Quantum Algorithms

Ashley Montanaro

School of Mathematics,
University of Bristol

13 January 2017

University of Bristol

EPSRC
Engineering and Physical Sciences Research Council

Introduction

What can we do with our (universal, general-purpose) quantum computers?

Introduction

What can we do with our (universal, general-purpose) quantum computers?

In this talk, I will discuss some classic applications, as well as more recent applications.

Introduction

What can we do with our (universal, general-purpose) quantum computers?

In this talk, I will discuss some classic applications, as well as more recent applications.

The Quantum Algorithm Zoo (`math.nist.gov/quantum/zoo/`) cites ~~279~~ 328 papers on quantum algorithms, so this is necessarily a partial view…

Integer factorisation

Problem

Given an n-digit integer $N = p \times q$ for primes p and q, determine p and q.

Integer factorisation

Problem

Given an n-digit integer $N = p \times q$ for primes p and q, determine p and q.

- The best (classical!) algorithm we have for factorisation (the number field sieve) runs in time

$$\exp(O(n^{1/3}(\log n)^{2/3}))$$

Integer factorisation

Problem

Given an n-digit integer $N = p \times q$ for primes p and q, determine p and q.

- The best (classical!) algorithm we have for factorisation (the number field sieve) runs in time

$$\exp(O(n^{1/3}(\log n)^{2/3}))$$

- The RSA cryptosystem that underlies Internet security is based around the hardness of this task.
- That is, if we can factorise large integers efficiently, we can break RSA.

Integer factorisation

Problem

Given an n-digit integer $N = p \times q$ for primes p and q, determine p and q.

- The best (classical!) algorithm we have for factorisation (the number field sieve) runs in time

$$\exp(O(n^{1/3}(\log n)^{2/3}))$$

- The RSA cryptosystem that underlies Internet security is based around the hardness of this task.
- That is, if we can factorise large integers efficiently, we can break RSA.

Theorem [Shor '97]

There is a quantum algorithm which finds the prime factors of an n-digit integer in time $O(n^3)$.

Shor's algorithm: complexity comparison

Very roughly (ignoring constant factors!):

Number of digits	Timesteps (quantum)	Timesteps (classical)
100	10^6	$\sim 4 \times 10^5$
1,000	10^9	$\sim 5 \times 10^{15}$
10,000	10^{12}	$\sim 1 \times 10^{41}$

Shor's algorithm: complexity comparison

Very roughly (ignoring constant factors!):

Number of digits	Timesteps (quantum)	Timesteps (classical)
100	10^6	$\sim 4 \times 10^5$
1,000	10^9	$\sim 5 \times 10^{15}$
10,000	10^{12}	$\sim 1 \times 10^{41}$

Based on these figures, a 10,000-digit number could be factorised by:

- A quantum computer with a clock speed of 1MHz in 11 days.

Shor's algorithm: complexity comparison

Very roughly (ignoring constant factors!):

Number of digits	Timesteps (quantum)	Timesteps (classical)
100	10^6	$\sim 4 \times 10^5$
1,000	10^9	$\sim 5 \times 10^{15}$
10,000	10^{12}	$\sim 1 \times 10^{41}$

Based on these figures, a 10,000-digit number could be factorised by:

- A quantum computer with a clock speed of 1MHz in 11 days.
- The fastest computer on the Top500 supercomputer list ($\sim 9.3 \times 10^{16}$ operations per second) in $\sim 3.4 \times 10^{16}$ years.

(see e.g. [Van Meter et al '05] for a more detailed comparison)

Grover's algorithm

One of the most basic problems in computer science is unstructured search.

Grover's algorithm

One of the most basic problems in computer science is unstructured search.

- Imagine we have access to a function $f : \{0,1\}^n \to \{0,1\}$ which we treat as a black box.

Grover's algorithm

One of the most basic problems in computer science is unstructured search.

- Imagine we have access to a function $f : \{0,1\}^n \to \{0,1\}$ which we treat as a black box.
- We want to find an x such that $f(x) = 1$.

0	0	1	0	0	0	1	0

Grover's algorithm

One of the most basic problems in computer science is unstructured search.

- Imagine we have access to a function $f : \{0,1\}^n \to \{0,1\}$ which we treat as a black box.

- We want to find an x such that $f(x) = 1$.

0	0	1	0	0	0	1	0

- On a classical computer, this task could require 2^n queries to f in the worst case. But on a quantum computer, Grover's algorithm [Grover '97] can solve the problem with $O(\sqrt{2^n})$ queries to f (and bounded failure probability).

