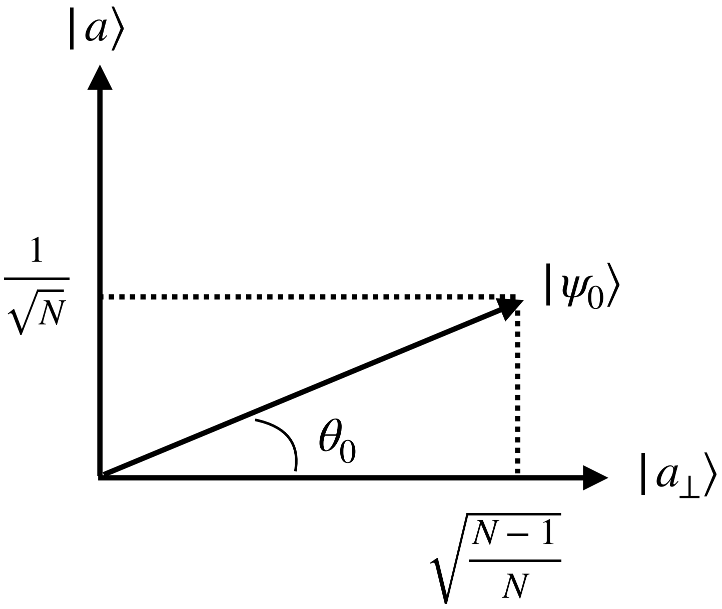
$$\langle a|a\rangle = 1$$

$$|\psi_0\rangle = \frac{1}{\sqrt{N}}|a\rangle + \sqrt{\frac{N-1}{N}}|a_\perp\rangle = \sin\theta_0|a\rangle + \cos\theta_0|a_\perp\rangle \qquad \langle a_\perp|a_\perp\rangle = 1$$

$$|a_\perp\rangle = \frac{1}{\sqrt{N-1}}\sum_{x\neq a,\, x=0}^{N-1}|x\rangle$$

$|a_\perp\rangle$ is the normalized uniform superposition of all basis states perpendicular to $|a\rangle$

$$\langle a|\psi_0\rangle = \frac{1}{\sqrt{N}} \equiv \sin\theta_0 \qquad \langle a_\perp|\psi_0\rangle = \sqrt{\frac{N-1}{N}} \equiv \cos\theta_0$$

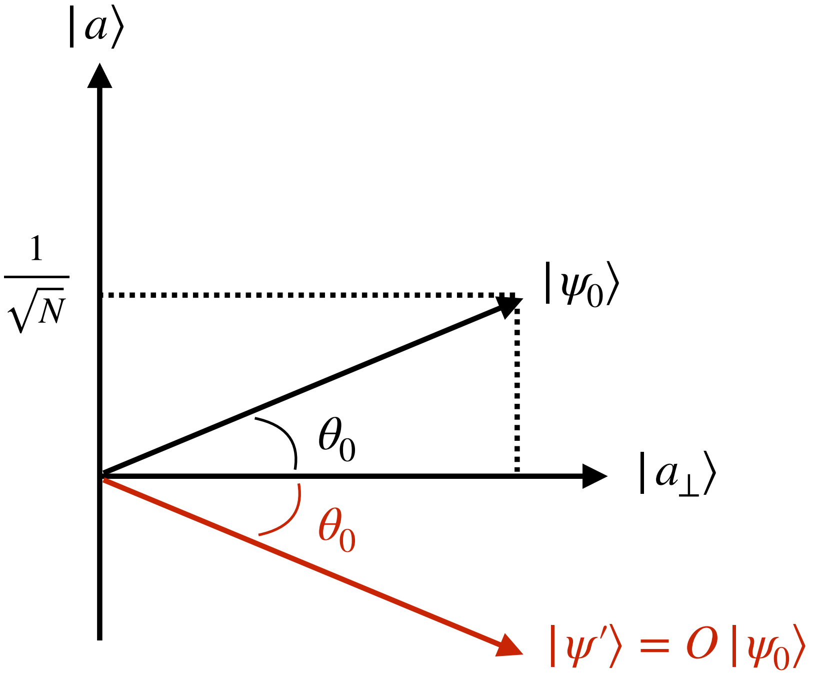


- Probability of $|\psi_0\rangle$ being in $|a\rangle$ = $|\langle a|\psi_0\rangle|^2 = \sin^2\theta_0 = \frac{1}{N}$

very small for a large N

- Grover's algorithm: iteratively rotate $|\psi_0\rangle$ (very close to $|a_\perp\rangle$ initially) to the direction close to $|a\rangle$ axis so that measurement returns a high probability → amplitude amplification

Grover’s search algorithm



- Reflection about $|a_\perp\rangle$

$$|\psi'\rangle = O|\psi_0\rangle = |\psi\rangle - 2|a\rangle\langle a|\psi\rangle$$

- O reflects $|\psi_0\rangle$ about $|a_\perp\rangle$ axis

- $O|x\rangle = |x\rangle$ for $x \neq a,\ O|a_\perp\rangle = |a_\perp\rangle$

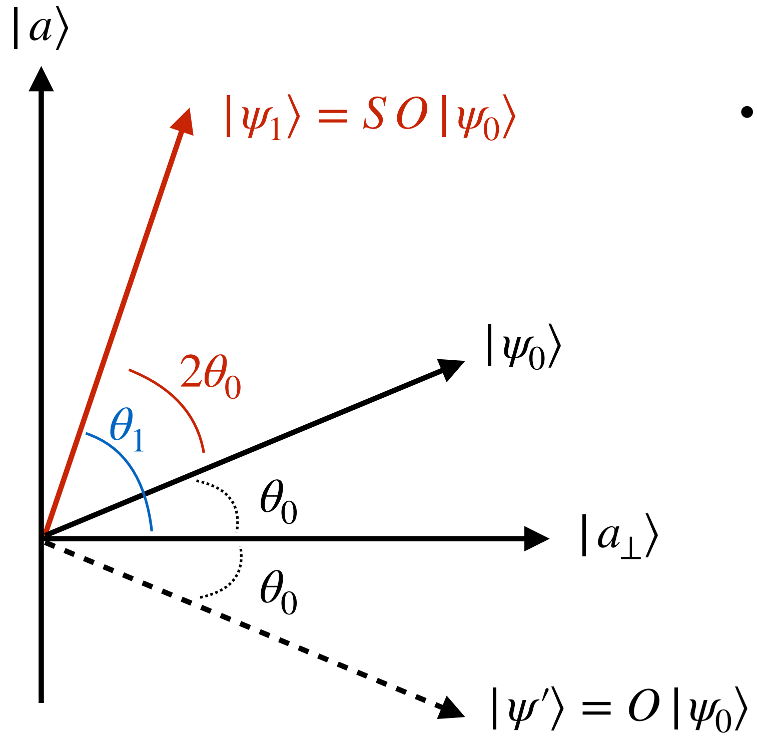
$O|a\rangle = -|a\rangle$ ⇨ flips $|a\rangle$ to $-|a\rangle$

$$|\psi_0\rangle = \sin\theta_0|a\rangle + \cos\theta_0|a_\perp\rangle$$

$$O|\psi_0\rangle = O\Big(\sin\theta_0|a\rangle + \cos\theta_0|a_\perp\rangle\Big)$$

$$= -\sin\theta_0|a\rangle + \cos\theta_0|a_\perp\rangle$$

Grover’s search algorithm



- Reflection about $|\psi_0\rangle$ (initial state)

$$|\phi\rangle \longrightarrow |\phi'\rangle = S|\phi\rangle = 2|\psi_0\rangle\langle\psi_0|\phi\rangle - |\phi\rangle$$

$$\langle\psi_0|\phi'\rangle = 2\langle\psi_0|\psi_0\rangle\langle\psi_0|\phi\rangle - \langle\psi_0|\phi\rangle = \langle\psi_0|\phi\rangle$$

→ component along $|\psi_0\rangle$ does not change.

$$\langle\psi_{0\perp}|\phi'\rangle = 2\langle\psi_{0\perp}|\psi_0\rangle\langle\psi_0|\phi\rangle - \langle\psi_{0\perp}|\phi\rangle = -\langle\psi_{0\perp}|\phi\rangle$$

→ component perpendicular to $|\psi_0\rangle$ changes the sign.

$G = SO$

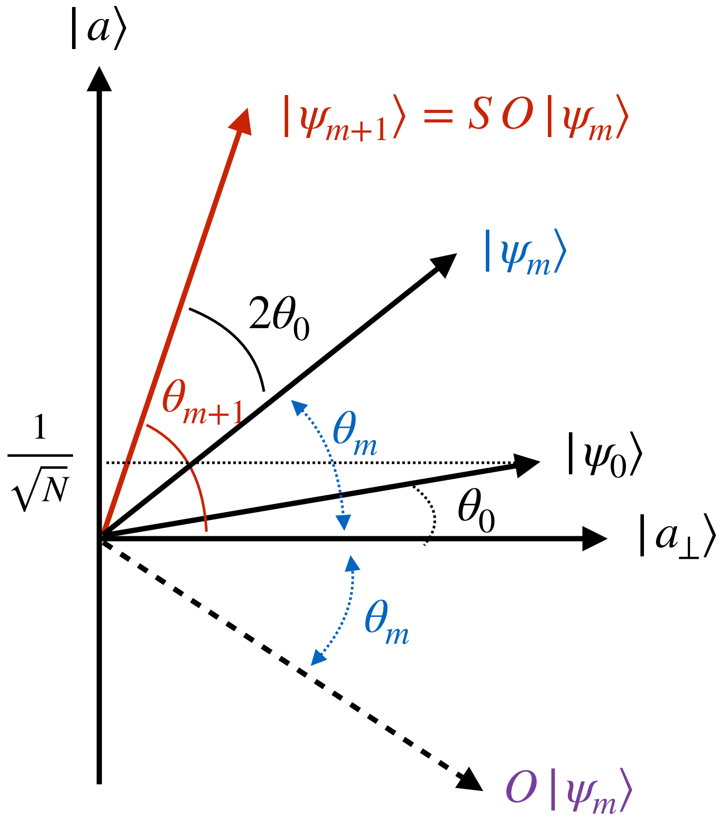
O: reflection of $|\psi_0\rangle$ about $|a_\perp\rangle$

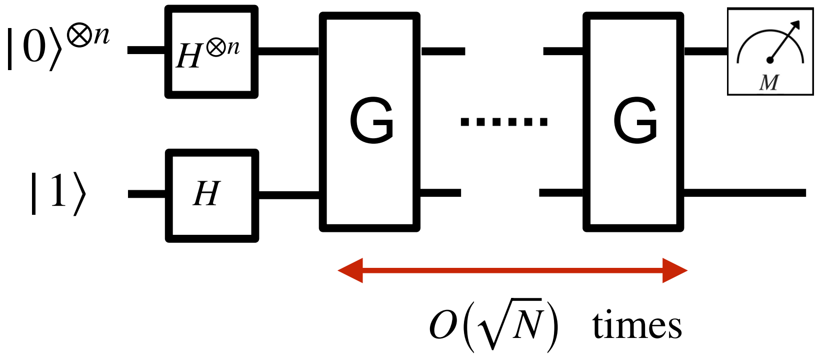
S: reflection of $|\psi'\rangle = O\,|\psi_0\rangle$ about initial state $|\psi_0\rangle$

G: Grover operator rotates the initial state $|\psi_0\rangle$ by $2\theta_0$ counterclockwise (toward the direction of $|a\rangle$ axis)

- Effect of 1st Grover iteration: rotate the initial state $|\psi_0\rangle$ by $2\theta_0$ counterclockwise.
- $|\psi_1\rangle$ making angle θ_1 to $|a_\perp\rangle$ axis, $\theta_1 = \theta_0 + 2\theta_0$

Grover's search algorithm





$|\psi_m\rangle = \cos\theta_m\,|a_\perp\rangle + \sin\theta_m\,|a\rangle$

$O\,|\psi_m\rangle$ rotates about $|a_\perp\rangle$ by angle $2\theta_m$

$SO\,|\psi_m\rangle$ rotates by angle $2(\theta_m + \theta_0)$ counterclockwise.

$$\theta_{m+1} = \theta_m + 2\theta_0 \qquad \theta_m = (2m+1)\,\theta_0$$

- $\langle a\,|\psi_m\rangle = \sin\theta_m = \sin\left[(2m+1)\theta_0\right]$
- Optimal number of Grover iteration: $\theta_m = \pi/2$

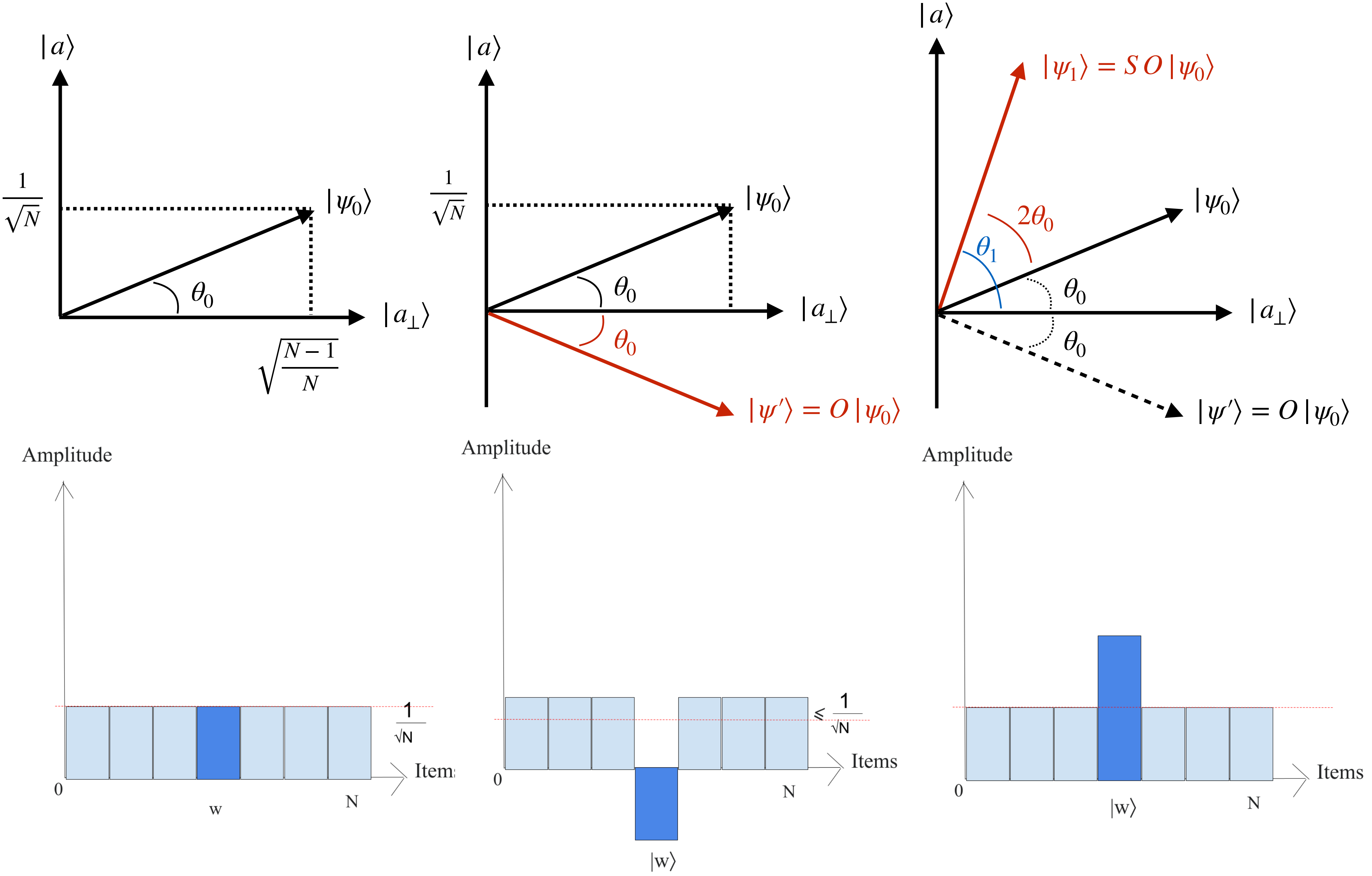
$$\frac{\pi}{2} = \theta_m = (2m+1)\,\theta_0 = (2m+1)\ \sin^{-1}\!\left(\frac{1}{\sqrt{N}}\right)$$

For a large N, $m = \frac{\pi}{4}\sqrt{N}$

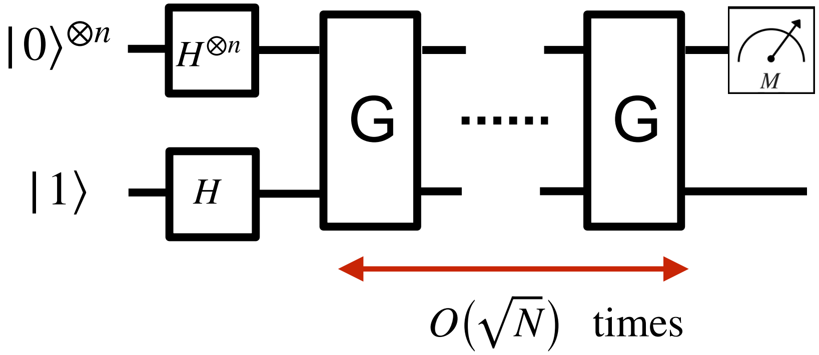
- When $\theta_m \approx \pi/2$, measurement gives a with high probability.
- For any value of θ_m such that $\frac{\pi}{4} < \theta_m \approx \frac{2m}{\sqrt{N}} < \frac{3\pi}{4}$

$\rightarrow \frac{\pi}{8}\sqrt{N} < m < \frac{3\pi}{8}\sqrt{N}$, Grover algorithm returns $|a\rangle$ with probability > 1/2.

