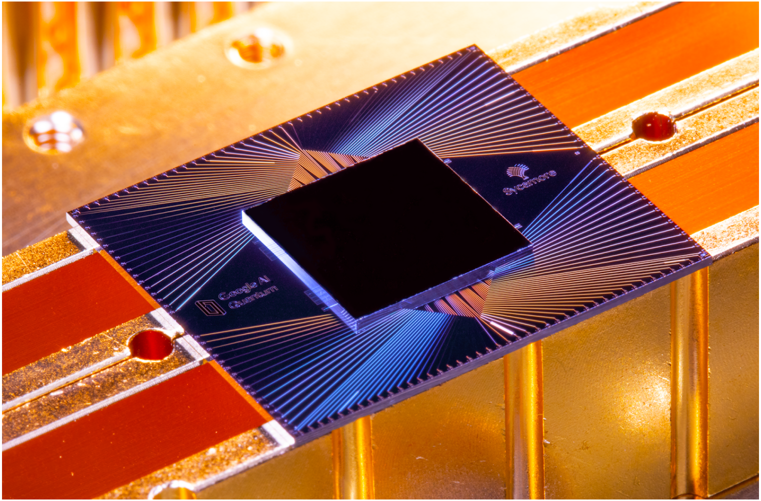
Quantum algorithms: an overview

Ashley Montanaro

School of Mathematics, University of Bristol

14 November 2019



Pic: Google

Quantum computers

Quantum computers are designed to do things that classical computers cannot. But to achieve a quantum speedup requires a quantum algorithm.

Quantum computers

Quantum computers are designed to do things that classical computers cannot. But to achieve a quantum speedup requires a quantum algorithm.

Most quantum algorithms can be divided into 5 categories:

Algorithm	**Speedup**	**Example**
Simulation of quantum systems	Exponential	Lloyd
Breaking cryptographic codes	Exponential	Shor
Optimisation / combinatorial search	Square-root	Grover
High-dimensional linear algebra	Exponential?	HHL
Quantum heuristics	Unknown	QAOA

Quantum computers

Quantum computers are designed to do things that classical computers cannot. But to achieve a quantum speedup requires a quantum algorithm.

Most quantum algorithms can be divided into 5 categories:

Algorithm	Speedup	Example
Simulation of quantum systems	Exponential	Lloyd
Breaking cryptographic codes	Exponential	Shor
Optimisation / combinatorial search	Square-root	Grover
High-dimensional linear algebra	Exponential?	HHL
Quantum heuristics	Unknown	QAOA

The Quantum Algorithm Zoo currently lists 404 papers on quantum algorithms.

Quantum simulation

The most important early application of quantum computers is likely to be quantum simulation: modelling a quantum-mechanical system on a quantum computer.

Applications include quantum chemistry, superconductivity, metamaterials, high-energy physics, . . . [Georgescu et al 1308.6253]

Quantum simulation

The most important early application of quantum computers is likely to be quantum simulation: modelling a quantum-mechanical system on a quantum computer.

Applications include quantum chemistry, superconductivity, metamaterials, high-energy physics, ... [Georgescu et al 1308.6253]

Different variants of this task include:

- Analogue vs. digital simulation
- Static vs. dynamics simulation

Analogue simulation

Problem

Given a Hamiltonian H describing a physical system, find a Hamiltonian H' that encodes H, and allows physically meaningful (static or dynamic) information about H to be determined.

Analogue simulation

Problem

Given a Hamiltonian H describing a physical system, find a Hamiltonian H' that encodes H, and allows physically meaningful (static or dynamic) information about H to be determined.

- H' should be "easier" to prepare in the lab than H.

Analogue simulation

Problem

Given a Hamiltonian H describing a physical system, find a Hamiltonian H' that encodes H, and allows physically meaningful (static or dynamic) information about H to be determined.

- H' should be "easier" to prepare in the lab than H.
- Even very simple quantum systems can be universal analogue quantum simulators [Cubitt, AM, Piddock, 1701.05182]

Analogue simulation

Problem

Given a Hamiltonian H describing a physical system, find a Hamiltonian H' that encodes H, and allows physically meaningful (static or dynamic) information about H to be determined.

- H' should be "easier" to prepare in the lab than H.
- Even very simple quantum systems can be universal analogue quantum simulators [Cubitt, AM, Piddock, 1701.05182]
- Analogue quantum simulators with > 50 qubits have been implemented experimentally (e.g. [Zhang et al, 1708.01044])

Digital simulation

Dynamics simulation

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.

Digital simulation

Dynamics simulation

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.
- A topic of very active research (e.g. [Childs et al 1711.10980])

Digital simulation

Static simulation (e.g.)

Given a Hamiltonian H describing a physical system, produce the ground (lowest energy) state of H.

