Concluding discussions for quantum algorithms

7. Quantum algorithms [4.5-5 lectures]
   (a) Quantum query complexity: black box model, phase kick back [KLM 9.2*, 6.2*]
   (b) Grover's search algorithm [NC 6.1, KLM 8.1-8.2, M 4]
   (c) Optimality of Grover's algorithm [reading] [NC 6.6]
   (d) Deutsch-Jozsa algorithm [NC 1.4.2-1.4.5, KLM 6.3-6.4, M 2.2]
   (e) Quantum fourier transform (I) [Quiz cut off] [NC 5.1, M 3.5, KLM p110-117]
   (f) Simon's algorithm [M 2.5, KLM 6.5]
   (g) Shor's factoring algorithm: Quantum fourier transform (II), period finding, classical postprocessing and error analysis, order finding, reduction of factoring to order finding, cryptographic consequences. [M 3.1-3.4, 3.7-3.10, NC 5.3, 5.4.1-5.4.2, KLM 7.1.2-7.1.3, 7.3.1-7.3.2, 7.3.4, 7.4]
   (h) Hidden subgroup framework [NC 5.4.3, KLM 7.5]
   (i) Quantum algorithm for simulating quantum physics (Hamiltonian simulation) [NC 4.7]
   (j) Concluding thoughts: quantum advantage and verification
   (k) Highlights on other quantum algorithms
   (l) Grand unification of quantum algorithms

All the quantum algorithms we have seen so far are quite similar. Is there a reason?

The problems turn out very similar if we rephrase them in a unified way.

Hidden subgroup framework

A group G: a set with
- an associative binary operation (op)
- an identity element under the group op
- closed under the group op and its inversion.

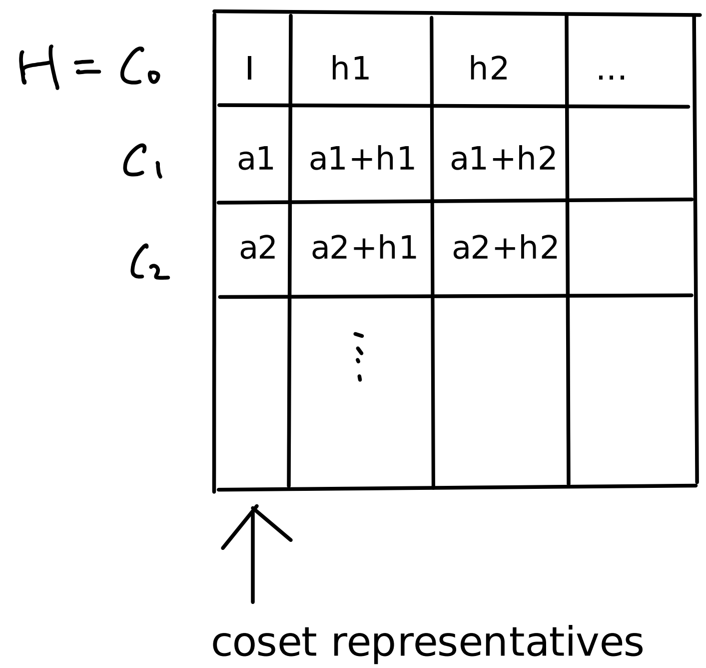
A subgroup H is a subset of G that is also a group.

Hidden subgroup framework

A group G: a set with
- an associative binary operation (op)
- an identity element under the group op
- closed under the group op and its inversion.

A subgroup H is a subset of G that is also a group.

H partitions G into cosets C0, C1, ... such that x,y are in the same coset Ci iff y-x is in H. We can take C0 = H.



e.g. $\mathbb{Z}$ is a group under addition.

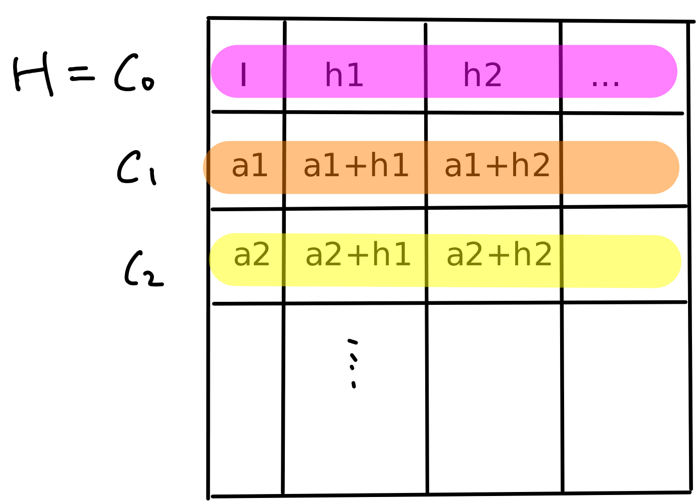
For any w, the multiples of w form a subgroup H. The cosets are labeled by elements of $\mathbb{Z}_w$ = {0,1,...,w-1} which is also a group under + (mod w).