Grover’s search algorithm



Grover’s search algorithm



- Optimal number of Grover iteration: $\theta_m = \pi/2$

$$\frac{\pi}{2} = \theta_m = (2m+1)\,\theta_0 = (2m+1)\ \sin^{-1}\left(\frac{1}{\sqrt{N}}\right)$$

For a large N, $m = \frac{\pi}{4}\sqrt{N}$

- When $\theta_m \approx \pi/2$, measurement gives a with high probability.
- For any value of θ_m such that $\frac{\pi}{4} < \theta < \frac{3\pi}{4} \rightarrow \frac{\pi}{8}\sqrt{N} < m < \frac{3\pi}{8}\sqrt{N}$, Grover algorithm returns $|a\rangle$ with probability > 1/2.
- Probability decreases for $m > \frac{\pi}{4}\sqrt{N}$.
- Operational count of the Grover algorithm $\approx O(\sqrt{N}) \rightarrow$ quadratic speed up compared with $O(N)$ count on a classical computer.
- Quantum advantage: superposition and $N = 2^n$ values of $f(x)$ evaluated in parallel
  - Operation count of $O(1)$?
  - Measurement returns only one $(x, f(x))$ value
  - Requires additional operations $\rightarrow O(\sqrt{N})$

nature COMMUNICATIONS

https://arxiv.org/pdf/1703.10535.pdf

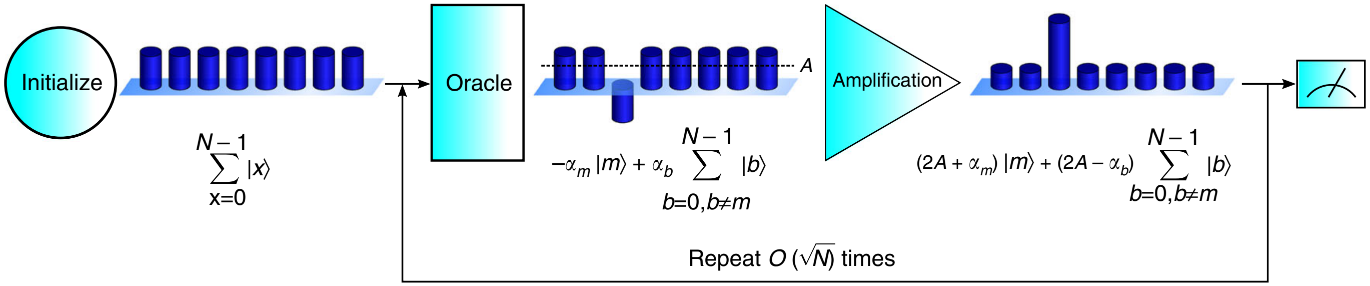
ARTICLE

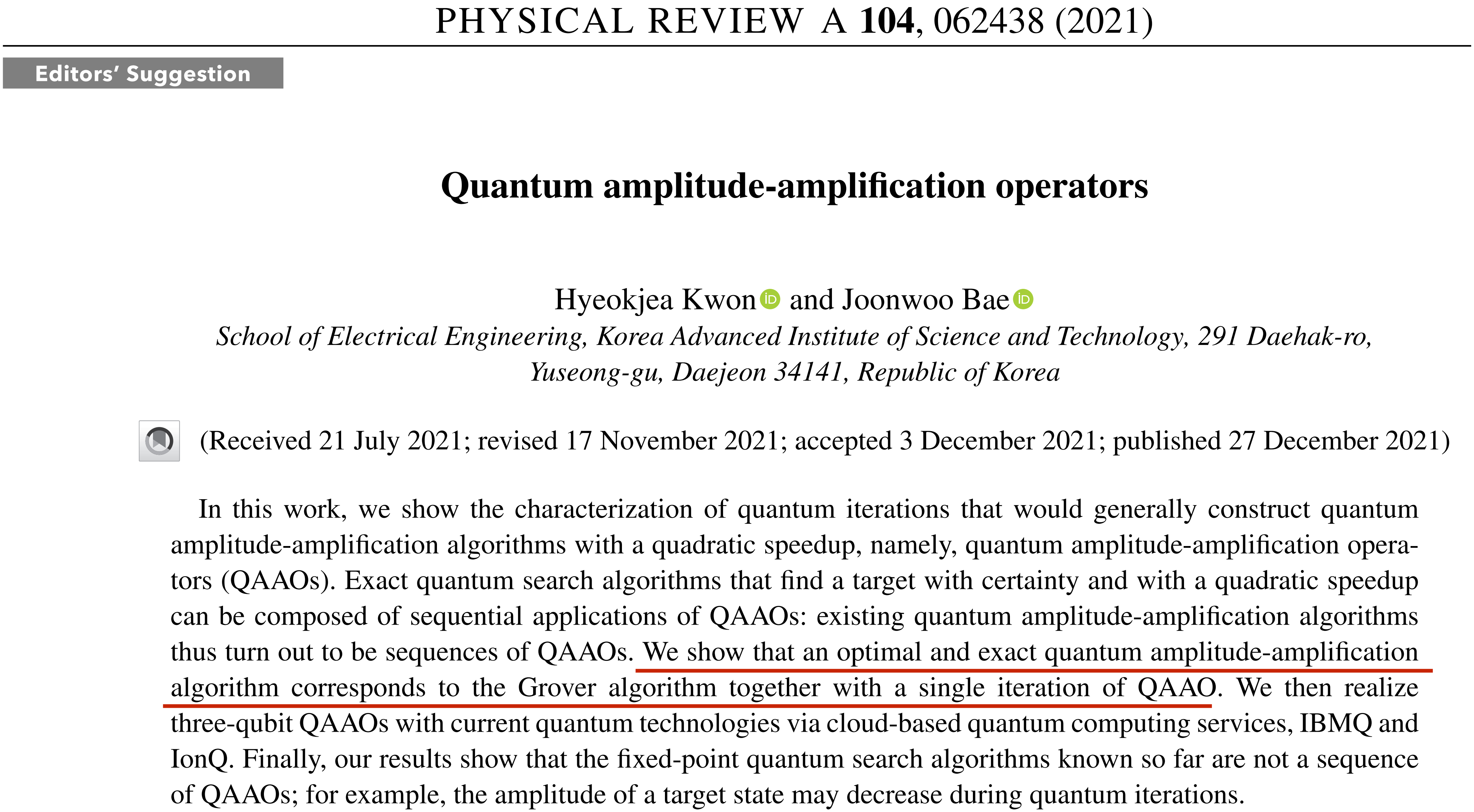
DOI: 10.1038/s41467-017-01904-7 OPEN

Complete 3-Qubit Grover search on a programmable quantum computer

C. Figgatt [1], D. Maslov [1,2], K.A. Landsman [1], N.M. Linke [1], S. Debnath [1] & C. Monroe [1,3]

The Grover quantum search algorithm is a hallmark application of a quantum computer with a well-known speedup over classical searches of an unsorted database. Here, we report results for a complete three-qubit Grover search algorithm using the scalable quantum computing technology of trapped atomic ions, with better-than-classical performance. Two methods of state marking are used for the oracles: a phase-flip method employed by other experimental demonstrations, and a Boolean method requiring an ancilla qubit that is directly equivalent to the state marking scheme required to perform a classical search. We also report the deterministic implementation of a Toffoli-4 gate, which is used along with Toffoli-3 gates to construct the algorithms; these gates have process fidelities of 70.5% and 89.6%, respectively.


Initialize
$\sum_{x=0}^{N-1} |x\rangle$
Oracle
A
$-\alpha_m |m\rangle + \alpha_b \sum_{b=0, b\neq m}^{N-1} |b\rangle$
Amplification
$(2A + \alpha_m) |m\rangle + (2A - \alpha_b) \sum_{b=0, b\neq m}^{N-1} |b\rangle$
Repeat $O(\sqrt{N})$ times



PHYSICAL REVIEW A **104**, 062438 (2021)

Editors' Suggestion

Quantum amplitude-amplification operators

Hyeokjea Kwon and Joonwoo Bae

School of Electrical Engineering, Korea Advanced Institute of Science and Technology, 291 Daehak-ro,
Yuseong-gu, Daejeon 34141, Republic of Korea

(Received 21 July 2021; revised 17 November 2021; accepted 3 December 2021; published 27 December 2021)

In this work, we show the characterization of quantum iterations that would generally construct quantum amplitude-amplification algorithms with a quadratic speedup, namely, quantum amplitude-amplification operators (QAAOs). Exact quantum search algorithms that find a target with certainty and with a quadratic speedup can be composed of sequential applications of QAAOs: existing quantum amplitude-amplification algorithms thus turn out to be sequences of QAAOs. We show that an optimal and exact quantum amplitude-amplification algorithm corresponds to the Grover algorithm together with a single iteration of QAAO. We then realize three-qubit QAAOs with current quantum technologies via cloud-based quantum computing services, IBMQ and IonQ. Finally, our results show that the fixed-point quantum search algorithms known so far are not a sequence of QAAOs; for example, the amplitude of a target state may decrease during quantum iterations.

https://arxiv.org/pdf/2105.09559.pdf

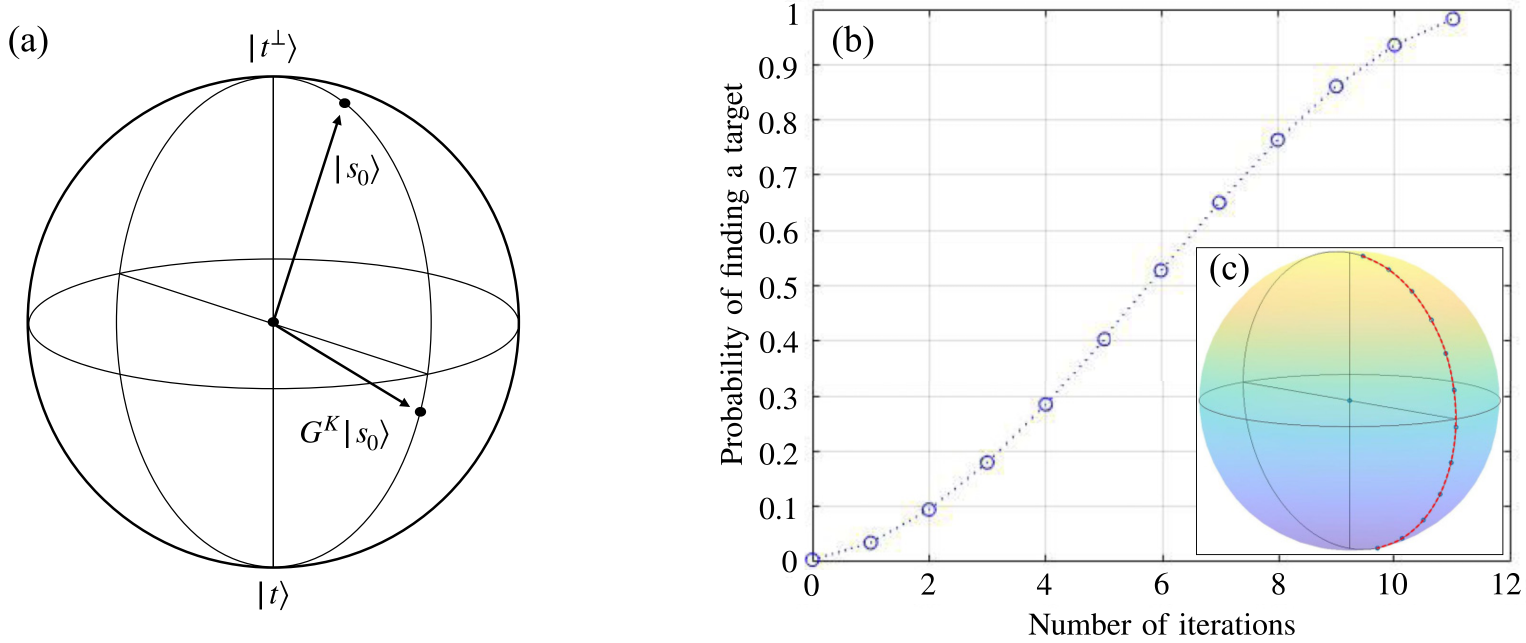


FIG. 1. (a) The Grover iteration corresponds to consecutive rotations in the space spanned by a target state $|t\rangle$ and its complement $|t^{\perp}\rangle$. (b) The probability of finding a target state is plotted in the case of eight qubits. The probability is monotonically increasing. (c) The path of an evolving state in the sphere is shown by Grover iterations from an initial to target states.

https://arxiv.org/pdf/2105.09559.pdf

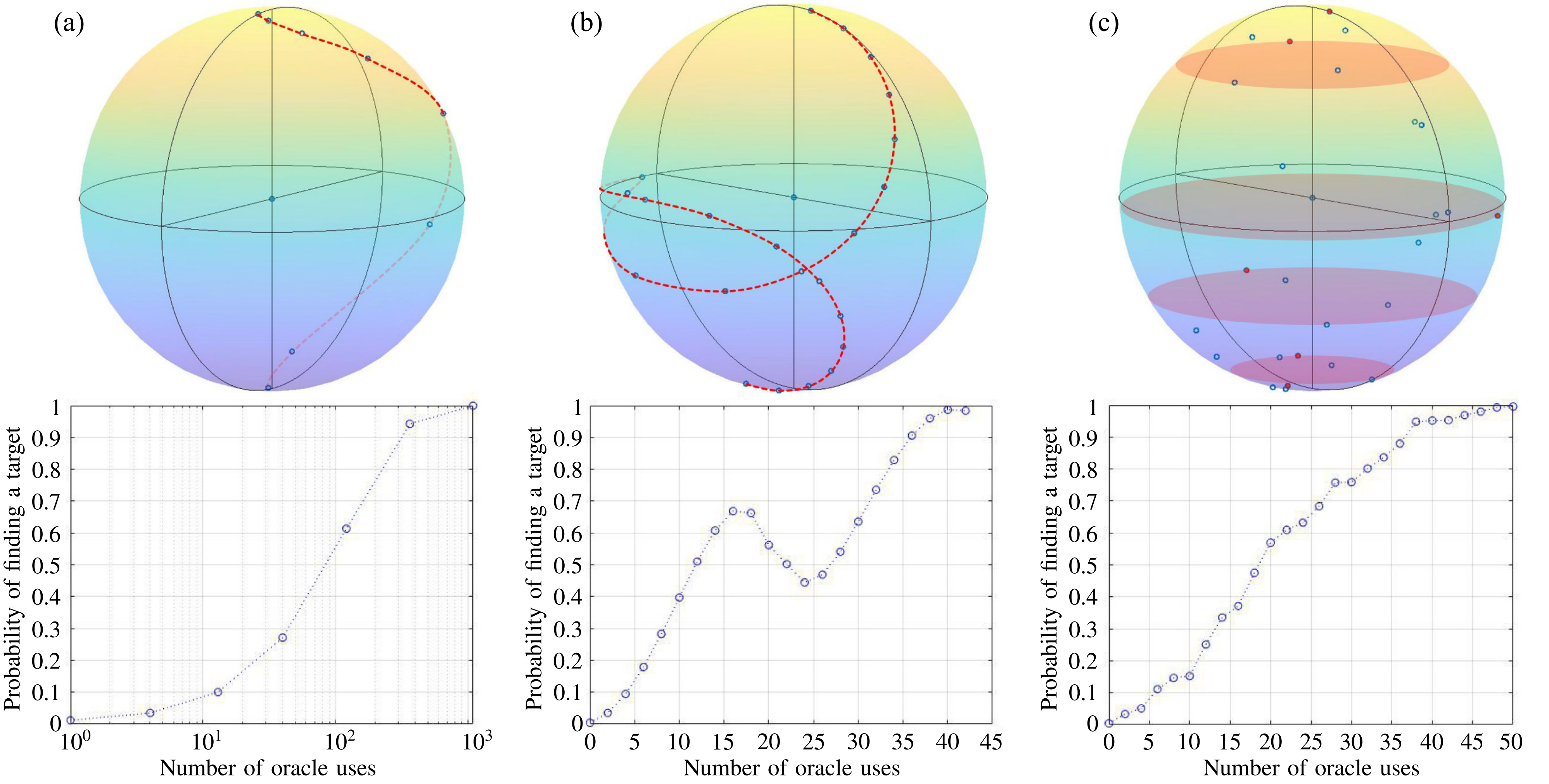


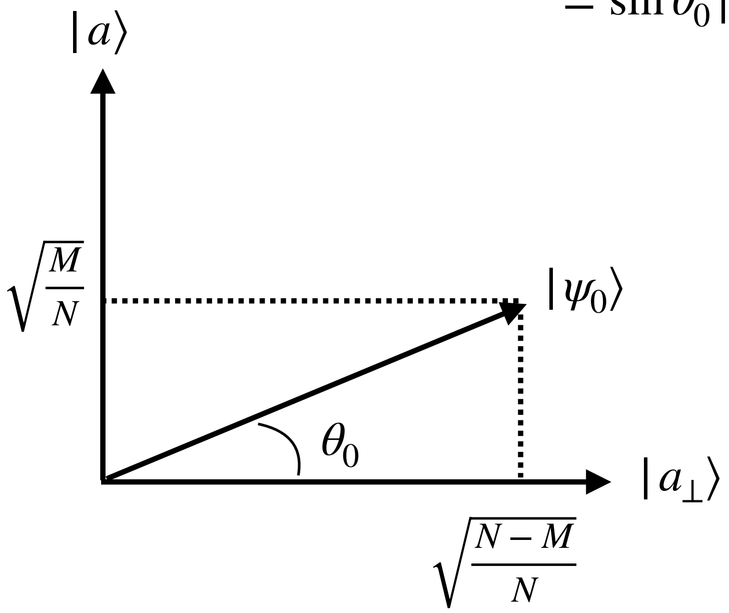
FIG. 4. Quantum amplitude amplification is performed in the case of eight qubits. The x axis shows the number of oracle uses, and the y axis shows the probability of finding a target state. (a) The $\pi/3$ algorithm is plotted [19]. The amplitude increases all the time until 10^3 oracle calls, without a quantum speedup. (b) A fixed-point quantum search with optimal query complexity is plotted [20]. The amplitude of the target state decreases in the meanwhile, and the oracle is called 45 times. (c) QAAOs are randomly generated and concatenated so that the amplitude keeps increasing until it reaches 1 after the oracle calls 50 times.

https://arxiv.org/pdf/2105.09559.pdf

Extension to more than one special value

- What if three are M solutions, $a_i, \quad i = 1, 2, \cdots, M$
- Superposition of all special states: $|a\rangle = \frac{1}{\sqrt{M}} \sum_{x \in \{a_i\}} |x\rangle$ $\quad \langle a|a\rangle = 1$ $\quad \langle a|a_\perp\rangle = 0$
- Uniform superposition of all other states: $|a_\perp\rangle = \frac{1}{\sqrt{N-M}} \sum_{x \notin \{a_i\}} |x\rangle$ $\quad \langle a_\perp|a_\perp\rangle = 1$
- Initial state: $|\psi_0\rangle = \sqrt{\frac{M}{N}}|a\rangle + \sqrt{\frac{N-M}{N}}|a_\perp\rangle$ $\quad \langle \psi_0|\psi_0\rangle = 1$

$$= \sin\theta_0|a\rangle + \cos\theta_0|a_\perp\rangle \qquad \sin\theta_0 = \langle a|\psi_0\rangle = \sqrt{\frac{M}{N}}$$



- $M \ll N, \; \theta \approx \frac{\pi}{2} \; \rightarrow \; m = \frac{\pi}{4}\sqrt{\frac{N}{M}}$

Quantum Counting

- What if we had no prior knowledge of M?
- Grover operator G rotates vectors in $|a\rangle - |a_\perp\rangle$ plane by angle $2\theta_0$

$$\sin\theta_0 = \sqrt{\frac{M}{N}}$$

$$G = \begin{pmatrix} \cos 2\theta_0 & -\sin 2\theta_0 \\ \sin 2\theta_0 & \cos 2\theta_0 \end{pmatrix} \longrightarrow \quad \text{eigenvalues} \quad e^{\pm 2i\theta_0}$$

$\longrightarrow$ Phase estimation

$\longrightarrow$ $\theta_0 \rightarrow M$

$\longrightarrow$ Quantum algorithm can tell us whether a special value exists at all, i.e., M=0.

