Quantum Algorithms: An overview of techniques

András Gilyén

Institute for Quantum Information and Matter

Caltech

The Quantum Wave in Computing Boot Camp
Berkeley, 28th January 2020

Outline

Main quantum tricks and techniques

- Quantum Fourier Transform
- The SWAP test
- Unitaries as representations
- Quantum simulation
- Dissipative & stochastic state preparation
- Quantum walks, Grover search

Quantum Fourier Transform

Discrete & Quantum Fourier Transform (QFT)

QFT over $\mathbb{Z}_N$

$$DFT_N = QFT_N = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{pmatrix}, \text{ where } \omega = e^{\frac{2\pi i}{N}}.$$

Discrete & Quantum Fourier Transform (QFT)

QFT over $\mathbb{Z}_N$

$$DFT_N = QFT_N = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{pmatrix}, \text{ where } \omega = e^{\frac{2\pi i}{N}}.$$

In particular $QFT_N : |j\rangle \mapsto \sum_{k=0}^{N-1} e^{\frac{2\pi ijk}{N}} |k\rangle$,

Discrete & Quantum Fourier Transform (QFT)

QFT over $\mathbb{Z}_N$

$$DFT_N = QFT_N = \frac{1}{\sqrt{N}}\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{pmatrix}, \text{ where } \omega = e^{\frac{2\pi i}{N}}.$$

In particular $QFT_N : |j\rangle \mapsto \sum_{k=0}^{N-1} e^{\frac{2\pi ijk}{N}} |k\rangle$, and $QFT_2 = H$.

Discrete & Quantum Fourier Transform (QFT)

QFT over $\mathbb{Z}_N$

$$DFT_N = QFT_N = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{pmatrix}, \text{ where } \omega = e^{\frac{2\pi i}{N}}.$$

In particular $QFT_N : |j\rangle \mapsto \sum_{k=0}^{N-1} e^{\frac{2\pi ijk}{N}} |k\rangle$, and $QFT_2 = H$.

For $N = 2^n$, QFT_N can be implemented using $O(n \log(n))$ two-qubit gates.

Discrete & Quantum Fourier Transform (QFT)

QFT over $\mathbb{Z}_N$

$$DFT_N = QFT_N = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{pmatrix}, \text{ where } \omega = e^{\frac{2\pi i}{N}}.$$

In particular $QFT_N \colon |j\rangle \mapsto \sum_{k=0}^{N-1} e^{\frac{2\pi ijk}{N}} |k\rangle$, and $QFT_2 = H$.

For $N = 2^n$, QFT_N can be implemented using $\mathcal{O}(n \log(n))$ two-qubit gates.
(The same construction as in FFT, which has complexity $\mathcal{O}(N \log(N)) = \mathcal{O}(2^n n)$.)

The Deutsch-Jozsa algorithm (1992)

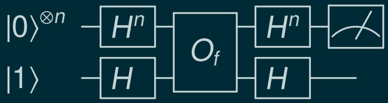
Problem

- Given a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ decide whether it is constant (0 or 1) or balanced (50% 0 and 1).
- The function is given as an oracle $O_f\colon |x\rangle|b\rangle \mapsto |x\rangle\big|b \oplus f(x)\rangle$.

The Deutsch-Jozsa algorithm (1992)

Problem

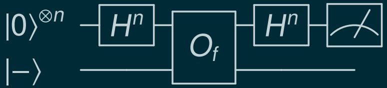
- Given a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ decide whether it is constant (0 or 1) or balanced (50% 0 and 1).
- The function is given as an oracle $O_f\colon |x\rangle|b\rangle \mapsto |x\rangle\big|b \oplus f(x)\big\rangle$.



The Deutsch-Jozsa algorithm (1992)

Problem

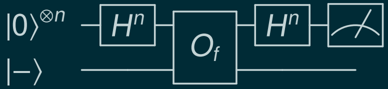
- Given a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ decide whether it is constant (0 or 1) or balanced (50% 0 and 1).
- The function is given as an oracle $O_f\colon |x\rangle|b\rangle \mapsto |x\rangle\big|b \oplus f(x)\rangle$.



The Deutsch-Jozsa algorithm (1992)

Problem

- Given a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ decide whether it is constant (0 or 1) or balanced (50% 0 and 1).
- The function is given as an oracle $O_f : |x\rangle|b\rangle \mapsto |x\rangle\big|b \oplus f(x)\big\rangle$.



Take away message

- Constructive interference can be used as a computational resource
- Studying problems in a black-box setting gives useful insights

The Bernstein-Vazirani algorithm (1992)

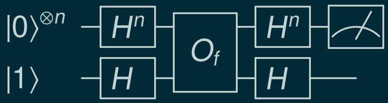
Problem

- Given a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ so that $f(x) = s \cdot x \pmod 2$; find s.
- The function is given as an oracle $O_f\colon |x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle$.