For any n in $\mathbb{Z}$, dividing n by w gives
- a quotient (which element in H)
- a remainder (which coset).

Hidden subgroup problem:

Given: a group G with operation "+", and
a black box for a function f: G -> X

Promise (partial information about f):
there exists a subgroup H of G s.t.
f(x) = f(y) iff y = x+h for some h in H.



elements in each row share
a common function value

Hidden subgroup problem:

Given: a group G with operation "+", and
a black box for a function f: G -> X

Promise (partial information about f):
there exists a subgroup H of G s.t.
f(x) = f(y) iff y = x+h for some h in H.

Problem: determine H (provide generators for H)

$H = C_0$	I	h1	h2	...
C_1	a1	a1+h1	a1+h2	
C_2	a2	a2+h1	a2+h2	
		⋮		

elements in each row share
a common function value

Problem	G	X	H	f
Deutsch	$\{0,1\}$ $\oplus$	$\{0,1\}$	$\{0\}$ if balanced $\{0,1\}$ if constant	balanced: $f(x)=x$, $f(x)=1-x$ constant: $f(x)=0$, $f(x)=1$
Simon	$\{0,1\}^n$ $\oplus$	any finite set	$\{0,s\}$, $s\in\{0,1\}^n$	$f(x\oplus s)=f(x)$
Period finding	$\mathbb{Z}, +$	any finite set	$r\mathbb{Z}=\{\dots,-r,0,r,2r,\dots\}$	$f(x+r)=f(x)$
Order finding	$\mathbb{Z}, +$	$\{a^j\}_{j\in r\mathbb{Z}}$ $a^r=1$	$r\mathbb{Z}$	$f(x)=a^x \bmod N$
Discrete log $b=a^k \bmod r$	$\mathbb{Z}_r\times\mathbb{Z}_r$ $+ \bmod r$	$\{a^j\}_{j\in r\mathbb{Z}}$ $a^r=1$	$(-\ell k,\ell)$ $\ell\in\mathbb{Z}_r$	$f(x_1,x_2)=a^{x_1}b^{x_2}$ $=a^{x_1+kx_2}$

eg Deutsch problem

$G = \{0,1\}$, $\oplus$, $X = \{0,1\}$

$f(x) = f(y) \iff x = y$ if f balanced

ie $H = \{0\} = \langle 0 \rangle$

$f(x) = f(y)$ $\forall$ x, y if f constant

ie $H = G = \{0,1\} = \langle 1 \rangle$

} finding the generators of H solves the problem

eg Simon's problem: $H = \langle s \rangle$.

Period finding: $H = r\mathbb{Z} = \langle r \rangle$.

eg Discrete log

Given a, b, r s.t $b = a^k \bmod r$, k unknown,

Find k.

$G = \mathbb{Z}_r \times \mathbb{Z}_r$, gp op : element wise + (mod r)

Find N s.t order of a mod N is r.

$$f(x_1, x_2) = \underbrace{a^{x_1} b^{x_2}}_{\text{known}} = a^{x_1 + k x_2}$$

$$f(x_1, x_2) = f(y_1, y_2) \Leftrightarrow (x_1, x_2) - (y_1, y_2) = l(k, -1).$$

$\therefore H = \langle (k, -1) \rangle$.

These groups are all Abelian.

Quantum algorithm:

1. Start with $|0\rangle^{\otimes n}$. Prepare $\frac{1}{\sqrt{|G|}} \sum_{x \in G} |x\rangle$ by applying QFT.

These groups are all Abelian.

Quantum algorithm:

1. Start with $|0\rangle^{\otimes n}$. Prepare $\frac{1}{\sqrt{|G|}}\sum_{x\in G}|x\rangle$ by applying QFT.

2. Apply U_f. The finite set X should have its own invertible binary operation "+".

$$U_f \frac{1}{\sqrt{|G|}}\sum_{x\in G}|x\rangle|0\rangle = \frac{1}{\sqrt{|G|}}\sum_{x\in G}|x\rangle|f(x)\rangle$$

identity for binary op on X

These groups are all Abelian.