Shor’s algorithm

Modular Exponentiation

- Taking powers of a number modulo some other number.

$$
\begin{aligned}
2^0 &\bmod 7 = 1 \bmod 7,\\
2^1 &\bmod 7 = 2 \bmod 7,\\
2^2 &\bmod 7 = 4 \bmod 7,\\
2^3 &\bmod 7 = 8 \bmod 7 = 1 \bmod 7,\\
2^4 &\bmod 7 = 16 \bmod 7 = 2 \bmod 7,\\
2^5 &\bmod 7 = 32 \bmod 7 = 4 \bmod 7,\\
2^6 &\bmod 7 = 64 \bmod 7 = 1 \bmod 7,\\
2^7 &\bmod 7 = 128 \bmod 7 = 2 \bmod 7,\\
2^8 &\bmod 7 = 256 \bmod 7 = 4 \bmod 7,\\
2^9 &\bmod 7 = 512 \bmod 7 = 1 \bmod 7,\\
&\vdots
\end{aligned}
$$

The period of order of the modular exponential = r = 3

$$f(x) = 2^x \pmod 7$$

$$
\begin{aligned}
f(x+r) &= 2^{x+r} \pmod 7\\
&= 2^x\, 2^r \pmod 7\\
&= 2^x \pmod 7 = f(x)
\end{aligned}
$$

Modular Exponentiation

$$
\begin{aligned}
&3^0 \bmod 10 = 1 \bmod 10,\\
&3^1 \bmod 10 = 3 \bmod 10,\\
&3^2 \bmod 10 = 9 \bmod 10,\\
&3^3 \bmod 10 = 27 \bmod 10 = 7 \bmod 10,\\
&3^4 \bmod 10 = 81 \bmod 10 = 1 \bmod 10,\\
&3^5 \bmod 10 = 243 \bmod 10 = 3 \bmod 10,\\
&3^6 \bmod 10 = 729 \bmod 10 = 9 \bmod 10,\\
&3^7 \bmod 10 = 2187 \bmod 10 = 7 \bmod 10,\\
&3^8 \bmod 10 = 6561 \bmod 10 = 1 \bmod 10,\\
&\qquad\vdots
\end{aligned}
$$

The period of order of the modular exponential = r = 4

$$f(x) = 3^x \pmod{10}$$

- Period finding or order finding plays an important role in number theory.
- Note the period r must be less than N, and so the challenge is to find the period for large N.

Single Modular Exponent

- Finding a single modular exponent is fast using the repeated squaring method. For example, say we want to find $91^{43} \pmod{131}$.
- we express the exponent in binary:

$$\begin{aligned} 43 &= 101011_2 \\ &= 1\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 \\ &= 1\cdot 32 + 0\cdot 16 + 1\cdot 8 + 0\cdot 4 + 1\cdot 2 + 1\cdot 1. \end{aligned}$$

$$\begin{aligned} 91^{43} \bmod 131 &= 91^{1\cdot 32+0\cdot 16+1\cdot 8+0\cdot 4+1\cdot 2+1\cdot 1} \bmod 131 \\ &= 91^{1\cdot 32}\,91^{0\cdot 16}\,91^{1\cdot 8}\,91^{0\cdot 4}\,91^{1\cdot 2}\,91^{1\cdot 1} \bmod 131 \\ &= \left(91^{32}\right)^1 \left(91^{16}\right)^0 \left(91^{8}\right)^1 \left(91^{4}\right)^0 \left(91^{2}\right)^1 \left(91^{1}\right)^1 \bmod 131 \end{aligned}$$

$$\begin{aligned} &91^1 \bmod 131 = 91 \bmod 131, \\ &91^2 \bmod 131 = 8281 \bmod 131 = 28 \bmod 131, \\ &91^4 \bmod 131 = (92^2)^2 \bmod 131 = 28^2 \bmod 131 = 784 \bmod 131 = 129 \bmod 131, \\ &91^8 \bmod 131 = (92^4)^2 \bmod 131 = 129^2 \bmod 131 = 16641 \bmod 131 = 4 \bmod 131, \\ &91^{16} \bmod 131 = (92^8)^2 \bmod 131 = 4^2 \bmod 131 = 16 \bmod 131, \\ &91^{32} \bmod 131 = (92^{16})^2 \bmod 131 = 16^2 \bmod 131 = 256 \bmod 131 = 125 \bmod 131. \end{aligned}$$

$$\begin{aligned} 91^{43} \bmod 131 &= (125)^1(16)^0(4)^1(129)^0(28)^1(91)^1 \bmod 131 \\ &= 125\cdot 4\cdot 28\cdot 91 \bmod 131 \\ &= 1\,274000 \bmod 131 \\ &= 25 \bmod 131. \end{aligned}$$

- Although calculating a single modular exponential using the previous repeated squares method is fast, finding the period is slow because, when N is large, we may need to calculate many individual modular exponentials before a pattern forms. There is no known efficient algorithm for period finding.

Shor’s algorithm: Period finding to factor an integer

- Example: take two large primes, p and q. Form the product $N = pq$. Goal is to find the two factors p and q, when only N is given. $\longrightarrow$ classically hard problem.
- For application in cryptography, p and q have around 600 digits (2000 bits)

(1) Choose a random integer a with no factor in common with N.

(Euclid algorithm can determine efficiently whether N and a have a common factor or not)

If they have a common factor (unlikely), we have found a factor of N and problem is solved.

(2) Compute $f(x) \equiv a^x \pmod{N}$

One can always find r such that $a^r \equiv 1 \bmod N$ for a and N, which are relatively primes (coprime). (may not be efficient, though.)

Euclid’s algorithm

- An efficient method for computing the greatest common divisor (GCD) of two integers (numbers).
- Suppose $a > b$ and $a = qb + r$, where q is a quotient and r is a remainder.
- The remainder theorem says $\gcd(a, b) = \gcd(b, r)$.
- Repeat until the remainder becomes 0.

$$\gcd(a, b) = \gcd(b, r) = \gcd(r, r')$$

$$a = qb + r\,, \qquad b = qr + r'$$

a=72, b=20, 72=2*30 + 12

b=20, r=12, 20=1*12+8

r=12, r’=8, gcd(12,8)=4

gcd(72, 20) = gcd(2*30+12, 20) = gcd(12, 20)

Period finding to factor an integer

(2) Compute $f(x) \equiv a^x \pmod{N}$

One can always find r such that $a^r \equiv 1 \bmod N$ for a and N, which are relatively primes. Then the function repeats

$f(x+r) \equiv a^{x+r} = a^x \pmod{N} = f(x)$, where r is the period (or order) of the function.

Take $N = pq = 91.$ (p=13 and q=7)

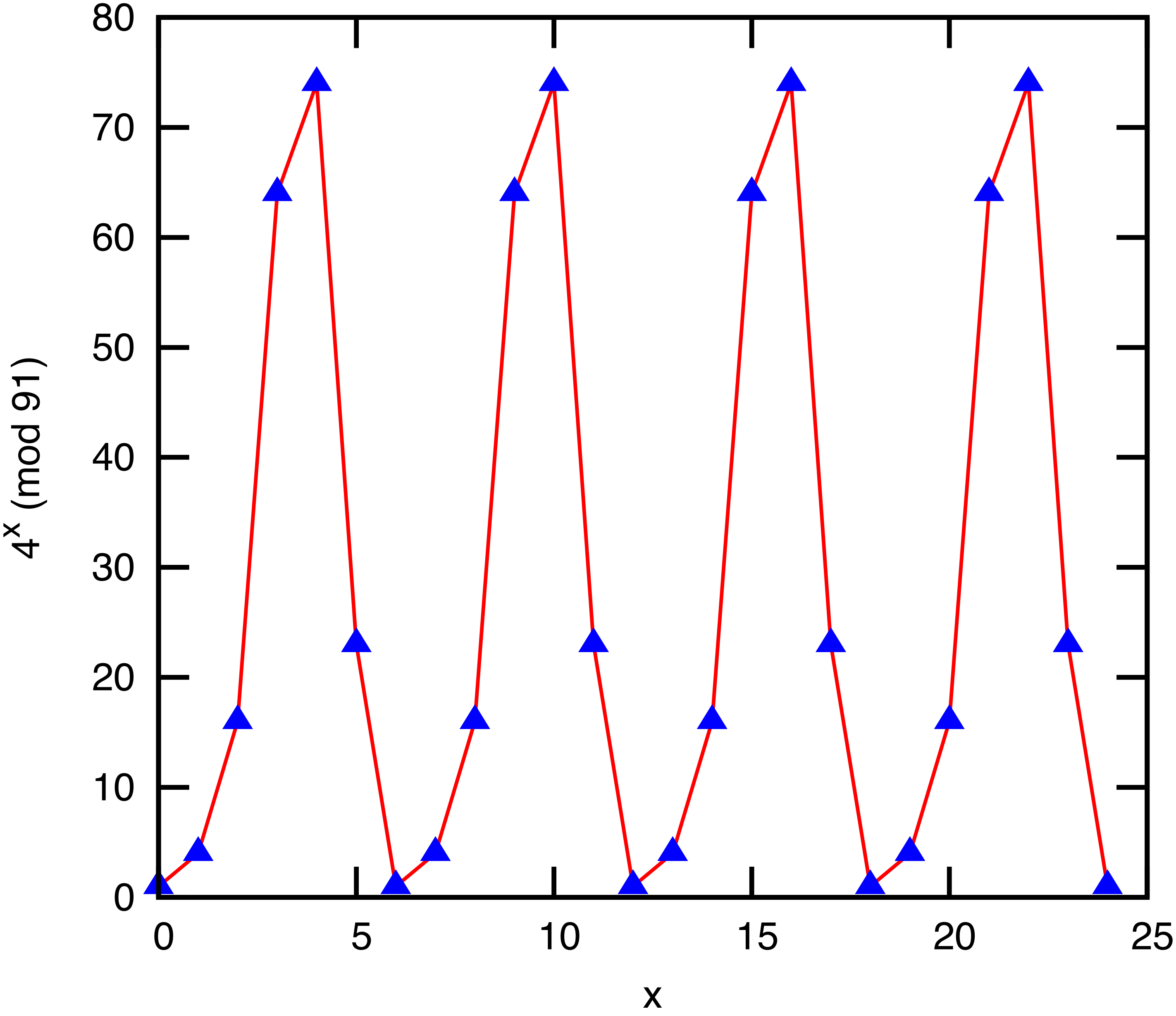
Take $a = 4$. No common factor with N=91. $\longrightarrow$ $f(x) = 4^x \pmod{91}$

$$\begin{aligned}
x &= 1, & a^x &= 4 \\
x &= 2, & a^x &= 16 \\
x &= 3, & a^x &= 64 \\
x &= 4, & a^x &= 64 \times 4 = 256 = 2 \times 91 + 74 = 74 \\
x &= 5, & a^x &= 74 \times 4 = 296 = 3 \times 91 + 23 = 23 \\
x &= 6, & a^x &= 23 \times 4 = 91 + 1 = 1 \pmod{91}
\end{aligned}$$

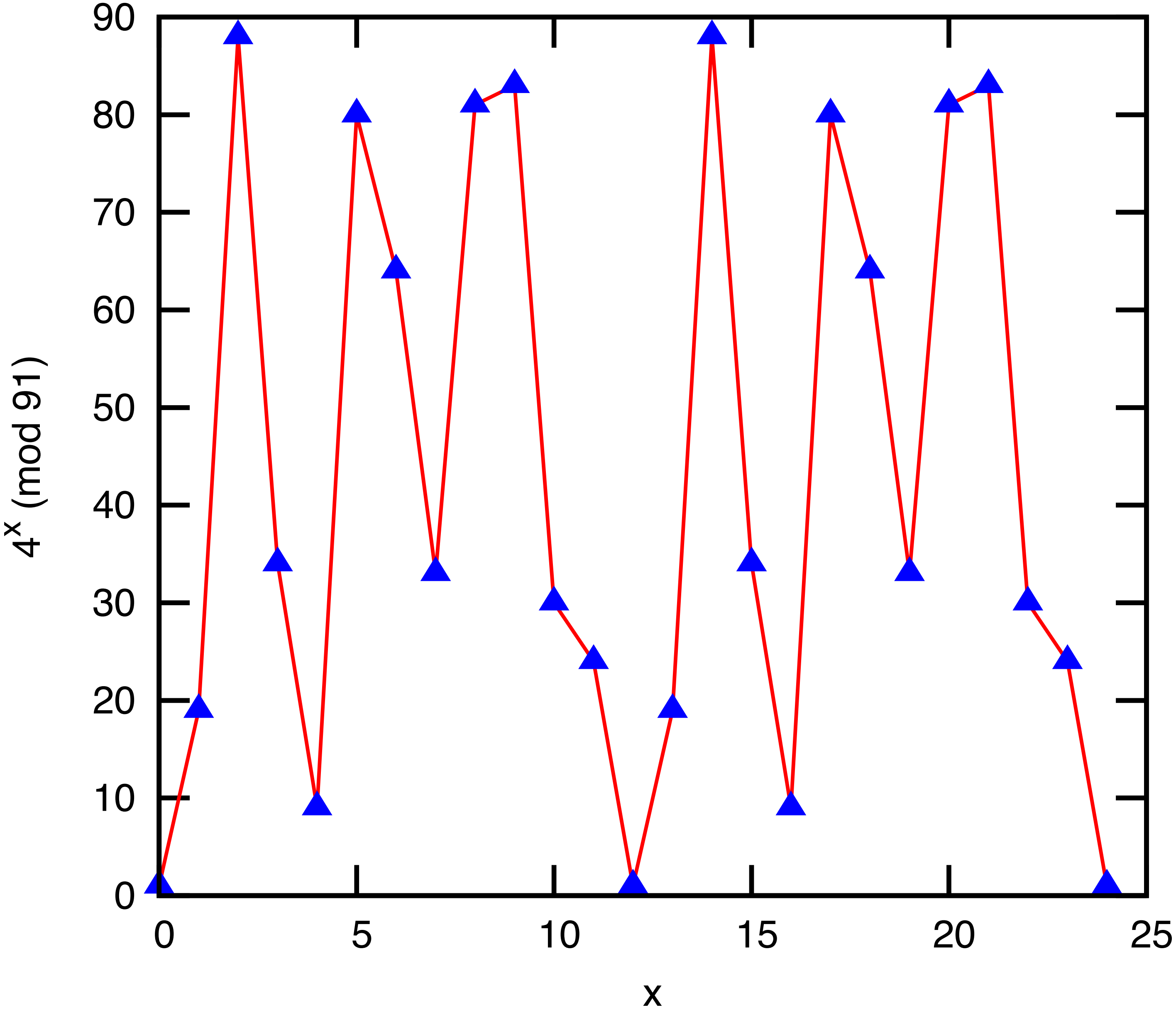
$\Rightarrow$

$a = 4$

$r = 6$



The function $f(x) \equiv 4^x \pmod{91}$. The period is seen by inspection to equal 6.



The function $f(x) \equiv 19^x \pmod{91}$. The period is seen by inspection to equal 12.

Period finding to factor an integer

$$f(x+r) \equiv a^{x+r} = a^x \pmod{N} = f(x)$$

$$f(x) \equiv a^x \pmod{N}$$

$$a^r \equiv 1 \bmod N$$

Two conditions:

(1) r must be even, so $r/2$ and $a^{r/2}$ are integers.

$$a^r = 1 \longrightarrow (a^{r/2}-1)(a^{r/2}+1) = 0$$

(2) $a^{r/2}-1$ and $a^{r/2}+1$ are not divisible by N but their product

$(a^{r/2}-1)(a^{r/2}+1)$ is divisible by N

so $a^{r/2}+1 = c \cdot p$ and $a^{r/2}-1 = c' \cdot q$

$$\begin{aligned} p &= gcd(a^{r/2}+1\,,N) \\ &= gcd(65\,,91) \\ &= gcd(26\,,65) \\ &= gcd(13\,,26) \\ &= gcd(0\,,13) \end{aligned}$$

$$\begin{aligned} q &= gcd(a^{r/2}-1\,,N) \\ &= gcd(63\,,91) \\ &= gcd(28\,,63) \\ &= gcd(7\,,28) \\ &= gcd(0\,,7) \end{aligned}$$

$N = pq = 91.$ (p=13 and q=7)

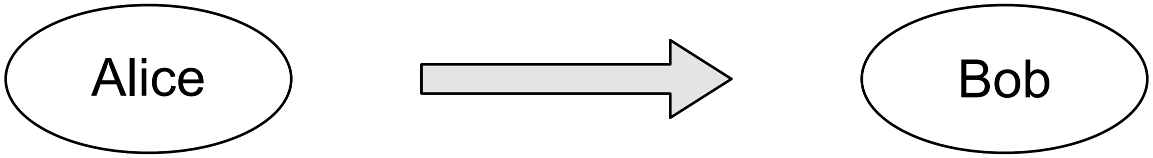
$$a^{r/2} = 4^{6/2} = 4^3 = 64$$

$$a = 4$$

$$r = 6$$

Factoring and RSA encryption

- Factoring is at the heart of RSA (Rivest-Shamir-Adleman) encryption.



(1) Bob picks two large prime numbers, p and q.

(2) Send to Alice their product $N = pq$ (on the public channel) not p and q separately.

- N: $\mathcal{O}(100)$ digit $\sim$ a few thousands bits

  Cannot be factored on a classical computer

- (ex) N is 400 digits (1000 bits)

$$\frac{1}{\ln N} = \frac{1}{\ln_{10}(400)} \approx 0.001$$

Probability of picking a prime number of N digits at random ~ 1/ln(N)

- Try to pick a number in $\mathcal{O}(100) \sim \mathcal{O}(1000)$ → may get a prime number
- can efficiently test if a number of prime or not,
- but no effective algorithm to do prime factorization of a composite number

Factoring and RSA encryption

(3) send a large “encoding number”, c which has no factors in common with $(p-1)(q-1) \rightarrow gcd(c,(p-1)(q-1)) = 1 \rightarrow$ coprime, relatively prime.