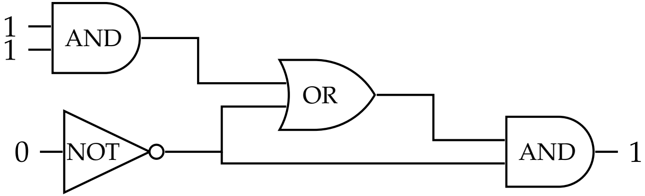
Applications of Grover's algorithm

Grover's algorithm gives a speedup over naïve algorithms for any decision problem in the complexity class NP, i.e. where we can verify the solution efficiently.

Applications of Grover's algorithm

Grover's algorithm gives a speedup over naïve algorithms for any decision problem in the complexity class NP, i.e. where we can verify the solution efficiently.

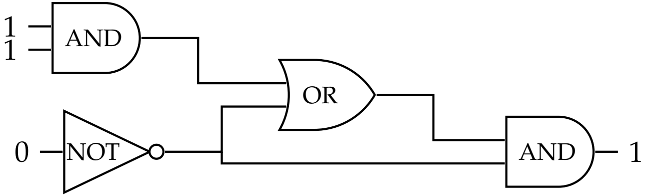
- For example, in the Circuit SAT problem we would like to find an input to a circuit on n bits such that the output is 1:



Applications of Grover's algorithm

Grover's algorithm gives a speedup over naïve algorithms for any decision problem in the complexity class NP, i.e. where we can verify the solution efficiently.

- For example, in the Circuit SAT problem we would like to find an input to a circuit on n bits such that the output is 1:



- Grover's algorithm improves the runtime from $O(2^n)$ to $O(2^{n/2})$: applications to design automation, circuit equivalence, model checking, ...

Applications of Grover's algorithm

An important generalisation of Grover's algorithm is known as amplitude amplification.

Amplitude amplification [Brassard et al '00]

Assume we are given access to a "checking" function f, and a probabilistic algorithm $\mathcal{A}$ such that

$$\Pr[\mathcal{A} \text{ outputs } w \text{ such that } f(w) = 1] = \epsilon.$$

Then we can find w such that $f(w) = 1$ with $O(1/\sqrt{\epsilon})$ uses of f.

Gives a quadratic speed-up over classical algorithms which are based on heuristics.

Applications of Grover's algorithm

These primitives can be used to obtain many speedups over classical algorithms, e.g.:

- Finding the minimum of n numbers in $O(\sqrt{n})$ time [Dürr and Høyer '96]

Applications of Grover's algorithm

These primitives can be used to obtain many speedups over classical algorithms, e.g.:

- Finding the minimum of n numbers in $O(\sqrt{n})$ time [Dürr and Høyer '96]
- Determining connectivity of an n-vertex graph in $O(n^{3/2})$ time [Dürr et al '04]

Applications of Grover's algorithm

These primitives can be used to obtain many speedups over classical algorithms, e.g.:

- Finding the minimum of n numbers in $O(\sqrt{n})$ time [Dürr and Høyer '96]
- Determining connectivity of an n-vertex graph in $O(n^{3/2})$ time [Dürr et al '04]
- Finding a collision in a 2-1 function $f : [n] \to [n]$ in $O(n^{1/3})$ time [Brassard et al '98]
- …

Applications of Grover's algorithm

These primitives can be used to obtain many speedups over classical algorithms, e.g.:

- Finding the minimum of n numbers in $O(\sqrt{n})$ time [Dürr and Høyer '96]
- Determining connectivity of an n-vertex graph in $O(n^{3/2})$ time [Dürr et al '04]
- Finding a collision in a 2-1 function $f : [n] \to [n]$ in $O(n^{1/3})$ time [Brassard et al '98]
- …

They can also speed up Monte Carlo methods [AM '15]:

- The mean of a random variable with variance σ^2 can be approximated up to ϵ in time roughly $O(\sigma/\epsilon)$, as opposed to the classical $O(\sigma^2/\epsilon^2)$.

Quantum simulation

The most important early application of quantum computers is likely to be quantum simulation.

Quantum simulation

The most important early application of quantum computers is likely to be quantum simulation.

- Here, we use "simulation" to mean approximating the dynamical properties of a quantum system.

