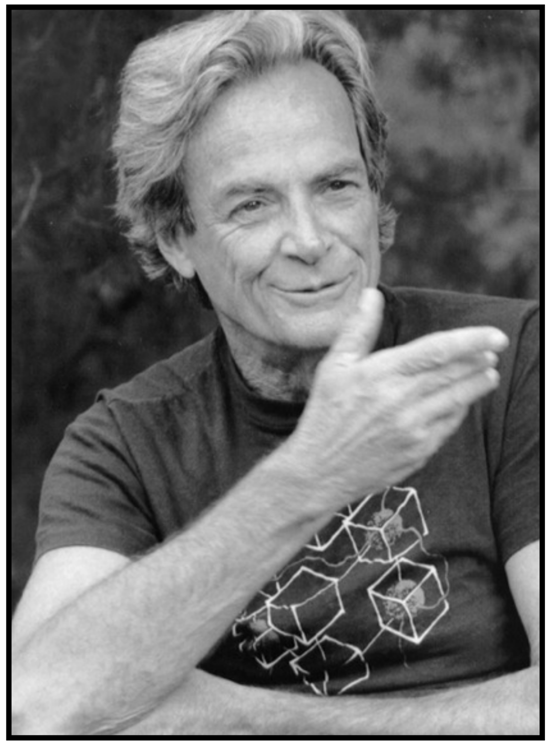
Quantum algorithms for simulating quantum mechanics

Andrew Childs

Department of Computer Science
Institute for Advanced Computer Studies
Joint Center for Quantum Information and Computer Science
University of Maryland

JOINT CENTER FOR
QUANTUM INFORMATION
AND COMPUTER SCIENCE



"... nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy."

Richard Feynman
Simulating physics with computers (1981)

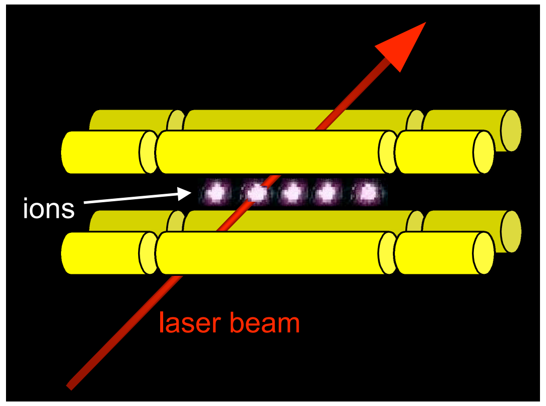
The ultimate quantum physics lab

Universal quantum computer

- Prepare system in a pure state of n qubits
- Apply 2-qubit unitary operations
- Measure in standard basis

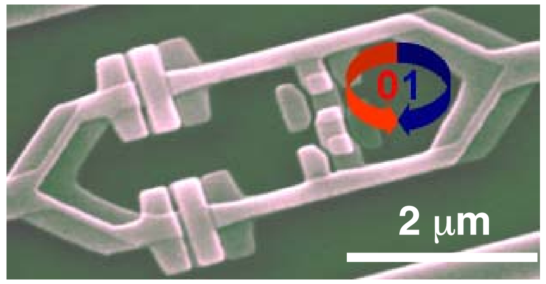
Many possible implementations

trapped ions



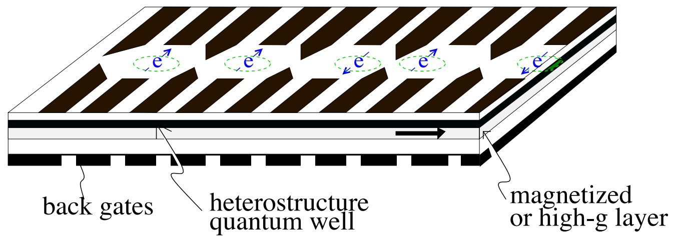
(Monroe & Wineland)

superconducting circuits



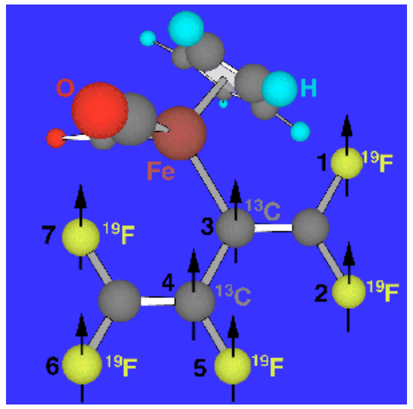
(Nakamura et al.)