- probability that two random integers have no common factors is greater than 1/2.
- not difficult to find c.
- Bob knows $p\,, q\,, (p-1)(q-1),$ therefore can determine d such that

$$c\,d \equiv 1 \mod (p-1)(q-1)$$

- algorithm to compute d is extension of Euclid’s algorithm and efficient.
- the private key d is unique.
- Alice (and anyone on public) knows N and c (not $p\,, q\,, d$).
- The private key (known only to Bob) is p and q (and hence d).

Factoring and RSA encryption

(3) $a =$ original message that Alice wants to send.

Alice computes $b = a^c \pmod N$ = the encoded message,

(b is a large number) and send it to Bob (on the public channel).

(4) Bob computes $a = b^d \pmod N = a^{cd} \pmod N$

(can crack the encryption if d is known.) $c\,d \equiv 1 \bmod (p-1)(q-1)$

(ex) $p = 7,\ q = 13,\ N = 91$

$(p-1)(q-1) = 6 \times 12 = 72.$ Take c=11, no common factor with 72.

$c\,d = 11 \times 59 = 649 = 9 \times 72 + 1 \pmod{72} \rightarrow$ d=59.

A random message: $a = 51$ with $c = 11,\ d = 59,\ N = 91$

Encoded message: $b = a^c\ (\text{mod N}) = 51^{11} \pmod{91} = 25$

Decoded message: $b^d \pmod N = 25^{59} = 51 \ \rightarrow \ a$

Example: classical factoring algorithm

Classical algorithm: try to factor N=15.

(1) Pick any number y less than 15: y=13

(2) Calculate $f(n) = y^n \pmod{15}$ and find the period (order) r of $f(n)$

$$n = 1: \qquad f(1) = 13^1 = 13 \qquad\qquad 13 \pmod{15}$$

$$n = 2: \qquad f(2) = 13^2 = 169 = 15 \times 11 + 4 \qquad\qquad 4 \pmod{15}$$

$$n = 3: \qquad f(3) = 13^3 = (15 \times 11 + 4) \times 13 = 4 \times 13 = 52 = 15 \times 3 + 7 \qquad\qquad 7 \pmod{15}$$

$$n = 4: \qquad f(4) = 13^4 = 7 \times 13 = 91 = 15 \times 6 + 1 \qquad\qquad 1 \pmod{15}$$

Period: $y^r = 1 \pmod{N}, \qquad y^{r+x} = y^x \pmod{N}, \qquad f(r+x) = f(x) \quad \longrightarrow \quad r = 4$

(3) Period is even: $r = 2s$. $y^r = 1 \pmod{N}$ and $y^{2s} = 1 \pmod{N}$ $\rightarrow (y^s - 1)(y^s + 1) = 0 \bmod 15)$

$$\longrightarrow \; (y^s - 1)(y^s + 1) = kN \qquad \longrightarrow \; \text{gcd}(y^s \pm 1, N) \text{ will give facotrs of N}.$$

$$13^2 - 1 = 168, \qquad \gcd(168, 15) = \gcd(15 \times 11 + 3, 15) = \gcd(3, 15) = 3 \qquad 168 \times 170 = 1904 \times 3 \times 5$$

$$13^2 + 1 = 170, \qquad \gcd(170, 15) = \gcd(15 \times 11 + 5, 15) = \gcd(5, 15) = 5$$

We assumed $y^s + 1 \neq 0 \pmod{N}$. If $y^s = -1 \pmod{N}$, algorithm fails. Pick a different y.

⇨ Problem of factoring is the problem of finding even period $r = 2s$ for which $y^s + 1$ is not equal to $0 \pmod{N}$

Shor’s algorithm

- Efficient factoring algorithm → security
- 1st step in Shor’s factoring algorithm is to reduce the problem of factoring an integer N to the problem of order finding.

- Assume N is odd.
- Suppose we find a solution to $x^2 = 1 \pmod{\mathrm{N}}$ where $x \neq 1,\ x \neq N \pm 1$.

$$(x-1)(x+1) = 0 \pmod{\mathrm{N}}$$

  - N must have a common factor with $x+1$ or with $x-1$.
  - Can not be N, since $x \neq 1$ (ignore trivial solution), $x \neq N \pm 1$
  - A factor of N is either $\gcd(x+1,\ N)$ or $\gcd(x-1,\ N)$
  - Use Euclid algorithm to find a gcd.
- Therefore, If we can find x such that $x^2 = 1 \pmod{\mathrm{N}}\ (x \neq 1,\ x \neq N \pm 1)$ then we can factor N.

Euclid’s algorithm

- Suppose $a > b$ and $a = qb + r$, where q is a quotient and r is a remainder.
- The remainder theorem says $\gcd(a, b) = \gcd(b, r)$.
- Repeat until the remainder becomes 0

$$\gcd(a, b) = \gcd(b, r) = \gcd(r, r')$$

$$a = qb + r, \qquad b = qr + r'$$

Shor’s algorithm

- If we can find x such that $x^2 = 1 \pmod{\mathrm{N}}$ $(x \neq 1,\ x \neq N \pm 1)$ then we can factor N.
- Pick a random y, $1 \leq y \leq N-1$
  - If $\gcd(c,d) \neq 1$, we found a factor.
  - If $\gcd(c,d) = 1$, no common positive factors.
    - » y and N are coprimes or relatively prime or strangers.
    - » y is coprime with N.
- Probability that two integers m and n picked at random are relatively primes = $P(\,(m,n) = 1\,) = \left[\zeta(2)\right]^{-1} = \dfrac{6}{\pi^2} = 0.60792\cdots$
- Probability that three integers k, m and n picked at random are relatively primes = $P(\,(k,m,n) = 1\,) = \left[\zeta(3)\right]^{-1} = 0.83190\cdots$
- If $\gcd(y,N) = 1$, y and N are coprime.
- The order of y is the smallest integer such that $y^r = 1 \pmod{\mathrm{N}}$

Order and Modular Exponentiation

- The order (r) of y is the smallest integer such that $y^r = 1 \pmod{\mathrm{N}}$ for two relatively prime y and N.

- The group of numbers coprime to N forms a cyclic group (?).
- Every element can be written as $g^t \pmod{\mathrm{N}}$ for a generator g.

- If r is even, $x = y^{r/2}$ and $x^2 = y^r = 1 \pmod{\mathrm{N}}$, since N is odd.
- If the probability of a random coprime number y having an even order is high, we see that we have reduced the factoring problem to the problem of finding the order of a number. (See Nielsen and Chuang, Quantum Computation and Quantum Information for details).

N=5, The group of numbers coprime to N forms a group, {1, 2, 3, 4}

$g = 2,\ g^0 = 1,\ g^1 = 1,\ g^2 = 2,\ g^3 = 3,\ g^4 = 1,\ g^5 = 2,\ g^6 = 4$

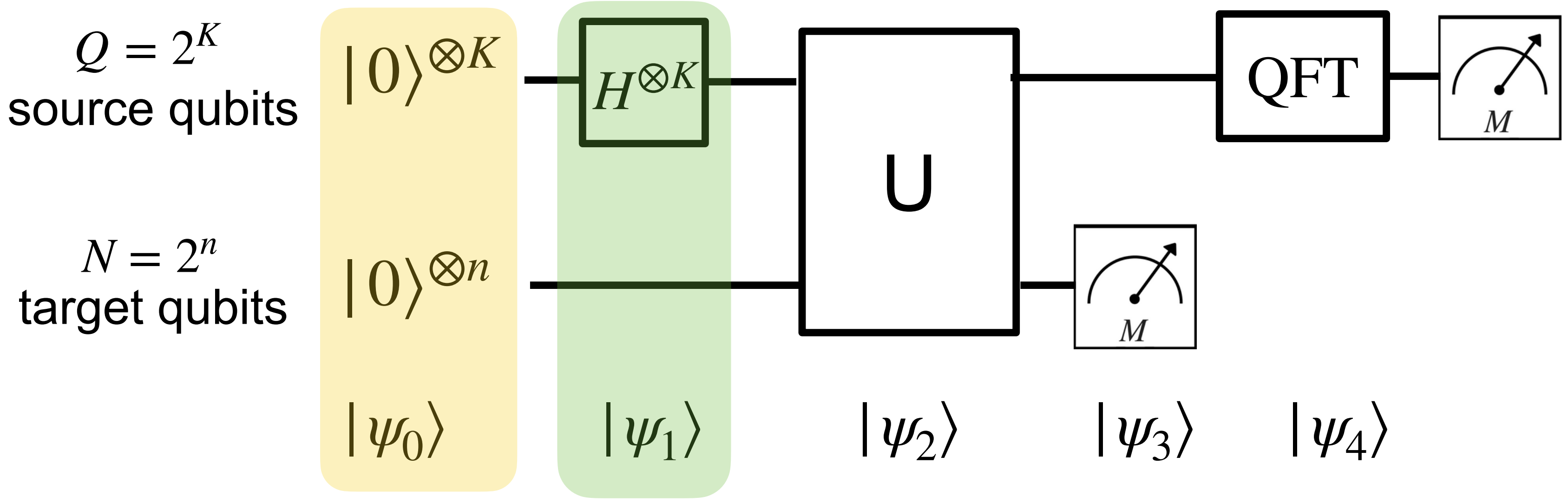
Order and Modular Exponentiation

- Modular exponentiation
  - For a modular exponentiation function $y = f(x) = a^x \pmod{\mathrm{N}}$, the order of the modular exponentiation (the order of $a \bmod \mathrm{N}$) is the smallest positive integer r such that $a^r = 1 \pmod{\mathrm{N}}$.

$$\begin{aligned} a^r &= kN + 1 \\ a^{r+1} &= kNa + a \\ a^{r+1} &= a \pmod{\mathrm{N}} \\ a^{r+x} &= a^x \pmod{\mathrm{N}} \end{aligned}$$

  - The r is the period of the function: $f(x+r) = f(x)$
- How do we find the order of a?
  - Calculate modular exponential function $f(x)$ for many values of x in parallel, and use QFT to detect the period in the sequence of function values.

Order Finding (Period Finding)



Begin with two registers:

$Q = 2^K$ for source qubits

$N = 2^n$ for target qubits

(1) Both registers are initialized to $|\psi_0\rangle = |0\rangle^{\otimes K} \otimes |0\rangle^{\otimes n}$

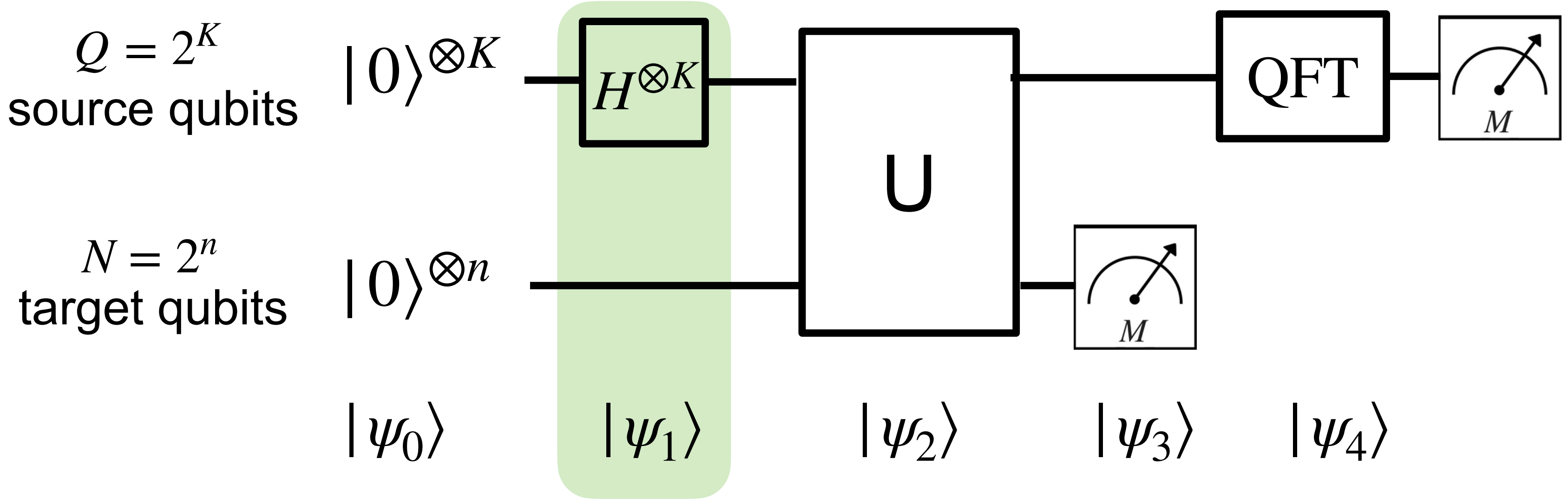
(2) Apply Hadamard to the source qubits:

$$H^{\otimes K}|x\rangle = \frac{1}{\sqrt{2^K}} \sum_{y=0}^{Q-1} (-1)^{x \cdot y} |y\rangle$$

$$|\psi_1\rangle = H^{\otimes K}|0\rangle = \frac{1}{\sqrt{2^K}} \sum_{y} |y\rangle$$

Superposition of all $Q = 2^K$ states

Order Finding (Period Finding)



(2) Apply Hadamard to the source qubits:

$$H^{\otimes K}|x\rangle = \frac{1}{\sqrt{2}^K}\sum_{y=0}^{Q-1}(-1)^{xy}|y\rangle$$

$$|\psi_1\rangle = H^{\otimes K}|0\rangle = \frac{1}{\sqrt{2}^K}\sum_y |y\rangle$$

Superposition of all $Q = 2^K$ states

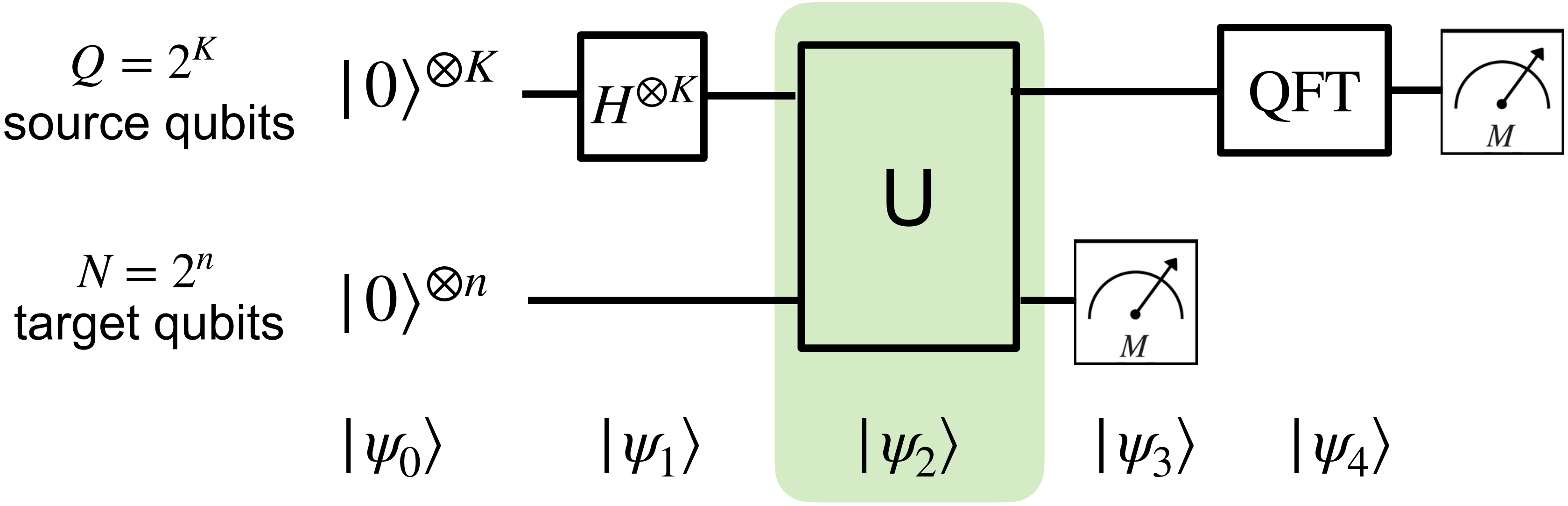
Equivalently apply QFT:

$$|q\rangle \longrightarrow \text{QFT}|q\rangle = \frac{1}{Q}\sum_{q'=0}^{Q-1}\exp\left(\frac{2\pi i q q'}{Q}\right)|q'\rangle$$

$$\text{Hadamard} \cong \text{multi} - \text{dimensional DFT}$$

$$|0\rangle \longrightarrow \text{QFT}|0\rangle = \frac{1}{Q}\sum_{q'=0}^{Q-1}|q'\rangle$$

Order Finding (Period Finding)



(3) Apply a quantum gate U_a that implements the modular exponentiation

$$q \longrightarrow f(q) = a^q \pmod{\mathrm{N}} \qquad \text{for a randomly chosen } a$$

$f(q)$ has r as its smallest period: $\quad f(q+r) = f(q) \qquad a^r = 1 \pmod{\mathrm{N}}$

$$a^{r+x} = 1 \pmod{\mathrm{N}}$$

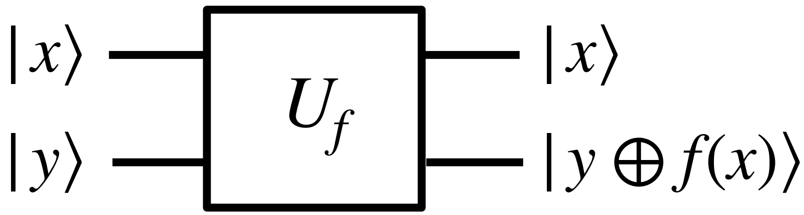
$f(q)$ is distinct on $[0, 1, 2,, \cdots, r-1]$ otherwise it would have a smaller period.

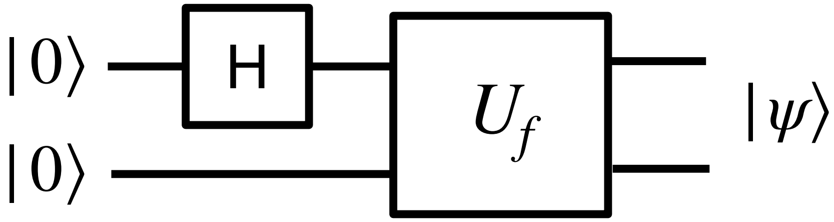
$$|\psi_2\rangle = U_a|\psi_1\rangle = U_a\left[\frac{1}{\sqrt{Q}}\sum_{q=0}^{Q-1}|q\rangle \otimes |0\rangle\right] = \frac{1}{\sqrt{Q}}\sum_{q=0}^{Q-1}|q\rangle \otimes |\underbrace{a^q \pmod{\mathrm{N}}}\rangle$$

There should be r different function values.

