

Quantum algorithms: an overview

Ashley Montanaro*

December 21, 2015

Abstract

Quantum computers are designed to outperform standard computers by running quantum algorithms. Areas in which quantum algorithms can be applied include cryptography, search and optimisation, simulation of quantum systems, and solving large systems of linear equations. Here we briefly survey some known quantum algorithms, with an emphasis on a broad overview of their applications rather than their technical details. We include a discussion of recent developments and near-term applications of quantum algorithms.

1 Introduction

A quantum computer is a machine designed to use quantum mechanics to do things which cannot be done by any machine based only on the laws of classical physics. Eventual applications of quantum computing range from breaking cryptographic systems to the design of new medicines. These applications are based on quantum algorithms – algorithms which run on a quantum computer and achieve a speedup, or other efficiency improvement, over any possible classical algorithm. Although large-scale general-purpose quantum computers do not yet exist, the theory of quantum algorithms has been an active area of study for over 20 years. Here we aim to give a broad overview of quantum algorithmics, focusing on algorithms with clear applications and rigorous performance bounds, and including recent progress in the field.

Contrary to a rather widespread popular belief that quantum computers have few applications, the field of quantum algorithms has developed into an area of study large enough that a brief survey such as this cannot hope to be remotely comprehensive. Indeed, at the time of writing the “Quantum Algorithm Zoo” website cites 278 papers on quantum algorithms [52]. There are now a number of excellent surveys about quantum algorithms [28, 71, 85, 8], and we defer to these for details of the algorithms we cover here, and many more. In particular, we omit all discussion of *how* the quantum algorithms mentioned work. We will also not cover the important topics of how to actually build a quantum computer [59] (in theory or in practice) and quantum error-correction [42], nor quantum communication complexity [23] or quantum Shannon theory [98].

1.1 Measuring quantum speedup

What does it mean to say that a quantum computer solves a problem more quickly than a classical computer? As is typical in computational complexity theory, we will generally consider asymptotic scaling of complexity measures such as runtime or space usage with problem size, rather than

*School of Mathematics, University of Bristol, UK; ashley.montanaro@bristol.ac.uk.

Class	Informal definition
P	Can be solved by a deterministic classical computer in polynomial time
BPP	Can be solved by a probabilistic classical computer in polynomial time
BQP	Can be solved by a quantum computer in polynomial time
NP	Solution can be checked by a deterministic classical computer in polynomial time
QMA	Solution can be checked by a quantum computer in polynomial time

Table 1: Some computational complexity classes of importance in quantum computation. “Polynomial time” is short for “in time polynomial in the input size”.

individual problems of a fixed size. In both the classical and quantum settings, we measure runtime by the number of elementary operations used by an algorithm. In the case of quantum computation, this can be measured using the quantum circuit model, where a quantum circuit is a sequence of elementary quantum operations called quantum gates, each applied to a small number of qubits (quantum bits). To compare the performance of algorithms, we use computer-science style notation $O(f(n))$, which should be interpreted as “asymptotically upper-bounded by $f(n)$ ”.

We sometimes use basic ideas from computational complexity theory [73], and in particular the notion of complexity classes, which are groupings of problems by difficulty. See Table 1 for informal descriptions of some important complexity classes. If a problem is said to be *complete* for a complexity class, this means that it is one of the “hardest” problems within that class: it is contained within that class, and every other problem within that class reduces to it.

2 The hidden subgroup problem and applications to cryptography

One of the first applications of quantum computers discovered was Shor’s algorithm for integer factorisation [89]. In the factorisation problem, given an integer $N = p \times q$ for some prime numbers p and q , our task is to determine p and q . The best classical algorithm known (the general number field sieve) runs in time $\exp(O((\log N)^{1/3}(\log \log N)^{2/3}))$ [22]¹, while Shor’s quantum algorithm solves this problem substantially faster, in time $O((\log N)^3)$. This result might appear only of mathematical interest, were it not for the fact that the widely-used RSA public-key cryptosystem [82] relies on the hardness of integer factorisation. Shor’s efficient factorisation algorithm implies that this cryptosystem is insecure against attack by a large quantum computer.

As a more specific comparison than the above asymptotic runtimes, in 2010 Kleinjung et al. [57] reported classical factorisation of a 768-bit number, using hundreds of modern computers over a period of two years, with a total computational effort of $\sim 10^{20}$ operations. A detailed analysis of one fault-tolerant quantum computing architecture [42], making reasonable assumptions about the underlying hardware, suggests that a *2000-bit* number could be factorised by a quantum computer using $\sim 3 \times 10^{11}$ quantum gates, and approximately a billion qubits, running for just over a day at a clock rate of 10MHz. This is clearly beyond current technology, but does not seem unrealistic as a long-term goal.

Shor’s approach to integer factorisation is based on reducing the task to a special case of a mathematical problem known as the hidden subgroup problem (HSP) [16, 19], then giving an efficient quantum algorithm for this problem.

¹In fact, this is a heuristic bound and this algorithm’s worst-case runtime has not been rigorously determined; the best proven bound is somewhat higher.

Problem	Group	Complexity	Cryptosystem
Factorisation	\mathbb{Z}	Polynomial [89]	RSA
Discrete log	$\mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1}$	Polynomial [89]	Diffie-Hellman, DSA, ...
Elliptic curve discrete log	Elliptic curve	Polynomial [76]	ECDH, ECDSA, ...
Principal ideal	\mathbb{R}	Polynomial [46]	Buchmann-Williams
Shortest lattice vector	Dihedral group	Subexponential [58, 80]	NTRU, Ajtai-Dwork, ...
Graph isomorphism	Symmetric group	Exponential	–

Table 2: Some problems which can be expressed as hidden subgroup problems (HSPs). The table lists the time complexity of the best quantum algorithms known for the HSPs, and the cryptosystems that are (or would be) broken by polynomial-time algorithms.

Hidden subgroup problem. Let G be a group and let X be a set. Given the ability to evaluate a function $f : G \rightarrow X$, where f is constant on the cosets of some unknown subgroup $H \leq G$, and distinct on each coset, identify H .

Shor’s algorithm solves the case $G = \mathbb{Z}$. Efficient solutions to the HSP for other groups G turn out to imply efficient algorithms to break other cryptosystems; we summarise some important cases of the HSP and some of their corresponding cryptosystems in Table 2. Two particularly interesting cases of the HSP for which polynomial-time quantum algorithms are not currently known are the dihedral and symmetric groups. A polynomial-time quantum algorithm for the former case would give an efficient algorithm for finding shortest vectors in lattices [79]; an efficient quantum algorithm for the latter case would give an efficient test for isomorphism of graphs (equivalence under relabelling of vertices).

3 Search and optimisation

One of the most basic problems in computer science is unstructured search. This problem can be formalised as follows:

Unstructured search problem. Given the ability to evaluate a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, find x such that $f(x) = 1$, if such an x exists; otherwise, output “not found”.