Digital simulation

Static simulation (e.g.)

Given a Hamiltonian H describing a physical system, produce the ground (lowest energy) state of H.

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

Digital simulation

Static simulation (e.g.)

Given a Hamiltonian H describing a physical system, produce the ground (lowest energy) state of H.

- Given such a state, measurements can be performed to determine quantities of interest about the state.
- There is good evidence that producing the ground state is hard (QMA-complete) in the worst case, but it may be easy for physical systems of interest.

Digital simulation

Static simulation (e.g.)

Given a Hamiltonian H describing a physical system, produce the ground (lowest energy) state of H.

- Given such a state, measurements can be performed to determine quantities of interest about the state.
- There is good evidence that producing the ground state is hard (QMA-complete) in the worst case, but it may be easy for physical systems of interest.
- One approach: optimise over quantum circuits using a variational algorithm [McClean et al 1509.04279].

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 quant-ph/9508027]

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 1MHz clock speed quantum computer 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 1MHz clock speed quantum computer in 11 days.
- The fastest computer on the Top500 supercomputer list ($\sim 10^{17}$ operations per second) in $\sim 3 \times 10^{16}$ years.

(see e.g. [Gidney+Ekerå 1905.09749] for a more detailed analysis, showing that a 2048-digit integer can be factorised in 8 hours with 23 million physical qubits)

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.

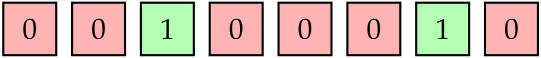
- Imagine we have access to a function $f : \{0,1\}^n \to \{0,1\}$ which we treat as a black box.

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 $f(x) = 1$.



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 quant-ph/9605043] can solve the problem with $O(\sqrt{2^n})$ queries to f (and bounded failure probability).

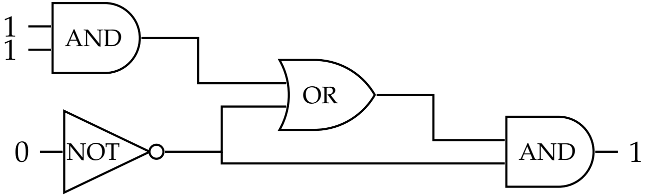
Applications of Grover's algorithm

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.

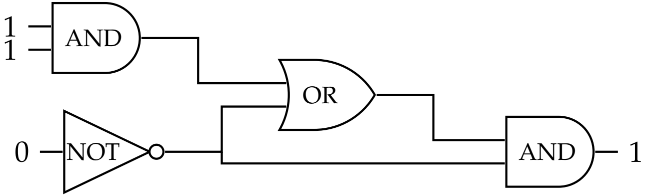
- For example, in the Circuit SAT problem we would like to find an input to a circuit on n bits such that the output is 1:



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 quant-ph/0005055]**

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+Høyer quant-ph/9607014]

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+Høyer quant-ph/9607014]
- Determining connectivity of an n-vertex graph in $O(n^{3/2})$ time [Dürr et al quant-ph/0401091]

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+Høyer quant-ph/9607014]
- Determining connectivity of an n-vertex graph in $O(n^{3/2})$ time [Dürr et al quant-ph/0401091]
- Finding a collision in a 2-1 function $f : [n] \to [n]$ in $O(n^{1/3})$ time [Brassard et al quant-ph/9705002]
- . . .

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+Høyer quant-ph/9607014]
- Determining connectivity of an n-vertex graph in $O(n^{3/2})$ time [Dürr et al quant-ph/0401091]
- Finding a collision in a 2-1 function $f : [n] \rightarrow [n]$ in $O(n^{1/3})$ time [Brassard et al quant-ph/9705002]
- ...

They can also speed up Monte Carlo methods [AM 1504.06987, Hamoudi+Magniez 1807.06456]:

- 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 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

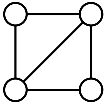
Backtracking is a general approach to solve constraint satisfaction problems (CSPs).

- 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).

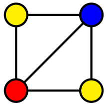
- An instance of a CSP on n variables $x_1, \dots, x_n$ is specified by a sequence of constraints, all of which must be satisfied by the variables.
- We might want to find one assignment to $x_1, \dots, x_n$ that satisfies all the constraints, or list all such assignments.
- A simple example: graph 3-colouring.



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 1509.02374] (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 $O(\sqrt{T}\,\mathsf{poly}(n))$.

Quantum speedup of backtracking algorithms

Theorem [AM 1509.02374] **(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 $O(\sqrt{T}\,\mathsf{poly}(n))$.

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

Quantum speedup of backtracking algorithms

Theorem [AM 1509.02374] (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 $O(\sqrt{T}\,\mathsf{poly}(n))$.

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

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

Quantum speedup of backtracking algorithms

Theorem [AM 1509.02374] (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 $O(\sqrt{T}\,\mathsf{poly}(n))$.