- $x \longrightarrow f(x)$ is not suitable because $f(x)$ is not unitary in general.
- $(x, y) \xrightarrow{U_f} (x, y \oplus f(x)) \xrightarrow{U_f} (x, y \oplus f(x) \oplus f(x)) = (x, y)$

$$U_f\Big(\,|x\rangle \otimes |y\rangle\Big) = |x\rangle \otimes |y \oplus f(x)\rangle$$

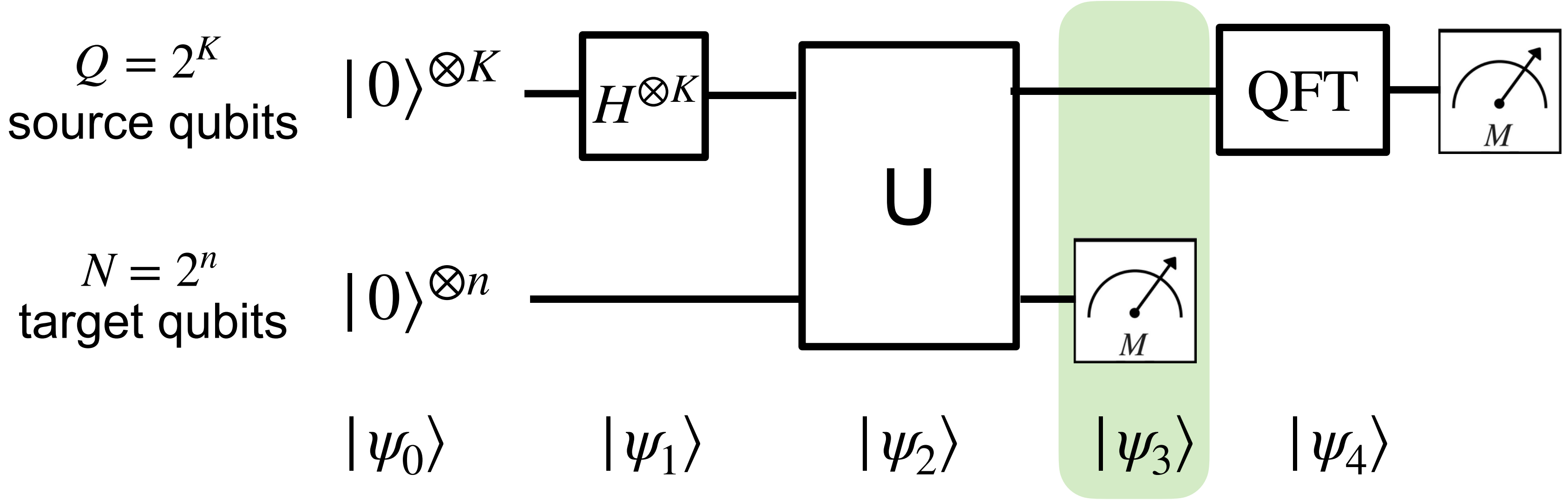




$$|\psi\rangle = \frac{1}{\sqrt{2}}\Big(\,|0\rangle \otimes |f(0)\rangle + |1\rangle \otimes |f(1)\rangle\Big) = \sum_{x=0,1} \frac{1}{\sqrt{2}} |x\rangle \otimes |f(x)\rangle$$

$$|q\rangle \longrightarrow \mathrm{QFT}\,|q\rangle = \frac{1}{Q}\sum_{q'=0}^{Q-1} \exp\Big(\frac{2\pi i q q'}{Q}\Big)\,|q'\rangle$$

Order Finding (Period Finding)



$Q = 2^K$ source qubits

$N = 2^n$ target qubits

$|0\rangle^{\otimes K}$

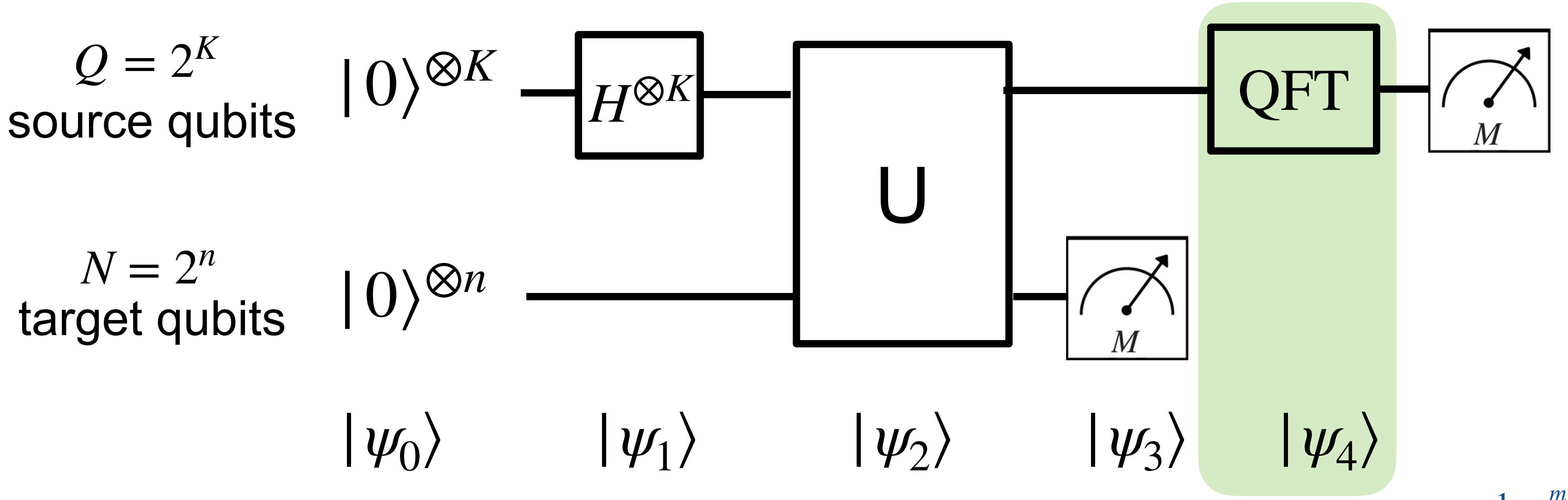
$|0\rangle^{\otimes n}$

(4) make a measurement on the second register. ⟶ must obtain a value which has to be one of r-distinct values of $f(q)$ ⟶ $f(q_0)$ ⟶ all superposed states of the 1st register inconsistent with the measured value must disappear.
⟶ for simplicity, assume $Q = mr$ ⟶ there are m-different values of q which have the same value of $f(q)$ ⟶ exactly $m = Q/r$ states of register 1 will contribute to the measured state of register 2.

$$|\psi_3\rangle = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |jr + q_0\rangle \otimes |f(q_0)\rangle$$

⟶ Periodic superposition of states in register 1 with period r (which is what we want to measure).
⟶ how do we measure r?

Order Finding (Period Finding)



(5) To measure r, use superposition before measurement!

$$|\psi_3\rangle = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |j\,r + q_0\rangle \otimes |f(q_0)\rangle$$

Define: $$g(a) = \begin{cases} \sqrt{\dfrac{r}{Q}} = \dfrac{1}{\sqrt{m}}, & a = j\,r + q_0 \\ 0, & \text{otherwise} \end{cases}$$

$(a - q_0$ is a multiple of $r)$

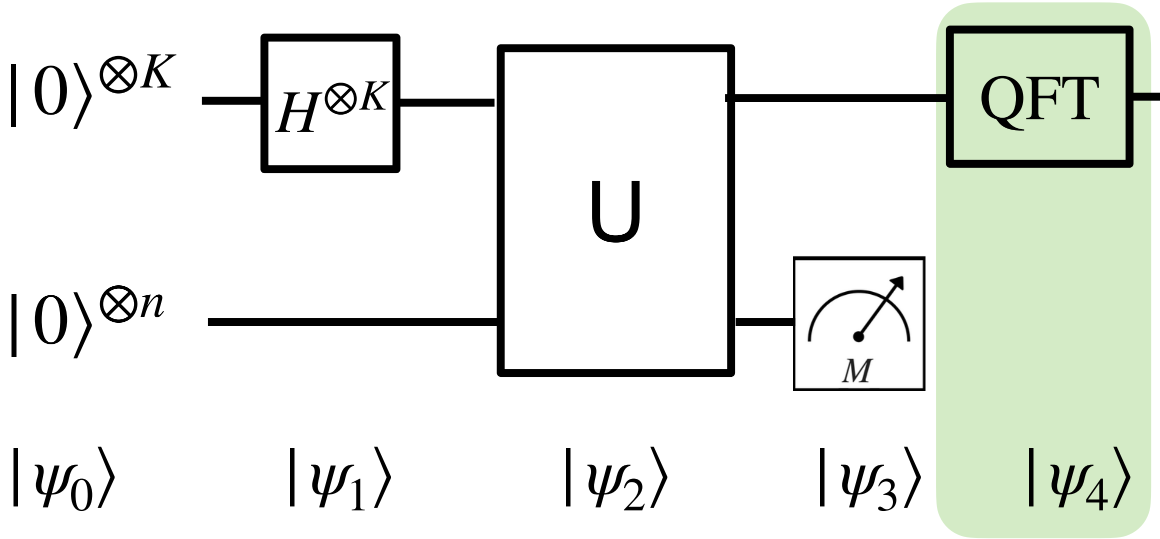
$$|\psi_3\rangle = \sum_{a=0}^{Q-1} g(a)\,|a\rangle$$

Consider source qubits only.

$$|\psi_4\rangle = \mathrm{QFT}\,|\psi_3\rangle = \frac{1}{\sqrt{Q}} \sum_c \sum_j g(jr + q_0)\,\exp\!\left(\frac{2\pi i(jr + q_0)c}{Q}\right)|c\rangle$$

$$= \frac{1}{\sqrt{Q}} \sum_c \left[\sum_j g(jr + q_0)\,\exp\!\left(\frac{2\pi i(jr)c}{Q}\right)\right] \exp\!\left(\frac{2\pi i q_0 c}{Q}\right)|c\rangle$$

Order Finding (Period Finding)



$$g(a) = \begin{cases} \sqrt{\frac{r}{Q}} = \frac{1}{\sqrt{m}}, & a = j\,r + q_0 \\ 0, & \text{otherwise} \end{cases}$$

$$|\psi_4\rangle = \frac{1}{\sqrt{Q}} \sum_c \left[\sum_j g(jr + q_0) \exp\left(\frac{2\pi i (jr)c}{Q}\right) \right] \exp\left(\frac{2\pi i q_0 c}{Q}\right) |c\rangle$$

If $\frac{rc}{Q}$ is not an integer, the sum goes to zero: $\sum_j g(jr + q_0) \exp\left(\frac{2\pi i (jr)c}{Q}\right) = 0.$

$\therefore \frac{rc}{Q} = k$ must be an integer, $\exp\left[2\pi i \left(\frac{rc}{Q}\right)\right] = 1.$

$$\sum_{j=0}^{Q-1} g(jr + q_0) = m \times \frac{1}{\sqrt{m}} = \sqrt{m}$$

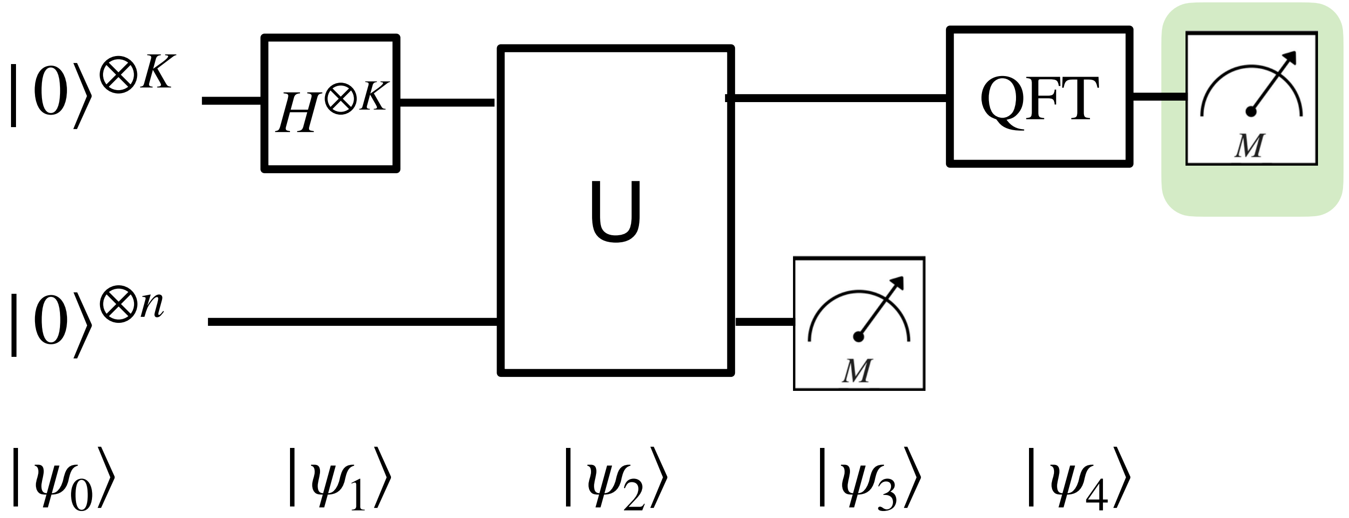
$$m = \frac{Q}{r}$$

$\downarrow$ m non-zero terms

$$|\psi_4\rangle = \sum_c \frac{1}{\sqrt{r}} \exp\left(\frac{2\pi i q_0 c}{Q}\right) |c\rangle = \sum_{k=0}^{r-1} \frac{1}{\sqrt{r}} \exp\left(\frac{2\pi i q_0 k}{r}\right) \left|k\frac{Q}{r}\right\rangle$$

$$c = k\frac{Q}{r}$$

Order Finding (Period Finding)



(6) We measure register 1.

$$|\psi_4\rangle = \sum_{k=0}^{r-1} \frac{1}{\sqrt{r}} \exp\left(\frac{2\pi i q_0 k}{r}\right) |k\frac{Q}{r}\rangle$$

Measurement gives a value of $c = kQ/r$ for a random k between 0 and $r-1$.

→ We know Q, c → $\frac{c}{Q} = \frac{k}{r}$

- If $\gcd(k,r) = 1$, k and r have no common factor.
- The ratio c/Q as an irreducible fraction and can read off values of k and r.
- k is chosen randomly by measurement.
- For a large r, the probability that $\gcd(k,r) = 1$ is greater than $1/\log(r)$.
- By repeating $\mathcal{O}(\log r) < \mathcal{O}(\log N)$ times, one can amplify the success probability of finding r.

Order Finding (Period Finding)

(7) Using order finding to factor a large number N

We have the order r of $a^x \pmod{N}$.

Check if r is even and $a^{r/2} \pmod{N} \neq -1$

$\rightarrow\ y \equiv a^{r/2}, y^2 = 1 \pmod{N} \ \rightarrow\ y^2 - 1 = (y+1)(y-1)$ is divisible by N.

N has **a common factor** with $y+1$ or $y-1$.

↳ must be one of gcds, $\gcd(N, y \pm 1)$

Use Euclid's algorithm for $\gcd(y, x)$.

Let us assume x, y : integers, $x > y$, and $z = \gcd(x, y)$.

$\rightarrow x, y$ and $x - y, x - 2y, \cdots$ are multiple of z.

$\rightarrow$ the remainder $r = x - ky < y$ is also a multiple of z.

$\rightarrow$ If $r = 0$, $z = y \rightarrow$ problem solved.

$$z = \gcd(x, y) = \gcd(y, r_1) = \gcd(r_1, r_2) = \gcd(r_2, r_3) = \cdots = \gcd(r_n, r_{n+1})$$

$r_1, r_2, \cdots$ are the successive remainders $r_i = r_{i-1} - k_i y$.

The last non-zero remainder is z.

Shor’s factoring algorithm

1. If N is even, return the factor 2 (check for other small prime factors such as 3, 5 …)
2. Check whether $N = a^b$ for $a > 1,\ b \geq 2$. If yes, return the factor a.
3. Randomly choose a between 1 and $N - 1$. If $z = \gcd(a, N) > 1$, return the factor z.
4. Use the order finding algorithm to find the order of $a \pmod N$. i.e., r such that $a^r = 1 \pmod N$.
5. If r is even and $a^{r/2} \neq -1 \pmod N$, then evaluate $\gcd(a^{r/2} \pm 1, N)$. If one of these is a non-trivial factor (other than 1), return that value as a factor. If not, go back to step 3 and repeat.

Shor’s factoring algorithm

- To factor an integer N, Shor's algorithm runs in polynomial time, meaning the time taken is polynomial in $\log N$, the size of the integer given as input. Specifically, it takes quantum gates of order $O\Big((\log N)^2(\log\log N)(\log\log\log N)\Big)$.
- This is significantly faster than the most efficient known classical factoring algorithm, the general number field sieve, which works in sub-exponential time: $O\Big(e^{1.9(\log N)^{1/3}(\log\log N)^{2/3}}\Big)$

Example: classical factoring algorithm

Classical algorithm: try to factor N=15.

(1) Pick any number y less than 15: y=13.   We want y and N are relatively primes.

(2) Calculate $f(n) = y^n \pmod{15}$ and find the period (order) r of $f(n)$

$$\begin{array}{lll}
n = 1: & f(1) = 13^1 = 13 & 13 \pmod{15} \\
n = 2: & f(2) = 13^2 = 169 = 15 \times 11 + 4 & 4 \pmod{15} \\
n = 3: & f(3) = 13^3 = (15 \times 11 + 4) \times 13 = 4 \times 13 = 52 = 15 \times 3 + 7 & 7 \pmod{15} \\
n = 4: & f(4) = 13^4 = 7 \times 13 = 91 = 15 \times 6 + 1 & 1 \pmod{15}
\end{array}$$

Period: $y^r = 1 \pmod{N}$, $y^{r+x} = y^x \pmod{N}$, $f(r+x) = f(x)$ $\longrightarrow$ $r = 4$

(3) Period is even: $r = 2s$. $y^r = 1 \pmod{N}$ and $y^{2s} = 1 \pmod{N}$ $\rightarrow (y^s - 1)(y^s + 1) = 0 \pmod{15}$

$$\longrightarrow\ (y^s - 1)(y^s + 1) = kN \qquad \longrightarrow\ \text{gcd}(y^s \pm 1, N) \text{ will give facotrs of N.}$$

$$13^2 - 1 = 168, \quad \gcd(168, 15) = \gcd(15 \times 11 + 3, 15) = \gcd(3, 15) = 3 \qquad 168 \times 170 = 1904 \times 3 \times 5$$

$$13^2 + 1 = 170, \quad \gcd(170, 15) = \gcd(15 \times 11 + 5, 15) = \gcd(5, 15) = 5$$

We assumed $y^s + 1 \neq 0 \pmod{N}$. If $y^s = -1 \pmod{N}$, algorithm fails. Pick a different y.