It is easy to see that any classical algorithm which solves the unstructured search problem with certainty must evaluate f $N = 2^n$ times in the worst case. Even if we seek a randomised algorithm which succeeds, say, with probability $1/2$ in the worst case, the number of evaluations required is of order N . However, remarkably, there is a quantum algorithm due to Grover [45] which solves this problem using $O(\sqrt{N})$ evaluations of f in the worst case². The algorithm is bounded-error; that is, it fails with probability ϵ , for arbitrarily small (but fixed) $\epsilon > 0$. Although f may have some kind of internal structure, Grover’s algorithm does not use this at all; we say that f is used as an *oracle* or *black box* in the algorithm.

Grover’s algorithm can immediately be applied to any problem in the complexity class NP. This class encapsulates decision problems whose solutions can be checked efficiently, in the following

²Grover’s original algorithm solved the special case where the solution is unique; the extension to multiple solutions came slightly later [18].

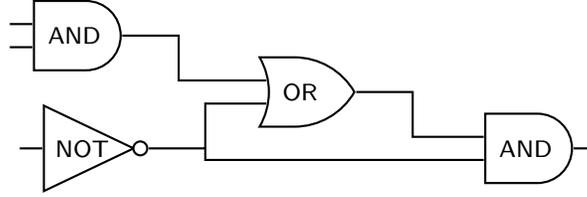


Figure 1: An instance of the Circuit SAT problem. The answer should be “yes” as there exists an input to the circuit such that the output is 1.

sense: There exists an efficient classical checking algorithm \mathcal{A} such that, for any instance of the problem where the answer should be “yes”, there is a *certificate* which can be input to \mathcal{A} such that \mathcal{A} accepts the certificate. In other words, a certificate is a proof that the answer is “yes”, which can be checked by \mathcal{A} . On the other hand, for any instance where the answer should be “no”, there should be no certificate that can make \mathcal{A} accept it. The class NP encompasses many important problems involving optimisation and constraint satisfaction.

Given a problem in NP that has a certificate of length m , by applying Grover’s algorithm to \mathcal{A} and searching over all possible certificates, we obtain an algorithm which uses time $O(2^{m/2} \text{poly}(m))$, rather than the $O(2^m \text{poly}(m))$ used by classical exhaustive search over all certificates. This (nearly) quadratic speedup is less dramatic than the super-polynomial speedup achieved by Shor’s algorithm, but can still be rather substantial. Indeed, if the quantum computer runs at approximately the same clock speed as the classical computer, this implies that problem instances of approximately twice the size can be solved in a comparable amount of time.

As a prototypical example of this, consider the fundamental NP-complete circuit satisfiability problem (Circuit SAT), which is illustrated in Figure 1. An instance of this problem is a description of an electronic circuit comprising AND, OR and NOT gates which takes n bits as input and produces 1 bit of output. The task is to determine whether there exists an input to the circuit such that the output is 1. Algorithms for Circuit SAT can be used to solve a plethora of problems related to electronic circuits; examples include design automation, circuit equivalence and model checking [75]. The best classical algorithms known for Circuit SAT run in worst-case time of order 2^n for n input variables, i.e. not significantly faster than exhaustive search [99]. By applying Grover’s algorithm to the function $f(x)$ which evaluates the circuit on input $x \in \{0, 1\}^n$, we immediately obtain a runtime of $O(2^{n/2} \text{poly}(n))$, where the $\text{poly}(n)$ comes from the time taken to evaluate the circuit on a given input.

3.1 Amplitude amplification

Grover’s algorithm speeds up the naïve classical algorithm for unstructured search. Quantum algorithms can also accelerate more complicated classical algorithms.

Heuristic search problem. Given the ability to execute a probabilistic “guessing” algorithm \mathcal{A} , and a “checking” function f , such that

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

output w such that $f(w) = 1$.

One way to solve the heuristic search problem classically is simply to repeatedly run \mathcal{A} and check the output each time using f , which would result in an average of $O(1/\epsilon)$ evaluations of f . However, a quantum algorithm due to Brassard et al. [20] can find w such that $f(w) = 1$ with only $O(1/\sqrt{\epsilon})$ uses of f , and failure probability arbitrarily close to 0, thus achieving a quadratic speedup. This algorithm is known as *amplitude amplification*, by analogy with classical probability amplification.

The unstructured search problem discussed above fits into this framework, by simply taking \mathcal{A} to be the algorithm which outputs a uniformly random n -bit string. Further, if there are k inputs $w \in \{0, 1\}^n$ such that $f(w) = 1$, then

$$\Pr[\mathcal{A} \text{ outputs } w \text{ such that } f(w) = 1] = \frac{k}{N},$$

so we can find a w such that $f(w) = 1$ with $O(\sqrt{N/k})$ evaluations of f . However, we could imagine \mathcal{A} being a more complicated algorithm or heuristic targeted at a particular problem we would like to solve. For example, one of the most efficient classical algorithms known for the fundamental constraint satisfaction problem 3-SAT³ is randomised and runs in time $O((4/3)^n \text{poly}(n))$ [86]. Amplitude amplification can be applied to this algorithm to obtain a quantum algorithm with runtime $O((4/3)^{n/2} \text{poly}(n))$, illustrating that quantum computers can speed up non-trivial classical algorithms for NP-complete problems.

An interesting future direction for quantum algorithms is finding accurate *approximate* solutions to optimisation problems. Recent work of Farhi, Goldstone and Gutmann [37] gave the first quantum algorithm for a combinatorial task (simultaneously satisfying many linear equations of a certain form) which outperformed the best classical algorithm known in terms of accuracy; in this case, measured by the fraction of equations satisfied. This inspired a more efficient classical algorithm for the same problem [9], leaving the question open of whether quantum algorithms for optimisation problems can substantially outperform the accuracy of their classical counterparts.

3.2 Applications of Grover’s algorithm and amplitude amplification

Grover’s algorithm and amplitude amplification are powerful subroutines which can be used as part of more complicated quantum algorithms, allowing quantum speedups to be obtained for many other problems. We list just a few of these speedups here.

1. Finding the **minimum** of an unsorted list of N integers (equivalently, finding the minimum of an arbitrary and initially unknown function $f : \{0, 1\}^n \rightarrow \mathbb{Z}$). A quantum algorithm due to Dürr and Høyer [35] solves this problem with $O(\sqrt{N})$ evaluations of f , giving a quadratic speedup over classical algorithms. Their algorithm is based on applying Grover’s algorithm to a function $g : \{0, 1\}^n \rightarrow \{0, 1\}$ defined by $g(x) = 1$ if and only if $f(x) < T$ for some threshold T . This threshold is initially random, and then updated as inputs x are found such that $f(x)$ is below the threshold.
2. Determining **graph connectivity**. To determine whether a graph on N vertices is connected requires time of order N^2 classically in the worst case. Dürr et al. [34] give a quantum algorithm which solves this problem in time $O(N^{3/2})$, up to logarithmic factors, as well as efficient algorithms for some other graph-theoretic problems (strong connectivity, minimum spanning tree, shortest paths).

