北京郵電大學

学　　院：　计算机学院

班　　级：　2018211302

小　组　成　员：

谯森　学　号:2018210895

杨成栋 学　号:2018210913

熊宇　学　号:2018210074

陈万鑫 学　号:2018210614

量子算法：概述

Ashley Montanaro

量子计算机因执行量子算法而优于标准计算机。量子算法可以应用的领域包括密码学、搜索和优化、量子系统模拟和求解大型线性方程组。这里，我们简要地调查了一些已知的量子算法，本文的重点是对它们的应用进行广泛概述，而不是介绍它们的技术细节。我们也对量子算法的最新发展和近期应用进行了讨论。

1 介绍

量子计算机被设计使用量子力学来解决所有基于经典物理定律的计算机解决不了的问题。量子计算的应用领域包括最开始产生进展的密码学系统到对新机器的设计。这些应用都是基于量子算法——一种运行在量子计算机上并相对于任何经典算法实现加速或者其他效率上提高的算法——实现的。虽然大规模通用量子计算机还不存在，但是对量子算法理论的研究领域已经活跃了超过 20 年。在这里，我们的目的是给出一个广泛的量子算法概述，重点放在具有明确的应用和严格的性能界限的算法上，并包括最近在该领域的进展。

与人们普遍认为量子计算机应用领域很少的观点相反，量子算法领域已经发展成为一个足够大的研究领域，这样一项简短的调查不能对其进行全面的概括。事实上，在编写‘Quantum Algorithm Zoo’这一网站时引用了 262 篇关于量子算法的文章。现在有许多关于量子算法的优秀调查，我们列举了一些，还有更多我们没有列举到的。特别地，我们忽略了所有关于所提到的量子算法是如何工作的讨论。我们也不会涉及如何实际构建量子计算机（在理论上或实践中）和量子纠错，也不涉及量子通信复杂性或量子香农理论。

2 测量量子加速

量子计算机相对于经典计算机可以更快地解决问题这句话意味着什么？正如计算复杂性理论中的典型问题那样，我们通常会考虑渐进复杂度（Asymptotic Complexity），例如具有不同大小的问题的运行时间或空间使用，而不是固定大小的单个问题。在经典和量子环境中，我们用算法使用的基本操作的数量来度量运行时间。 在量子计算的情况下，这可以使用量子电路模型来测量，其中量子电路是一系列称为量子门的基本量子操作，每个量子操作都应用于少量的量子比特（quantum bits）。为了比较算法的性能，我们使用具有计算机科学风格的来表示，它被解释为‘的渐近上界’。

我们有时使用计算复杂性理论的基本思想，特别是复杂性类的概念，它们是按困难程度分组的问题。一些重要复杂类的非正式描述见表 1。如果一个问题被认为是一个完整复杂类的问题，那么这意味着它是该类中“最难”的问题之一：它包含在该类中，该类中的每一个其他问题都可以归约到它。

Table 1. Some computational complexity classes of importance in quantum computation

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

Abbreviation: QMA, Quantum Merlin–Arthur.
'Polynomial time' is short for 'in time polynomial in the input size'.

3 隐藏的子群问题和密码学的应用

量子计算机的第一批应用之一是 Shor 的整数的因式分解算法。在因式分解问题中，给定一个整数 $N = p \times q$，我们的任务是确定 p 和 q，其中 p, q 是素数。已知的最快的经典算法（一般数域筛法）运行的时间复杂度是 $\exp(O(\log N)^{1/3}(\log\log N)^{2/3}))^{12}$ (事实上，这是一个启发式界限；严格的界限会稍高一些），而 Shor 的量子算法在时间 $O((\log N)^3)$ 中更快地解决了这个问题。如果不是因为广泛使用的 RSA 公钥密码系统依赖于整数分解，这个结论只会在数学领域出现。Shor 的有效的因式分解算法意味着这个密码系统面对对大型量子计算机的攻击是不安全的。

有一个比上述渐近运行时间更具体的比较，2010 年 Kleinjung 报告了 768 位数字的经典因式分解，在 2 年内使用了数百台现代计算机，总共计算了约 10^{20} 个操作。在一个对一种容错量子计算体系结构的详细分析中，作者对底层硬件做出了合理的假设，表明使用约 3×10^{11} 量子门的量子计算机可以分解一个 2000 位的数字，以及大约 10 亿个量子位，并且可以以 10MHz 的时钟速率运行一天以上。这显然超出了目前的技术，但作为一个长期目标似乎并不不切实际。