The Bernstein-Vazirani algorithm (1992)

Problem

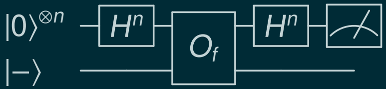
- Given a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ so that $f(x) = s \cdot x \pmod 2$; find s.
- The function is given as an oracle $O_f\colon |x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle$.



The Bernstein-Vazirani algorithm (1992)

Problem

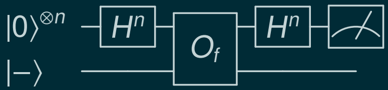
- Given a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ so that $f(x) = s \cdot x \pmod 2$; find s.
- The function is given as an oracle $O_f\colon |x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle$.



The Bernstein-Vazirani algorithm (1992)

Problem

- Given a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ so that $f(x) = s \cdot x \pmod 2$; find s.
- The function is given as an oracle $O_f\colon |x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle$.



Take away message

- Shows the power of Fourier transform (over the group $\mathbb{Z}_2^n$)
- (+1 Phase kickback is a surprising and useful quantum effect)

Jordan's quantum algorithm for gradients (2004)

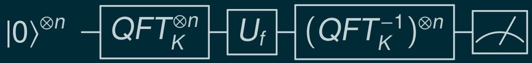
A generalization of the Bernstein-Vazirani algorithm ($\mathbb{Z}_2 \rightsquigarrow \mathbb{Z}_K$)

- Given a function $f\colon \mathbb{Z}_K^n \to \mathbb{Z}_K$ so that $f(x) = s \cdot x \pmod{K}$; find s.
- The function is given as a phase oracle $U_f\colon |x\rangle \mapsto e^{\frac{2\pi i}{K} f(x)}|x\rangle = e^{2\pi i \frac{sx}{K}}|x\rangle$.

Jordan's quantum algorithm for gradients (2004)

A generalization of the Bernstein-Vazirani algorithm ($\mathbb{Z}_2 \rightsquigarrow \mathbb{Z}_K$)

- Given a function $f\colon \mathbb{Z}_K^n \to \mathbb{Z}_K$ so that $f(x) = s \cdot x \pmod K$; find s.
- The function is given as a phase oracle $U_f\colon |x\rangle \mapsto e^{\frac{2\pi i}{K} f(x)}|x\rangle = e^{2\pi i \frac{sx}{K}}|x\rangle$.



Jordan's quantum algorithm for gradients (2004)

A generalization of the Bernstein-Vazirani algorithm ($\mathbb{Z}_2 \rightsquigarrow \mathbb{Z}_K$)

- Given a function $f\colon \mathbb{Z}_K^n \to \mathbb{Z}_K$ so that $f(x) = s \cdot x \pmod{K}$; find s.
- The function is given as a phase oracle $U_f\colon |x\rangle \mapsto e^{\frac{2\pi i}{K} f(x)}|x\rangle = e^{2\pi i \frac{sx}{K}}|x\rangle$.

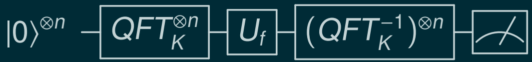
$$|0\rangle^{\otimes n} \; - \boxed{QFT_K^{\otimes n}} - \boxed{U_f} - \boxed{(QFT_K^{-1})^{\otimes n}} - \boxed{\text{measure}}$$

$\Big($Recall: $QFT_K\colon |j\rangle \mapsto \frac{1}{\sqrt{K}} \sum_{\ell=0}^{K-1} e^{2\pi i \frac{j\ell}{K}}|\ell\rangle\Big)$

Jordan's quantum algorithm for gradients (2004)

A generalization of the Bernstein-Vazirani algorithm ($\mathbb{Z}_2 \rightsquigarrow \mathbb{Z}_K$)

- Given a function $f\colon \mathbb{Z}_K^n \to \mathbb{Z}_K$ so that $f(x) = s \cdot x \pmod{K}$; find s.
- The function is given as a phase oracle $U_f\colon |x\rangle \mapsto e^{\frac{2\pi i}{K} f(x)}|x\rangle = e^{2\pi i \frac{sx}{K}}|x\rangle$.



$\Big($Recall: $QFT_K\colon |j\rangle \mapsto \frac{1}{\sqrt{K}} \sum_{\ell=0}^{K-1} e^{2\pi i \frac{j\ell}{K}}|\ell\rangle\Big)$

Jordan's algorithm ($\mathbb{Z}_K \rightsquigarrow \mathbb{R}$)

- For a differentiable function $f\colon \mathbb{R}^n \to \mathbb{R}$ we have $f(x_0 + \delta_x) \approx f(x_0) + \nabla f \cdot \delta_x$
- Discretize $\mathbb{R}$ and run the above algorithm for large enough K (resolution is $\approx \frac{1}{K}$)
- Implement $U_f\colon |\delta_x\rangle \mapsto e^{\frac{2\pi i}{K} f(x_0+\delta_x)}|\delta_x\rangle \approx e^{\frac{2\pi i f(x_0)}{K}} e^{\frac{2\pi i (\nabla f \cdot \delta_x)}{K}}|\delta_x\rangle$ with one evaluation of f