Quantum algorithm:

1. Start with $|0\rangle^{\otimes n}$. Prepare $\frac{1}{\sqrt{|G|}}\sum_{x\in G}|x\rangle$ by applying QFT.

2. Apply U_f. The finite set X should have its own invertible binary operation "+".

$$U_f \frac{1}{\sqrt{|G|}}\sum_{x\in G}|x\rangle|0\rangle = \frac{1}{\sqrt{|G|}}\sum_{x\in G}|x\rangle|f(x)\rangle$$

($|0\rangle$: identity for binary op on X)

3. Measure 2nd register. Get "coset" state (such as the periodic state) in 1st register. $\frac{1}{\sqrt{|H|}}\sum_{x\in C_i}|x\rangle$

4. Invert QFT on 1st register.

5. Measure 1st register.

6. Repeat steps 1-5 enough # times, process classically to obtain all generators of H.

4. Invert QFT on 1st register.

5. Measure 1st register.

6. Repeat steps 1-5 enough # times, process classically to obtain all generators of H.

NB QFT is defined in terms of the group character.
Left as reading assignment.

The HSP for a nonabelian gp has a similar definition.

Example (graph automorphism problem):

G = permutation group S_n of n items, w/ composition as group op.
g = graph with n vertices.

$\pi(g)$: new graph with vertex set $\pi(V(G)) = V(G)$,

$\pi(v_1) \sim \pi(v_2) \iff v_1 \sim v_2$

$X_g = \{\pi(g) : \pi \in S_n\}$

$f_g : S_n \mapsto X_g$, $f_g(\pi) = \pi(g)$

$H = \{\pi : \pi(g) = g\}$. $f_g(\pi_1) = f_g(\pi_2) \iff \pi_1 = \pi_2 \pi$ for some $\pi \in H$.

the automorphism gp

Generalizing the hidden subgroup problem to the nonabelian will solve the graph automorphism problem, which is believed not to be in BPP, but not NP-complete.

Solving the graph automorphism problem solves the graph isomorphism problem as well (given g1, g2, is g1 = π (g2) for some $\pi \in S_n$?

Elusive for the last 25 years ...

Classes of quantum algorithms "covered":

1. Those for Hidden subgroup problems
   (Deutsch-Jozsa, Simon's, Shor's)
   based on quantum fourier transform.

Classes of quantum algorithms "covered":

1. Those for Hidden subgroup problems
   (Deutsch-Jozsa, Simon's, Shor's)
   based on quantum fourier transform.

2. Those for unstructured search, counting, collisions
   (Grover's + variations) A3 Q3
   based on "amplitude amplification".

Classes of quantum algorithms "covered":

1. Those for Hidden subgroup problems
   (Deutsch-Jozsa, Simon's, Shor's)
   based on quantum fourier transform.

2. Those for unstructured search, counting, collisions
   (Grover's + variations) A3 Q3
   based on "amplitude amplification".

3. Hamiltonian simulation
   based on approximation formula for matrix
   exponentiation.

Classes of quantum algorithms "covered":

1. Those for Hidden subgroup problems
   (Deutsch-Jozsa, Simon's, Shor's)
   based on quantum fourier transform.

2. Those for unstructured search, counting, collisions
   (Grover's + variations)
   based on "amplitude amplification".

   A3 Q3

3. Hamiltonian simulation
   based on approximation formula for matrix
   exponentiation.

The technique "phase estimation" can be used for both 1 and 2.

Other major classes of algorithms:

1. Quantum walks

Other major classes of algorithms:

1. Quantum walks

2. Adiabatic quantum computation

Other major classes of algorithms:

1. Quantum walks

2. Adiabatic quantum computation

3. Quantum annealing
   (for example, used in DWave for optimization,
   unclear how to manage noise)

Other major classes of algorithms:

1. Quantum walks

2. Adiabatic quantum computation

3. Quantum annealing
   (for example, used in DWave for optimization, unclear how to manage noise)

4. Sampling hard distributions
   (mostly proof of principle for quantum computational advantage over classical)

Other major classes of algorithms:

1. Quantum walks

2. Adiabatic quantum computation

3. Quantum annealing
   (for example, used in DWave for optimization,
   unclear how to manage noise)

4. Sampling hard distributions
   (mostly proof of principle for quantum
   computational advantage over classical)

5. Speedup for solving linear systems of equations
   (Harrow, Hassidim, Lloyd: output |x> s.t. Ax=b,
   for special A's. App: Q algs for machine learning.)