Problem

Given a Hamiltonian H describing a physical system, and an initial state $|\psi_0\rangle$ of that system, produce the state

$$|\psi_t\rangle = e^{-iHt}|\psi_0\rangle.$$

Given such an output state, measurements can be performed to determine quantities of interest about the state.

Quantum simulation

The most important early application of quantum computers is likely to be quantum simulation.

- Here, we use "simulation" to mean approximating the dynamical properties of a quantum system.

Problem

Given a Hamiltonian H describing a physical system, and an initial state $|\psi_0\rangle$ of that system, produce the state

$$|\psi_t\rangle = e^{-iHt}|\psi_0\rangle.$$

Given such an output state, measurements can be performed to determine quantities of interest about the state.

- No efficient classical algorithm is known for this task (in full generality), but efficient quantum algorithms exist for many physically reasonable cases.

Quantum simulation

Applications of quantum simulation include quantum chemistry, superconductivity, metamaterials, high-energy physics, . . . [Georgescu et al '13]

Some recent examples:

- The Hubbard model used in the study of superconductivity [Wecker et al '15]
- Quantum chemistry [Hastings et al '14] [Wecker et al '14]
- Quantum field theories [Jordan et al '11]

Quantum simulation

Applications of quantum simulation include quantum chemistry, superconductivity, metamaterials, high-energy physics, . . . [Georgescu et al '13]

Some recent examples:

- The Hubbard model used in the study of superconductivity [Wecker et al '15]
- Quantum chemistry [Hastings et al '14] [Wecker et al '14]
- Quantum field theories [Jordan et al '11]

Many static properties of quantum systems are also interesting (e.g. ground-state energy).

- There is good evidence that these are hard to compute in the worst case, but may be easy for physical systems of interest.

“Solving” linear equations

A basic task in mathematics and engineering:

Solving linear equations

Given access to a d-sparse $N \times N$ matrix A, and $b \in \mathbb{R}^N$, output x such that $Ax = b$.

“Solving” linear equations

A basic task in mathematics and engineering:

Solving linear equations

Given access to a d-sparse $N \times N$ matrix A, and $b \in \mathbb{R}^N$, output x such that $Ax = b$.

One “quantum” way of thinking about the problem:

“Solving” linear equations

Given the ability to produce the quantum state $|b\rangle = \sum_{i=1}^{N} b_i |i\rangle$, and access to A as above, produce the state $|x\rangle = \sum_{i=1}^{N} x_i |i\rangle$.

“Solving” linear equations

A basic task in mathematics and engineering:

Solving linear equations

Given access to a d-sparse $N \times N$ matrix A, and $b \in \mathbb{R}^N$, output x such that $Ax = b$.

One “quantum” way of thinking about the problem:

“Solving” linear equations

Given the ability to produce the quantum state $|b\rangle = \sum_{i=1}^{N} b_i |i\rangle$, and access to A as above, produce the state $|x\rangle = \sum_{i=1}^{N} x_i |i\rangle$.

Theorem: If A has condition number κ $(= \|A^{-1}\| \|A\|)$, $|x\rangle$ can be approximately produced in time $\mathrm{poly}(\log N, d, \kappa)$ [Harrow et al ’08] [Ambainis ’10] [Berry et al ’15].

Notes on this algorithm

The algorithm (approximately) produces a state $|x\rangle$ such that we can extract some information from $|x\rangle$. Is this useful?

Notes on this algorithm

The algorithm (approximately) produces a state $|x\rangle$ such that we can extract some information from $|x\rangle$. Is this useful?

- We could use this to e.g. determine whether two sets of linear equations have (approximately) the same solution – not clear how to do this classically.

Notes on this algorithm

The algorithm (approximately) produces a state $|x\rangle$ such that we can extract some information from $|x\rangle$. Is this useful?

- We could use this to e.g. determine whether two sets of linear equations have (approximately) the same solution – not clear how to do this classically.
- Achieving a similar runtime classically would imply that all quantum computations could be simulated!

Notes on this algorithm

The algorithm (approximately) produces a state $|x\rangle$ such that we can extract some information from $|x\rangle$. Is this useful?

- We could use this to e.g. determine whether two sets of linear equations have (approximately) the same solution – not clear how to do this classically.
- Achieving a similar runtime classically would imply that all quantum computations could be simulated!

Some applications of this algorithm include:

- Computing electromagnetic scattering cross-sections using the finite element method [Clader et al ’13] [AM and Pallister ’16]

Notes on this algorithm

The algorithm (approximately) produces a state $|x\rangle$ such that we can extract some information from $|x\rangle$. Is this useful?