Shor 的整数因式分解方法是基于将任务减少到一个被称为隐藏子群问题(HSP)的数学问题的特殊情况，然后给出一个有效的量子算法来解决这个问题。HSP 有一个为群 G 的参数，Shor 算法解决了 G=$\mathbb{Z}$的情况。对于其他群 G 的 HSP 的有效解决方案意味着存在有效的算法来破坏其他密码系统我们在表 2 中总结了 HSP 的一些重要案例及其相应的密码系统。HSP 特别有趣的情况是多项式时间的量子算法目前还不能解决二面体和对称群。对于前一种情况，多项式时间量子算法将给出一种有效的算法来在格中寻找最短的向量；对于后一种情况，有效的量子算法将给出图的同构的有效检验。

4 搜索和优化

计算机科学中最基本的问题之一是非结构化搜索。 这个问题可以形式化为如下问题：

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

很容易看出，在没有关于 f 的先验信息的情况下，任何经典算法，都必须在最坏的情况下进行 f $N = 2^n$ 次评估，从而确定性地解决非结构化搜索问题。即使我们找到了一个随机算法可以成功地运行，在最坏的情况下也有 1/2 的概率得到其所需的评估量级是 N 阶。然而，值得注意的是，Grover 提出的一种量子算法，在最坏的情况下，它使用 f 的 $O(\sqrt{N})$ 时间来解决这个问题(Grover 的原始算法解决了只有一种解决方案的特殊情况；对多个解决方案的扩展稍晚)。该算法是有一定误差的，即对任意小（但一定的） $\varepsilon > 0$，它有 ε 的概率失效。

虽然 f 可能有某种内部结构，但 Grover 的算法

根本没有使用它；我们说 f 在算法中被当作黑匣子。

Grover 的算法可以立即应用于 NP 难的任何问题。这个类封装了决策问题，其解决方案可以在以下意义上被有效地检验：存在一个有效的经典检查算法 A，对于答案应该是“yes”的问题的任何实例，A 都有一个证明，也就是说 A 接受这个证明。换句话说，该证明是一个答案是“yes”的，可以由 A 检查的证明。另一方面，对于答案应该是“no”的任何情况，都不应该有任何证明可以使 A 接受它。NP 难包含许多重要的问题，涉及优化和约束满意度。

给定具有长度为 m 项的 NP 中的一个问题，通过将 Grover 的算法应用于 A 并搜索所有可能的项，我们得到了一个使用时间为 $O(2^{m/2}\,poly(m))$ 的算法，而不是经典穷举搜索在所有项上使用的 $O(2^{m}\,poly(m))$ 时间。这种二次加速比 Shor 算法实现的超多项式加速要慢，但仍然可以相当可观。事实上，如果量子计算机以与经典计算机大致相同的时钟速度运行，那么这意味着大约两倍规模的问题可以在相当的时间内得到解决。

作为一个典型的例子，考虑如图 1 所示的基本的 NP 完全电路的可满足性问题(SAT 电路)。这个问题的一个实例是：一个电子电路，包括 AND，OR 和 NOT 门，以 n 位作为输入，产生 1 位输出。问题的目的是是确定电路是否存在一个使输出为 1 的输入。SAT 电路算法可以用来解决例如设计自动化、电路等效和模型检查这样与电子电路有关的大量问题，已知的 SAT 电路的最佳经典算法在 n 个输入变量时最坏情况下运行时间为 2^{n}，即不比穷举搜索快得多。通过将 Grover 算法应用于在输入为 $x\in\{0,1\}^{n}$、计算电路的函数 $f(x)$ 上，我们可以得到运行时间为 $O(2^{n/2}\,poly(n))$，其中 $poly(n)$ 来自在给定输入上计算电路所花费的时间。

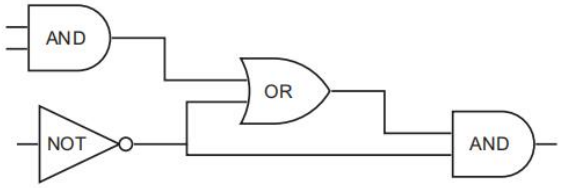

Figure 1. An instance of the Circuit SAT problem. The answer should be ‘yes’ as there exists an input to the circuit such that the output is 1.

5 振幅放大

Grover 算法加快了非结构化搜索的朴素经典算法。 量子算法也可以加速更复杂的经典算法。

Heuristic search problem: Given the ability to execute a probabilistic ‘guessing’ algorithm $\mathcal{A}$, and a ‘checking’ function f, such that $\Pr[\mathcal{A}$ outputs w such that $f(w)=1]=\varepsilon$, output w such that $f(w)=1$.