6. Speedup for semidefinite programming

7. Quantum algorithms for "recommendation systems" (Kerenidis and Prakash 2016, ++) and dequantization (Ewin Tang PhD thesis, ++)

8. Quantum algorithms for learning problems concerning quantum systems (Robert Huang, Richard Kueng, John Preskill, ++)

Resources for learning more:

An overview for quantum algorithms
(by Ashley Montanaro)
https://arxiv.org/pdf/1511.04206.pdf

Algorithm zoo (by Stephen Jordan)
quantumalgorithmzoo.org

Lecture notes on advanced level quantum algorithms
(by Andrew Childs)
https://www.cs.umd.edu/~amchilds/qa/

IQC grad course by Prof David Gosset:
https://uwaterloo.ca/scholar/dgosset/classes/co-781qic-823cs-867-quantum-algorithms

An interesting recent development:

A unified framework to understand most of the existing quantum algorithms!

Main idea:

Almost all known quantum algorithms can be viewed as special polynomial transformations to the singular values of some matrices.

These transformations can be performed with some efficient quantum circuits -- most importantly, WITHOUT performing the singular value decomposition!

A bit out of scope but I want to share a tiny bit of this insight! The results are largely due to:

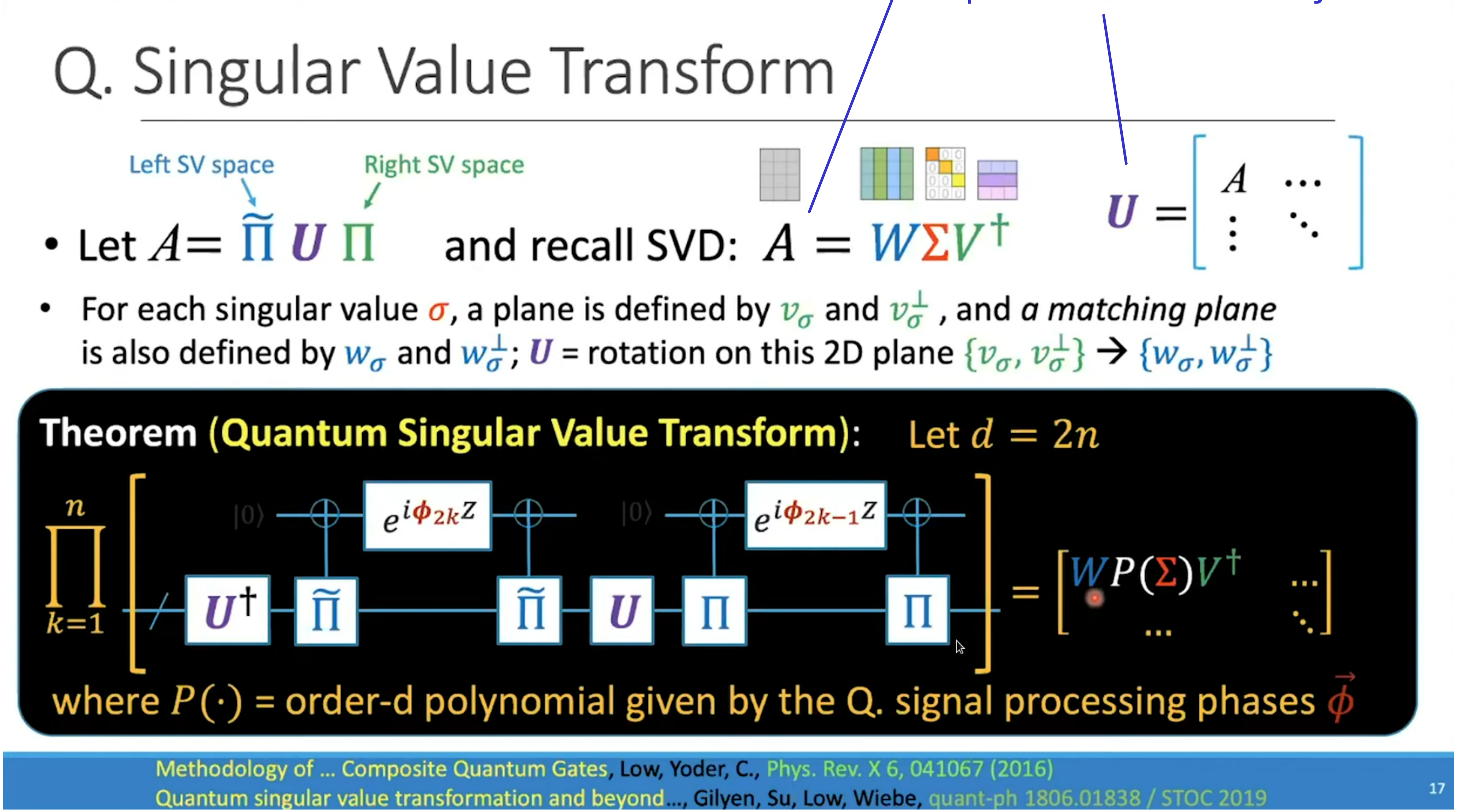
Low, Yoder, Chuang (2016)
Gilyen, Su, Low, Wiebe (2019)