quantum dots



(Loss & DiVincenzo)

nuclear spins

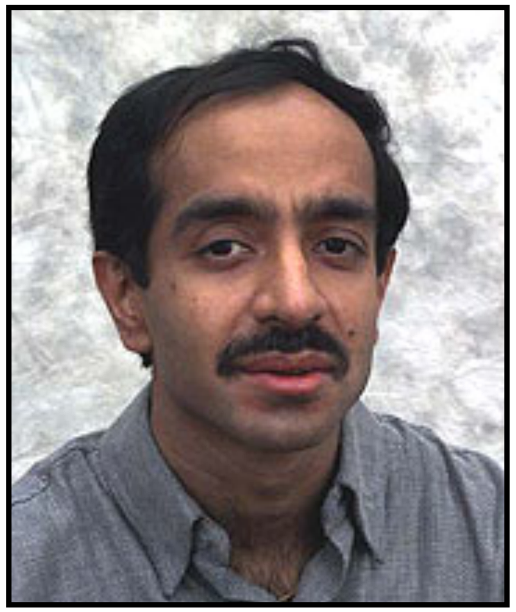


(Chuang et al.)

Fault-tolerance Threshold Theorem: If we can manipulate qubits sufficiently well (constant error rate, say 10^{-4}), we can effectively make them perfect through an encoding with reasonable overhead.

Fast algorithms for classically hard problems

3107418240490043721350750035888567930037346022842727545720161948823206440518081504556346829671723286782437916272838033415471073108501919548529007337724822783525742386454014691736602477652346609
=
16347336458092538484431338838650908598417836700330923121811108523893331001045081512121181675115 79
×
190087128166482211312685157393541397547189678996851549366663853908802710380210449895719126146557 1



```
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000001000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
000000000000000000000000000000000
```

- Computing discrete logarithms
- Decomposing Abelian groups
- Computations in number fields
- Approximating Gauss sums
- Shifted Legendre symbol
- Counting points on algebraic curves
- Approximating the Jones polynomial (and other topological invariants)
- Simulating quantum mechanics
- Linear systems
- Computing effective resistance
- …

- Formula evaluation
- Collision finding (k-distinctness, k-sum, etc.)
- Minimum spanning tree, connectivity, shortest paths, bipartiteness of graphs
- Network flows, maximal matchings
- Finding subgraphs
- Minor-closed graph properties
- Property testing (distance between distributions, bipartiteness/expansion of graphs, etc.)
- Checking matrix multiplication
- Group commutativity
- Subset sum
- …

Two kinds of quantum simulation

Analog simulation: Build a device whose Hamiltonian effectively models a desired target system

Ex: Use an optical lattice to simulate spin models

Digital simulation: Build a universal, fault-tolerant quantum computer and perform a controlled approximation of the dynamics of the target system

Why simulate quantum mechanics?

Computational chemistry/physics

- chemical reactions
- properties of materials

Implementing quantum algorithms

- continuous-time quantum walk (e.g., for formula evaluation)
- adiabatic quantum computation (e.g., for optimization)
- linear/differential equations

Quantum dynamics

The dynamics of a quantum system are determined by its *Hamiltonian*.

$$i \frac{\mathrm{d}}{\mathrm{d}t} |\psi(t)\rangle = H |\psi(t)\rangle$$

$$\Downarrow$$

$$|\psi(t)\rangle = e^{-iHt} |\psi(0)\rangle$$

Quantum simulation problem: Given a description of the Hamiltonian H, an evolution time t, and an initial state $|\psi(0)\rangle$, produce the final state $|\psi(t)\rangle$ (to within some error tolerance ϵ)

A classical computer cannot even represent the state efficiently

A quantum computer cannot produce a complete description of the state, but by performing measurements on the state, it can answer questions that (apparently) a classical computer cannot

Local Hamiltonians

Simple class of systems that can be simulated efficiently [Lloyd 96]:

$H = \sum_{j=1}^{m} H_j$ where each H_j acts on $k = O(1)$ qubits

Ex: Spin system on a lattice

Lie Product Formula:

$$\lim_{r\to\infty} \left(e^{-iAt/r} e^{-iBt/r}\right)^r = e^{-i(A+B)t}$$

Approximate version:

$$\left(e^{-iAt/r} e^{-iBt/r}\right)^r = e^{-i(A+B)t} + O(t^2/r)$$

Complexity (number of elementary gates): $\mathrm{poly}(\log N)\,(\|H\|t)^2/\epsilon$

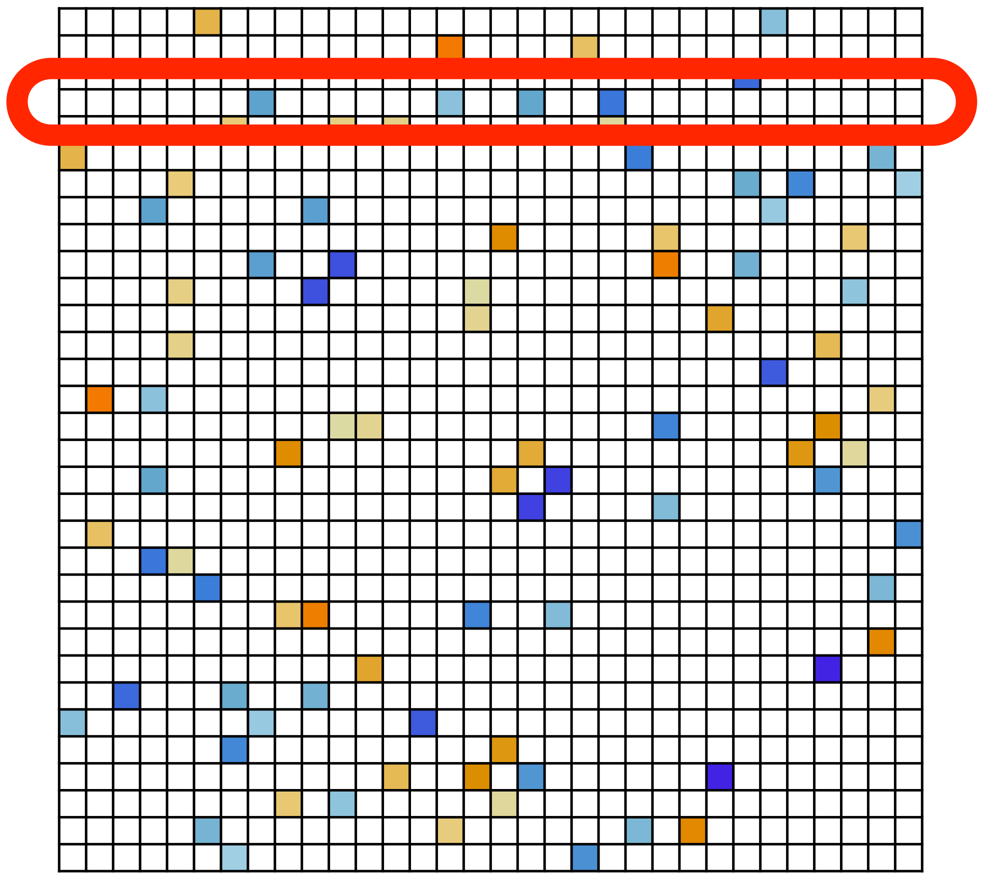
where H is $N \times N$

Sparse Hamiltonians

More general class of Hamiltonians that can be simulated efficiently [Aharonov, Ta-Shma 03]:

At most d nonzero entries per row, $d = \mathrm{poly}(\log N)$ (where H is $N \times N$)

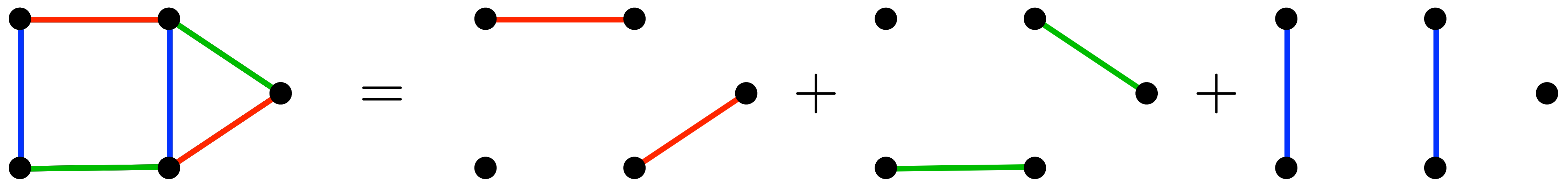
In any given row, the location of the jth nonzero entry and its value can be computed efficiently (or is given by a black box)