解决启发式搜索问题的一种经典方法是重复运行 A，每次使用 f 检查输出，这将导致 f 平均需要 $O(1/\varepsilon)$ 时间。然而，Brassard、Høyer、Mosca 和 Tapp 的量子算法可以在 $O(1/\sqrt{\varepsilon})$ 找到使得 $f(w)=1$ 的 w，出错概率接近 0，从而实现二次加速。类比经典概率放大，这种算法被称为振幅放大。

通过将输出一个均匀随机的 n 位字符串的 A 作为算法，这个框架适合上面讨论的非结构化搜索问题。进一步地，如果有 k 个输入，w∈{0，1}n，使得 f（w）=1，然后

Pr[A 输出 w，以便 f（w）=1]=$\frac{k}{N}$。

所以我们可以找到一个 w，使得 f（w）=1，对 f 进行 O($\sqrt{N/k}$)查询。然而，我们可以想象 A 是针对我们想要解决的特定问题的一个更复杂的算法或启发式算法。例如，基本 NP 完全约束满足问题 3-SAT 的一个最有效的经典算法是随机化，运行时间为 O($(4/3)^{n}$poly(n))。振幅放大可应用于该算法以获得运行时间为 O($(4/3)^{n}$poly(n))的量子算法，说明量子计算机可以加速求解 NP 完全问题的非平凡经典算法。

量子算法的一个有趣的未来方向是寻找优化问题的精确近似解。Farhi、Goldstone 和 Gutmann

最近的工作给出了组合任务（同时满足许多特定形式的线性方程组）的第一个量子算法，在精度方面优于已知的最佳有效经典算法；在这种情况下，用满足的方程组的分数来衡量。这启发了一个更加有效的经典算法来解决同样的问题，但是留下了一个悬而未决的问题，即在准确性上，用于优化问题的量子算法是否能实质上超越经典算法。

Grover 算法和振幅放大 Grover 算法以及振幅放大的应用是强大的子程序，可以作为更为复杂的量子算法的一部分使用，从而可以获得许多问题的量子加速。我们在这里只列出其中的一些加速例子。

1. 求 N 个未排序的整数列表的最小值（等价地，求任意且初值未知的函数 f:{0,1}n->ℤ)的最小值。由 Dürr 和 Høyer 所给出的量子算法在 O() 时间复杂度上通过对 f 求值来解决这个问题，相对于经典算法给出了一个二次加速。他们的算法基于 Grover 算法在由 g（x）=1 定义的函数 g:{0，1}n→{0，1}的应用，当且仅当 f（x）<T 为某个阈值 T 时，该阈值最初是随机的，然后随着输入 x 的发现而更新，使得 f（x）低于阈值。
2. 确定图的连通性。在最坏的情况下，要确定 N 个顶点的图是否连通，通常需要 N2 阶时间。Dürr，Heiligman，Høyer 和 Mhalla 给出了一个在时间 O（N3/2）内求解该问题的量子算法，以及其它一些图论问题（强连通性、最小生成树、最短路径）的有效算法。
3. 模式匹配，文本处理和生物信息学中的一个基本问题。这里的任务是在长度为 N 的文本 T 中找到长度为 M 的给定模式 P，其中模式和文本是一些字母表上的字符串。Ramesh 和 Vinay 给出了一个量子算法，该算法与最佳经典复杂度 O（N+M）相比能在时间 O(+)解决该问题。上述俩个时间复杂度都是最坏情况下的时间界限，但也可以考虑平均情况，其中文本和模式都是随机选取的。这的量子加速更明显：有一种结合了振幅放大和二面体隐子群问题思想的量子算法，运行时间复杂度为 O()到对数因子，同最好的经典运行时间复杂度 O(N/M+)相比，当 M 很大时，是一个超多项式加速。

6 绝热优化

量子绝热算法提供了量子组合优化的另一种方法。绝热算法可应用于任何约束满足问题（CSP），问题被赋予一系列约束，要求问题为输入位输出一个分配，最大化地满足约束数量。许多这样的问题是 NP 完全的，具有重要的实际意义。算法背后的基本思想是物理激励，而且还基于 csp 和物理系统之间的对应关系。我们从量子态开始，它是 CSP 所有可能解的均匀叠加。这是哈密顿量的基态（最低能量），可以很容易地制备出来。然后对该哈密顿量进行逐步修正，给出一个新的哈密顿量，其基态对满足约束数最大化的解进行编码。量子绝热定理保证，如果这个过程进行得足够慢，系统将始终保持在基态。特别地，最终状态给出了 CSP 的最优解。这里的关键是“足够慢”，对于 n 比特上的 CSP 的一些实例，所需的时间可能是 n 的指数。

与本调查其余部分描述的算法不同，绝热算法在运行时缺乏一般的、严格的最坏情况上界。虽然在较小的情况下对其性能进行快速评估是不可行的。我们可以构造问题实例，在这些实例上，标准绝热算法可证明需要指数时间，然而，改变算法可以回避其中一

Table 2. Some problems which can be expressed as hidden subgroup problems

Problem	*Group*	*Complexity*	*Cryptosystem*
Factorisation	$\mathbb{Z}$	Polynomial[11]	RSA
Discrete log	$\mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1}$	Polynomial[11]	Diffie-Hellman, DSA,…
Elliptic curve discrete log	Elliptic curve	Polynomial[92]	ECDH, ECDSA,…
Principal ideal	$\mathbb{R}$	Polynomial[93]	Buchmann-Williams
Shortest lattice vector	Dihedral group	Subexponential[94,95]	NTRU, Ajtai-Dwork,…
Graph isomorphism	Symmetric group	Exponential	—

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.