- We could use this to e.g. determine whether two sets of linear equations have (approximately) the same solution – not clear how to do this classically.
- Achieving a similar runtime classically would imply that all quantum computations could be simulated!

Some applications of this algorithm include:

- Computing electromagnetic scattering cross-sections using the finite element method [Clader et al '13] [AM and Pallister '16]
- "Solving" differential equations [Leyton and Osborne '08] [Berry '14]

Notes on this algorithm

The algorithm (approximately) produces a state $|x\rangle$ such that we can extract some information from $|x\rangle$. Is this useful?

- We could use this to e.g. determine whether two sets of linear equations have (approximately) the same solution – not clear how to do this classically.
- Achieving a similar runtime classically would imply that all quantum computations could be simulated!

Some applications of this algorithm include:

- Computing electromagnetic scattering cross-sections using the finite element method [Clader et al '13] [AM and Pallister '16]
- "Solving" differential equations [Leyton and Osborne '08] [Berry '14]
- Recommendation systems [Kerenidis and Prakash '16]

Notes on this algorithm

The algorithm (approximately) produces a state $|x\rangle$ such that we can extract some information from $|x\rangle$. Is this useful?

- We could use this to e.g. determine whether two sets of linear equations have (approximately) the same solution – not clear how to do this classically.
- Achieving a similar runtime classically would imply that all quantum computations could be simulated!

Some applications of this algorithm include:

- Computing electromagnetic scattering cross-sections using the finite element method [Clader et al '13] [AM and Pallister '16]
- "Solving" differential equations [Leyton and Osborne '08] [Berry '14]
- Recommendation systems [Kerenidis and Prakash '16]
- Space-efficient matrix inversion [Ta-Shma '13]

Quantum walks

A quantum walk on a graph is a quantum generalisation of a classical random walk.

Quantum walks

A quantum walk on a graph is a quantum generalisation of a classical random walk.

- Two variants: continuous-time and discrete-time.
- A discrete-time quantum walk is the quantum analogue of the simple discrete-time random walk.

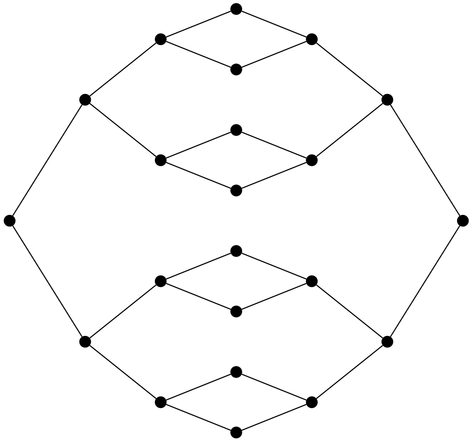
Quantum walks

A quantum walk on a graph is a quantum generalisation of a classical random walk.

- Two variants: continuous-time and discrete-time.
- A discrete-time quantum walk is the quantum analogue of the simple discrete-time random walk.
- A continuous-time quantum walk for time t on a graph with adjacency matrix A is the application of the unitary operator e^{-iAt}.
- Continuous-time quantum walks can be efficiently implemented as quantum circuits using Hamiltonian simulation.

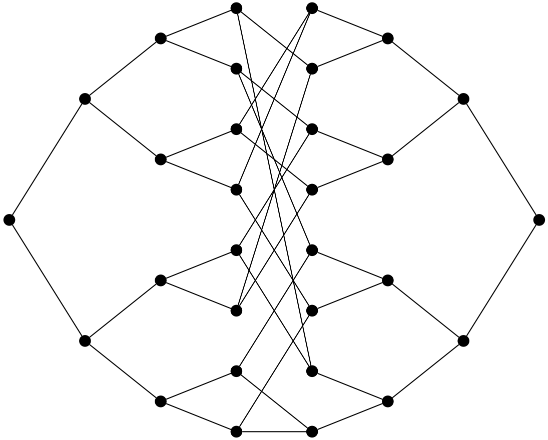
Quantum walks

Consider the graph formed by gluing two binary trees with N vertices together, e.g.:



Quantum walks

Now add a random cycle in the middle:



Quantum walk on the glued trees graph

Theorem [Childs et al '02]

- A continuous-time quantum walk which starts at the entrance (on the LHS) and runs for time $O(\log N)$ finds the exit (on the RHS) with probability at least $1/\operatorname{poly}(\log N)$.