³The input is a boolean formula on n variables written in conjunctive normal form with at most 3 variables per clause; our task is to determine whether the formula is satisfiable. For example, on input $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_1)$ we should output “yes”, as witnessed by $x_1 = 0, x_2 = 1, x_3 = 1$.

3. **Pattern matching**, a fundamental problem in text processing and bioinformatics. Here the task is to find a given pattern P of length M within a text T of length N , where the pattern and the text are strings over some alphabet. Ramesh and Vinay have given a quantum algorithm [78] which solves this problem in time $O(\sqrt{N} + \sqrt{M})$, up to logarithmic factors, as compared with the best possible classical complexity $O(N + M)$. These are both worst-case time bounds, but one could also consider an average-case setting where the text and pattern are both picked at random. Here the quantum speedup is more pronounced: there is a quantum algorithm which combines amplitude amplification with ideas from the dihedral hidden subgroup problem and runs in time $O(\sqrt{N/M}2^{O(\sqrt{\log M})})$ up to logarithmic factors, as compared with the best possible classical runtime $O(N/M + \sqrt{N})$ [70]. This is a super-polynomial speedup when M is large.

3.3 Adiabatic optimisation

An alternative approach to quantum combinatorial optimisation is provided by the quantum adiabatic algorithm [40]. The adiabatic algorithm can be applied to any constraint satisfaction problem (CSP) where we are given a sequence of constraints applied to some input bits, and are asked to output an assignment to the input bits which maximises the number of satisfied constraints. Many such problems are NP-complete and of significant practical interest. The basic idea behind the algorithm is physically motivated, and based around a correspondence between CSPs and physical systems. We start with a quantum state which is the uniform superposition over all possible solutions to the CSP. This is the ground (lowest energy) state of a Hamiltonian which can be prepared easily. This Hamiltonian is then gradually modified to give a new Hamiltonian whose ground state encodes the solution maximising the number of satisfied constraints. The quantum adiabatic theorem guarantees that, if this process is carried out slowly enough, the system will remain in its ground state throughout; in particular, the final state gives an optimal solution to the CSP. The key phrase here is “slowly enough”; for some instances of CSPs on n bits, the time required for this evolution might be exponential in n .

Unlike the algorithms described in the rest of this survey, the adiabatic algorithm lacks general, rigorous worst-case upper bounds on its runtime. Although numerical experiments can be carried out to evaluate its performance on small instances [38], this rapidly becomes infeasible for larger problems. One can construct problem instances on which the standard adiabatic algorithm provably takes exponential time [94, 39]; however, changing the algorithm can evade some of these arguments [36, 29].

The adiabatic algorithm can be implemented on a universal quantum computer. However, it also lends itself to direct implementation on a physical system whose Hamiltonian can be varied smoothly between the desired initial and final Hamiltonians. The most prominent exponent of this approach is the company D-Wave Systems, Inc., which has built large machines designed to implement this algorithm [51], with the most recent such machine (“D-Wave 2X”) announced as having up to 1152 qubits. For certain instances of CSPs, these machines have been demonstrated to outperform classical solvers running on a standard computer [69, 55], although the speedup (or otherwise) seems to have a rather subtle dependence on the problem instance, classical solver compared, and measure of comparison [83, 55].

As well as the theoretical challenges to the adiabatic algorithm mentioned above, there are also some significant practical challenges faced by the D-Wave system. In particular, these machines do not remain in their ground state throughout, but are in a thermal state above absolute zero. Because of this, the algorithm actually performed has some similarities to classical simulated annealing, and

is hence known as “quantum annealing”. It is unclear at present whether a quantum speedup predicted for the adiabatic algorithm would persist in this setting.

4 Quantum simulation

In the early days of classical computing, one of the main applications of computer technology was the simulation of physical systems⁴. Similarly, the most important early application of quantum computers is likely to be the simulation of quantum systems [24, 21, 44]. Applications of quantum simulation include quantum chemistry, superconductivity, metamaterials and high-energy physics. Indeed, one might expect that quantum simulation would help us understand any system where quantum mechanics plays a role.

The word “simulation” can be used to describe a number of problems, but in quantum computation is often used to mean the problem of calculating the dynamical properties of a system. This can be stated more specifically as: Given a Hamiltonian H describing a physical system, and a description of an initial state $|\psi\rangle$ of that system, output some property of the state $|\psi_t\rangle = e^{-iHt}|\psi\rangle$ corresponding to evolving the system according to that Hamiltonian for time t . As all quantum systems obey the Schrödinger equation, this is a fundamentally important task; however, the exponential complexity of completely describing general quantum states suggests that it should be impossible to achieve efficiently classically, and indeed no efficient general classical algorithm for quantum simulation is known. This problem originally motivated Feynman to ask whether a *quantum* computer could efficiently simulate quantum mechanics [41].

A general-purpose quantum computer can indeed efficiently simulate quantum mechanics in this sense for many physically realistic cases, such as systems with locality restrictions on their interactions [64]. Given a description of a quantum state $|\psi\rangle$, a description of H , and a time t , the quantum simulation algorithm produces an approximation to the state $|\psi_t\rangle$. Measurements can then be performed on this state to determine quantities of interest about it. The algorithm runs in time polynomial in the size of the system being simulated (the number of qubits) and the desired evolution time, giving an exponential speedup over the best general classical algorithms known. However, there is still room for improvement and quantum simulation remains a topic of active research. Examples include work on increasing the accuracy of quantum simulation while retaining a fast runtime [14]; optimising the algorithm for particular applications such as quantum chemistry [48]; and exploring applications to new areas such as quantum field theory [53].

The above, very general, approach is sometimes termed *digital* quantum simulation: we assume we have a large-scale, general-purpose quantum computer, and run the quantum simulation algorithm on it. By contrast, in *analogue* quantum simulation we mimic one physical system directly using another. That is, if we would like to simulate a system with some Hamiltonian H , we build another system which can be described by a Hamiltonian approximating H . We have gained something by doing this if the second system is easier to build, to run or to extract information from than the first. For certain systems analogue quantum simulation may be significantly easier to implement than digital quantum simulation, at the expense of being less flexible. It is therefore expected that analogue simulators outperforming their classical counterparts will be implemented first [24].

⁴Such applications arguably go back at least as far as the Antikythera mechanism from the 2nd century BC.