Generalizations and applications of Jordan's algorithm

Convex functions

- Have at least one subgradient at every point
- Around most points can be well approximated by a linear function

Generalizations and applications of Jordan’s algorithm

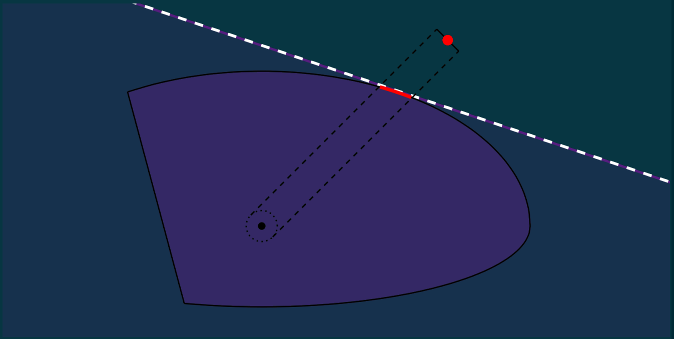
Convex functions

- Have at least one subgradient at every point
- Around most points can be well approximated by a linear function

Separating hyperplanes

Exponential speed-up for finding separating hyperplanes (2018):

- Apeldoorn, **G**, Gribling, de Wolf
- Chakrabarti, Childs, Li, Wu



Generalizations and applications of Jordan's algorithm

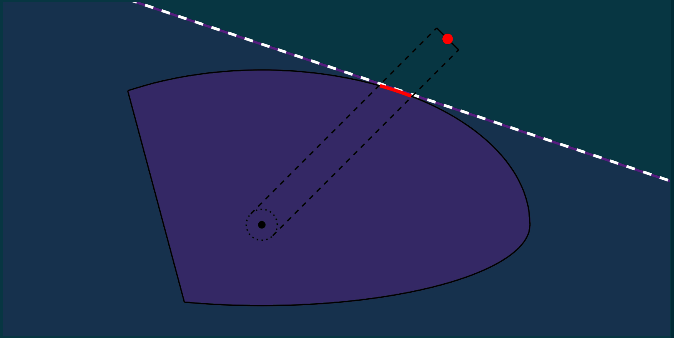
Convex functions

- Have at least one subgradient at every point
- Around most points can be well approximated by a linear function

Separating hyperplanes

Exponential speed-up for finding separating hyperplanes (2018):

- Apeldoorn, **G**, Gribling, de Wolf
- Chakrabarti, Childs, Li, Wu



Gradient computation for variational qauntum circits (QAOA)

- $\frac{1}{\varepsilon}$ Quadratic speed-up for computing the gradient (**G**, Arunachalam, Wiebe 2017)

Phase estimation ($\mathbb{Z}_2^n \rightsquigarrow \mathbb{Z}_{2^n}$)

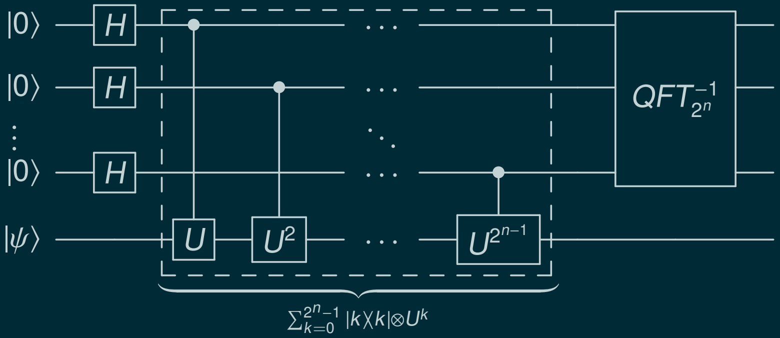
Phase estimation problem

Given $U = \sum_\lambda e^{2\pi i\lambda}|\psi_\lambda\rangle\langle\psi_\lambda|$ and an eigenstate $|\psi_\lambda\rangle$ output λ.

Phase estimation ($\mathbb{Z}_2^n \rightsquigarrow \mathbb{Z}_{2^n}$)

Phase estimation problem

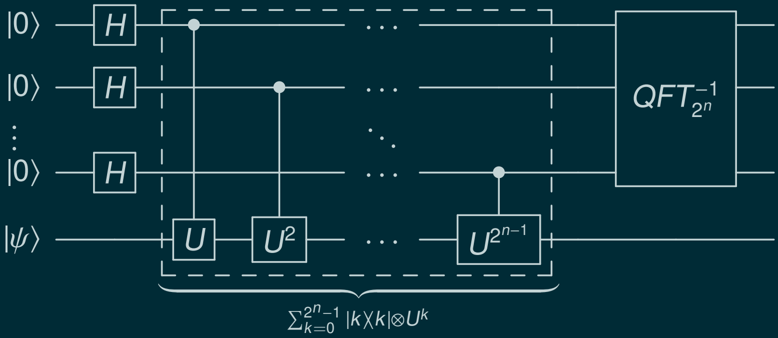
Given $U = \sum_\lambda e^{2\pi i\lambda}|\psi_\lambda\rangle\langle\psi_\lambda|$ and an eigenstate $|\psi_\lambda\rangle$ output λ.