and our discussion is drawn largely from
an IQC colloquium given by Isaac Chuang:
https://www.youtube.com/watch?v=GFRojXdrVXI
(including 2 slides)

with additional mathematical details from
an invited talk at TQC in QuICS by Andras Gilyen:
https://www.youtube.com/watch?v=SMdLc36ysJE

Singular value trsf w/o SV decomposition:

A encodes the problem, embed in efficiently implementable unitary U



Credit: screenshot taken from Chuang's talk

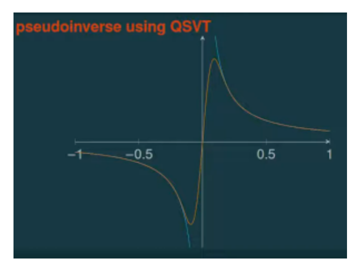
Q algorithms as singular value transformations:

Grand Unification of Q. Algorithms

Algorithm	Singular Vectors	Singular Values	Embedding	Transform
Search find t	$\lvert s\rangle, \lvert t\rangle$ start & target	$c = \langle s \lvert t \rangle$	$\begin{bmatrix} c & \cdots \\ \cdots & \ddots \end{bmatrix}$	$c \to 1$
Simulation approx. e^{-iHt}	$\lvert E \rangle$ Energy eigenstates	E Eigenenergies	$\begin{bmatrix} H & \cdots \\ \cdots & \ddots \end{bmatrix}$	$H \to \cos(H)$
Factoring eigenphase λ of U	Eigenvectors of $(I + U^\dagger)(I + U)$	$\cos(\lambda)$ Cosine of eigenphases	$\begin{bmatrix} \frac{I+U}{\sqrt{2}} & \cdots \\ \cdots & \ddots \end{bmatrix}$	$\lambda \to \begin{cases} 0 \text{ if } \lambda < 1/2 \\ \lambda \text{ otherwise} \end{cases}$

period finding is related to "eigenvalue (phase) estimation"

$Ax = b$ a (sv of A)   a -> 1/a



Credit: above screenshot from Chuang's talk
right-side screenshot from Gilyen

There are also known limitations on quantum algorithms. e.g., for unstructured problems, we cannot beat quadratic speed-up:

arXiv > quant-ph > arXiv:quant-ph/9701001

Search...

Help |

Quantum Physics

[Submitted on 1 Jan 1997]

Strengths and Weaknesses of Quantum Computing

Charles H. Bennett, Ethan Bernstein, Gilles Brassard, Umesh Vazirani

Recently a great deal of attention has focused on quantum computation following a sequence of results suggesting that quantum computers are more powerful than classical probabilistic computers. Following Shor's result that factoring and the extraction of discrete logarithms are both solvable in quantum polynomial time, it is natural to ask whether all of NP can be efficiently solved in quantum polynomial time. In this paper, we address this question by proving that relative to an oracle chosen uniformly at random, with probability 1, the class NP cannot be solved on a quantum Turing machine in time $o(2^{n/2})$. We also show that relative to a permutation oracle chosen uniformly at random, with probability 1, the class $NP \cap coNP$ cannot be solved on a quantum Turing machine in time $o(2^{n/3})$. The former bound is tight since recent work of Grover shows how to accept the class NP relative to any oracle on a quantum computer in time $O(2^{n/2})$.