些争议点。

绝热算法可以在通用量子计算机上实现。然而，它也有助于绝热算法直接在物理系统上实现，其哈密顿量可以在所期望的初始和最终哈密顿量之间平滑地变化。D-Wave Systems 公司是这种方法的最突出的倡导者，它设计并制造了用于实现该算法的大型机器，最新的此类机器（“D-Wave 2X”）拥有高达 1152 个量子比特。对于 CSP 的某些实例，这些机器已经被证明在性能上优于在标准计算机上运行的经典解算器，虽然加速(或其他)似乎有相当微妙的依赖于问题实例、经典求解器比较和比较度量。

在上述绝热算法的理论挑战的同时，D 波系统还面临一些重大的实际挑战。特别地，这些机器不会始终保持在基态，而是处于绝对零度以上的热态。因此，实际执行的算法与经典模拟退火有一些相似之处，因此被称为“量子退火”。 目前还不清楚绝热算法所预测的量子加速是否会在这种情况下持续存在。

7 量子模拟

在古典计算的早期，计算机技术的一个主要应用是物理系统的模拟（这种应用至少可以追溯到公元前 2 世纪的 Antikythera 机制）。

同样，量子计算机最重要的早期应用可能是量子系统的模拟。量子模拟的应用包括量子化学、超导电性、超材料和高能物理。事实上，人们期望量子模拟能帮助我们理解任何量子力学作用的系统。

“模拟”一词可以用来描述许多问题，但在量子计算中，通常是指计算系统动力学性质的问题。这可以更具体地表述为：给定一个描述物理系统的哈密顿量 H，以及该系统的初始状态$|\psi>$的描述，输出状态$|\psi t>=|\psi>$的一些性质，对应于根据该哈密顿量演化系统的时间 t。因为所有量子系统都服从薛定谔方程，这是一项很重要且基础的性质。然而，完全描述一般量子态的指数复杂度表明，要有效地实现经典的量子模拟是不可能的，而且目前还不知道有效的量子模拟的一般经典算法。这个问题最初促使了费曼去寻求量子计算机能否有效地模拟量子力学。

一般用途的量子计算机确实可以在许多物理现实的情况下有效地模拟量子力学，例如相互作用受局域性限制的系统。给出量子状态$|\psi>$的描述、H 的描述和时间 t，量子模拟算法产生一个近似于$|\psi t>$的状态。然后可以对该状态进行测量，以确定有关它的量。该算法以多项式时间运行，模拟系统的大小（即量子比特数）和期望的进化时间，比已知的最好的一般经典算法有指数级的加速。然而，量子模拟仍有改进的空间，量子模拟仍然是一个活跃的研究课题。例如，在保持快速运行时间的同时提高量子模拟的精确度，针对量子化学等特殊应用优化算法，以及探索量子场论等新领域的应用。

上述非常普遍的方法有时被称为数字量子模拟：我们假设我们有一台大型通用量子计算机，并在其上运行量子模拟算法。相比之下，在模拟量子模拟中，我们直接用另一个物理系统来模拟。也就是说，如果我们想用哈密顿量 H 来模拟一个系统，那么我们就可以建立另一个可以用近似于 H 的哈密顿量来描述的系统。如果第二个系统比第一个系统更易于构建、运行或从中提取信息，我们就获得了一些成果。对于某些系统，模拟量子模拟可能比数字量子模拟更容易实现，但其灵活性较差。因此，预计将首先实现性能优于经典模拟器的模拟量子模拟器。

8 量子行走

在经典的计算机科学中，随机游走或马尔可夫链的概念是一个强大的算法工具，经常被应用于搜索和采样问题。量子行走为设计快速量子算法提供了一个同样强大和通用的框架。正如随机游走算法是基于一个粒子在某种底层图结构中随机运动的模拟运动，量子行走是基于在图上运动的粒子的模拟相干量子演

