Hamiltonian simulation and solving linear systems

Robin Kothari

Center for Theoretical Physics

MIT

Quantum Optimization Workshop
Fields Institute
October 28, 2014

“Ask not what you can do for quantum computing—ask what quantum computing can do for you”

Polynomial vs exponential speedups

Polynomial speedup

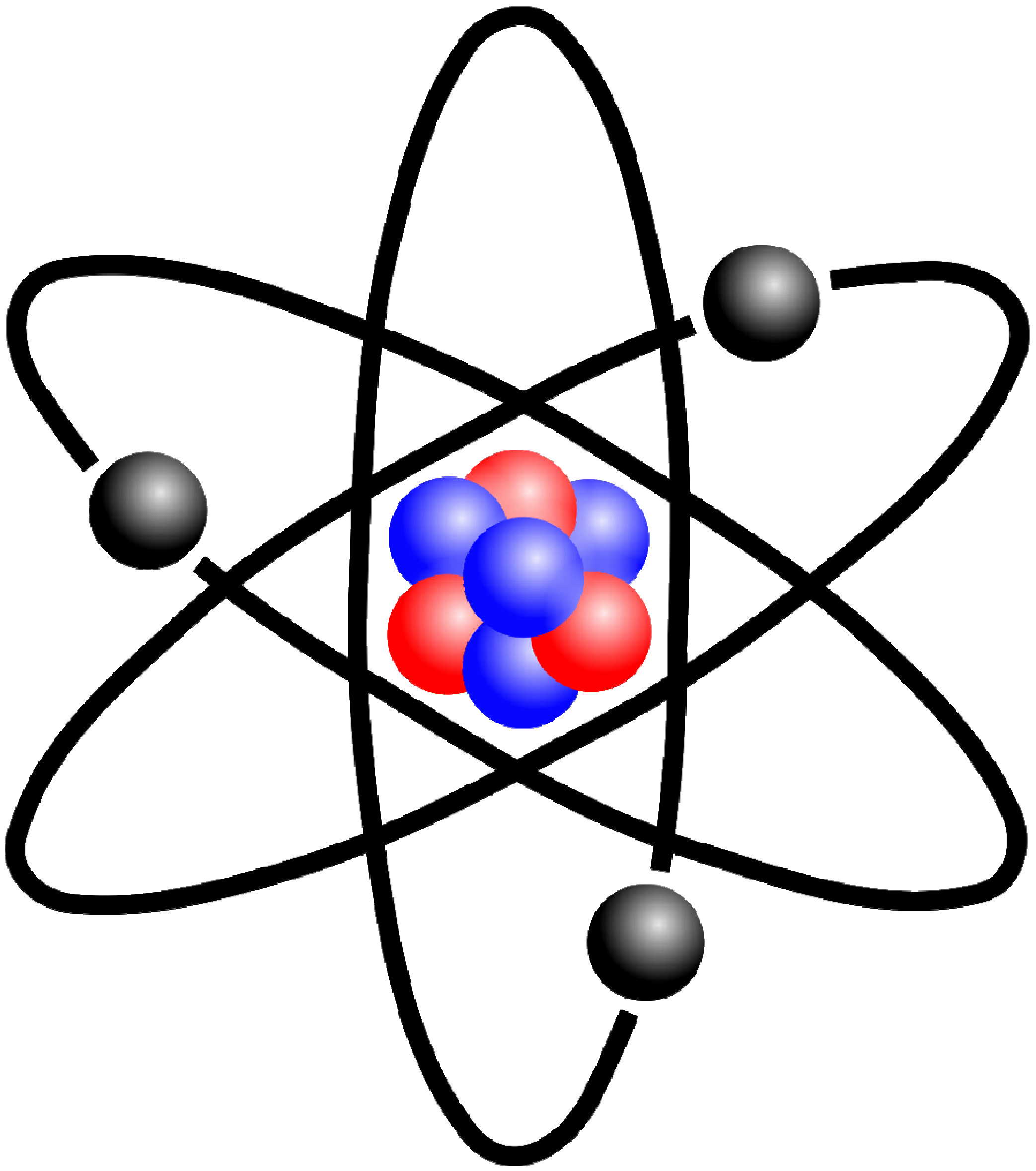
- Grover's algorithm
- Amplitude amplification
- Algorithms based on quantum walk search
- Triangle finding and other graph properties, element distinctness, matrix multiplication, formula evaluation, etc.

Exponential speedup

- Shor's algorithm (for factoring and discrete log)
- Abelian hidden subgroup
- Hamiltonian simulation
- Solving linear systems of equations (explained later)
- Computing topological invariants

Part I: Hamiltonian Simulation

Simulating physical systems



Simulating physical systems

General problem: Given the description of a physical system and an initial state, compute the final state of the system after some time.