$H =$ 

Note: A k-local Hamiltonian with m terms is d-sparse with $d = 2^k m$

Sparse Hamiltonians and coloring

Strategy [Childs, Cleve, Deotto, Farhi, Gutmann, Spielman 03; Aharonov, Ta-Shma 03]: Color the edges of the graph of H. Then the simulation breaks into small pieces that are easy to handle.



A sparse graph can be efficiently colored using only local information [Linial 87], so this gives efficient simulations.

Can also use other decompositions (e.g., stars [Childs, Kothari 10])

Higher-order product formulas

$$\left(e^{-iAt/r}e^{-iBt/r}\right)^r = e^{-i(A+B)t} + O(t^2/r)$$
$$\left(e^{-iAt/2r}e^{-iBt/r}e^{-iAt/2r}\right)^r = e^{-i(A+B)t} + O(t^3/r^2)$$
$$\vdots$$

[Suzuki 91]: Systematic construction of arbitrarily high-order formulas

Number of terms in the formula grows exponentially with order

Complexity of best known simulation, using pth order:

$$O\left(5^{2p}d^3\|H\|t\left(\frac{d\|H\|t}{\epsilon}\right)^{1/2p}\right)$$

[Berry, Ahokas, Cleve, Sanders 07; Childs, Kothari 11; Berry, Childs, Cleve, Kothari, Somma 14]

High-precision simulation

We have recently developed a novel approach that directly implements the Taylor series of the evolution operator

New tools:

- Implementing linear combinations of unitary operations
- Oblivious amplitude amplification

Dependence on simulation error is $\mathrm{poly}(\log(1/\epsilon))$, an exponential improvement over previous work

Algorithms are also simpler, with less overhead

[Berry, Childs, Cleve, Kothari, Somma 14 & 15]

Linear combinations of unitaries

LCU Lemma: Given the ability to perform unitaries V_j with unit complexity, one can perform the operation $U = \sum_j \beta_j V_j$ with complexity $O(\sum_j |\beta_j|)$. Furthermore, if U is (nearly) unitary then this implementation can be made (nearly) deterministic.

Main ideas:

- Using controlled-V_j operations, implement U with some amplitude:

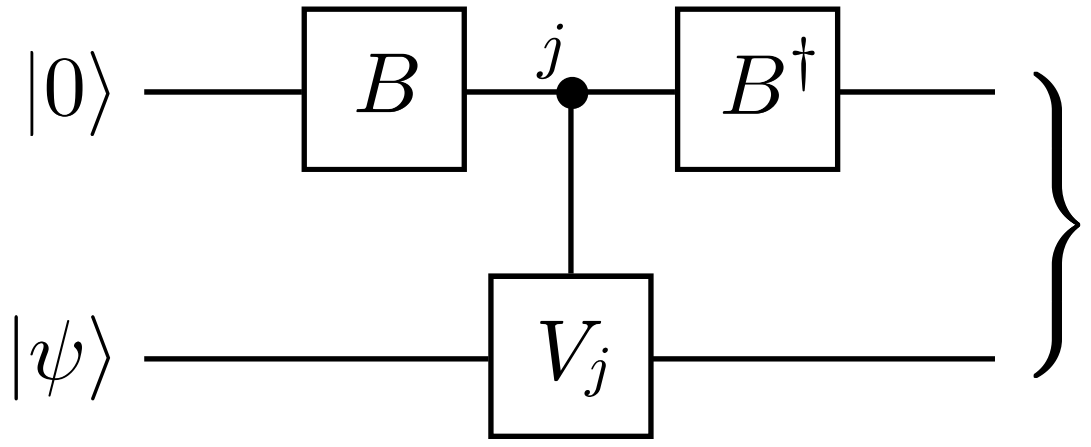
$$|0\rangle|\psi\rangle \mapsto \sin\theta|0\rangle U|\psi\rangle + \cos\theta|\Phi\rangle$$