Phase estimation ($\mathbb{Z}_2^n \rightsquigarrow \mathbb{Z}_{2^n}$)

Phase estimation problem

Given $U = \sum_\lambda e^{2\pi i\lambda}|\psi_\lambda\rangle\langle\psi_\lambda|$ and an eigenstate $|\psi_\lambda\rangle$ output λ.



$$|0\rangle^{\otimes n}|\psi\rangle \overset{H^{\otimes n}}{\mapsto} \sum_{k=0}^{2^n-1} |k\rangle|\psi\rangle \overset{U^k}{\mapsto} \left(\sum_{k=0}^{2^n-1} e^{2\pi i\lambda k}|k\rangle\right)|\psi\rangle \overset{QFT_{2^n}^{-1}}{\mapsto} |\approx 2^n\lambda\rangle|\psi\rangle$$

The Hidden subgroup problem (HSP) ($\mathbb{Z}_{2^n} \rightsquigarrow G$)

Problem

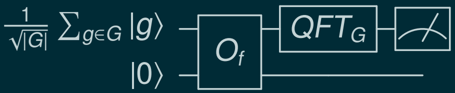
- **Input:** Oracle access to a function $f\colon G \to S$ for some group G and (finite) set S
- **Promise:** There is a subgroup $H \leq G$ such that $f(x) = f(y)$ iff $x^{-1}y \in H$
- **Goal:** Find H (and a system of its generators)

The Hidden subgroup problem (HSP) ($\mathbb{Z}_{2^n} \rightsquigarrow G$)

Problem

- **Input:** Oracle access to a function $f\colon G \to S$ for some group G and (finite) set S
- **Promise:** There is a subgroup $H \leq G$ such that $f(x) = f(y)$ iff $x^{-1}y \in H$
- **Goal:** Find H (and a system of its generators)

Algorithm for solving the problem – Kitaev (1995)

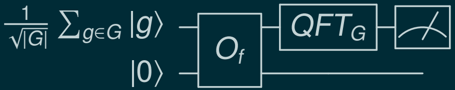


The Hidden subgroup problem (HSP) ($\mathbb{Z}_{2^n} \rightsquigarrow G$)

Problem

- **Input:** Oracle access to a function $f\colon G \to S$ for some group G and (finite) set S
- **Promise:** There is a subgroup $H \leq G$ such that $f(x) = f(y)$ iff $x^{-1}y \in H$
- **Goal:** Find H (and a system of its generators)

Algorithm for solving the problem – Kitaev (1995)



Works well for Abelian groups

- Samples a uniformly random character / irrep. of G that is trivial on H
- One can find a generator system of H after a few repetitions
- We can implement QFT_G efficiently

Some examples of the Abelian HSP

Simon's problem

- **Function:** $f\colon \{0,1\}^n \to \{0,1\}$ (the group is $\mathbb{Z}_2^n$)
- **Subgroup:** $\{0, s\}$, i.e., $f(x) = f(y)$ iff $x - y \in \{0, s\}$
- **Output:** s

Some examples of the Abelian HSP

Simon's problem

- **Function:** $f\colon \{0,1\}^n \to \{0,1\}$ (the group is $\mathbb{Z}_2^n$)
- **Subgroup:** $\{0, s\}$, i.e., $f(x) = f(y)$ iff $x - y \in \{0, s\}$
- **Output:** s

Period finding (and Shor's algorithm)

- **Function:** $f\colon \mathbb{Z} \to \mathbb{Z}_N$ (in Shor's algorithm $f(x) = a^x \mod N$ for some a)
- **Subgroup:** $p \cdot \mathbb{Z}$, i.e, $f(x) = f(y)$ iff $x - y \in p \cdot \mathbb{Z}$
- **Output:** p

Some examples of the Abelian HSP

Simon's problem

- **Function:** $f\colon \{0,1\}^n \to \{0,1\}$ (the group is $\mathbb{Z}_2^n$)
- **Subgroup:** $\{0, s\}$, i.e., $f(x) = f(y)$ iff $x - y \in \{0, s\}$
- **Output:** s

Period finding (and Shor's algorithm)

- **Function:** $f\colon \mathbb{Z} \to \mathbb{Z}_N$ (in Shor's algorithm $f(x) = a^x \mod N$ for some a)
- **Subgroup:** $p \cdot \mathbb{Z}$, i.e, $f(x) = f(y)$ iff $x - y \in p \cdot \mathbb{Z}$
- **Output:** p

Discrete log (for given γ, A find a such that $A = \gamma^a$)

- **Function:** $f\colon \mathbb{Z}_N \times \mathbb{Z}_N \to \mathbb{Z}_N$ mapping $(x, y) \mapsto \gamma^x A^{-y} \mod N$
- **Subgroup:** $\langle(a, 1)\rangle$, i.e., $f(x, y) = f(x', y')$ iff $\exists c \in \mathbb{Z}_N\colon (x - x', y - y') = (ac, c)$
- **Output:** a