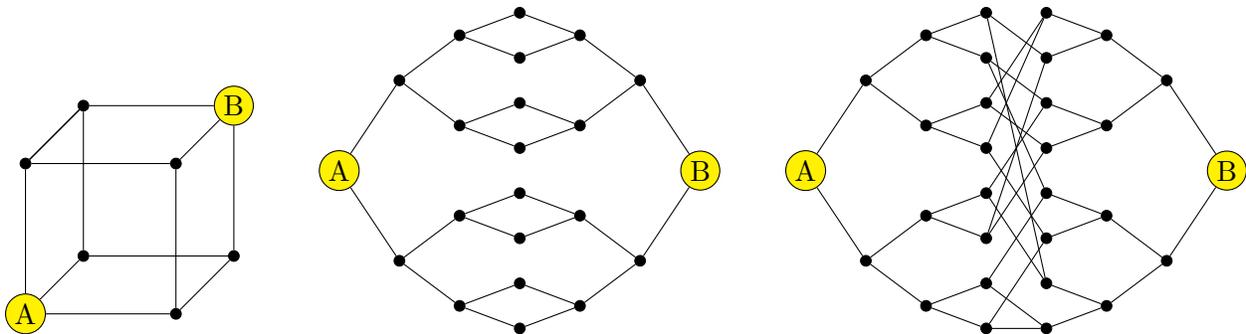


Figure 2: Three graphs for whose natural generalisations to N vertices a classical random walk requires exponentially more time than a quantum walk to reach the exit (B) from the entrance (A). However, on the first two graphs there exist efficient classical algorithms to find the exit which are not based on a random walk.

5 Quantum walks

In classical computer science the concept of the random walk or Markov chain is a powerful algorithmic tool, and is often applied to search and sampling problems. Quantum walks provide a similarly powerful and general framework for designing fast quantum algorithms. Just as a random walk algorithm is based on the simulated motion of a particle moving randomly within some underlying graph structure, a quantum walk is based on the simulated coherent quantum evolution of a particle moving on a graph.

Quantum walk algorithms generally take advantage of one of two ways in which quantum walks outperform random walks: faster hitting (the time taken to find a target vertex from a source vertex), and faster mixing (the time taken to spread out over all vertices after starting from one source vertex). For some graphs, hitting time of quantum walks can be exponentially less than their classical counterparts [27, 54]. The separation between quantum and classical mixing time can be quadratic, but no more than this [3] (approximately). Nevertheless, fast mixing has proven to be a very useful tool for obtaining general speedups over classical algorithms.

Figure 2 illustrates special cases of three families of graphs for which quantum walks display faster hitting than random walks: the hypercube, the “glued trees” graph, and the “glued trees” graph with a random cycle added in the middle. This third example is of particular interest because quantum walks can be shown to outperform any classical algorithm for navigating the graph, even one not based on a random walk. A continuous-time quantum walk which starts at the entrance (on the left-hand side) and runs for time $O(\log N)$ finds the exit (on the right-hand side) with probability at least $1/\text{poly}(\log N)$. However, any classical algorithm requires time of order $N^{1/6}$ to find the exit [26]. Intuitively, the classical algorithm can progress quickly at first, but then gets “stuck” in the random part in the middle of the graph. The coherence and symmetry of the quantum walk make it essentially blind to this randomness, and it efficiently progresses from the left to the right.

A possibly surprising application of quantum walks is fast evaluation of boolean formulae. A boolean formula on N binary inputs x_1, \dots, x_N is a tree whose internal vertices represent AND (\wedge), OR (\vee) or NOT (\neg) gates applied to their child vertices, and whose N leaves are labelled with the bits x_1, \dots, x_N . Two such formulae are illustrated in Figure 3. There is a quantum algorithm which allows any such formula to be evaluated in slightly more than $O(N^{1/2})$ operations [5], while it is

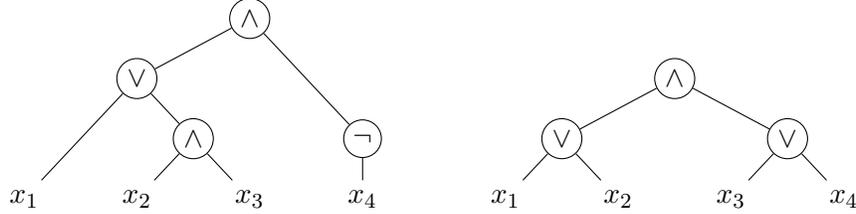


Figure 3: Two boolean formulae on 4 bits. For $x_1 = 1, x_2 = x_3 = x_4 = 0$, for example, the first formula evaluates to 1 and the second to 0. The second formula is an AND-OR tree.

known that for a wide class of boolean formulae, any randomised classical algorithm requires time of order $N^{0.753\dots}$ in the worst case [84]. The quantum algorithm is based around the use and analysis of a quantum walk on the tree graph corresponding to the formula’s structure. A particularly interesting special case of the formula evaluation problem which displays a quantum speedup is evaluating AND-OR trees, which corresponds to deciding the winner of certain two-player games.

Quantum walks can also be used to obtain a very general speedup over classical algorithms based on Markov chains. A discrete-time Markov chain is a stochastic linear map defined in terms of its transition matrix P , where P_{xy} is the probability of transitioning from state x to state y . Many classical search algorithms can be expressed as simulating a Markov chain for a certain number of steps, and checking whether a transition is made to a “marked” element for which we are searching. A key parameter which determines the efficiency of this classical algorithm is the spectral gap δ of the Markov chain (i.e. the difference between the largest and second-largest eigenvalues of P).

There are analogous algorithms based on quantum walks which improve the dependence on δ quadratically, from $1/\delta$ to $1/\sqrt{\delta}$ [6, 92, 66]. This framework has been used to obtain quantum speedups for a variety of problems [85], ranging from determining whether a list of integers are all distinct [6] to finding triangles in graphs [61].

6 Solving linear equations and related tasks

A fundamental task in mathematics, engineering and many areas of science is solving systems of linear equations. We are given an $N \times N$ matrix A , and a vector $b \in \mathbb{R}^N$, and are asked to output x such that $Ax = b$. This problem can be solved in time polynomial in N by straightforward linear-algebra methods such as Gaussian elimination. Can we do better than this? This appears difficult, because even to write down the answer x would require time of order N . The quantum algorithm of Harrow, Hassidim and Lloyd [47] (HHL) for solving systems of linear equations sidesteps this issue by “solving” the equations in a peculiarly quantum sense: Given the ability to create the quantum state $|b\rangle = \sum_{i=1}^N b_i|i\rangle$, and access to A , the algorithm outputs a state approximately proportional to $|x\rangle = \sum_{i=1}^N x_i|i\rangle$. This is an N -dimensional quantum state which can be stored in $O(\log N)$ qubits.

The algorithm runs efficiently, assuming that the matrix A satisfies some constraints. First, it should be sparse – each row should contain at most d elements, for some $d \ll N$. We should be given access to A via an function to which we can pass a row number r and an index i , with $1 \leq i \leq d$, and which returns the i ’th nonzero element in the r ’th row. Also, the condition number $\kappa = \|A^{-1}\| \|A\|$, a parameter measuring the numerical instability of A , should be small. Assuming these constraints, $|x\rangle$ can be approximately produced in time polynomial in $\log N$, d , and κ [47, 4].