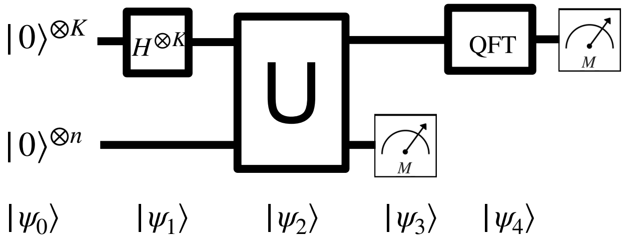
⇨ Problem of factoring is the problem of finding even period $r = 2s$ for which $y^s + 1$ is not equal to $0 \pmod{N}$

Example: Shor’s factoring algorithm

The idea of Shor’s algorithm

1. Evaluate all values of periodic function $y^n \pmod{N}$ simultaneously.
2. Adjust the probability amplitude to get a value of the period r with high probability.

(In some cases, 1/2 is good enough. The finite FT can transform cyclic behavior of the periodic function into the enhanced amplitude of some states.)



(1) Choose the number of qubits so $2^n > N$. $n = 4,\ 2^4 > N = 15.$

Pick y such that $\gcd(y, N) = 1$. Pick y=13.

(2) Initialize two quantum registers of n=4 qubits to $|0\rangle$ state.

$$|\psi\rangle = |0000\rangle \otimes |0000\rangle = |0\rangle^{\otimes 4} \otimes |0\rangle^{\otimes 4}$$

(3) Randomize 1st register. Make the superposition of states with all possible four-qubit basis states.

$$|0000\rangle \ \rightarrow \ \frac{1}{\sqrt{16}}\big(|0000\rangle + |0001\rangle + \cdots + |1111\rangle\big) = \frac{1}{\sqrt{16}}\sum_{k=0}^{15}|k\rangle$$

$|0\rangle = |0000\rangle$	$|4\rangle = |0100\rangle$	$|8\rangle = |1000\rangle$	$|12\rangle = |1011\rangle$
$|1\rangle = |0001\rangle$	$|5\rangle = |0101\rangle$	$|9\rangle = |1001\rangle$	$|13\rangle = |1100\rangle$
$|2\rangle = |0010\rangle$	$|6\rangle = |0110\rangle$	$|10\rangle = |1010\rangle$	$|14\rangle = |1101\rangle$
$|3\rangle = |0011\rangle$	$|7\rangle = |0111\rangle$	$|11\rangle = |1011\rangle$	$|15\rangle = |1111\rangle$

Example: Shor’s factoring algorithm

(4) Compute the function $f(k) = 13^k \pmod{15} = y^k \pmod{N}$ on the second register.

$$|\psi_2\rangle = \frac{1}{\sqrt{16}} \sum_{k=0}^{15} |k\rangle \otimes |f(k)\rangle = \frac{1}{\sqrt{16}} \Big(|0\rangle \otimes |f(0)\rangle + |1\rangle \otimes |f(1)\rangle + \cdots + |15\rangle \otimes |f(15)\rangle \Big)$$

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$f(k)$	1	13	4	7	1	13	4	7	1	13	4	7	1	13	4	7

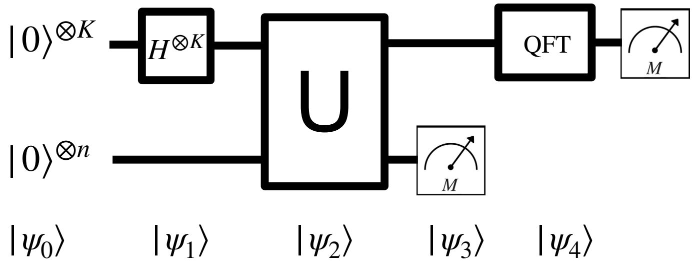
$$\begin{aligned}|\psi_2\rangle = \frac{1}{\sqrt{16}} \Big(& |0\rangle \otimes |1\rangle + |1\rangle \otimes |13\rangle + |2\rangle \otimes |4\rangle + |3\rangle \otimes |7\rangle \\ & + |4\rangle \otimes |1\rangle + |5\rangle \otimes |13\rangle + |6\rangle \otimes |4\rangle + |7\rangle \otimes |7\rangle \\ & + |8\rangle \otimes |1\rangle + |9\rangle \otimes |13\rangle + |10\rangle \otimes |4\rangle + |11\rangle \otimes |7\rangle \\ & + |12\rangle \otimes |1\rangle + |13\rangle \otimes |13\rangle + |14\rangle \otimes |4\rangle + |15\rangle \otimes |7\rangle \Big)\end{aligned}$$

← Computed in one operation

(5) Perform measurement on 2nd register. Superposition $|\psi_2\rangle$ will collapse and four terms survive.

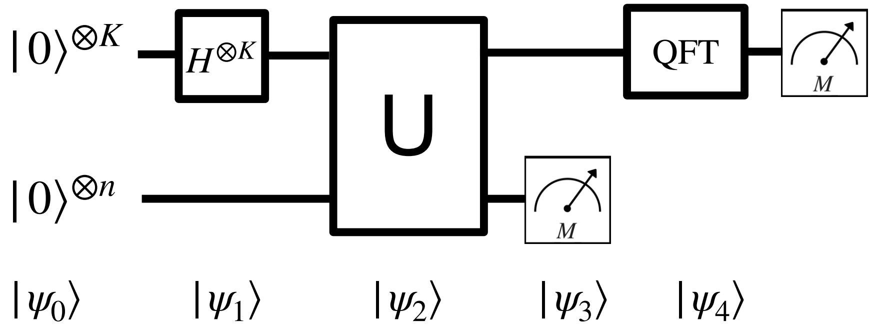
$$|\psi_3\rangle = \sqrt{\frac{4}{16}} \Big(|2\rangle \otimes |4\rangle + |6\rangle \otimes |4\rangle + |10\rangle \otimes |4\rangle + |14\rangle \otimes |4\rangle \Big)$$

Suppose we get $|4\rangle = |0100\rangle$.



Example: Shor's factoring algorithm

(4) Perform QFT: $|k\rangle \longrightarrow \frac{1}{\sqrt{16}} \sum_{u=0}^{15} \exp\left(\frac{2\pi i u k}{16}\right) |u\rangle$



$$|2\rangle \longrightarrow \frac{1}{\sqrt{16}} \sum_{u=0}^{15} \exp\left(\frac{2\pi i u 2}{16}\right) |u\rangle$$

$$|6\rangle \longrightarrow \frac{1}{\sqrt{16}} \sum_{u=0}^{15} \exp\left(\frac{2\pi i u 6}{16}\right) |u\rangle$$

$$|10\rangle \longrightarrow \frac{1}{\sqrt{16}} \sum_{u=0}^{15} \exp\left(\frac{2\pi i u 10}{16}\right) |u\rangle$$

$$|14\rangle \longrightarrow \frac{1}{\sqrt{16}} \sum_{u=0}^{15} \exp\left(\frac{2\pi i u 14}{16}\right) |u\rangle$$

$$|\psi_4\rangle = \sqrt{\frac{4}{16}}\sqrt{\frac{1}{16}} \sum_{u=0}^{15} |u\rangle \left[e^{2\pi i u 2/16} + e^{2\pi i u 6/16} + e^{2\pi i u 10/16} + e^{2\pi i u 14/16} \right] = \frac{1}{8} \sum_{u=0}^{15} |u\rangle A_u$$

Probability pf getting results $|u\rangle$ after 1st register is measured: $P_u = \left| \frac{1}{8} A_u \right|^2$ $P_0 = P_4 = P_8 = P_{12} = \frac{1}{4}$

We obtain states, $|0\rangle, |4\rangle, |8\rangle, |12\rangle$ with equal probability.
Probabilities are non-zero only if 16 divides ur where r is the period.

$$\frac{uc}{N} = k$$

Results of Shor's algorithm is one of states $|0\rangle, |4\rangle, |8\rangle, |12\rangle$ with equal probability, and the period satisfies $ur = 16\,k$.

Example: Shor’s factoring algorithm

Results of Shor’s algorithm is one of states $|0\rangle, |4\rangle, |8\rangle, |12\rangle$ with equal probability, and the period satisfies $ur = 16k$.

What is the probability to get the correct period from the first try?

$|u\rangle = |0\rangle$ Does not give any information. Rerun algorithm.

$|u\rangle = |4\rangle$ $4r = 16k$. Lowest $k = 1$. Period is $r = 4$

$|u\rangle = |8\rangle$ $8r = 16k$. $r = 2$ incorrect. Rerun algorithm.

$|u\rangle = |12\rangle$ $12r = 16k$. $k = 3$. $r = 4$

Algorithm has 1/2 probability of success from the 1st run.

- **Generalization**: Shor’s original paper contained
  - Quantum factoring algorithm
  - Algorithm for the discrete logarithm problem: generalization of Shor’s algorithm has been obtained for problems falling in the general class of hidden subgroup problems.

Basic Group Theory

- Group: a set G with an associative binary operation $\bullet$ satisfying
  - For any two elements g_1 and $g_2 \in G$, $g_1 \bullet g_2 \in G$ (closure)
  - $\exists\, e \in G$ such that $e \bullet g = g \bullet e = g$ for $\forall g \in G$ (identity)
  - $\exists\, g^{-1} \in G$ such that $g^{-1} \bullet g = g \bullet g^{-1} = e$ for $\forall g \in G$ (inverse)
  - $g_1 \bullet (g_2 \bullet g_3) = (g_1 \bullet g_2) \bullet g_3$ (associativity)
- $(\mathbb{Z}_n, + \pmod{n}) = \{0, 1, 2, \cdots, n-1\}$ forms a group under addition modulo n.
- Set of k-bit string, $\mathbb{Z}_2^k$ forms a group under bitwise addition modulo 2.
- For a prime p, $\{1, 2, \cdots, n-1\}$ forms a group $\mathbb{Z}_p^*$ under multiplication modulo p.
- $U(n)$: all unitary operators on an n-dimensional vector space V.
- Order = # of elements = $|G|$
- Finite group if $|G| < \infty$. Otherwise G is an infinite group.
- The order of an element g = the size of the subgroup of G that it generates.
  - The order of an element g must divide the order of the group.
- A set of generators of a group G is a subset of G such that all elements of G can be written as a finite product of the generators and their inverse.

Basic Group Theory

- A set of generators of a group is independent, if no generator can be written as a product of the other generators.
- A group is finitely generated if a finite group of generators exists.
- If a group can be generated by a single element, it is cyclic.
- The centralizer $Z(H)$ of a subgroup H of G is the set of elements of G that commute with all elements of H: $Z(H) = \{g \in G \,|\, gh = hg \text{ for all } h \in H\}$.
- For $H < G$, $Z(H)$ is a subgroup of G.
- If $g_1 \bullet g_2 = g_2 \bullet g_1$, G is Abelian or commutative.
- Every finite Abelian group is isomorphic to a product of one or more cyclic groups, $\mathbb{Z}_n$.
- If $n = pq$ and p and q are relatively prime, $\mathbb{Z}_n \cong \mathbb{Z}_p \times \mathbb{Z}_q$, $\mathbb{Z}_n$ is isomorphic to $\mathbb{Z}_p \times \mathbb{Z}_q$.
- Any Abelian group A has the unique decomposition (up to ordering of factors) into cyclic groups of prime power order: $A \cong \mathbb{Z}_{c_1} \times \mathbb{Z}_{c_2} \times \cdots \times \mathbb{Z}_{c_K}$
- $|A| = \prod_i c_i$ (prime factorization) where $c_i = p_i^{s_i}$ and p_i are distinct primes
- Product group $G \times H$ with operations $\bullet$ and $\circ$ = $\{(g, h) \,|\, g \in G,\ h \in H\}$ with $(g_1, h_1) \star (g_2, h_2) = (g_1 \bullet g_2, h_1 \circ h_2)$.

Discrete Logarithm Problem

- All standard public key encryption system and digital signature schemes are based on either factoring or discrete logarithm problem.
- $\mathbb{Z}_p^*$: group of integers $\{1, 2, \cdots, p-1\}$ under multiplication modulo p.
  - b: generator of $\mathbb{Z}_p^*$ (any b relatively prime to $p-1$ will work)
  - The discrete logarithm of $y \in \mathbb{Z}_p^*$ with respect to base b is the element $x \in \mathbb{Z}_p^*$ such that $b^x = y \pmod{p}$.
- Discrete logarithm problem: Given a prime p, a base $b \in \mathbb{Z}_p^*$ and an arbitrary element $y \in \mathbb{Z}_p^*$, find an $x \in \mathbb{Z}_p^*$ such that $b^x = y \pmod{p}$
  - Find the discrete logarithm of $y \in \mathbb{Z}_p^*$ with respect to base b such that $b^x = y \pmod{p}$
  - For a large p, this problem is computationally difficult to solve.
  - It is a special case of Abelian hidden subgroup problem.
  - Can be generalized to arbitrary finite cyclic groups.

Hidden Subgroup Problem

- Hidden subgroup problem: Let G be a group. Suppose a subgroup $H < G$ is implicitly defined by a function f on G in that f is constant and distinct on every coset of H. Find a set of generators of H.
    - Aim is to find an algorithm that computes a set of generators for H in $\mathcal{O}((\log|G|)^k)$ steps for some integer k.
- Finite Abelian hidden subgroup problem: Let G be a finite Abelian group with cyclic decomposition $G = \mathbb{Z}_{n_0} \times \cdots \times \mathbb{Z}_{n_L}$. Suppose G contains a subgroup $H < G$ that is implicitly defined by a function f on G that is constant and distinct on every coset of H. Find a set of generators.
- Period finding as a hidden subgroup problem: f is a periodic function on $\mathbb{Z}_N$ with period r that divides N. The subgroup $H < \mathbb{Z}_N$ generated by r is the hidden subgroup. Once a generator h for H has been found, the period r can be found by taking the greatest common divisor of h and N.

- H is a subgroup of G. The left coset is defined as $gH = \{gh \mid h \in H, \text{ for all } g \in G\}$

Hidden Subgroup Problem

- The discrete logarithm problem as a hidden subgroup problem
- For a given group $G = \mathbb{Z}_p^*$ where p is a prime and base $b \in G$ and an arbitrary $y \in G$, find $x \in G$ such that $y = b^x \pmod{p}$.
  - Consider $f: \ G \times G \ \longrightarrow \ G$ where $f(g, h) = b^{-g} y^h$.
  - The set of elements satisfying $f(g, h) = 1$ is the hidden subgroup H of $G \times G$, consisting of tuples of the form (mx, m).
  - From any generator of H, the element $(x, 1)$ can be computed.
  - Therefore solving the hidden subgroup problem yields x, which is the solution of the discrete logarithm problem.

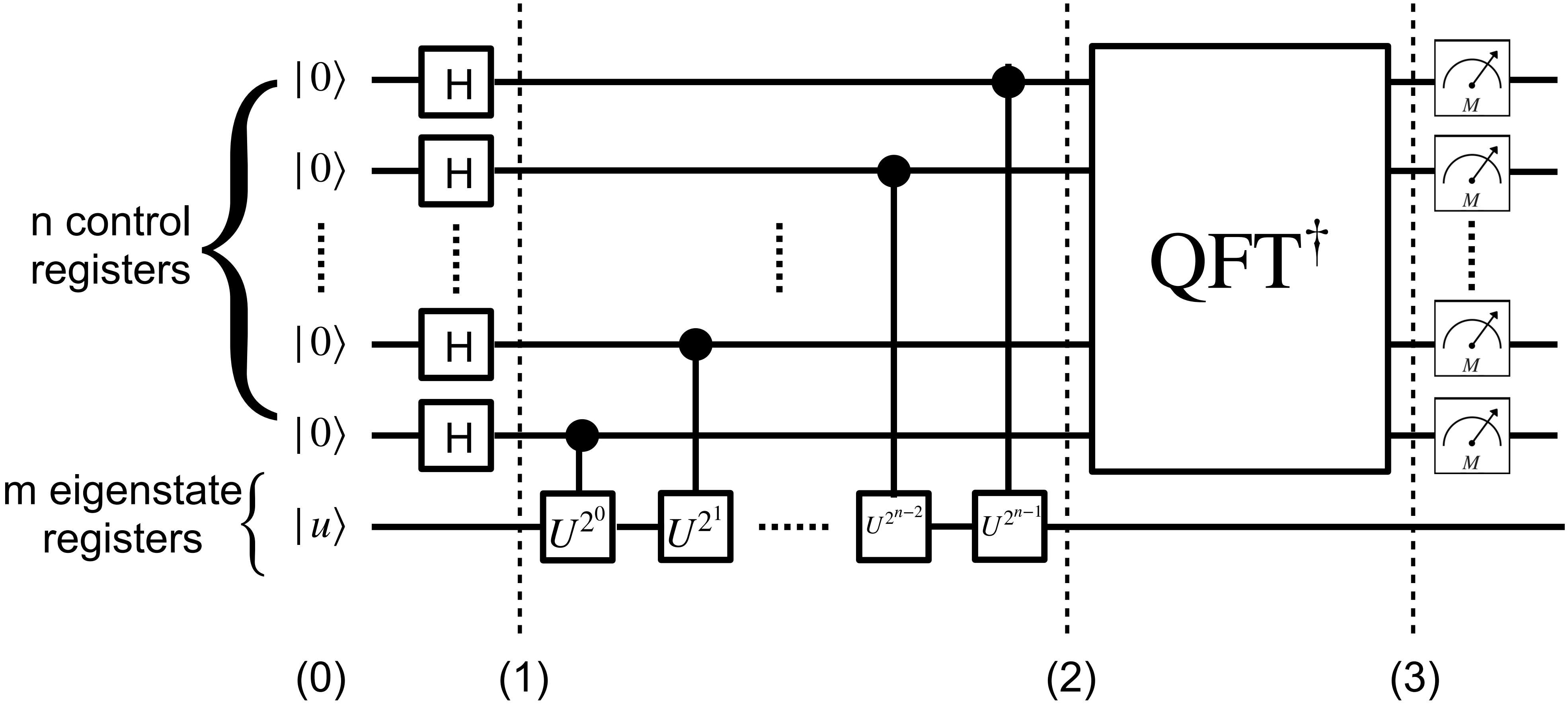
Quantum Phase Estimation and Finding Eigenvalues

Quantum Phase Estimation and Finding Eigenvalues

- Good example of phase kickback and use of QFT
- Unitary operator $U: \; U|u\rangle = e^{i\phi}|u\rangle\,, \qquad 0 \leq \phi < 2\pi$
- How to find eigenvalue? = How to measure the phase?
- How to find ϕ to a given level of precision?
- Find the best n-bit estimate of the phase ϕ
- Given a unitary matrix U and one of its eigenvectors $|u\rangle$, find or estimate its eigenvalue.