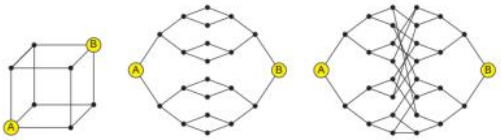

图 2. 三个图，其自然推广到 N 个顶点的经典随机游走比量子行走需要更多的时间才能从入口（a）到达出口（B）。然而，在前两个图上，存在有效的经典算法来寻找出口，这些算法不是基于随机游走的。

化。

量子行走算法通常利用量子行走优于随机行走的两种方式之一：更快的命中（即从源顶点找到目标顶点所需的时间）和更快的混合（即从一个源顶点开始后在所有顶点上展开所需的时间）。对于某些图来说，量子行走的命中时间可能比经典的图要少。量子混合时间和经典混合时间的间隔可以是二次的，但不超过这个（近似值）。

量子和经典混合时间可以是二次的，但不能超过这个(大致上)。然而，快速混合已被证明是一个非常有用的工具，以获得一般的速度比经典算法。

图 2 说明了三种图族的特殊情况，在这三种图族中，量子行走示出比随机行走更快的撞击速度:超立方体,“粘树”图，以及中间添加了一个随机循环的“粘树”图。第三个例子特别有趣，因为在导航图方面，量子行走可以显示出优于任何经典算法，即使不是基于随机行走。从入口(左边)开始，运行时间 0(log N)的连续时间量子路径找到出口(右边)的概率至少为 1/poly(log N).然而,任何经典算法都需要(N ^1/6)阶的时间来找到出口。直观地说，经典算法开始可以快速前进，但随后会”卡在”图中间的随机部分。量子行走的相干性和对称性使它基本上看不到这种随机性，它有效地从左向右前进。

量子行走的一个可能令人惊讶的应用是布尔公式的快速评估。 关于 N 个二进制输入 x1，...，xN 的布尔公式是一棵树，其内部顶点表示与(^)，或,非（¬）门应用于它们的子顶点，其 N 叶被标记为位 x1，...，xN。两个这样的公式如图 3 所示。有一 种量子算法允许任何这样的公式在略大于 0(N)的情况下计算)操作,众所周知，对于一大类布尔公式，任何随机经典算法都需要 N 阶时间在最坏的情况下。量子算法是基于使用和分析一个量子行走在树图上对应的公式的结构。在显示量子加速的公式计算问题中，一个特别有趣的特殊情况是计算与非树，它对应于在某些双人游戏中决定胜者。量子行走也可以用来获得一个非常普遍的加速比传统的基于马尔可夫链的算法。

离散时间马尔科夫链是由其转移矩阵 P 定义 的随机线性映射，其中 P 为状态 x 转换到状态 y 的概率。许多经典的搜索算法可以表示为模拟一个马尔可夫链，进行一定的步骤, 并检查是否转换到我们正在搜索的“标记”元素。决定该经 典算法效率的一个关键参数是马尔可夫链的谱间隙分布(即马尔可夫链的谱间隙分布)。P 的最大特征值和第二大特征值之间的差)。

基于量子游走的类似算法,该算法通过改进对 δ 值空间的依赖，从 $1/\delta$ 到 $1/\sqrt{\delta}$ 。这个框架被用来获得各种问题的量子加速，从确定整数列表是否都是不同的到在图中找到三角形。

9 解线性方程组及相关任务

数学、工程学和许多科学领域的一项基本任务是求解线性方程组。我们有一个 N×N 矩阵 A，和向量 b 属于 R^N，并被要求输出 x，使 Ax=b。 该问题可以用高斯消去等线性代数方法在 N 中的时间多项式求解。 我们能做得更好吗？ 这似乎很困难，因为即使写下答案 x 也需要 N 阶时间。Harrow、Hassidim 和 Lloyd(HHL)的量子算法用于求解系统线性方程的 Tems 通过在一种特殊的量子意义上“求解”方程来回避这一 问 题：给 定 了 创 建 量 子 态 $|b\rangle = \sum_{i=1}^{N} b_i|i\rangle$ 的能力，以及对 A 的访问，算法输出一个与近似成正比的状态 $|x\rangle = \sum_{i=1}^{N} x_i|i\rangle$。这是一个 N 维量子态，可以存储在 O(logN)量子位元中。

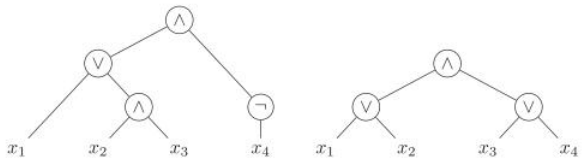

图 3.两个布尔公式在 4 位上。对于 x1=1，x2=x3，x4=0，例如，第一个公式计算为 1，第二个公式计算为 0。 第二个公式是 AND-OR 树。

假设矩阵 A 满足一定的约束条件，算法可以有效地运行。首先，它应该 sparse- each 行应该包含最多 d 元素,对于一些 d<<n 我们应该得到一个通过一个函数 我们应该被允许访问到 A 通过一个函数，我们可以传递一个行号 r 和一个索引 i，其中 1⩽ i⩽ d，并返回 r 行中的 i 非零元素。此外，条件数 $K = \| A^{-1} \| \| A \|$ 一个参数测量 A 的数值不稳定性，应该很小.假设这些约束条件，|X> 可以在 logN，d 和 k 中近似产生时间多项式。如果 d 和 κ