- Boost the amplitude for success by oblivious amplitude amplification

Implementing U with some amplitude

$$U = \sum_j \beta_j V_j \qquad \text{(WLOG } \beta_j > 0\text{)}$$

Ancilla state: $B|0\rangle = \frac{1}{\sqrt{s}} \sum_j \sqrt{\beta_j}|j\rangle \qquad s := \sum_j \beta_j$



$$\frac{1}{s}|0\rangle\, U|\psi\rangle + \sqrt{1 - \frac{1}{s^2}}\,|\Phi\rangle$$

with $\langle 0|\Phi\rangle = 0$

Oblivious amplitude amplification

Suppose W implements U with amplitude $\sin\theta$:

$$W|0\rangle|\psi\rangle = \sin\theta|0\rangle U|\psi\rangle + \cos\theta|\Phi\rangle$$

To perform U with amplitude close to 1: use amplitude amplification?

But the input state is unknown!

Using ideas from [Marriott, Watrous 05], we can show that a $|\psi\rangle$-independent reflection suffices to do effective amplitude amplification.

With this *oblivious amplitude amplification*, we can perform the ideal evolution with only about $1/\sin\theta$ steps.

We also give a robust version that works even when U is not exactly unitary.

Simulating the Taylor series

Taylor series of the dynamics generated by H:

$$e^{-iHt} = \sum_{k=0}^{\infty} \frac{(-iHt)^k}{k!}$$
$$\approx \sum_{k=0}^{K} \frac{(-iHt)^k}{k!}$$

Write $H = \sum_\ell \alpha_\ell H_\ell$ where each H_ℓ is unitary

Then $e^{-iHt} \approx \sum_{k=0}^{K} \sum_{\ell_1,\ldots,l_k} \frac{(-it)^k}{k!} \alpha_{\ell_1} \cdots \alpha_{\ell_k} H_{\ell_1} \cdots H_{\ell_k}$

is a linear combination of unitaries

Decomposing sparse Hamiltonians

To express H as a linear combination of unitaries:

- Edge coloring: $H = \sum_{j=1}^{d^2} H_j$ where each H_j is 1-sparse

  new trick: H is bipartite wlog since it suffices to simulate $H \otimes \sigma_x$

  d^2-coloring: $\mathrm{color}(\ell, r) = (\mathrm{idx}(\ell, r), \mathrm{idx}(r, \ell))$

- Approximately decompose into terms with all nonzero entries equal

Ex: $$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 3 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

- Remove zero blocks so that all terms are rescaled unitaries

Ex: $$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Why $\mathrm{poly}(\log(1/\epsilon))$?

Lowest-order product formula:

$$(e^{-iA/r}e^{-iB/r})^r = e^{-i(A+B)} + O(1/r)$$

so we must take $r = O(1/\epsilon)$ to achieve error at most ϵ

Higher-order formulas exist, but they only improve the power of ϵ

The approximation $e^{-iHt} \approx \sum_{k=0}^{K} \frac{(-iHt)^k}{k!}$ has error ϵ provided

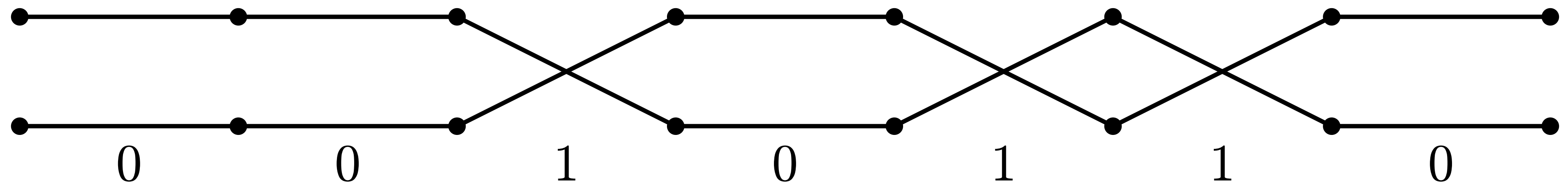
$$K = O\left(\frac{\log(1/\epsilon)}{\log\log(1/\epsilon)}\right)$$

Lower bounds

No-fast-forwarding theorem [BACS 07]: $\Omega(t)$

Main idea:

- Query complexity of computing the parity of n bits is $\Omega(n)$.
- There is a Hamiltonian that can compute parity by running for time $O(n)$.