Cryptographic consequences of quantum algorithms if a reliable large-scale quantum computer can be built.

1. Public key cryptosystem:

   a. Cryptosystems relying on hardness of factoring (RSA, Rabin's cryptosystem, etc) will be broken.

Cryptographic consequences of quantum algorithms if a reliable large-scale quantum computer can be built.

1. Public key cryptosystem:

   a. Cryptosystems relying on hardness of factoring (RSA, Rabin's cryptosystem, etc) will be broken.

   b. Cryptosystems relying on discrete log in cyclic abelian groups (digital signature algorithm, Diffie-Hellman encryption, ElGamal encryption system, Diffie-Hellman encryption based on elliptic curves) will be broken.

Cryptographic consequences of quantum algorithms if a reliable large-scale quantum computer can be built.

2. Private key (symmetric) cryptosystem:

a. For encryption, an n-bit key can be searched with Grover's algorithm in $O(2^{n/2})$ time, twice the key length needed for the same security.

e.g., a 128-bit AES (Advanced Encryption Standard) key is reduced to the strength of a 64-bit key.

Cryptographic consequences of quantum algorithms if a reliable large-scale quantum computer can be built.

2. Private key (symmetric) cryptosystem:

a. For encryption, an n-bit key can be searched with Grover's algorithm in $O(2^{n/2})$ time, twice the key length needed for the same security.

e.g., a 128-bit AES (Advanced Encryption Standard) key is reduced to the strength of a 64-bit key.

b. Hash functions (used in authentication etc) with an n-bit output will have reduced security, equivalent to n/2-bit output against preimage attacks and 2n/3-bit output against collisions.

Do we have to worry about an attack that may happen in the future?

Do we have to worry about an attack that may happen in the future?

Absolutely! Encrypted ciphertext today can be archived (think FB or google) and decrypted late (by quantum computation, or other advances such as a BPP algorithm for NP-complete problems).

Do we have to worry about an attack that may happen in the future?

Absolutely! Encrypted ciphertext today can be archived (think FB or google) and decrypted late (by quantum computation, or other advances such as a BPP algorithm for NP-complete problems).

Need to protect against potential attack in the next 50 years ... so won't live to be embarrassed by exposed secrets.

Do we have to worry about an attack that may happen in the future?

Absolutely! Encrypted ciphertext today can be archived (think FB or google) and decrypted late (by quantum computation, or other advances such as a BPP algorithm for NP-complete problems).

Need to protect against potential attack in the next 50 years ... so won't live to be embarrassed by exposed secrets.

Even if I have nothing to hide, privacy is a basic human right ... and I benefit from an open in which others can speak with protection.

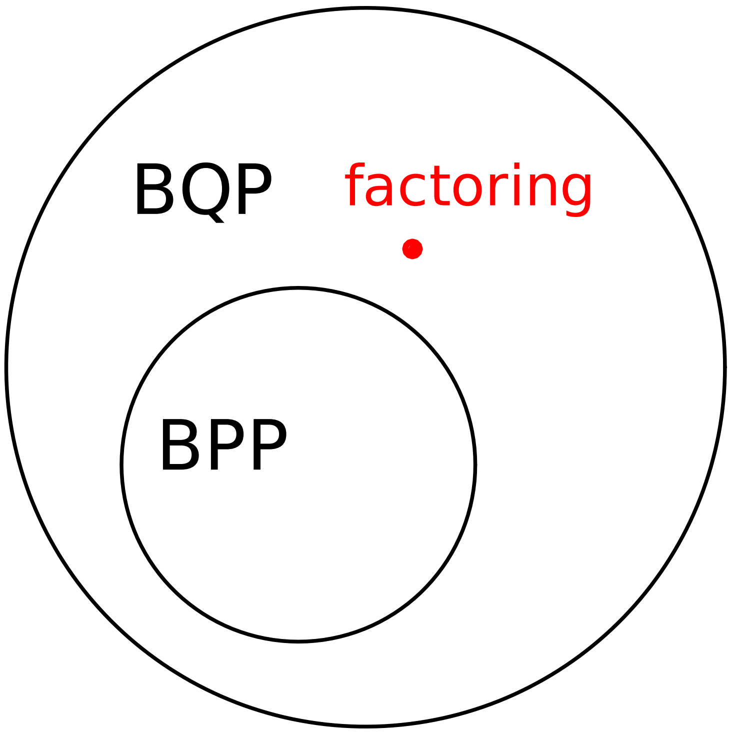
The NSA (National security agency, US) issued an information assurance directorate in August 2015 with plans to replace schemes like RSA to those resistant to quantum attacks, such as lattice-based cryptography.