很小，那么这个 i 这是标准经典算法的指数改进。事实上，人们甚至可以证明，经典地实现类似的运行时意味着经典计算机可以有效地模拟一个多项式时间量子计算。

当然，该算法不会输出整个 x,而是生成一个 n 维量子态|X>;要输出解 x 本身，就需要进行许多测量完全描述状态，一般需要 N 阶的时间。但是，我们可能对解决方案的整体不感兴趣，而是对它的某些全局属性感兴趣。这些属性可以通过在 $|x\rangle$ 上执行度来确定。例如，HHL 算法允许人们有效地确定两组线性方程是否具有相同的解，以及许多其他简单的全局属性。

HHl 算法很可能在矩阵 A 和向量是算法生成的环境中找到应用，而不是显式地写下来。工程中的有限元法(EM)就是这样一种设置。Clader. Jcooroo 近的研究表明,HHl 算法在与预处理相结合的情况下，可以通过 FEM 来解决电磁散射问题。同样的算法， 或紧密相关的思想，也可以应用于线性方程之外的问题。这些包括解决大系统的微分方程，数据匹配，以及机器学习中的各种任务。需要强调的是，在所有这些情况下，量子算法解决“这些问题，就像 HL 算法解决它们一样它从一个子态开始， 护生一个最子态作为输出。这是否是“解决方案”的合理定义取决于应用程序，同样也可能取决于输入是通过算法产生的还是作为任意数据显式提供的。

10 少慢子位的应用和实验实现

虽然在实验量子计算方面取得了迅速的进展，但在我们拥有一台大规模的通用量子计算机之前，还有一些路要走，目前的实现方式只有几个最子位元。任何在标准量子电路模型中最多工作 20-30 个量子比特的量子计算都可以很容易地在现代经典计算机上进行模拟。 因此，存在量子算法的实现通常应该被看作是原理的证明，而不是在经典的艺术状态上展示真正的加速。 在表 3 中，这里我们突出了一些实验讨论的算法的实现，重点是迄今为止考虑的最大问题大小(尽管注意，当使用“问题大小”作为“困难”的代理时，必须小心在量子计算机上求解)。

本表省略的一个重要算法是量子仿真。 这一主题从量子计算的早期就被研究过(也许有第一次实现可以追溯到 199967 年，量子模拟现在已经在某种形式上实现了，基本上是在量子计算的每个技术平台上。 一个突出的例子是利用 6 位离子阱系统现通用数字量子模拟:我们听从调查报告以作进一步参考。从测量可控子系统的性质的意义上说，子模拟已经完成，超出了当前经典模拟技术的范围,这是有争议的。

量子化学是数字量子模拟的一个应用领域，也是目前研究的重点。经典的分 子模拟技术目前仅限于 50-70 自旋轨道的分子。由于每个自旋轨道对应于子模拟算法中的一个量子位元，一台只有 100 个逻辑量子位元的量子计算机就可以执行经典计算所无法执行的计算。在这种情况下的挑战是优化模拟时间，虽然轨道的数量是多项式，但一开始看起来很长，但是通过详细的分析得到了迅速的改进。

在不久的将来，量子算法的表现优于经典计算,这自然是相当有兴趣的。玻色子采样问题就是为了解决这个问题而设计的。玻色子抽样概率分布的抽样问题得到喂养光子通过线性光网络在 m 模式其中 m>n。这个任务被推测是古典电脑很难解决。然而，利用线性光学可以很容易地进行玻色子的采样，事实上已经进行了几个小规模的实验演示。虽然玻色子取样最初并没有考虑到实际应用，但随后的工作探索了分子振动和振动谱之间的联系。

量子算法可以被有效地应用于甚至非常小规模的系统的一种方式是“量子算法思维:将量子算法的设计思想应用于物理问题。子计量领域的一个例子是基于子相位估计算法的高精度量子测量方案的发展。