Example (classical)

Physical system: n bodies under gravitational force

Initial state: initial positions and velocities of all n bodies

Final state: final positions and velocities of all n bodies

Example (quantum)

Physical system: n qubits with Hamiltonian H

Initial state: $|\Psi_i\rangle$

Final state: $|\Psi_f\rangle$

For a time-independent Hamiltonian H, $|\Psi_f\rangle = e^{-iHt} |\Psi_i\rangle$

Hamiltonian simulation problem (informal): Given a Hamiltonian H and a time t, (approximately) perform e^{-iHt} on an input state.

Hamiltonian simulation: motivation

- Simulating physical quantum systems
  - Original application of quantum computers [Feynman82]
  - Significant fraction of world's computing power devoted to simulating physical systems that arise in quantum chemistry, condensed matter physics, materials science, etc.
  - No known efficient classical algorithm (and we don't expect one, unless quantum computers are useless)

- Algorithmic applications: can be used as a subroutine to
  - Implement continuous-time quantum walks [CCDFGS03]
  - Evaluate the output of game trees [FGG08]
  - Solve linear equations [HHL09]

Simulating quantum systems

Hamiltonian simulation problem

Given a Hamiltonian (a Hermitian matrix) H of size N x N, a time t, and $\epsilon>0$, perform the unitary e^{-iHt} with error at most ϵ.

We would like an efficient quantum algorithm for this problem

But what is an efficient algorithm?

Polynomial time (in the size of the system), i.e., poly(log N, t)

Scaling with ϵ?	poly($1/\epsilon$)	OK
	log($1/\epsilon$)	much better

<u>Quantum computers cannot simulate all Hamiltonians efficiently!</u>

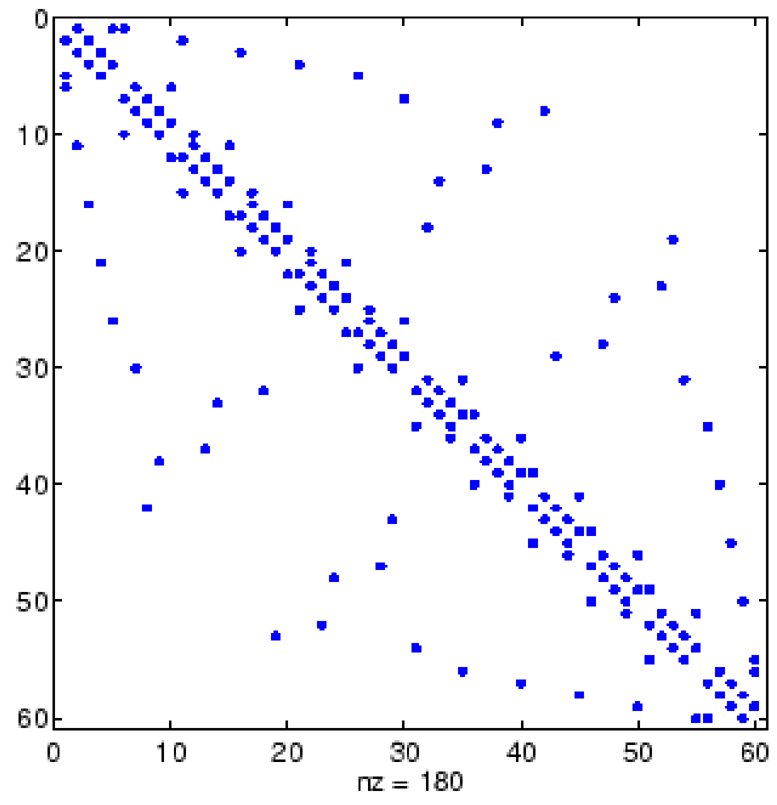
Quantum computers can efficiently simulate, for example,

Local Hamiltonians: Sum of terms each acting on O(1) qubits.

Sparse Hamiltonians: Each row of H has poly(log N) nonzero entries.

How is the input represented?

Input: H, t, and ϵ.



Local Hamiltonians

•Specify H by listing all terms.

Sparse Hamiltonians

•Can have exponentially many (exponential in log N) nonzero entries. No explicit polynomial size description.

•Assume Hamiltonian is <u>row computable</u>, i.e., there is an efficient algorithm to determine the jth nonzero entry of the ith row of H.