For more info see, e.g., Ronald de Wolf's lecture notes: `arXiv:1907.09415`

More advanced algorithms based on Abelian HSPs

- Solving Pell's equation (Hallgren 2002)

$$x^2 - dy^2 = 1$$

- Solving the principal ideal problem (Hallgren 2002)
- Period finding over $\mathbb{R}$ and $\mathbb{R}^n$
- Computing the unit group of number fields
- Breaking elliptic curve based cryptography
- $\vdots$

More advanced algorithms based on Abelian HSPs

- Solving Pell's equation (Hallgren 2002)

$$x^2 - dy^2 = 1$$

- Solving the principal ideal problem (Hallgren 2002)
- Period finding over $\mathbb{R}$ and $\mathbb{R}^n$
- Computing the unit group of number fields
- Breaking elliptic curve based cryptography
- $\vdots$

More advanced algorithms based on Abelian HSPs

- Solving Pell's equation (Hallgren 2002)

$$x^2 - dy^2 = 1$$

- Solving the principal ideal problem (Hallgren 2002)
- Period finding over $\mathbb{R}$ and $\mathbb{R}^n$
- Computing the unit group of number fields
- Breaking elliptic curve based cryptography
- $\vdots$

See Sean Hallgren's talk on Thursday for more on this direction!

The non-Abelian HSP

What works and what does not

- QFT_G is somewhat harder to define and implement
- Unclear how to efficiently recover the subgroup
- However, the same algorithm is actually query efficient (Barnum & Knill 2002)

The non-Abelian HSP

What works and what does not

- QFT_G is somewhat harder to define and implement
- Unclear how to efficiently recover the subgroup
- However, the same algorithm is actually query efficient (Barnum & Knill 2002)
- Some cases can be solved efficiently, e.g., normal subgroups (Hallgren, Russell, Ta-Shma 2000), solvable groups (Watrous 2001), nil-2 groups (Ivanyos, Sanselme, Sántha 2007), and certain semidirect product p-groups of constant nilpotency class (Ivanyos, Sántha 2015)
- Kuperberg's algorithm (2003) solves HSP in the dihedral group in time

$$\mathcal{O}\left(2^{\sqrt{\log(G)}}\right)$$

The non-Abelian HSP

What works and what does not

- QFT_G is somewhat harder to define and implement
- Unclear how to efficiently recover the subgroup
- However, the same algorithm is actually query efficient (Barnum & Knill 2002)
- Some cases can be solved efficiently, e.g., normal subgroups (Hallgren, Russell, Ta-Shma 2000), solvable groups (Watrous 2001), nil-2 groups (Ivanyos, Sanselme, Sántha 2007), and certain semidirect product p-groups of constant nilpotency class (Ivanyos, Sántha 2015)
- Kuperberg's algorithm (2003) solves HSP in the dihedral group in time

$$\mathcal{O}\left(2^{\sqrt{\log(G)}}\right)$$

Important example: Graph isomorphism (i.e., deciding whether $G \simeq G'$)

- **Group:** S_{2n}, **Function:** permute the vertices of $G \cup G'$
- **Subgroup:** Automorphisms of $G \cup G'$
- **Output:** whether there is a generator interchanging vertices of G and G'

The SWAP test

A simpler algorithm for graph isomorphism

Prepare a uniform superposition

- Let $|\psi_0\rangle \propto \sum_{s \in S_n} \left|s(G)\right\rangle$
- Let $|\psi_1\rangle \propto \sum_{s \in S_n} \left|s(G')\right\rangle$
- Observe that

$$\langle\psi_0|\psi_1\rangle = \begin{cases} 1 & \text{if } G \simeq G' \\ 0 & \text{otherwise} \end{cases}$$

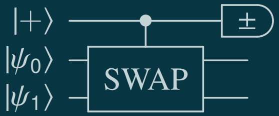
A simpler algorithm for graph isomorphism

Prepare a uniform superposition

- Let $|\psi_0\rangle \propto \sum_{s \in S_n} \left|s(G)\right\rangle$
- Let $|\psi_1\rangle \propto \sum_{s \in S_n} \left|s(G')\right\rangle$
- Observe that

$$\langle \psi_0 | \psi_1 \rangle = \begin{cases} 1 & \text{if } G \simeq G' \\ 0 & \text{otherwise} \end{cases}$$

The SWAP test



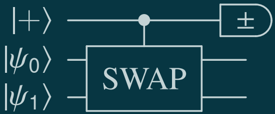
A simpler algorithm for graph isomorphism

Prepare a uniform superposition

- Let $|\psi_0\rangle \propto \sum_{s \in S_n} \left|s(G)\right\rangle$
- Let $|\psi_1\rangle \propto \sum_{s \in S_n} \left|s(G')\right\rangle$
- Observe that

$$\langle\psi_0|\psi_1\rangle = \begin{cases} 1 & \text{if } G \simeq G' \\ 0 & \text{otherwise} \end{cases}$$