11 ZERO_QUBIT 应用程序

我们最后提到一些方式，其中量子计算是有用的现在，而不需要一个实际的大规模量子计算机。这些可以概括为从子计算理论到其他科学和数学领域的想法的应用。

首先，哈密顿复杂度领域旨在描述量子力学系统中计算感兴趣量的复杂性。一个典型的例子，也是量子化学和凝聚态物理的基本任务，是近似计算由局部哈密顿函数描述的物理系统的基态能量的问题。现在我们知道，这个问题-----以及其他许多问题------是量子梅林亚瑟(QMA)-------完全解决的。 QMA 类中的问题是那些可以被量子计算机有效地解决的，给予量

子“证书”的访问权。我们想象这个证书是由全能(但不可信)的巫师梅林颁发的，然后交给多项式时间的人类亚瑟来验证;因此量子 Merlin Arthur。经典地说，如果一个问题被证明是 np 完备的，那么这就被认为是没有有效算法来解决它的很好的证据。类似地，qma 完全问题被认为不太可能有有效的量子(或经典)算法。人们甚至可以更进一步，尝试描述出，对于哪些物理系统，计算基态能量是困难的，而对于哪些物理系统，这个问题是容易的。虽然这个计划还没有完成，但它已经提供了一些正式的理由，以经验观察冷凝物质物理学有关这些问题的相对硬度。

其次，使用量子信息模型作为数学工具可以提供对纯古典性质的其他问题的洞察。例如，基于子信息理论原理，可以得到经典内积函数通信复杂度的强下界。来自量子计算的想法也被用来证明经典数据结构、代码和公式的新限制。

12 前景

我们已经描述了相当多的量子算法，解决相当多的问题。然而，人们可能仍然会问，为什么还有更多的算法不为人所知-----特别是，更多的指数速度?

一个原因是，在查询复杂度模型中，量子计算的能力已经证明了强下界，其中只考虑输入的查询数量作为复杂度的度量。例如，在保持相同的成功概率的情况下，即使是一个查询也不能提高 Grover 算法所实现的复杂性。更普遍的是，为了在查询复杂性模型中实现比经典计算的指数加速，必须对输入做出承诺，即必须禁止某些可能的输入。这是量子算法在密码学领域取得成功的原因之一.量子计算机可以利用传统计算机无法利用的隐藏问题结构。

在其他具有实际意义的问题中找到这种隐性结构仍然是一个重要的开放性问题。

此外，激进的读者可能会指出，已知的量子算法大多是基于少的量子基元(此如量子傅里叶变换和量子行走).van Dam 的观察结果（见 http://dabacon.org/pontiff/？p=1291）提供了一些理由。众所周知，任何量子电路都可以用 Toffoli 和 Hadamard 量子门来近似。其中第一个是纯经典门，第二个等效于群$\mathbb{Z}$ 2 上的傅里叶变换。因此，任何量子算法都可以用量子傅立叶变换与经典处理相结合来表达。然而，上面描述的量子算法的背后要比这个结合结果所暗示的要复杂得多。其他量子算法，在这里没有讨论，包括拓扑量子场论；量子电路和自旋模型之间的连接；Elitzur‐Vaidman 量子炸弹测试仪和直接解决半定规划问题的量子查询复杂性,以及新的量子算法。未来研究的一个重要方向是将已知的量子算法（和算法原语）应用于新的问题领域。这很可能需要来自其他领域的从业者的大量投入和交流。

13 致谢

这项工作得到了英国 EPSRC 早期职业奖学金 EP/L021005/1 的支持。

14 利益竞争

作者声明没有利益冲突。

npj | Quantum Information

www.nature.com/npjqi

REVIEW ARTICLE OPEN

Quantum algorithms: an overview

Ashley Montanaro[1]

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.

npj Quantum Information (2016) **2,** 15023; doi:10.1038/npjqi.2015.23; published online 12 January 2016

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 that 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 262 papers on quantum algorithms.[1] There are now a number of excellent surveys about quantum algorithms,[2–5] 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[6] (in theory or in practice) and quantum error-correction,[7] nor quantum communication complexity[8] or quantum Shannon theory.[9]

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 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,[10] 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, then 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.

THE HIDDEN SUBGROUP PROBLEM AND APPLICATIONS TO CRYPTOGRAPHY

One of the first applications of quantum computers discovered was Shor's algorithm for integer factorisation.[11] 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 $\exp(O(\log N)^{1/3}(\log\log N)^{2/3}))$[12] (in fact, this is a heuristic bound; the best rigorous bound is somewhat higher), 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[13] 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.*[14] reported classical factorisation of a 768-bit number, using hundreds of modern computers over a period of 2 years, with a total computational effort of $\sim 10^{20}$ operations. A detailed analysis of one fault-tolerant quantum computing architecture,[7] making reasonable assumptions about the underlying hardware, suggests that a 2,000-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

[1]School of Mathematics, University of Bristol, Bristol, UK.
Correspondence: A Montanaro (ashley.montanaro@bristol.ac.uk)
Received 18 August 2015; revised 2 October 2015; accepted 30 October 2015

Table 1. Some computational complexity classes of importance in quantum computation

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

Abbreviation: QMA, Quantum Merlin–Arthur.
'Polynomial time' is short for 'in time polynomial in the input size'.

day at a clock rate of 10 MHz. 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),[15,16] then giving an efficient quantum algorithm for this problem. The HSP is parametrised by a group G, and 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;[17] an efficient quantum algorithm for the latter case would give an efficient test for isomorphism of graphs (equivalence under relabelling of vertices).