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

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

Some applications:

- Quantum speedup of the Travelling Salesman Problem on bounded-degree graphs [Moylett, Linden and AM 1612.06203]
- Finding shortest vectors in lattices for cryptographic applications [Alkim et al. '15, del Pino et al. '16]
- Accelerating classical branch-and-bound algorithms for optimisation problems [AM 1906.10375]

“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 0811.3171]

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:

- Electromagnetic scattering cross-sections using the finite element method [Clader et al 1301.2340] [AM+Pallister 1512.05903]

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:

- Electromagnetic scattering cross-sections using the finite element method [Clader et al 1301.2340] [AM+Pallister 1512.05903]
- "Solving" differential equations [Leyton+Osborne 0812.4423] [Berry 1010.2745]

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:

- Electromagnetic scattering cross-sections using the finite element method [Clader et al 1301.2340] [AM+Pallister 1512.05903]
- "Solving" differential equations [Leyton+Osborne 0812.4423] [Berry 1010.2745]
- Recommendation systems and other problems in machine learning (e.g. [Kerenidis+Prakash 1603.08675]) – but note "quantum-inspired" competition [Tang 1807.04271]!

Quantum heuristics

Some quantum optimisation algorithms might be more efficient than our best classical algorithms, but we can't prove this rigorously…

Quantum heuristics

Some quantum optimisation algorithms might be more efficient than our best classical algorithms, but we can't prove this rigorously…

Examples:

- The adiabatic algorithm / quantum annealing [Farhi et al quant-ph/0001106]
- The Quantum Approximate Optimisation Algorithm (QAOA) [Hogg+Portnov quant-ph/0006090, Farhi et al 1411.4028]

Quantum heuristics

Some quantum optimisation algorithms might be more efficient than our best classical algorithms, but we can't prove this rigorously...

Examples:

- The adiabatic algorithm / quantum annealing [Farhi et al quant-ph/0001106]
- The Quantum Approximate Optimisation Algorithm (QAOA) [Hogg+Portnov quant-ph/0006090, Farhi et al 1411.4028]

These algorithms try to find good solutions to hard combinatorial optimisation problems (e.g. MAX-CUT).

Quantum heuristics

Some quantum optimisation algorithms might be more efficient than our best classical algorithms, but we can't prove this rigorously…

Examples:

- The adiabatic algorithm / quantum annealing [Farhi et al quant-ph/0001106]
- The Quantum Approximate Optimisation Algorithm (QAOA) [Hogg+Portnov quant-ph/0006090, Farhi et al 1411.4028]

These algorithms try to find good solutions to hard combinatorial optimisation problems (e.g. MAX-CUT).

Evidence that they outperform classical algorithms is mixed, but we at least know they are probably hard to simulate classically [Farhi+Harrow 1602.07674].

Analysing real quantum algorithm complexity

Analysing real quantum algorithm complexity

Some fully worked-out applications with large speedups (for quantum runtime ~ 1 day) include:

- Nitrogen fixation [Reiher et al 1605.03590]
- Many-body localisation [Childs et al 1711.10980]
- Other problems in quantum chemistry and condensed-matter physics, e.g. [Babbush et al 1805.03662]
- Integer factorisation [Kutin quant-ph/0609001] [Gidney and Ekerå 1905.09749]

Analysing real quantum algorithm complexity

Some fully worked-out applications with large speedups (for quantum runtime ~ 1 day) include:

- Nitrogen fixation [Reiher et al 1605.03590]
- Many-body localisation [Childs et al 1711.10980]
- Other problems in quantum chemistry and condensed-matter physics, e.g. [Babbush et al 1805.03662]
- Integer factorisation [Kutin quant-ph/0609001] [Gidney and Ekerå 1905.09749]

In constraint satisfaction the speedups are smaller and quantum hardware requirements larger…

- Graph colouring / boolean satisfiability: speedup factor of $\sim 10^5$ (ignoring cost of fault-tolerance processing) but $\sim 10^{12}$ physical qubits required [Campbell et al 1810.05582]

Conclusions

There are many quantum algorithms, solving many different problems, some of which achieve substantial speedups over their classical counterparts.

Conclusions

There are many quantum algorithms, solving many different problems, some of which achieve substantial speedups over their classical counterparts.

Important future research directions include:

- Finding more practical applications for these algorithms;
- Analysing their complexity in detail;
- New ideas for quantum algorithm design.

Conclusions

There are many quantum algorithms, solving many different problems, some of which achieve substantial speedups over their classical counterparts.

Important future research directions include:

- Finding more practical applications for these algorithms;
- Analysing their complexity in detail;
- New ideas for quantum algorithm design.

Further reading:

Quantum algorithms: an overview [AM, 1511.04206]

Thanks!