Quantum walk on the glued trees graph

Theorem **[Childs et al '02]**

- A continuous-time quantum walk which starts at the entrance (on the LHS) and runs for time $O(\log N)$ finds the exit (on the RHS) with probability at least $1/\operatorname{poly}(\log N)$.
- Any classical algorithm given black-box access to the graph requires $O(N^{1/6})$ queries to find the exit.

Quantum walk on the glued trees graph

Theorem [Childs et al '02]

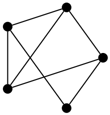
- A continuous-time quantum walk which starts at the entrance (on the LHS) and runs for time $O(\log N)$ finds the exit (on the RHS) with probability at least $1/\operatorname{poly}(\log N)$.
- Any classical algorithm given black-box access to the graph requires $O(N^{1/6})$ queries to find the exit.

This is an exponential separation, but no application of this result is known…

Applications of quantum walks

Quantum walks can be used to solve many different search problems, such as:

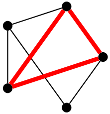
- Finding a triangle in a graph: $O(n^{1.25})$ queries, vs. classical $O(n^2)$ [Le Gall '14] [Jeffery et al '12] [Magniez et al '03]



Applications of quantum walks

Quantum walks can be used to solve many different search problems, such as:

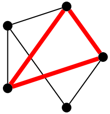
- Finding a triangle in a graph: $O(n^{1.25})$ queries, vs. classical $O(n^2)$ [Le Gall '14] [Jeffery et al '12] [Magniez et al '03]



Applications of quantum walks

Quantum walks can be used to solve many different search problems, such as:

- Finding a triangle in a graph: $O(n^{1.25})$ queries, vs. classical $O(n^2)$ [Le Gall '14] [Jeffery et al '12] [Magniez et al '03]



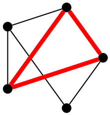
- Matrix product verification: $O(n^{5/3})$ queries, vs. classical $O(n^2)$ [Buhrman and Špalek '04]

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 3 \\ -2 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 5 & -2 \\ -1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \stackrel{?}{=} \begin{pmatrix} -1 & 4 & -3 \\ 1 & 5 & 4 \\ 1 & -9 & 5 \end{pmatrix}$$

Applications of quantum walks

Quantum walks can be used to solve many different search problems, such as:

- Finding a triangle in a graph: $O(n^{1.25})$ queries, vs. classical $O(n^2)$ [Le Gall '14] [Jeffery et al '12] [Magniez et al '03]



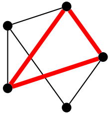
- Matrix product verification: $O(n^{5/3})$ queries, vs. classical $O(n^2)$ [Buhrman and Špalek '04]

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 3 \\ -2 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 5 & -2 \\ -1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \stackrel{?}{=} \begin{pmatrix} -1 & 4 & -3 \\ 1 & 5 & 4 \\ 1 & -9 & 5 \end{pmatrix}$$

Applications of quantum walks

Quantum walks can be used to solve many different search problems, such as:

- Finding a triangle in a graph: $O(n^{1.25})$ queries, vs. classical $O(n^2)$ [Le Gall '14] [Jeffery et al '12] [Magniez et al '03]



- Matrix product verification: $O(n^{5/3})$ queries, vs. classical $O(n^2)$ [Buhrman and Špalek '04]

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 3 \\ -2 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 5 & -2 \\ -1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \stackrel{?}{=} \begin{pmatrix} -1 & 4 & -3 \\ 1 & 5 & 4 \\ 1 & -9 & 5 \end{pmatrix}$$

- Whether n integers are all distinct: $O(n^{2/3})$ queries, vs. classical $O(n)$ [Ambainis '03]

Quantum speedup of backtracking algorithms

Backtracking is a general approach to solve constraint satisfaction problems (CSPs).

Quantum speedup of backtracking algorithms

Backtracking is a general approach to solve constraint satisfaction problems (CSPs).

- An instance of a CSP on n variables $x_1, \ldots, x_n$ is specified by a sequence of constraints, all of which must be satisfied by the variables.

Quantum speedup of backtracking algorithms

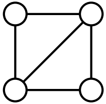
Backtracking is a general approach to solve constraint satisfaction problems (CSPs).

- An instance of a CSP on n variables $x_1, \dots, x_n$ is specified by a sequence of constraints, all of which must be satisfied by the variables.
- We might want to find one assignment to $x_1, \dots, x_n$ that satisfies all the constraints, or list all such assignments.