The SWAP test



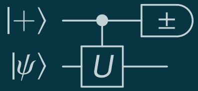
The probability of getting outcome $+$ is

$$\frac{1}{2} + \frac{1}{2}|\langle\psi_0|\psi_1\rangle|^2$$

Unitaries as representations

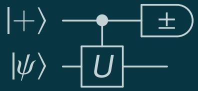
Towards approximating the Jones polynomial

The Hadamard test



Towards approximating the Jones polynomial

The Hadamard test

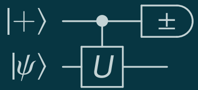


The probability of getting outcome $+$ is

$$\frac{1}{2} + \frac{1}{2}\mathrm{Re}(\langle\psi|U|\psi\rangle)$$

Towards approximating the Jones polynomial

The Hadamard test



The probability of getting outcome $+$ is

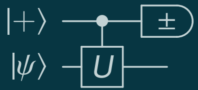
$$\frac{1}{2} + \frac{1}{2}\mathrm{Re}(\langle\psi|U|\psi\rangle)$$

The Jones polynomial – a link invariant

A *link* is a collection of loops embedded into $\mathbb{R}^3$, in a possibly intertwined way. A *link invariant* is a quantity associated to links that is invariant under smooth transformations of the embedding.

Towards approximating the Jones polynomial

The Hadamard test



The probability of getting outcome $+$ is

$$\frac{1}{2} + \frac{1}{2}\mathrm{Re}(\langle\psi|U|\psi\rangle)$$

The Jones polynomial – a link invariant

A *link* is a collection of loops embedded into $\mathbb{R}^3$, in a possibly intertwined way. A *link invariant* is a quantity associated to links that is invariant under smooth transformations of the embedding.

Approximating the Jones polynomial

Links from braids

A *braid* is a collection of parallel strands, where adjacent strands are allowed to cross under or over each other. One can get a link by connecting the bottom and top ends of the strands.

Approximating the Jones polynomial

Links from braids

A *braid* is a collection of parallel strands, where adjacent strands are allowed to cross under or over each other. One can get a link by connecting the bottom and top ends of the strands.

Braids form a group under the operation of concatenation. The Jones polynomial of various links formed by a braid can be expressed in terms of the Temperley-Lieb algebra – a representation of the braid group.

Approximating the Jones polynomial

Links from braids

A *braid* is a collection of parallel strands, where adjacent strands are allowed to cross under or over each other. One can get a link by connecting the bottom and top ends of the strands.

Braids form a group under the operation of concatenation. The Jones polynomial of various links formed by a braid can be expressed in terms of the Temperley-Lieb algebra – a representation of the braid group.

Quantum algorithms and connections to field theory

- For a root of unity $e^{2\pi i/k}$, the relevant representation is unitary; the corresponding value of the Jones polynomial can be approx. evaluated via estimating $\langle\psi|U|\psi\rangle$. This (BQP-complete) algorithm is due to Aharonov, Jones, and Landau (2006).

Approximating the Jones polynomial

Links from braids

A *braid* is a collection of parallel strands, where adjacent strands are allowed to cross under or over each other. One can get a link by connecting the bottom and top ends of the strands.

Braids form a group under the operation of concatenation. The Jones polynomial of various links formed by a braid can be expressed in terms of the Temperley-Lieb algebra – a representation of the braid group.

Quantum algorithms and connections to field theory

- For a root of unity $e^{2\pi i/k}$, the relevant representation is unitary; the corresponding value of the Jones polynomial can be approx. evaluated via estimating $\langle\psi|U|\psi\rangle$. This (BQP-complete) algorithm is due to Aharonov, Jones, and Landau (2006).
- Witten showed that the Jones polynomial is closely related to topological quantum field theory (TQFT).

Approximating the Jones polynomial

Links from braids

A *braid* is a collection of parallel strands, where adjacent strands are allowed to cross under or over each other. One can get a link by connecting the bottom and top ends of the strands.

Braids form a group under the operation of concatenation. The Jones polynomial of various links formed by a braid can be expressed in terms of the Temperley-Lieb algebra – a representation of the braid group.

Quantum algorithms and connections to field theory

- For a root of unity $e^{2\pi i/k}$, the relevant representation is unitary; the corresponding value of the Jones polynomial can be approx. evaluated via estimating $\langle\psi|U|\psi\rangle$. This (BQP-complete) algorithm is due to Aharonov, Jones, and Landau (2006).
- Witten showed that the Jones polynomial is closely related to topological quantum field theory (TQFT).
- Friedman, Kitaev, Larsen, and Wang (2001) showed that quantum computers can efficiently simulate TQFTs.

Quantum simulation

(Dynamical) Hamiltonian simulation

Time-independent Hamiltonians

Schrödinger's equation ($\hbar = 1$) for time-independent quantum systems:

$$\frac{d}{dt}|\psi\rangle = -iH|\psi\rangle \implies |\psi(t)\rangle = e^{-itH}|\psi(0)\rangle$$

Recap – matrix functions

Any Hermitian matrix H can be diagonalised using some unitary V such that $H = V^\dagger DV = \sum_\lambda \lambda|\lambda\rangle\langle\lambda|$.