If d and κ are small, this is an exponential improvement on standard classical algorithms. Indeed, one can even show that achieving a similar runtime classically would imply that classical computers could efficiently simulate any polynomial-time quantum computation [47].

Of course, rather than giving as output the entirety of x , the algorithm produces an N -dimensional quantum state $|x\rangle$; to output the solution x itself would then involve making many measurements to completely characterise the state, requiring time of order N in general. However, we may not be interested in the entirety of the solution, but rather in some global property of it. Such properties can be determined by performing measurements on $|x\rangle$. For example, the HHL algorithm allows one to efficiently determine whether two sets of linear equations have the same solution [4], as well as many other simple global properties [30].

The HHL algorithm is likely to find applications in settings where the matrix A and the vector b are generated algorithmically, rather than being written down explicitly. One such setting is the finite element method (FEM) in engineering. Recent work by Clader, Jacobs and Sprouse has shown that the HHL algorithm, when combined with a preconditioner, can be used to solve an electromagnetic scattering problem via the FEM [30]. The same algorithm, or closely related ideas, can also be applied to problems beyond linear equations themselves. These include solving large systems of differential equations [62, 13], data fitting [97] and various tasks in machine learning [65]. It should be stressed that in all these cases the quantum algorithm “solves” these problems in the same sense as the HHL algorithm solves them: it starts with a quantum state and produces a quantum state as output. Whether this is a reasonable definition of “solution” depends on the application, and again may depend on whether the input is produced algorithmically or is provided explicitly as arbitrary data [1].

7 Few-qubit applications and experimental implementations

Although progress in experimental quantum computation has been rapid, there is still some way to go before we have a large-scale, general-purpose quantum computer, with current implementations consisting of only a few qubits. Any quantum computation operating on at most 20-30 qubits in the standard quantum circuit model can be readily simulated on a modern classical computer. Therefore, existing implementations of quantum algorithms should usually be seen as proofs of principle rather than demonstrating genuine speedups over the classical state-of-the-art. In Table 3 we highlight some experimental implementations of algorithms discussed here, focusing on the largest problem sizes considered thus far⁵.

An important algorithm omitted from this table is quantum simulation. This topic has been studied since the early days of quantum computation (with perhaps the first implementation dating from 1999 [91]), and quantum simulations have now been implemented, in some form, on essentially every technological platform for quantum computing. One salient example is the use of a 6-qubit ion trap system [60] to implement general digital quantum simulation; we defer to survey papers [24, 44, 15, 7] for many further references. It is arguable that quantum simulations, in the sense of measuring the properties of a controllable quantum system, have *already* been performed which are beyond the reach of current classical simulation techniques [93].

One application of digital quantum simulation which is currently the object of intensive study is quantum chemistry [48, 74, 96]. Classical techniques for molecular simulation are currently limited to molecules with 50-70 spin orbitals [74]. As each spin orbital corresponds to a qubit in

⁵Although note that one has to be careful when using “problem size” as a proxy for “difficulty in solving on a quantum computer” [90].

Algorithm	Technology	Problem solved
Shor’s algorithm	Bulk optics [68]	Factorisation of 21
Grover’s algorithm	NMR [95]	Unstructured search, $N = 8$
Quantum annealing	D-Wave 2X [55]	Ising model on a “Chimera” graph with 1097 vertices
HHL algorithm	Bulk optics [25, 10], NMR [72]	2×2 system of linear equations

Table 3: Some proof-of-concept experimental implementations of quantum algorithms. Table only includes some “largest” problem instances solved thus far.

the quantum simulation algorithm, a quantum computer with as few as 100 logical qubits could perform calculations beyond the reach of classical computation. The challenge in this context is optimising the simulation time; although polynomial in the number of orbitals, this initially seemed prohibitively long [96], but was rapidly improved via detailed analysis [74].

The demonstration of quantum algorithms which outperform classical computation in the more immediate future is naturally of considerable interest. The Boson Sampling problem was designed specifically to address this [2]. Boson Sampling is the problem of sampling from the probability distribution obtained by feeding n photons through a linear-optical network on m modes, where $m \gg n$. This task is conjectured to be hard for a classical computer to solve [2]. However, Boson Sampling can be performed easily using linear optics, and indeed several small-scale experimental demonstrations with a few photons have already been carried out [77]. Although Boson Sampling was not originally designed with practical applications in mind, subsequent work has explored connections to molecular vibrations and vibronic spectra [50, 67].

One way in which quantum algorithms can be profitably applied for even very small-scale systems is “quantum algorithmic thinking”: applying ideas from the design of quantum algorithms to physical problems. An example of this from the field of quantum metrology is the development of high-precision quantum measurement schemes based on quantum phase estimation algorithms [49].

8 Zero-qubit applications

We finally mention some ways in which quantum computing is useful now, without the need for an actual large-scale quantum computer. These can be summarised as the application of ideas from the theory of quantum computation to other scientific and mathematical fields.

First, the field of Hamiltonian complexity aims to characterise the complexity of computing quantities of interest about quantum-mechanical systems. A prototypical example, and a fundamental task in quantum chemistry and condensed-matter physics, is the problem of approximately calculating the ground-state energy of a physical system described by a local Hamiltonian. It is now known that this problem – along with many others – is QMA-complete [56, 17]. Problems in the class QMA are those which can be efficiently solved by a quantum computer given access to a quantum “certificate”⁶. Classically, if a problem is proven NP-complete, this is considered good evidence that there is no efficient algorithm to solve it. Similarly, QMA-complete problems are considered unlikely to have efficient quantum (or classical) algorithms. One can even go further than this, and attempt to characterise for which families of physical systems calculating ground-

⁶We imagine that the certificate is produced by an all-powerful (yet untrustworthy) wizard Merlin, and given to a polynomial-time human Arthur to check; hence Quantum Merlin-Arthur.

state energies is hard, and for which the problem is easy [87, 70]. Although this programme is not yet complete, it has already provided some formal justification for empirical observations in condensed-matter physics about relative hardness of these problems.

Second, using the model of quantum information as a mathematical tool can provide insight into other problems of a purely classical nature. For example, a strong lower bound on the classical communication complexity of the inner product function can be obtained based on quantum information-theoretic principles [31]. Ideas from quantum computing have also been used to prove new limitations on classical data structures, codes and formulae [33].

9 Outlook

We have described a rather large number of quantum algorithms, solving a rather large number of problems. However, one might still ask why more algorithms are not known – and in particular, more exponential speedups?