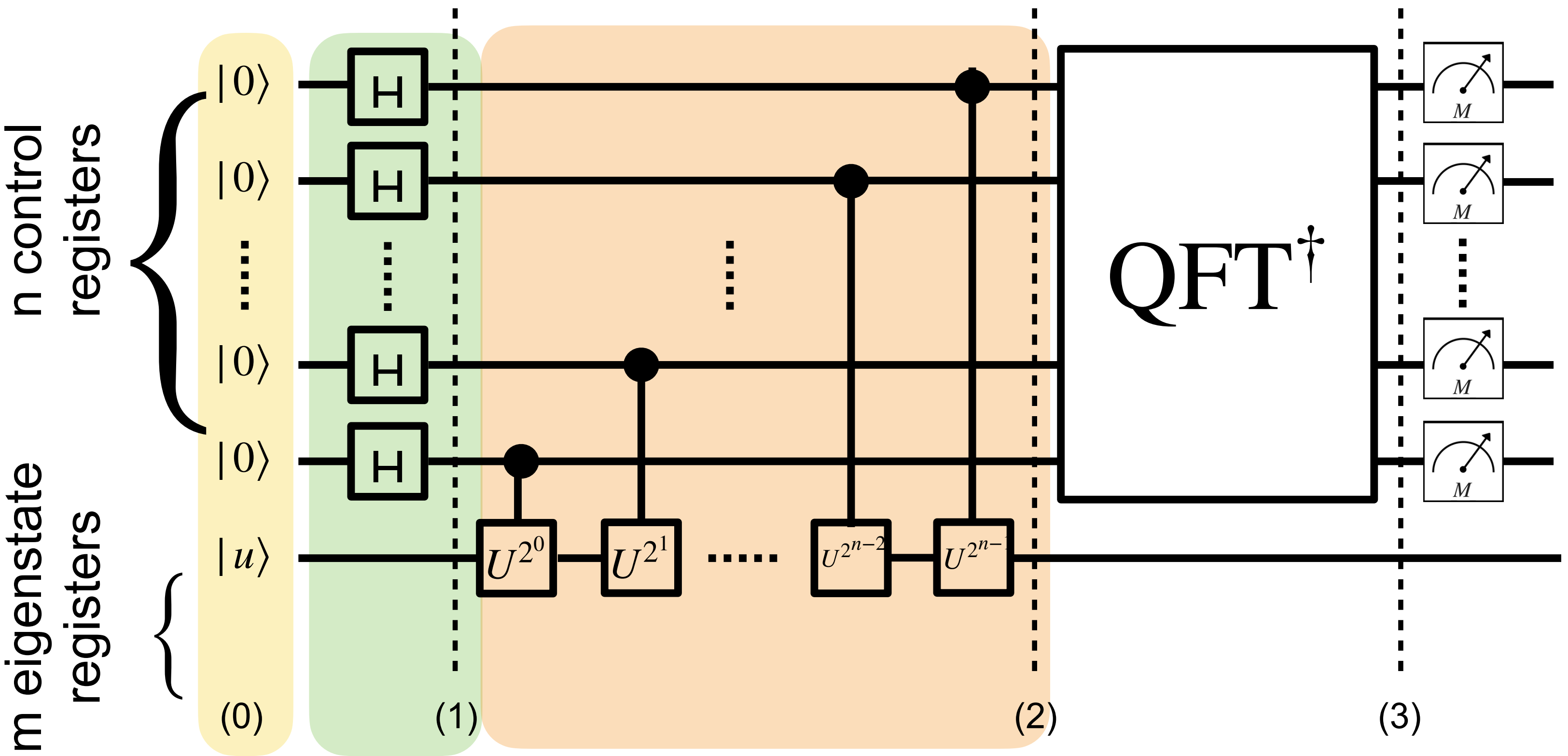
$$U^{2j}|u\rangle = \left(e^{i\phi}\right)^{2^j}|u\rangle = e^{i\phi\,2^j}|u\rangle$$

Quantum Circuit for QPE



$$\mathrm{QPE} \; = H + \text{controlled} - U^{2^j} + \mathrm{QFT}^\dagger$$

Quantum Circuit for QPE



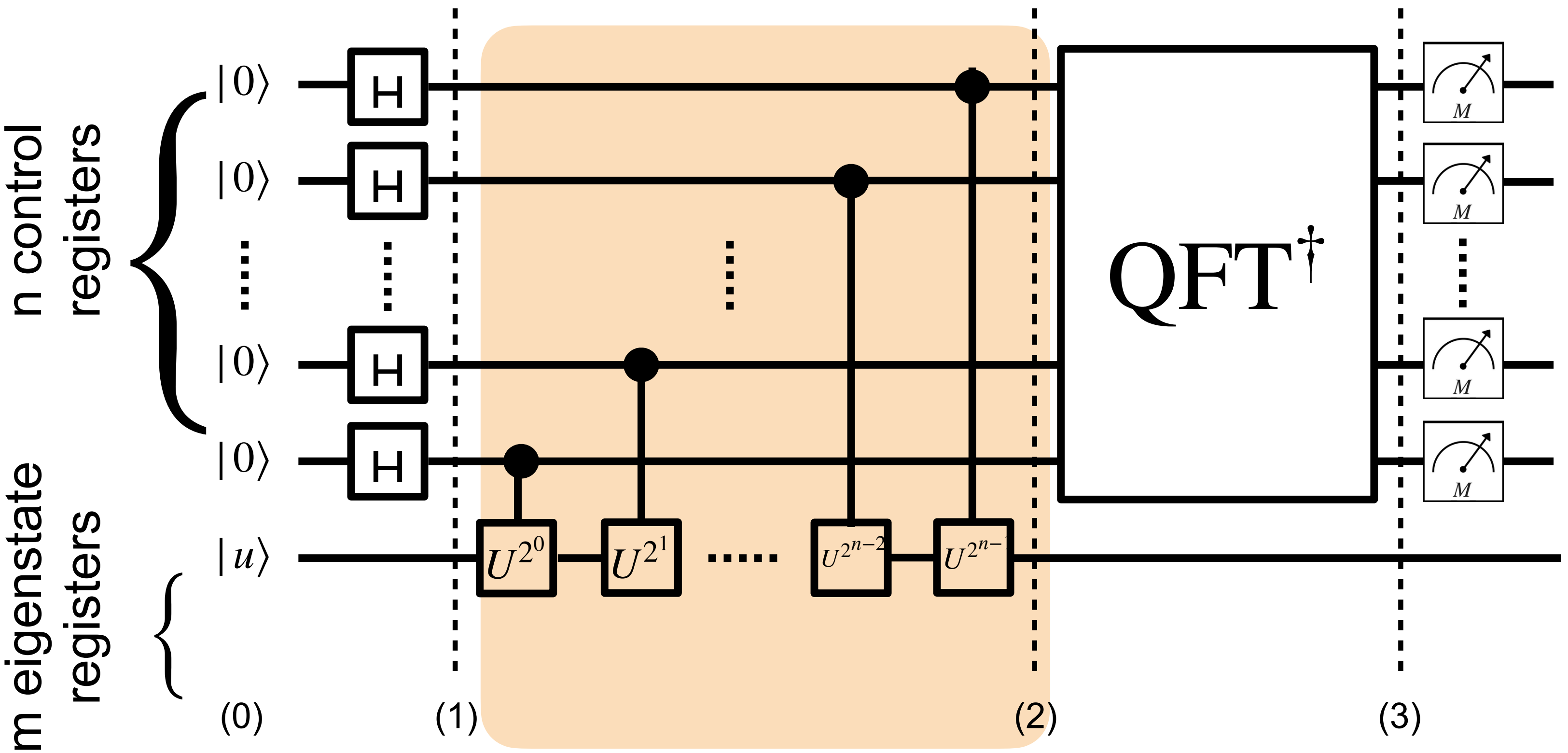
$$\mathrm{QPE} \;= H + \text{controlled} - U^{2^j} + \mathrm{QFT}^\dagger$$

$$|\psi_0\rangle = |0\rangle^{\otimes n} \otimes |u\rangle$$

$$|\psi_1\rangle = \Big(H|0\rangle\Big)^{\otimes n} \otimes |u\rangle = \frac{1}{\sqrt{2}^n}\Big(|0\rangle + |1\rangle\Big)^{\otimes n} \otimes |u\rangle$$

$$|\psi_2\rangle = \prod_{j=0}^{n-1} \mathrm{CU}^{2^j} \frac{1}{\sqrt{2}^n}\Big(|0\rangle + |1\rangle\Big)^{\otimes n} \otimes |u\rangle$$

Quantum Circuit for QPE



$$|\psi_2\rangle = \prod_{j=0}^{n-1} \mathrm{CU}^{2^j} \frac{1}{\sqrt{2^n}} \Big(|0\rangle + |1\rangle \Big)^{\otimes n} \otimes |u\rangle$$

$$\frac{1}{\sqrt{2}} \Big(|0\rangle + |1\rangle \Big) \otimes |u\rangle \xrightarrow{\mathrm{CU}^{2^j}} \frac{1}{\sqrt{2}} \Big(|0\rangle \otimes |u\rangle + U^{2^j} |1\rangle \otimes |u\rangle \Big)$$

$$= \frac{1}{\sqrt{2}} \Big(|0\rangle + e^{i\phi\, 2^j} |1\rangle \Big) \otimes |u\rangle$$

Quantum Circuit for QPE

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}}\left(|0\rangle + e^{i\phi\, 2^{n-1}}|1\rangle\right)\left(|0\rangle + e^{i\phi\, 2^{n-2}}|1\rangle\right)\cdots\left(|0\rangle + e^{i2\phi}|1\rangle\right)\left(|0\rangle + e^{i\phi}|1\rangle\right)\otimes|u\rangle$$

$$= \frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}\underbrace{e^{i\phi y}}\,|y\rangle\otimes|u\rangle$$

Phase kick-back: phase factor $e^{i\phi y}$ has been propagated back from the second eigenstate register to the first control register

$$\mathrm{QFT}\,|a\rangle = \frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{2\pi i a k/2^n}|k\rangle \longrightarrow \frac{2\pi i a}{2^n} = i\phi \longrightarrow \boxed{\phi = 2\pi\left(\frac{a}{2^n}+\delta\right)}$$

$$a = a_{n-1}a_{n-2}\cdots a_0$$

- $\dfrac{2\pi a}{2^n}$ is the best n-bit binary approximation of ϕ .
- $0 \le |\delta| \le \dfrac{1}{2^{n+1}}$ is the associated error.

$$F|j\rangle = \frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1} w^{jk}|k\rangle$$

$$w = \exp\left(\frac{2\pi i}{2^n}\right)$$

$$\mathrm{QFT}^{-1}|y\rangle = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1} e^{-2\pi i x y)/2^n}|x\rangle$$

$$|\psi_3\rangle = \mathrm{QFT}^{-1}|\psi_2\rangle = \frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{y=0}^{2^n-1} e^{2\pi i(a-x)y/2^n}e^{2\pi i\delta y}|x\rangle\otimes|u\rangle$$

Operate only n control register.

Quantum Circuit for QPE

$$|\psi_3\rangle = \mathrm{QFT}^{-1}|\psi_2\rangle = \frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{y=0}^{2^n-1} e^{2\pi i(a-x)y/2^n} e^{2\pi i\delta y}|x\rangle\otimes|u\rangle$$

Operate only n control register.

(1) If $\delta = 0$, $\frac{1}{2^n}\sum_{y=0}^{2^n-1}\exp\left(\frac{2\pi i(a-x)y}{2^n}\right) = \delta_{ax} \longrightarrow |\psi_3\rangle = |a\rangle\otimes|u\rangle \longrightarrow \phi = \frac{2\pi a}{2^n}$

(2) If $\delta \neq 0$, Measuring 1st register and getting the state $|x\rangle = |a\rangle$ is the best n-bit estimate of ϕ. The corresponding probability is $P_a = |C_a|^2 \geq \frac{4}{\pi^2} \approx 0.405$

Quantum Circuit for QPE

$$\phi = \frac{2\pi a}{2^n}$$

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} e^{2\pi i x\phi} |x\rangle \otimes |u\rangle$$

$$\text{QFT}^{-1}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{-2\pi i x y/2^n} |y\rangle$$

$$|\psi_3\rangle = \text{QFT}^{-1}|\psi_2\rangle = \frac{1}{2^n} \sum_{x=0}^{2^n-1} \sum_{y=0}^{2^n-1} e^{2\pi i x(\phi - y/2^n)} |y\rangle \otimes |u\rangle$$

Probability of observing $|y\rangle$ = $P(y) = \left| \frac{1}{2^n} \sum_{x=0}^{2^n-1} e^{2\pi i x(\phi - y/2^n)} \right|^2 = \frac{1}{2^{2n}} \left| \frac{1 - r^{2^n}}{1 - r} \right|^2, \quad r \equiv \exp\left[2\pi i\left(\phi - \frac{y}{2^n}\right)\right]$

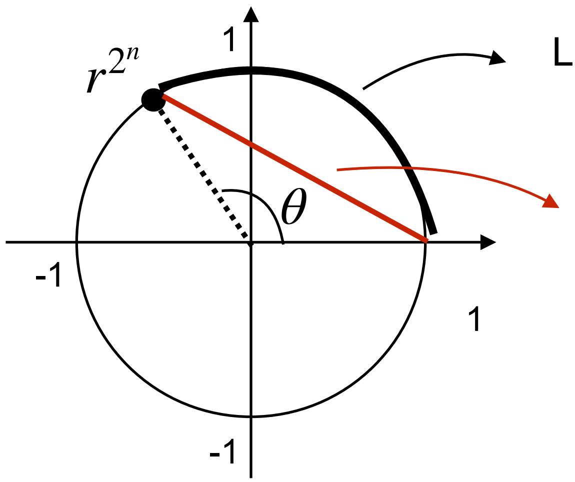
(1) If $\phi = \frac{y}{2^n}$, $\quad |\psi_3\rangle = |y\rangle \otimes |u\rangle \qquad P(\phi = \frac{y}{2^n}) = 100\,\%$

(2) If $\phi \neq \frac{y}{2^n}$, $\quad \text{closest n} - \text{bit approximation to } \phi = 0.\nu_1\nu_2\cdots\nu_n \equiv \nu \qquad \phi - \nu \equiv \delta, \quad 0 \leq |\delta| \leq \frac{1}{2^{n+1}}$

$$r \equiv \exp\left[2\pi i\left(\phi - \frac{y}{2^n}\right)\right] = \exp(2\pi i \delta)$$

$$P(y) = \frac{1}{2^{2n}} \left| \frac{1 - r^{2^n}}{1 - r} \right|^2,$$

$$r^{2^n} = \left[\exp(2\pi i \delta)\right]^{2^n} = \exp(2\pi i \delta 2^n) = e^{i\theta}$$

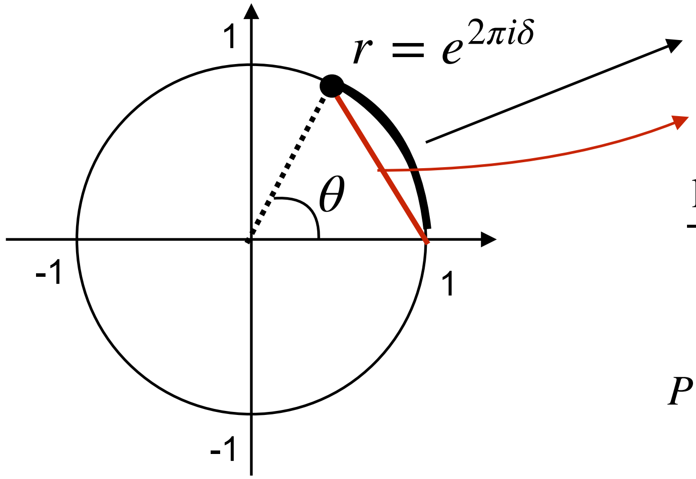


Length of minor arc $= \theta = 2\pi\delta 2^n$

Length of a cord from 1 to r^{2^n} $= |1 - r^{2^n}|$

$$\frac{\text{length of minor arc}}{\text{length of cord}} = \frac{2\pi\delta 2^n}{|1 - r^{2^n}|} \leq \frac{\text{half circumference}}{\text{diameter}} \leq \frac{\pi R}{2R} = \frac{\pi}{2} \longrightarrow |1 - r^{2^n}| \geq 4\delta 2^n$$

Quantum Circuit for QPE



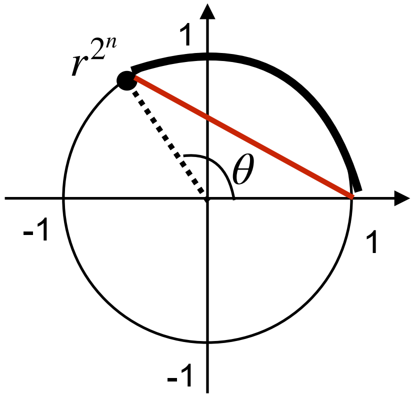
Length of minor arc = $\theta = 2\pi\delta$

Length of a cord from 1 to r = $|1-r|$

$$\frac{\text{length of minor arc}}{\text{length of cord}} = \frac{2\pi\delta}{|1-r|} > 1\,, \qquad |1-r| < 2\pi\delta$$

$$P(y) = \frac{1}{2^{2n}}\left|\frac{1-r^{2^n}}{1-r}\right|^2 \geq \frac{1}{2^{2n}}\left(\frac{4\delta 2^n}{2\pi\delta}\right)^2 = \frac{4}{\pi^2} > 0.405$$

- We will get the correct answer with probability greater than a constant.
- Probability of getting incorrect outcome can be calculated using $|\delta| > \dfrac{1}{2^{n+1}}$



$$|1-r^{2^n}| < 2 \qquad \frac{\text{length of minor arc}}{\text{length of cord}} = \frac{2\pi\delta}{|1-r|} < \frac{\pi}{2}\,, \qquad |1-r| > 4\delta$$

$$P(y) = \frac{1}{2^{2n}}\left|\frac{1-r^{2^n}}{1-r}\right|^2 \leq \frac{1}{2^{2n}}\left(\frac{2}{4\delta}\right)^2 = \frac{1}{2^{2n}(2\delta)^2}$$

If $\delta = \dfrac{c}{2^n}$, $\quad P(c) \leq \dfrac{1}{4c^2}$

- N-bit estimate of phase ϕ is obtained with a high probability.
- Need to repeat the calculation multiple times.
- Increasing n will increase the probability of success (not obvious but true).
- Increasing n (# of qubits) will improve the precision of the phase estimate.

$$\frac{\text{length of minor arc}}{\text{length of cord}} \leq \frac{\text{half circumference}}{\text{diameter}} \leq \frac{\pi R}{2R} = \frac{\pi}{2}$$

Classical Solution

Since we are promised that $|v\rangle$ is an eigenvector of U, and its eigenvalue takes the form $e^{i\theta}$, then we know that multiplying $|v\rangle$ by U will result in $|v\rangle$ multiplied by $e^{i\theta}$, i.e.,

$$U|v\rangle = e^{i\theta}|v\rangle.$$

If $|v\rangle$ is an N-dimensional vector and U is an $N \times N$ matrix, we can write out this equation as

$$\begin{pmatrix} U_{11} & U_{12} & \dots & U_{1N} \\ U_{21} & U_{22} & \dots & U_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ U_{N1} & U_{N2} & \dots & U_{NN} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix} = e^{i\theta} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix}.$$

Multiplying out the left-hand side,

$$\begin{pmatrix} U_{11}v_1 + U_{12}v_2 + \cdots + U_{1N}v_N \\ U_{21}v_1 + U_{22}v_2 + \cdots + U_{2N}v_N \\ \vdots \\ U_{N1}v_1 + U_{N2}v_2 + \cdots + U_{NN}v_N \end{pmatrix} = e^{i\theta} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix}.$$

We can use any row to find $e^{i\theta}$. For example, using the first row,

$$U_{11}v_1 + U_{12}v_2 + \cdots + U_{1N}v_N = e^{i\theta}v_1.$$

Thus the eigenvalue is

$$e^{i\theta} = \frac{U_{11}v_1 + U_{12}v_2 + \cdots + U_{1N}v_N}{v_1}.$$

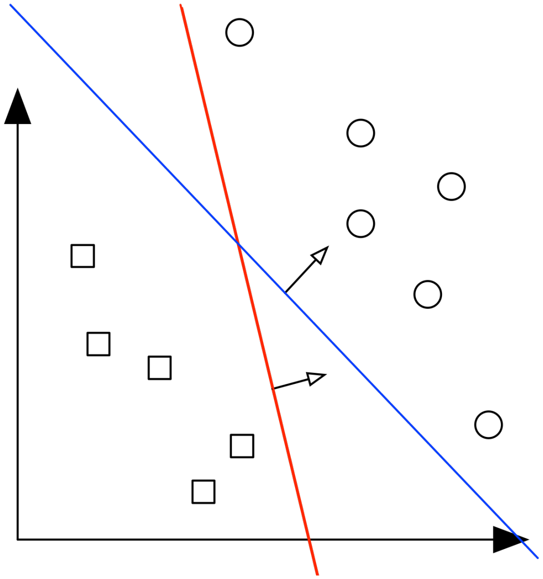
This takes N multiplications, $N-1$ additions, and one division, for a total of $2N = O(N)$ elementary arithmetic operations.