The key length for private key cryptosystems will be increased accordingly.

Part (j): quantum advantage and verification

BPP: class of problems solvable in poly time by classical computers.

BQP: class of problems solvable in poly time by quantum computers.

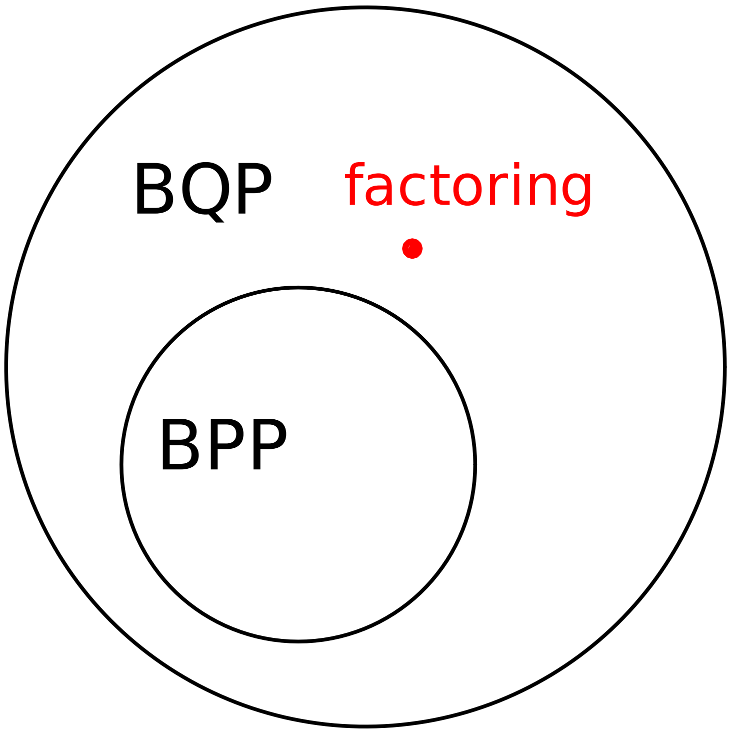


Widely held belief:

BQP $\supsetneq$ BPP

BPP: class of problems solvable in poly time by classical computers.

BQP: class of problems solvable in poly time by quantum computers.



Q1. Expts testing QM are compared to BPP-computed predictions.

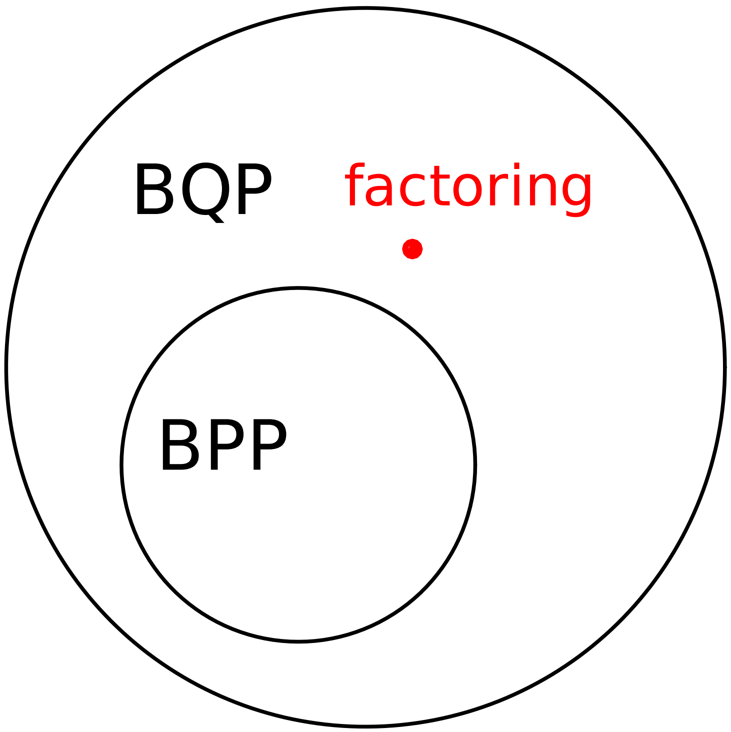
Can we test QM beyond what's predictable by BPP?