Hamiltonian simulation algorithms

Algorithms based on

1. Product formulas [Llo96], [AT03], [BACS07]. Best: [Childs-K. 2011]
2. Quantum walks [Chi10]. Best: [Berry-Childs 2012]
3. Fractional queries. [Berry-Childs-Cleve-K.-Somma 2013]
4. Linear combination of quantum walks. [Berry-Childs-K. 2014]

d = sparsity t = time ϵ = allowed error

Dependence	**On d**	**On t**	**On ϵ**
Best possible	d	t	$\log(1/\epsilon)/\log\log(1/\epsilon)$

Simulation vs. finding ground states:

Two problems that are often confused, but are very different.

Simulate a system

- Predict the behavior of a system
- Easy (in P, BQP, etc.)
- Examples:

1. Predict output of a given Boolean circuit on input x
2. Predict output of quantum circuit on input $|\Psi\rangle$

Simulate [v]: To model, replicate, duplicate the behavior, appearance or properties of

Find a ground state

- Optimize a global property of a system
- Hard (NP-hard, QMA-hard)
- Examples:

1. Find an input that satisfies a given Boolean circuit
2. Compute max. acceptance probability of quantum circuit

Find ground state = solve an optimization problem over an exponentially large set

Part II: Solving linear equations

Solving linear equations

Input: An N x N matrix A and a vector b in $\mathbb{C}^N$.

Goal: To solve the equation

$$Ax = b$$

i.e., to compute (approximately) $x = A^{-1}b$

Explicit representation

The inputs A and b are written out explicitly

Best classical and quantum algorithms necessarily run in time poly(N).

Quantum computers cannot give exponential speedup for this!

Solving linear equations (modified)

Goal: To solve the equation

$$Ax = b$$

i.e., to compute (approximately) $x = A^{-1}b$

Modified problem

Assume A is d-sparse and has an efficient black-box representation for the entries (same black box as before)

Assume b is a vector for which the quantum state $|b\rangle := b/\|b\|$ can be created efficiently (in time polylog N)

New objective: Create the quantum state corresponding to x, i.e.,

$|x\rangle := x/\|x\|$.

Solving linear equations (modified)

New objective: Output an approximation to $|x\rangle := x/\|x\|$.

Best quantum algorithm [Harrow–Hassidim–Lloyd 2009] runs in time $O(\log(N)\ \mathrm{poly}(d,\kappa)\ \epsilon^{-1})$, where

N: number of rows or columns of the matrix A

d: sparsity of A (max number of nonzero entries per row/column)

κ: condition number of A, i.e., $\kappa := \|A\|\ \|A^{-1}\|$

ϵ: approximation error (output is ϵ-close to ideal output)

Tools used: Hamiltonian simulation and phase estimation

Classical matrix inversion algorithms run in poly(N) time. Thus we have an exponential speedup if d, κ, and ϵ^{-1} are all polylog(N).

Classically, a poly(log N, κ, ϵ^{-1}) algorithm is impossible, unless quantum computers are useless.

Solving linear equations: summary

What we can do on a quantum computer

Given A (a sparse matrix) and b (a vector that can be created efficiently on a quantum computer), we can approximately create the quantum state $|x\rangle = A^{-1}b \,/\, \|A^{-1}b\|$ in time poly(log N, d, κ, ϵ^{-1})

This bring up (at least) two obvious questions

1. Which states $|b\rangle$ can we create efficiently?

Difficult to characterize precisely. Examples include

- All ones vector
- All entries b_i satisfy $|b_i|=1$ and can be computed efficiently
- Entries such that partial sums of b_i are efficiently computable
- Only polylog(N) nonzero entries in b

2. What can we do with $|x\rangle$?

Solving linear equations: summary

What we can do on a quantum computer

Given A (a sparse matrix) and b (a vector that can be created efficiently on a quantum computer), we can approximately create the quantum state $|x\rangle = A^{-1}b / \|A^{-1}b\|$ in time poly(log N, d, κ, ϵ^{-1})

This bring up (at least) two obvious questions

1. Which states $|b\rangle$ can we create efficiently?
2. What can we do with $|x\rangle$?
    - Measure. If the amplitudes are x_i, we have $\Pr(i)=|x_i|^2$
    - Apply a unitary and then measure, e.g., Fourier transform.
    - Swap test. Given two states $|x\rangle$ and $|y\rangle$, the swap test allows us to distinguish between $|x\rangle \approx |y\rangle$ and $|x\rangle \perp |y\rangle$.

Open problems

Hamiltonian simulation

- Further improve current algorithms and simplify them.
- Customize algorithms to Hamiltonians that arise in practice.
- Precise estimates for gate complexity of these algorithms.
- Real implementations?

Solving linear equations

- Find applications!
- Some known applications:
  - Solving linear differential equations [Berry 2010]
  - Quantum algorithms for data fitting [Wiebe-Braun-Lloyd 2012]
  - Machine learning problems [Lloyd-Mohseni-Rebentrost 2013]

Thank you