New lower bound: $\Omega(\frac{\log(1/\epsilon)}{\log\log(1/\epsilon)})$

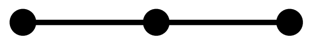
Main idea:

- Query complexity of parity is $\Omega(n)$ even for *unbounded error*.
- The same Hamiltonian as above computes parity with unbounded error by running for any positive time. Running for constant time gives the parity with probability $\Theta(1/n!)$.

Discrete-time quantum walk

Quantum walk: quantum analog of a random walk on a graph

In general, locality is inconsistent with unitarity [Severini 03]

Ex:  no unitary matrix has this pattern: $\begin{pmatrix} 0 & \bullet & 0 \\ \bullet & 0 & \bullet \\ 0 & \bullet & 0 \end{pmatrix}$

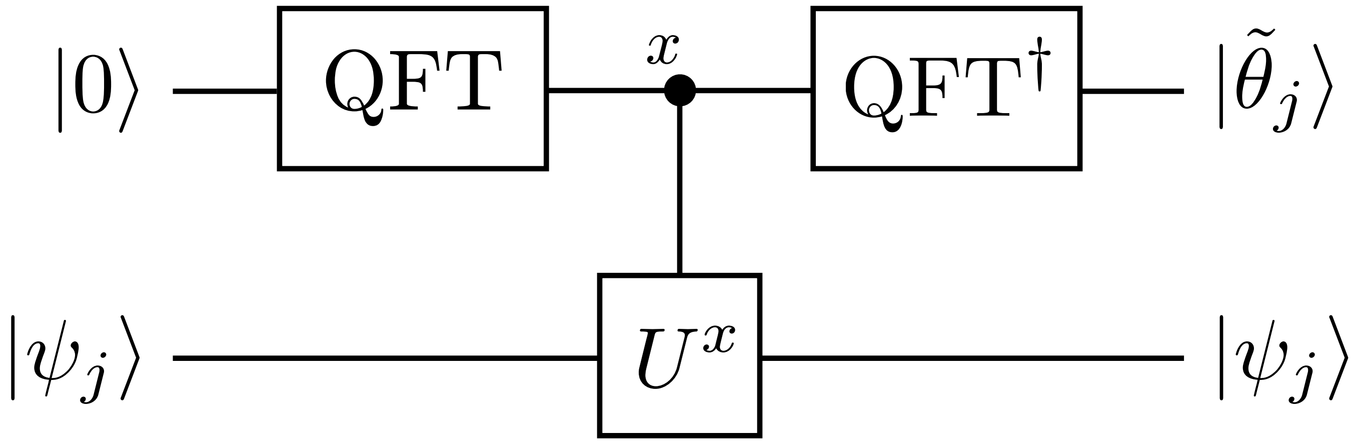
Natural definition of a discrete-time quantum walk [Szegedy 04]:

- Represent state by two locations: $|u, v\rangle$ (u is the current vertex; v is the next vertex)
- Conditioned on u, reflect about some superposition of its neighbors
- Swap the two registers

Such walks have many nice properties and have been used extensively to construct quantum algorithms

Phase estimation

Problem: Given a unitary operator U with eigenvectors $|\psi_j\rangle$, where $U|\psi_j\rangle = e^{i\theta_j}|\psi_j\rangle$, produce an estimate of θ_j



$$\sum_j \alpha_j |0\rangle|\psi_j\rangle \mapsto \sum_j \alpha_j |\tilde{\theta}_j\rangle|\psi_j\rangle$$

To get an estimate with precision ϵ, we need $O(1/\epsilon)$ uses of U.

[Kitaev 95]

Quantum walk simulation

Define a Szegedy walk operator for any given Hamiltonian H

Each eigenvalue λ of H corresponds to two eigenvalues $\pm e^{\pm i \arcsin \lambda}$ of the walk operator (with eigenvectors closely related to those of H)

Strategy: Use phase estimation to determine and correct the phase

$$\begin{aligned} |\psi\rangle &\mapsto |\psi\rangle|\widetilde{\arcsin \lambda}\rangle \\ &\mapsto e^{-i\lambda t}|\psi\rangle|\widetilde{\arcsin \lambda}\rangle \\ &\mapsto e^{-i\lambda t}|\psi\rangle \end{aligned}$$

Complexity: $O(\tau/\sqrt{\epsilon})$ where $\tau := d\|H\|_{\max} t$

This matches the no fast-forwarding bound: real-time simulation!