Open ...

Widely held belief:

BQP $\supsetneq$ BPP

BPP: class of problems solvable in poly time by classical computers.
BQP: class of problems solvable in poly time by quantum computers.

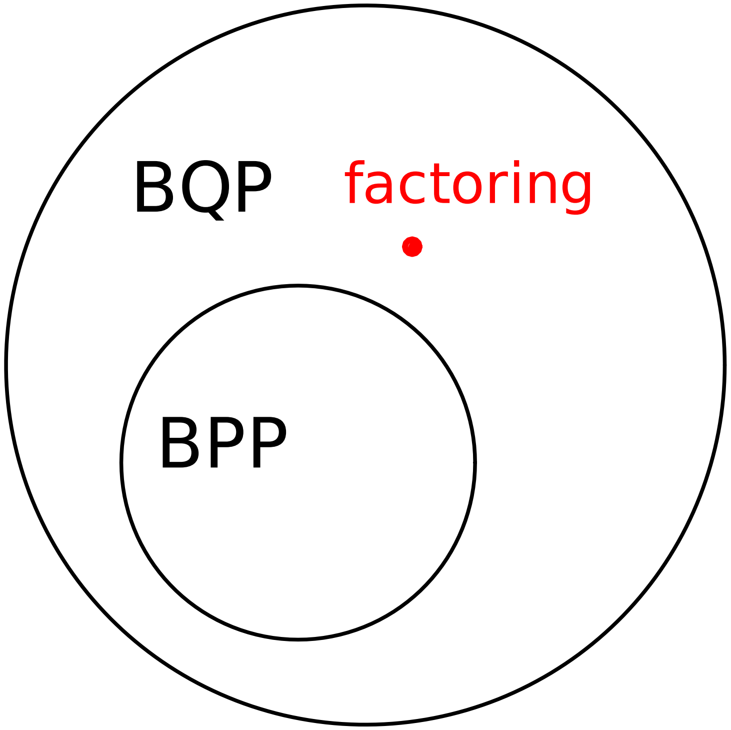


Q2. A quantum advantage is a BQP computation outside the capability of BPP. How can we verify the correctness of such quantum computation?

Widely held belief:

$BQP \supsetneq BPP$

BPP: class of problems solvable in poly time by classical computers.
BQP: class of problems solvable in poly time by quantum computers.



Q2. A quantum advantage is a BQP computation outside the capability of BPP. How can we verify the correctness of such quantum computation?

e.g., if we simulate physics (say, Ising model with 3000 qubits) by Hamiltonian simulation and measuring properties of the sys, how to tell if we actually get the right answer?

Widely held belief:

BQP $\supsetneq$ BPP

Q2 is partially addressed by Aharonov, Ben-Or, Eban, Mahadev https://arxiv.org/abs/1704.04487 :

Result: a verifier Victor, who wants to verify that a Prover Paul performs a quantum circuit in BQP correctly, can engage Paul in an "interactive protocol" in which Victor only runs BPP computations, transmits and stores a constant # of qubits and poly many classical bits.

Relies on
(1) quantum error correcting codes
(2) the power of interaction

Example how interaction can be useful:

Paul claims to have an algorithm to count the leaves on any tree in 1 second.
Victor can only count up to 100 leaves in 10 mins.

Example how interaction can be useful:

Paul claims to have an algorithm to count the leaves on any tree in 1 second.
Victor can only count up to 100 leaves in 10 mins.

To test Paul's claim, Victor asks Paul to count the leaves of a big tree, receives the answer, pulls off $0 \leqslant x \leqslant 100$ leaves from the same tree, and asks Paul to count the leaves again.

Example how interaction can be useful:

Paul claims to have an algorithm to count the leaves on any tree in 1 second.
Victor can only count up to 100 leaves in 10 mins.

To test Paul's claim, Victor asks Paul to count the leaves of a big tree, receives the answer, pulls off $0 \leqslant x \leqslant 100$ leaves from the same tree, and asks Paul to count the leaves again.

Paul knows what Victor is doing between the 2 questions, but does not know x. Victor believes Paul if the two counts differ by x. Here interaction allows Victor to verify a computation too hard for him.

e.g. PSPACE = IP (Interactive polynomial time).

Students interested can view Dorit Aharonov's talk
at KITP in 2017:
https://online.kitp.ucsb.edu/online/qinfo_c17/aharonov/