Quantum speedup of backtracking algorithms

Backtracking is a general approach to solve constraint satisfaction problems (CSPs).

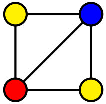
- An instance of a CSP on n variables $x_1, \ldots, x_n$ is specified by a sequence of constraints, all of which must be satisfied by the variables.
- We might want to find one assignment to $x_1, \ldots, x_n$ that satisfies all the constraints, or list all such assignments.
- A simple example: graph 3-colouring.



Quantum speedup of backtracking algorithms

Backtracking is a general approach to solve constraint satisfaction problems (CSPs).

- An instance of a CSP on n variables $x_1, \ldots, x_n$ is specified by a sequence of constraints, all of which must be satisfied by the variables.
- We might want to find one assignment to $x_1, \ldots, x_n$ that satisfies all the constraints, or list all such assignments.
- A simple example: graph 3-colouring.



Backtracking algorithms solve CSPs by "trial and error": exploring a tree of partial solutions.

Quantum speedup of backtracking algorithms

Theorem [AM '16] (informal)

If there is a classical backtracking algorithm which solves a CSP by exploring a tree of partial solutions of size T, there is a quantum algorithm that solves the CSP in time $O(\sqrt{T}\,\mathrm{poly}(n))$.

Quantum speedup of backtracking algorithms

Theorem [AM '16] (informal)

If there is a classical backtracking algorithm which solves a CSP by exploring a tree of partial solutions of size T, there is a quantum algorithm that solves the CSP in time $O(\sqrt{T}\,\mathrm{poly}(n))$.

This is a near-quadratic speedup, assuming that $T \gg \mathrm{poly}(n)$.

Quantum speedup of backtracking algorithms

Theorem [AM '16] (informal)

If there is a classical backtracking algorithm which solves a CSP by exploring a tree of partial solutions of size T, there is a quantum algorithm that solves the CSP in time $O(\sqrt{T}\,\mathrm{poly}(n))$.

This is a near-quadratic speedup, assuming that $T \gg \mathrm{poly}(n)$.

Backtracking is one of the most useful classical algorithmic techniques known in practice.

Quantum speedup of backtracking algorithms

Theorem [AM '16] **(informal)**

If there is a classical backtracking algorithm which solves a CSP by exploring a tree of partial solutions of size T, there is a quantum algorithm that solves the CSP in time $O(\sqrt{T}\,\mathrm{poly}(n))$.

This is a near-quadratic speedup, assuming that $T \gg \mathrm{poly}(n)$.

Backtracking is one of the most useful classical algorithmic techniques known in practice.

Applications (so far):

- Quantum speedup of the Travelling Salesman Problem on bounded-degree graphs [Moylett, Linden and AM '16]
- Finding shortest vectors in lattices for cryptographic applications [Alkim et al. '15, del Pino et al. '16]

Summary and further reading

There are many quantum algorithms, solving many different problems, using many different techniques.

Summary and further reading

There are many quantum algorithms, solving many different problems, using many different techniques.

Some further reading:

- "Quantum algorithms for algebraic problems" [Childs and van Dam '08]
- "Quantum walk based search algorithms" [Santha '08]
- "Quantum algorithms" [Mosca '08]
- "New developments in quantum algorithms" [Ambainis '10]

Quantum algorithms: an overview,
AM, *npj Quantum Information* 2, 2016
www.nature.com/articles/npjqi201523

Quadratic speedup

Is a quadratic speedup significant?

Quadratic speedup

Is a quadratic speedup significant?

A concrete example: Circuit SAT with different clock speeds.

	Classical		Quantum		
Input bits	1MHz	1GHz	1KHz	10KHz	1MHz
30	18s	1s	32s	3s	0.03s
40	13d	18m	17m	104s	1s
50	36y	13d	9h	55m	33s
60	37M	36y	12d	1d	18m

Speeds listed are approximate, effective speeds (i.e. number of circuit evaluations per second) after overhead for fault-tolerance.

Yet more algorithms

There are a number of other quantum algorithms which I don't have time to go into:

- Hidden subgroup problems (e.g. [Bacon et al '05])
- Number-theoretic problems (e.g. [Fontein and Wocjan '11], ...)
- Formula evaluation (e.g. [Reichardt and Špalek '07])
- Tensor contraction (e.g. [Arad and Landau '08])
- Hidden shift problems (e.g. [Gavinsky et al '11])
- Adiabatic optimisation (e.g. [Farhi et al '00])
- ...

...as well as the entire field of quantum communication complexity.