(Dynamical) Hamiltonian simulation

Time-independent Hamiltonians

Schrödinger's equation ($\hbar = 1$) for time-independent quantum systems:

$$\frac{d}{dt}|\psi\rangle = -iH|\psi\rangle \implies |\psi(t)\rangle = e^{-itH}|\psi(0)\rangle$$

Recap – matrix functions

Any Hermitian matrix H can be diagonalised using some unitary V such that $H = V^\dagger D V = \sum_\lambda \lambda|\lambda\rangle\langle\lambda|$. For any $f\colon \mathbb{R} \to \mathbb{C}$ we can define

$$f(H) := V^\dagger f(D) V = \sum_\lambda f(\lambda)|\lambda\rangle\langle\lambda|$$

(Dynamical) Hamiltonian simulation

Time-independent Hamiltonians

Schrödinger's equation ($\hbar = 1$) for time-independent quantum systems:

$$\frac{d}{dt}|\psi\rangle = -iH|\psi\rangle \implies |\psi(t)\rangle = e^{-itH}|\psi(0)\rangle$$

Recap – matrix functions

Any Hermitian matrix H can be diagonalised using some unitary V such that $H = V^\dagger DV = \sum_\lambda \lambda|\lambda\rangle\langle\lambda|$. For any $f\colon \mathbb{R} \to \mathbb{C}$ we can define

$$f(H) := V^\dagger f(D)V = \sum_\lambda f(\lambda)|\lambda\rangle\langle\lambda|$$

Wait a minute, don't we build quantum computers using Hamiltonian simulation???

Product formula approach (Lloyd 1996)

Time-independent local Hamiltonians

Let $H = \sum_{k=1}^{K} H_k$, where each term H_k acts on a constant (say 2) number of qubits.

Product formula approach (Lloyd 1996)

Time-independent local Hamiltonians

Let $H = \sum_{k=1}^{K} H_k$, where each term H_k acts on a constant (say 2) number of qubits. WLOG. assume $\forall k: \|H_k\| \leq 1$. We can approximate the time-evolution by

$$e^{-itH} = (e^{-\frac{itH}{r}})^r = (e^{-\frac{itH_1}{r}} e^{-\frac{itH_2}{r}} \cdots e^{-\frac{itH_K}{r}})^r + \mathcal{O}\left(\frac{(tK)^2}{r}\right).$$

Choosing $r = \Theta((tK)^2/\varepsilon)$ guarantees an ε-approximation.

Product formula approach (Lloyd 1996)

Time-independent local Hamiltonians

Let $H = \sum_{k=1}^{K} H_k$, where each term H_k acts on a constant (say 2) number of qubits. WLOG. assume $\forall k : \|H_k\| \leq 1$. We can approximate the time-evolution by

$$e^{-itH} = (e^{-\frac{itH}{r}})^r = (e^{-\frac{itH_1}{r}} e^{-\frac{itH_2}{r}} \cdots e^{-\frac{itH_K}{r}})^r + \mathcal{O}\left(\frac{(tK)^2}{r}\right).$$

Choosing $r = \Theta((tK)^2/\varepsilon)$ guarantees an ε-approximation.

(Query) Optimal Hamiltonian simulation of sparse matrices

- $\vdots$
- Quantum Signal Processing (QSP): (Low & Chuang 2016)

$$\mathcal{O}(t\|H\|_{\max}s + \log(1/\varepsilon))$$

For a recent survey see: Childs, Maslov, Nam, Ross, Su – `arXiv:1711.10980`

More generalizations and improvements

A few more recent generic results (without being exhaustive)

- Time-dependent sparse Hamiltonians: (Berry, Child, Su, Wang, Wiebe 2019)

$$\widetilde{\mathcal{O}}\left(s \int_0^t \|H(\tau)\|_{\max} d\tau\right)$$

- Quantum chemistry: (Babbush, Berry, McClean, Neven 2019)

$$\widetilde{\mathcal{O}}\left(N^{\frac{1}{3}}\eta^{\frac{8}{3}}\right), \text{ with } N: \#\text{plane wave orbitals}, \eta: \#\text{electrons}$$

- Lattice Hamiltonians: (Haah, Hastings, Kothari, Low: QIP'19)

$$\widetilde{\mathcal{O}}(nt)$$

- …, multi-product formulas, interaction picture simulation, …

More generalizations and improvements

A few more recent generic results (without being exhaustive)

- Time-dependent sparse Hamiltonians: (Berry, Child, Su, Wang, Wiebe 2019)

$$\widetilde{\mathcal{O}}\left(s\int_0^t \|H(\tau)\|_{\max} d\tau\right)$$

- Quantum chemistry: (Babbush, Berry, McClean, Neven 2019)

$$\widetilde{\mathcal{O}}\left(N^{\frac{1}{3}}\eta^{\frac{8}{3}}\right), \text{ with } N : \#\text{plane wave orbitals}, \eta : \#\text{electrons}$$