[Childs 10], [Berry, Childs 12]

Linear combination of quantum walk steps

Another approach: find coefficients so that

$$e^{-iH} \approx T^\dagger \sum_{k=-K}^{K} \beta_k U^k \, T$$

and implement this using the LCU Lemma

By a generating series for Bessel functions,

$$e^{-i\lambda t} = \sum_{k=-\infty}^{\infty} J_k(-t)\, e^{ik \arcsin \lambda}$$

Coefficients drop off rapidly for large k, so we can truncate the series

Query complexity of this approach: $O\left(\tau \frac{\log(\tau/\epsilon)}{\log\log(\tau/\epsilon)}\right)$

$$\tau := d\|H\|_{\max} t$$

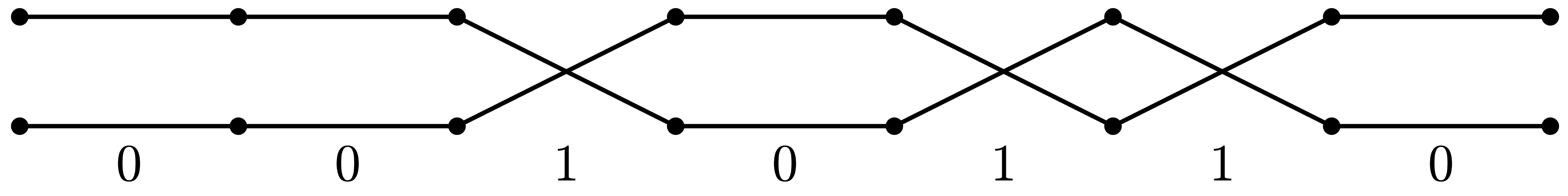
[Berry, Childs, Kothari 15]

Lower bounds revisited

No-fast-forwarding theorem [BACS 07]: $\Omega(t)$

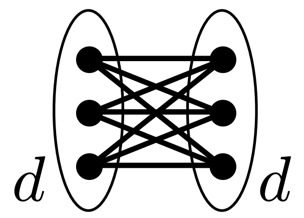
Main idea:

- Query complexity of computing the parity of n bits is $\Omega(n)$.
- There is a Hamiltonian that can compute parity by running for time $O(n)$.



New lower bound: $\Omega(dt)$

- Replacing each edge with $K_{d,d}$ effectively boosts Hamiltonian by d.



$\Rightarrow$ Our algorithm is (nearly) optimal with respect to each of t, d, and ϵ

Query complexity of sparse Hamiltonian simulation

Quantum walk + phase estimation [BC 10]: $O\left(\frac{\tau}{\sqrt{\epsilon}}\right)$ $\tau := d\|H\|_{\max}t$

Quantum walk + LCU [BCK 15]: $O\left(\tau\frac{\log(\tau/\epsilon)}{\log\log(\tau/\epsilon)}\right)$

or for $\alpha \in (0,1]$: $O\big(\tau^{1+\alpha/2} + \tau^{1-\alpha/2}\log(1/\epsilon)\big)$

Lower bound: $\Omega\big(\tau + \frac{\log(1/\epsilon)}{\log\log(1/\epsilon)}\big)$

Notes:

- Gate complexity is only slightly larger than query complexity
- These techniques assume time-independent Hamiltonians (otherwise, use fractional queries/LCU on Dyson series [BCCKS 14])

Summary

Product formulas are the obvious approach to quantum simulation, but they are suboptimal!

Recently-developed algorithms are both
- asymptotically faster
- likely to be competitive in practice

Quantum simulation will probably be the first practical application of quantum computers; recent advances make this all the more likely

Outlook

Improved simulation algorithms

- Optimal tradeoff for sparse Hamiltonian simulation
- Faster algorithms for structured problems
- Simulating open quantum systems

Applications to simulating physics

- What is the cost in practice for simulating molecular systems?
- How do recent algorithms compare to naive methods?

New quantum algorithms

- Improved algorithms for linear systems
- New applications of linear systems
- Other quantum algorithms from quantum simulation