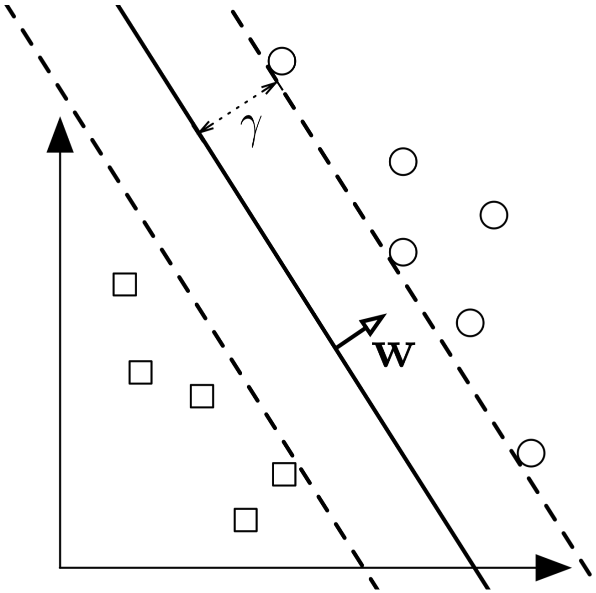
QPE

- More precisely, the algorithm returns an approximation for the phase, with high probability within additive error ϵ, using $\mathcal{O}(\log(1/\epsilon))$ qubits (without counting the ones used to encode the eigenvector state) and $\mathcal{O}(1/\epsilon)$ controlled-U operations.

Support Vector Machine

- SVM is a linear classifier that can be viewed as an extension of the perceptron (Rosenblatt 1958). The perceptron guarantees that we can find a hyperplane, if it exists. The SVM finds the maximum margin separating hyperplane.
- Setup: Define a linear classifier, $h(\vec{x}) = \text{sign}(\vec{w} \cdot \vec{x} + b)$ and assume a binary classification with labels $\{+1, -1\}$.
- Typically, if a data set is linearly separable, there are infinitely many separating hyperplanes. A natural question is:
- Q: What is the best separating hyperplane?
- SVM answer: The one that maximizes the distance to the closed data points from both classes.

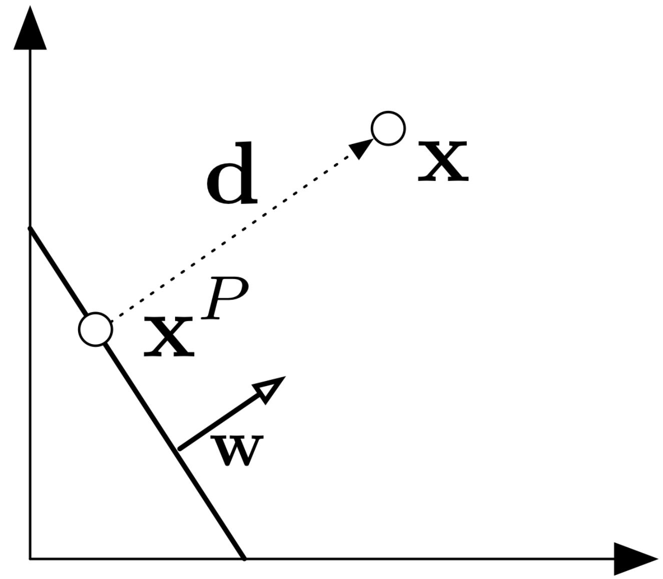




Margin

- Margin: A hyperplane is defined through $\overrightarrow{w}, b$ as a set of points such that $H = \{\vec{x} \mid \overrightarrow{w} \cdot \vec{x} + b = 0\}$. Define the margin γ as the distance from the hyperplane to the closest point across both classes.

- Distance of a point $\vec{x}$ to the hyperplane H?



$$\vec{x}^P = \vec{x} - \vec{d}$$

$$\vec{d} \parallel \overrightarrow{w} \quad \rightarrow \quad \vec{d} = \alpha\overrightarrow{w} \text{ for } \alpha \in \mathbb{R}$$

$$\vec{x}^P \in H \quad \rightarrow \quad \overrightarrow{w}^P \cdot \vec{x} + b = 0$$

$$0 = \overrightarrow{w}^P \cdot \vec{x} + b = \overrightarrow{w} \cdot (\vec{x} - \vec{d}) + b = \overrightarrow{w} \cdot (\vec{x} - \alpha\overrightarrow{w}) + b$$

$$\Rightarrow \alpha = \frac{\overrightarrow{w} \cdot \vec{x} + \vec{b}}{\overrightarrow{w} \cdot \overrightarrow{w}} \quad \Rightarrow |\vec{d}| = \sqrt{\vec{d} \cdot \vec{d}} = \sqrt{\alpha^2 \overrightarrow{w} \cdot \overrightarrow{w}} = \frac{|\overrightarrow{w} \cdot \vec{x} + \vec{b}|}{\sqrt{\overrightarrow{w} \cdot \overrightarrow{w}}}$$

$$\text{Margin of } H = \gamma(\overrightarrow{w}, b) = \min_{\vec{x} \in D} \frac{|\overrightarrow{w} \cdot \vec{x} + b|}{|\overrightarrow{w}|} \qquad \gamma(\beta\overrightarrow{w}, \beta b) = \gamma(\overrightarrow{w}, b),\ \forall \beta \neq 0$$

Scale invariance

Max Margin Classifier

- We can formulate our search for the maximum margin separating hyperplane as a constrained optimization problem. The objective is to maximize the margin under the constraints that all data points must lie on the correct side of the hyperplane:

$$\max_{\vec{w},b} \gamma(\vec{w},b) \quad \text{such that } \forall i \; y_i(\vec{w}\cdot\vec{x}_i + b) \geq 0 \qquad \gamma(\vec{w},b) = \min_{\vec{x}\in D} \frac{|\vec{w}\cdot\vec{x}+b|}{|\vec{w}|}$$

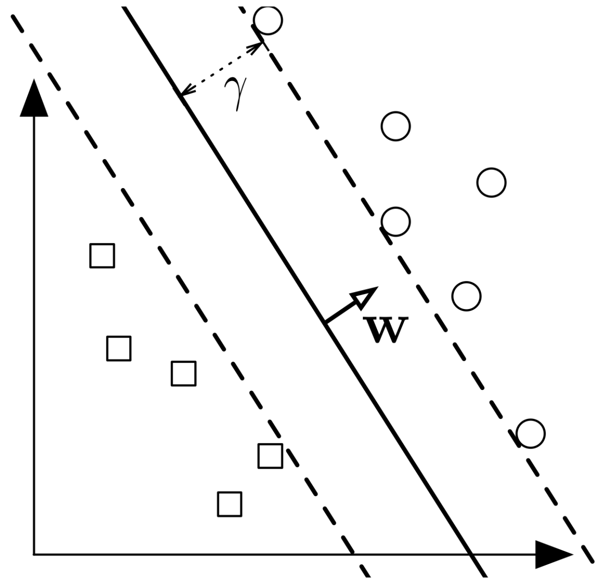
↓ Maximize the margin

↓ Separating hyperplanes

$$\max_{\vec{w},b} \frac{1}{|\vec{w}|} \min_{\vec{x}\in D} |\vec{w}\cdot\vec{x}+b| \quad \text{such that } \forall i \; y_i(\vec{w}\cdot\vec{x}_i + b) \geq 0$$

- Hyperplane is scale-invariant so we can choose $\vec{w}, b$ such that $\vec{w}\cdot\vec{x}+b=1$. The problem becomes a quadratic optimization problem.

$$\min_{\vec{w}} |\vec{w}\cdot\vec{w}|^2 \quad \text{such that } \forall i \; y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1$$

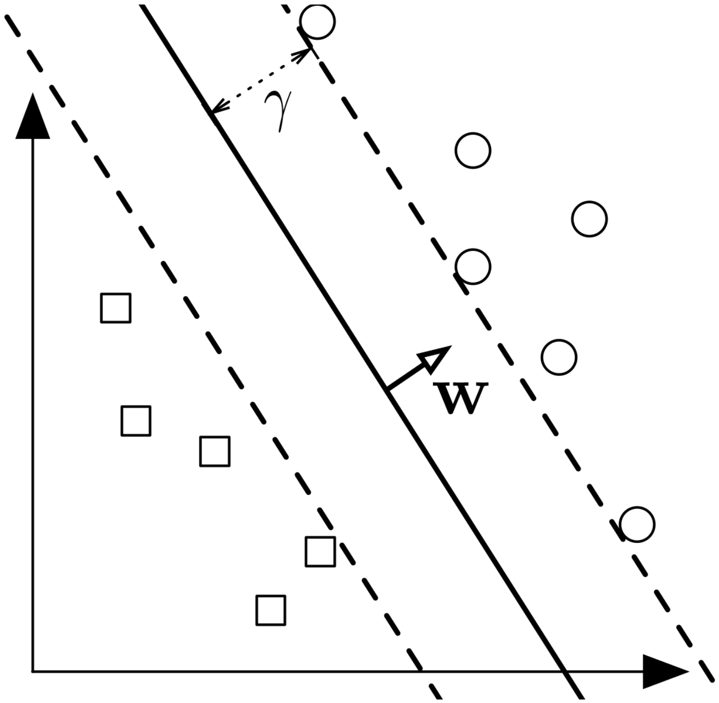


Support Vectors

- For optimal $\overrightarrow{w}, b$, some training points will have tight constraints,

$$y_i(\overrightarrow{w} \cdot \vec{x}_i + b) = 1$$

- Such training points are called support vectors.
- Support vectors are special because they are the training points that define the maximum margin of the hyperplane to the data set and they therefore determine the shape of the hyperplane. If you were to move one of them and retrain the SVM, the resulting hyperplane would change. The opposite is the case for non-support vectors (provided you don't move them too much, or they would turn into support vectors themselves). This will become particularly important in the dual formulation for Kernel-SVMs



Support Vector Machine with soft constraints

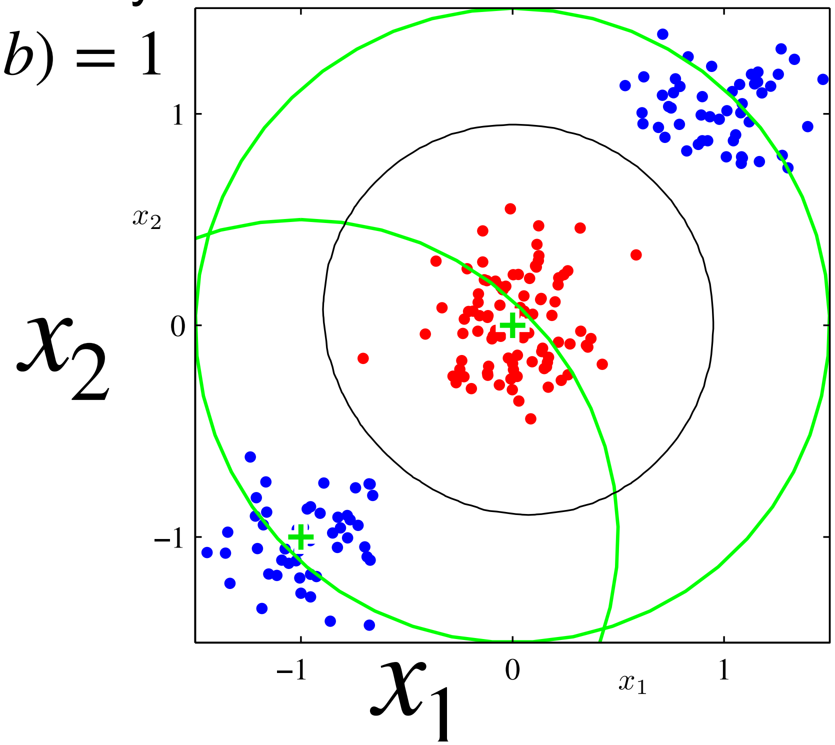
$$L = \min_{\vec{w},b} |\vec{w}\cdot\vec{w}|^2 + C\sum_{i=1}^{n} \xi_i \quad \text{such that } \forall i \; y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1 - \xi_i \;\; \text{and } \forall i \; \xi_i \geq 0$$

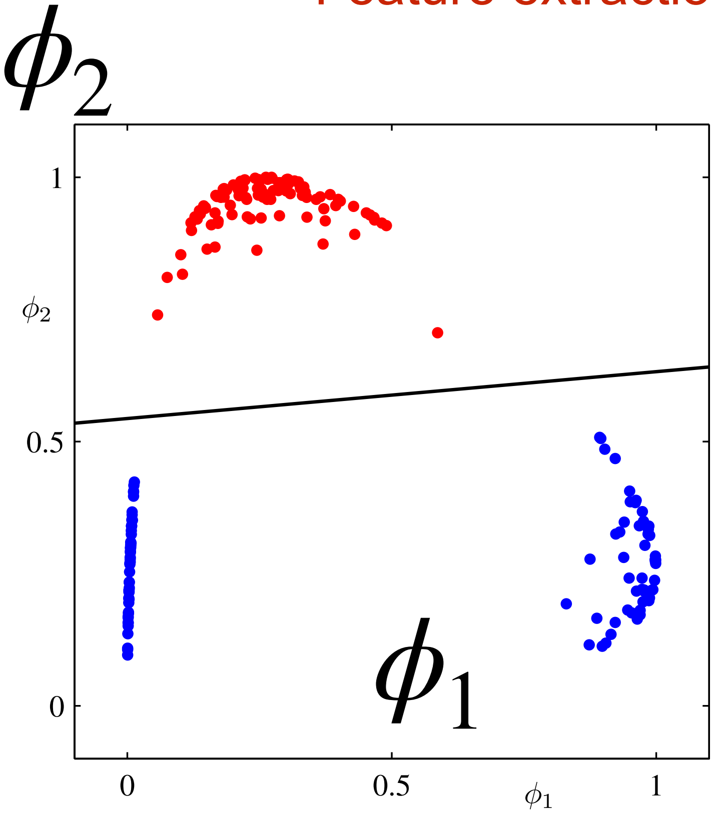
$$\Rightarrow \; L = \min_{\vec{w},b} |\vec{w}\cdot\vec{w}|^2 + C\sum_{i=1}^{n} \max\left[1 - y_i(\vec{w}\cdot x + b),\, 0\right]$$

- Kernel can be defined as $K_{ij} = k(\vec{x}_i, \vec{x}_j) = \overrightarrow{\phi}(\vec{x}_i)\cdot\overrightarrow{\phi}(\vec{x}_j)$, where $\overrightarrow{\phi}$ represents feature vector: $\vec{x} \in \mathbb{R}^d \longrightarrow \phi(\vec{x}) \in \mathbb{R}^D$

"Feature extraction"

- Support vectors satisfy: $y_i(\vec{w}\cdot\overrightarrow{\phi}(\vec{x}_i) + b) = 1$





Kernel Support Vector Machine

- Then the Support Vector Machine with soft constraints has the dual form:

$$L = \min_{\alpha_k} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K_{ij} - \sum_i \alpha_i \quad \text{such that } o \le \alpha_i \le C \text{ and } \sum_i \alpha_i y_i = 0$$

- Here $\overrightarrow{w} = \sum_i \alpha_i y_i \phi_i(\vec{x}_i)$ and $h(\vec{x}) = \operatorname{sign}\left(\sum_i \alpha_i y_i k(\vec{x}_i, \vec{x}) + b \right)$

$$K_{ij} = k(\vec{x}_i, \vec{x}_j) = \overrightarrow{\phi}(\vec{x}_i) \cdot \overrightarrow{\phi}(\vec{x}_j)$$

$$\min_{\overrightarrow{w}} |\overrightarrow{w} \cdot \overrightarrow{w}|^2 \quad \text{such that } \forall\, i \; y_i(\overrightarrow{w} \cdot \vec{x}_i + b) \ge 1$$

$$h(\vec{x}) = \operatorname{sign}(\overrightarrow{w} \cdot \vec{x} + b)$$

- Linear models for classification
- Support Vector Machines
- Quantum Support Vector Machines
- Quantum Kernel Methods