- Lattice Hamiltonians: (Haah, Hastings, Kothari, Low: QIP'19)

$$\widetilde{\mathcal{O}}(nt)$$

- …, multi-product formulas, interaction picture simulation, …

Simulating quantum field theory? See Preskill's recent survey: `arXiv:1811.10085`

Dissipative & stochastic state preparation

Ground state preparation of frustration-free Hamiltonians

Ground state preparation of frustration-free Hamiltonians

The resampling algorithm

while not all constraints checked **do**

- pick an unchecked constraint and check (measure) it
- **if** unsatisfied **then**
  randomly resample all adjacent (qu)bits
  mark all adjacent constraints as unchecked

Ground state preparation of frustration-free Hamiltonians

The resampling algorithm

while not all constraints checked **do**

- pick an unchecked constraint and check (measure) it
- **if** unsatisfied **then**
  randomly resample all adjacent (qu)bits
  mark all adjacent constraints as unchecked

- Ground state preparation by dissipation (Verstraete, Wolf, Cirac 2008)

Ground state preparation of frustration-free Hamiltonians

The resampling algorithm

while not all constraints checked **do**

- pick an unchecked constraint and check (measure) it
- **if** unsatisfied **then**
  randomly resample all adjacent (qu)bits
  mark all adjacent constraints as unchecked

- Ground state preparation by dissipation (Verstraete, Wolf, Cirac 2008)
- Efficient version in the classical version (Moser & Tardos 2009)

Ground state preparation of frustration-free Hamiltonians

The resampling algorithm

while not all constraints checked **do**

- pick an unchecked constraint and check (measure) it
- **if** unsatisfied **then**
  randomly resample all adjacent (qu)bits
  mark all adjacent constraints as unchecked

- Ground state preparation by dissipation (Verstraete, Wolf, Cirac 2008)
- Efficient version in the classical version (Moser & Tardos 2009)
- Efficient commuting quantum Lovász Local Lemma
  (Sattath & Arad; Schwarz, Cubitt, Verstraete – 2013)
- Efficient non-commuting version for uniformly gapped systems (**G** & Sattath 2016)

Ground state preparation of frustration-free Hamiltonians

The resampling algorithm

while not all constraints checked **do**

- pick an unchecked constraint and check (measure) it
- **if** unsatisfied **then**
  randomly resample all adjacent (qu)bits
  mark all adjacent constraints as unchecked

- Ground state preparation by dissipation (Verstraete, Wolf, Cirac 2008)
- Efficient version in the classical version (Moser & Tardos 2009)
- Efficient commuting quantum Lovász Local Lemma
  (Sattath & Arad; Schwarz, Cubitt, Verstraete – 2013)
- Efficient non-commuting version for uniformly gapped systems (**G** & Sattath 2016)

A loosely related result

- Quant. Metropolis samp. (Temme, Osborne, Vollbrecht, Poulin, Verstraete 2009)

Quantum walks

Continuous-time quantum / random walks

Laplacian of a weighted graph

Let $G = (V, E)$ be a finite simple graph, with non-negative edge-weights $w\colon E \to \mathbb{R}_+$. The Laplacian is defined as

$$u \neq v\colon\ L_{uv} = w_{uv}, \text{ and } L_{uu} = -\sum_{v} w_{uv}.$$

Continuous-time quantum / random walks

Laplacian of a weighted graph

Let $G = (V, E)$ be a finite simple graph, with non-negative edge-weights $w\colon E \to \mathbb{R}_+$. The Laplacian is defined as

$$u \neq v\colon\ L_{uv} = w_{uv},\ \text{and } L_{uu} = -\sum_{v} w_{uv}.$$

Continuous-time walks

Evolution of the state:

$$\frac{d}{dt} p_u(t) = \sum_{v \in V} L_{uv} p_v(t) \qquad \Longrightarrow \qquad p(t) = e^{tL} p(0)$$

Continuous-time quantum / random walks

Laplacian of a weighted graph

Let $G = (V, E)$ be a finite simple graph, with non-negative edge-weights $w\colon E \to \mathbb{R}_+$. The Laplacian is defined as

$$u \neq v\colon \; L_{uv} = w_{uv}, \; \text{and } L_{uu} = -\sum_{v} w_{uv}.$$

Continuous-time walks

Evolution of the state:

$$\frac{d}{dt} p_u(t) = \sum_{v\in V} L_{uv} p_v(t) \qquad \Longrightarrow \qquad p(t) = e^{tL} p(0)$$

$$i\frac{d}{dt} \psi_u(t) = \sum_{v\in V} L_{uv} \psi_v(t) \qquad \Longrightarrow \qquad \psi(t) = e^{-itL} \psi(0)$$

Exponential speedup by a quantum walk

Childs, Cleve, Deotto, Farhi, Gutmann, and Spielman: `quant-ph/0209131`