One reason is that strong lower bounds have been proven on the power of quantum computation in the query complexity model, where one considers only the number of queries to the input as the measure of complexity. For example, the complexity achieved by Grover’s algorithm cannot be improved by even one query while maintaining the same success probability [100]. More generally, in order to achieve an exponential speedup over classical computation in the query complexity model there has to be a promise on the input, i.e. some possible inputs must be disallowed [11]. This is one reason behind the success of quantum algorithms in cryptography: the existence of hidden problem structure which quantum computers can exploit in ways that classical computers cannot. Finding such hidden structure in other problems of practical interest remains an important open problem.

In addition, a cynical reader might point out that known quantum algorithms are mostly based on a rather small number of quantum primitives (such as the quantum Fourier transform and quantum walks). An observation attributed to van Dam⁷ provides some justification for this. It is known that any quantum circuit can be approximated using only Toffoli and Hadamard quantum gates [88]. The first of these is a purely classical gate, and the second is equivalent to the Fourier transform over the group \mathbb{Z}_2 . Thus any quantum algorithm whatsoever can be expressed as the use of quantum Fourier transforms interspersed with classical processing! However, the intuition behind the quantum algorithms described above is much more varied than this observation would suggest. The inspiration for other quantum algorithms, not discussed here, includes topological quantum field theory [43]; connections between quantum circuits and spin models [32]; the Elitzur-Vaidman quantum bomb tester [63]; and directly solving the semidefinite programming problem characterising quantum query complexity [81, 12].

As well as the development of new quantum algorithms, an important direction for future research seems to be the application of known quantum algorithms (and algorithmic primitives) to new problem areas. This is likely to require significant input from, and communication with, practitioners in other fields.

⁷See <http://dabacon.org/pontiff/?p=1291>.

Acknowledgements

This work was supported by the UK EPSRC under Early Career Fellowship EP/L021005/1. Thanks to many people including Patrick Birchall, Steve Brierley, Aram Harrow and Tom Wong for comments which improved previous versions of the paper.

References

- [1] S. Aaronson. Quantum machine learning algorithms: Read the fine print. *Nature Physics*, 11:291–293, 2015.
- [2] S. Aaronson and A. Arkhipov. The computational complexity of linear optics. *Theory of Computing*, 9(4):143–252, 2013. [arXiv:1011.3245](#).
- [3] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani. Quantum walks on graphs. In *Proc. 33rd Annual ACM Symp. Theory of Computing*, pages 50–59, 2001. [quant-ph/0012090](#).
- [4] A. Ambainis. Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations. In *Proc. 29th Annual Symp. Theoretical Aspects of Computer Science*, pages 636–647, 2012. [arXiv:1010.4458](#).
- [5] A. Ambainis, A. Childs, B. Reichardt, R. Špalek, and S. Zhang. Any AND-OR formula of size n can be evaluated in time $n^{1/2+o(1)}$ on a quantum computer. *SIAM J. Comput.*, 39(6):2513–2530, 2010. [quant-ph/0703015](#).
- [6] A. Ambainis, L. J. Schulman, A. Ta-Shma, U. Vazirani, and A. Wigderson. The quantum communication complexity of sampling. *SIAM J. Comput.*, 32(6):1570–1585, 2003.
- [7] A. Aspuru-Guzik and P. Walther. Photonic quantum simulators. *Nature Physics*, 8:285–291, 2012.
- [8] D. Bacon and W. van Dam. Recent progress in quantum algorithms. *C. ACM*, 53(2):84–93, 2010.
- [9] B. Barak, A. Moitra, R. O’Donnell, P. Raghavendra, O. Regev, D. Steurer, L. Trevisan, A. Vijayaraghavan, D. Witmer, and J. Wright. Beating the random assignment on constraint satisfaction problems of bounded degree, 2015. [arXiv:1505.03424](#).
- [10] S. Barz, I. Kassal, M. Ringbauer, Y. Ole Lipp, B. Dakic, A. Aspuru-Guzik, and P. Walther. Solving systems of linear equations on a quantum computer. *Scientific Reports*, 4:115, 2014. [arXiv:1302.1210](#).
- [11] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. [quant-ph/9802049](#).
- [12] A. Belovs. Quantum algorithms for learning symmetric juntas via adversary bound. In *Proc. 29th Annual IEEE Conf. Computational Complexity*, pages 22–31, 2014. [arXiv:1311.6777](#).
- [13] D. Berry. High-order quantum algorithm for solving linear differential equations. *J. Phys. A: Math. Gen.*, 47:105301, 2014. [arXiv:1010.2745](#).
- [14] D. Berry, A. Childs, and R. Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters, 2015. [arXiv:1501.01715](#).
- [15] R. Blatt and C. Roos. Quantum simulations with trapped ions. *Nature Physics*, 8:277–284, 2012.
- [16] D. Boneh and R. Lipton. Quantum cryptanalysis of hidden linear functions. In *Proc. CRYPTO’95*, pages 424–437, 1995.
- [17] A. Bookatz. QMA-complete problems. *Quantum Inf. Comput.*, 14(5&6):361–383, 2014. [arXiv:1212.6312](#).
- [18] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortschr. Phys.*, 46(4–5):493–505, 1998. [quant-ph/9605034](#).
- [19] G. Brassard and P. Høyer. An exact quantum polynomial-time algorithm for Simon’s problem. In *Theory of Computing and Systems, Proceedings of the Fifth Israeli Symposium on*, pages 12–23, 1997. [quant-ph/9704027](#).
- [20] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. *Quantum Computation and Quantum Information: A Millennium Volume*, pages 53–74, 2002. [quant-ph/0005055](#).

- [21] K. Brown, W. Munro, and V. Kendon. Using quantum computers for quantum simulation. *Entropy*, 12(11):2268–2307, 2010. [arXiv:1004.5528](#).
- [22] J. Buhler, H. W. Lenstra Jr., and C. Pomerance. Factoring integers with the number field sieve. In *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*, pages 50–94. Springer, 1993.
- [23] H. Buhrman, R. Cleve, S. Massar, and R. de Wolf. Non-locality and communication complexity. *Rev. Mod. Phys.*, 82(1):665–698, 2010. [arXiv:0907.3584](#).
- [24] I. Bulata and F. Nori. Quantum simulators. *Science*, 326(5949):108–111, 2009.
- [25] X.-D. Cai, C. Weedbrook, Z.-E. Su, M.-C. Chen, M. Gu, M.-J. Zhu, L. Li, N.-L. Liu, C.-Y. Lu, and J.-W. Pan. Experimental quantum computing to solve systems of linear equations. *Phys. Rev. Lett.*, 110:230501, 2013. [arXiv:1302.4310](#).
- [26] A. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. Spielman. Exponential algorithmic speedup by a quantum walk. In *Proc. 35th Annual ACM Symp. Theory of Computing*, pages 59–68, 2003. [quant-ph/0209131](#).
- [27] A. Childs, E. Farhi, and S. Gutmann. An example of the difference between quantum and classical random walks. *Quantum Information Processing*, 1:35–43, 2002. [quant-ph/0103020](#).
- [28] A. Childs and W. van Dam. Quantum algorithms for algebraic problems. *Reviews of Modern Physics*, 82:1–52, 2010. [arXiv:0812.0380](#).
- [29] V. Choi. Different adiabatic quantum optimization algorithms for the NP-complete exact cover and 3SAT problems. *Quantum Inf. Comput.*, 11(7&8):0638–0648, 2011. [arXiv:1010.1221](#).
- [30] B. Clader, B. Jacobs, and C. Sprouse. Preconditioned quantum linear system algorithm. *Phys. Rev. Lett.*, 110:250504, 2013. [arXiv:1301.2340](#).
- [31] R. Cleve, W. van Dam, M. Nielsen, and A. Tapp. Quantum entanglement and the communication complexity of the inner product function. In *Selected papers from the First NASA International Conference on Quantum Computing and Quantum Communications*, pages 61–74, 1998. [quant-ph/9708019](#).
- [32] G. De las Cuevas, W. Dür, M. van den Nest, and M. Martin-Delgado. Quantum algorithms for classical lattice models. *New J. Phys.*, 13:093021, 2011. [arXiv:1104.2517](#).
- [33] A. Drucker and R. de Wolf. Quantum proofs for classical theorems. *Theory of Computing Graduate Surveys*, 2:1–54, 2011. [arXiv:0910.3376](#).
- [34] C. Dürr, M. Heiligman, P. Høyer, and M. Mhalla. Quantum query complexity of some graph problems. In *Proc. 31st International Conference on Automata, Languages and Programming (ICALP’04)*, pages 481–493, 2004. [quant-ph/0401091](#).
- [35] C. Dürr and P. Høyer. A quantum algorithm for finding the minimum, 1996. [quant-ph/9607014](#).
- [36] E. Farhi, J. Goldstone, D. Gosset, S. Gutmann, H. Meyer, and P. Shor. Quantum adiabatic algorithms, small gaps, and different paths. *Quantum Inf. Comput.*, 11(3):0181–0214, 2011. [arXiv:0909.4766](#).
- [37] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem, 2014. [arXiv:1412.6062](#).
- [38] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science*, 292(5516):472–475, 2001. [quant-ph/0104129](#).
- [39] E. Farhi, J. Goldstone, S. Gutmann, and D. Nagaj. How to make the quantum adiabatic algorithm fail. *Int. J. Quantum Inform.*, 6:503, 2008. [quant-ph/0512159](#).
- [40] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Quantum computation by adiabatic evolution. Technical Report MIT-CTP-2936, MIT, 2000. [quant-ph/0001106](#).
- [41] R. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6–7):467–488, 1982.
- [42] A. Fowler, M. Mariantoni, J. Martinis, and A. Cleland. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A*, 86:032324, 2012. [arXiv:1208.0928](#).
- [43] M. Freedman, M. Larsen, and Z. Wang. A modular functor which is universal for quantum computation. *Comm. Math. Phys.*, 227(3):605–622, 2002. [quant-ph/0001108](#).

- [44] I. Georgescu, S. Ashhab, and F. Nori. Quantum simulation. *Rev. Mod. Phys.*, 86:153, 2014. [arXiv:1308.6253](#).
- [45] L. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.*, 79(2):325–328, 1997. [quant-ph/9706033](#).
- [46] S. Hallgren. Polynomial-time quantum algorithms for Pell’s equation and the principal ideal problem. *J. ACM*, 54(1):4:1–4:19, 2007.
- [47] A. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 15(103):150502, 2009. [arXiv:0811.3171](#).
- [48] M. Hastings, D. Wecker, B. Bauer, and M. Troyer. Improving quantum algorithms for quantum chemistry. *Quantum Inf. Comput.*, 15(1–2):1–21, 2015. [arXiv:1403.1539](#).
- [49] B. Higgins, D. Berry, S. Bartlett, H. Wiseman, and G. Pryde. Entanglement-free Heisenberg-limited phase estimation. *Nature*, 450:396–396, 2007. [arXiv:0709.2996](#).
- [50] J. Huh, G. Guerreschi, B. Peropadre, J. McClean, and A. Aspuru-Guzik. Boson sampling for molecular vibronic spectra, 2014. [arXiv:1412.8427](#).
- [51] M. Johnson, M. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. Berkley, J. Johansson, P. Bunyk, E. Chapple, C. Enderud, J. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. Thom, E. Tolkacheva, C. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–198, 2011.
- [52] S. Jordan. The quantum algorithm zoo. <http://math.nist.gov/quantum/zoo/>.
- [53] S. Jordan, K. Lee, and J. Preskill. Quantum algorithms for quantum field theories. *Science*, 336(6085):1130–1133, 2012. [arXiv:1111.3633](#).
- [54] J. Kempe. Quantum random walks hit exponentially faster. *Probability Theory and Related Fields*, 133(2):215–235, 2005. [quant-ph/0205083](#).
- [55] J. King, S. Yarkoni, M. Nevisi, J. Hilton, and C. McGeoch. Benchmarking a quantum annealing processor with the time-to-target metric, 2015. [arXiv:1508.05087](#).
- [56] A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation*, volume 47 of *Graduate Studies in Mathematics*. AMS, 2002.
- [57] T. Kleinjung, K. Aoki, J. Franke, A. Lenstra, E. Thomé, J. Bos, P. Gaudry, A. Kruppa, P. Montgomery, D. Osvik, H. te Riele, A. Timofeev, and P. Zimmermann. Factorization of a 768-bit RSA modulus. In *CRYPTO’10*, pages 333–350, 2010.
- [58] G. Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput.*, 35(1):170–188, 2005. [quant-ph/0302112](#).
- [59] T. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, and J. O’Brien. Quantum computing. *Nature*, 464:45–53, 2010. [arXiv:1009.2267](#).
- [60] B. Lanyon, C. Hempel, D. Nigg, M. Müller, R. Gerritsma, F. Zähringer, P. Schindler, J. Barreiro, M. Rambach, G. Kirchmair, M. Hennrich, P. Zoller, R. Blatt, and C. Roos. Universal digital quantum simulations with trapped ions. *Science*, 334(6052):57–61, 2011. [arXiv:1109.1512](#).
- [61] F. Le Gall. Improved quantum algorithm for triangle finding via combinatorial arguments. In *Proc. 55th Annual Symp. Foundations of Computer Science*, pages 216–225, 2014. [arXiv:1407.0085](#).
- [62] S. Leyton and T. Osborne. A quantum algorithm to solve nonlinear differential equations, 2008. [arXiv:0812.4423](#).
- [63] C. Lin and H. Lin. Upper bounds on quantum query complexity inspired by the Elitzur-Vaidman bomb tester. In *Proc. 30th Annual IEEE Conf. Computational Complexity*, pages 537–566, 2015. [arXiv:1410.0932](#).
- [64] S. Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996.
- [65] S. Lloyd, M. Mohseni, and P. Rebentrost. Quantum algorithms for supervised and unsupervised machine learning, 2013. [arXiv:1307.0411](#).
- [66] F. Magniez, A. Nayak, J. Roland, and M. Santha. Search via quantum walk. *SIAM J. Comput.*, 40(1):142–164, 2011. [quant-ph/0608026](#).

- [67] E. Martín-López, C. Harrold, C. Sparrow, N. Russell, J. Carolan, N. Matsuda, M. Oguma, M. Itoh, T. Hashimoto, D. Tew, J. O’Brien, and A. Laing. Simulating molecular vibrations with photons, 2015. in preparation.
- [68] E. Martín-López, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. O’Brien. Experimental realisation of Shor’s quantum factoring algorithm using qubit recycling. *Nature Photonics*, 6:773–776, 2012. [arXiv:1111.4147](#).
- [69] C. McGeoch and C. Wang. Experimental evaluation of an adiabatic quantum system for combinatorial optimization. In *Proc. ACM International Conference on Computing Frontiers (CF’13)*, pages 23:1–23:11, 2013.
- [70] A. Montanaro. Quantum pattern matching fast on average. *Algorithmica*, pages 1–24, 2015. [arXiv:1408.1816](#).
- [71] M. Mosca. Quantum algorithms. In *Computational Complexity*, pages 2303–2333. Springer, 2012. [arXiv:0808.0369](#).
- [72] J. Pan, Y. Cao, X. Yao, Z. Li, C. Ju, X. Peng, S. Kais, and J. Du. Experimental realization of quantum algorithm for solving linear systems of equations. *Phys. Rev. A*, 89:022313, 2014. [arXiv:1302.1946](#).
- [73] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
- [74] D. Poulin, M. Hastings, D. Wecker, N. Wiebe, A. Doherty, and M. Troyer. The Trotter step size required for accurate quantum simulation of quantum chemistry. *Quantum Inf. Comput.*, 15(5&6):361–384, 2015. [arXiv:1406.4920](#).
- [75] M. Prasad, A. Biere, and A. Gupta. A survey of recent advances in SAT-based formal verification. *International Journal on Software Tools for Technology Transfer*, 7(2):156–173, 2005.
- [76] J. Proos and C. Zalka. Shor’s discrete logarithm quantum algorithm for elliptic curves. *Quantum Inf. Comput.*, 3(4):317–344, 2003. [quant-ph/0301141](#).
- [77] T. Ralph. Quantum computation: Boson sampling on a chip. *Nature Photonics*, 7:514–515, 2013.
- [78] H. Ramesh and V. Vinay. String matching in $\tilde{O}(\sqrt{n} + \sqrt{m})$ quantum time. *Journal of Discrete Algorithms*, 1:103–110, 2003. [quant-ph/0011049](#).
- [79] O. Regev. Quantum computation and lattice problems. *SIAM J. Comput.*, 33(3):738–760, 2004. [cs/0304005](#).
- [80] O. Regev. A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space, 2004. [quant-ph/0406151](#).
- [81] B. Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function. In *Proc. 50th Annual Symp. Foundations of Computer Science*, pages 544–551, 2009. [arXiv:0904.2759](#).
- [82] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *C. ACM*, 21(2):120–126, 1978.
- [83] T. Rønnow, Z. Wang, J. Job, S. Boixo, S. Isakov, D. Wecker, J. Martinis, D. Lidar, and M. Troyer. Defining and detecting quantum speedup. *Science*, 345(6195):420–424, 2014. [arXiv:1401.2910](#).
- [84] M. Santha. On the Monte Carlo boolean decision tree complexity of read-once formulae. *Randomized Structures and Algorithms*, 6(1):75–87, 1995.
- [85] M. Santha. Quantum walk based search algorithms. In *Theory and Applications of Models of Computation*, pages 31–46, 2008. [arXiv:0808.0059](#).
- [86] U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proc. 40th Annual Symp. Foundations of Computer Science*, pages 410–414, 1999.
- [87] N. Schuch and F. Verstraete. Computational complexity of interacting electrons and fundamental limitations of Density Functional Theory. *Nature Physics*, 5:732–735, 2009. [arXiv:0712.0483](#).
- [88] Y. Shi. Both Toffoli and controlled-NOT need little help to do universal quantum computing. *Quantum Inf. Comput.*, 3(1):84–92, 2003. [quant-ph/0205115](#).
- [89] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26:1484–1509, 1997. [quant-ph/9508027](#).
- [90] J. Smolin, G. Smith, and A. Vargo. Oversimplifying quantum factoring. *Nature*, 499:163–165, 2013. [arXiv:1301.7007](#).

- [91] S. Somaroo, C. Tseng, T. Havel, R. Laflamme, and D. Cory. Quantum simulations on a quantum computer. *Phys. Rev. Lett.*, 82:5381, 1999. [quant-ph/9905045](#).
- [92] M. Szegedy. Quantum speed-up of Markov chain based algorithms. In *Proc. 45th Annual Symp. Foundations of Computer Science*, pages 32–41, 2004. [quant-ph/0401053](#).
- [93] S. Trotzky, Y-A. Chen, A. Flesch, I. McCulloch, U. Schollwöck, J. Eisert, and I. Bloch. Probing the relaxation towards equilibrium in an isolated strongly correlated one-dimensional Bose gas. *Nature Physics*, 8:325–330, 2012. [arXiv:1101.2659](#).
- [94] W. van Dam, M. Mosca, and U. Vazirani. How powerful is adiabatic quantum computation? In *Proc. 42nd Annual Symp. Foundations of Computer Science*, pages 279–287. IEEE, 2001. [quant-ph/0206003](#).
- [95] L. Vandersypen, M. Steffen, M. Sherwood, C. Yannoni, G. Breyta, and I. Chuang. Implementation of a three-quantum-bit search algorithm. *Appl. Phys. Lett.*, 76(5):646–648, 2000. [quant-ph/9910075](#).
- [96] D. Wecker, B. Bauer, B. Clark, M. Hastings, and M. Troyer. Gate count estimates for performing quantum chemistry on small quantum computers. *Phys. Rev. A*, 90:022305, 2014. [arXiv:1312.1695](#).
- [97] N. Wiebe, D. Braun, and S. Lloyd. Quantum algorithm for data fitting. *Phys. Rev. Lett.*, 109:050505, 2012. [arXiv:1204.5242](#).
- [98] M. Wilde. *Quantum Information Theory*. Cambridge University Press, 2013.
- [99] R. Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013.
- [100] C. Zalka. Grover’s quantum searching algorithm is optimal. *Phys. Rev. A*, 60(4):2746–2751, 1999. [quant-ph/9711070](#).