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 $f:\{0,1\}^n \to \{0,1\}$, find x such that $f(x)=1$, if such an x exists; otherwise, output 'not found'.

It is easy to see that, with no prior information about f, 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, then the number of evaluations required is of order N. However, remarkably, there is a quantum algorithm due to Grover,[18] which solves this problem using $O(\sqrt{N})$ evaluations of f in the worst case (Grover's original algorithm solved the special case where the solution is unique; the extension to multiple solutions came slightly later.[19]). 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 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 that 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 marked 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, then 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.[20] 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.[21] 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 poly(n) comes from the time taken to evaluate the circuit on a given input.

Amplitude amplification

Grover's algorithm speeds up the naive 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

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

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, Høyer, Mosca and Tapp[22] can find w such that $f(w)=1$ with only $O(1/\sqrt{\varepsilon})$ 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

Table 2. Some problems which can be expressed as hidden subgroup problems

Problem	*Group*	*Complexity*	*Cryptosystem*
Factorisation	$\mathbb{Z}$	Polynomial[11]	RSA
Discrete log	$\mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1}$	Polynomial[11]	Diffie-Hellman, DSA,…
Elliptic curve discrete log	Elliptic curve	Polynomial[92]	ECDH, ECDSA,…
Principal ideal	$\mathbb{R}$	Polynomial[93]	Buchmann-Williams
Shortest lattice vector	Dihedral group	Subexponential[94,95]	NTRU, Ajtai-Dwork,…
Graph isomorphism	Symmetric group	Exponential	—

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.

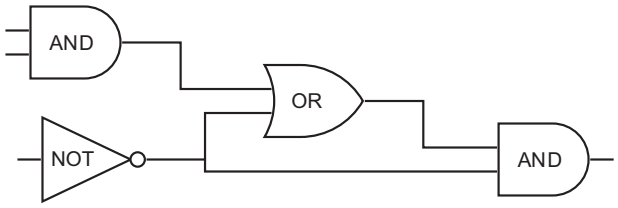

Figure 1. An instance of the Circuit SAT problem. The answer should be 'yes' as there exists an input to the circuit such that the output is 1.

$w \in \{0, 1\}^n$ 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 $O(\sqrt{N/k})$ queries to 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 NP-complete constraint satisfaction problem 3-SAT is randomised and runs in time $O((4/3)^n \text{poly}(n))$.[23] 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 speedup 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[24] gave the first quantum algorithm for a combinatorial task (simultaneously satisfying many linear equations of a certain form) which outperformed the best efficient 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,[25] leaving the question open of whether quantum algorithms for optimisation problems can substantially outperform the accuracy of their classical counterparts.

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:$\{0,1\}^n \to \mathbb{Z}$). A quantum algorithm due to Dürr and Høyer[26] 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:$\{0, 1\}^n \to \{0, 1\}$ 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, Heiligman, Høyer and Mhalla[27] 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).
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[28] 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 $O(N/M + \sqrt{N})$.[29] This is a super-polynomial speedup when M is large.

Adiabatic optimisation

An alternative approach to quantum combinatorial optimisation is provided by the quantum adiabatic algorithm.[30] 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 that is the uniform superposition over all possible solutions to the CSP. This is the ground (lowest energy) state of a Hamiltonian that 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,[31] this rapidly becomes infeasible for larger problems. One can construct problem instances on which the standard adiabatic algorithm provably takes exponential time;[32,33] however, changing the algorithm can evade some of these arguments.[34,35]

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, which has built large machines designed to implement this algorithm,[36] with the most recent such machine ('D-Wave 2X') announced as having up to 1,152 qubits. For certain instances of CSPs, these machines have been demonstrated to outperform classical solvers running on a standard computer,[37,38] although the speedup (or otherwise) seems to have a rather subtle dependence on the problem instance, classical solver compared, and measure of comparison.[38,39]

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.

QUANTUM SIMULATION

In the early days of classical computing, one of the main applications of computer technology was the simulation of physical systems (such applications arguably go back at least as far as the Antikythera mechanism from the 2nd century BC.). Similarly, the most important early application of quantum computers is likely to be the simulation of quantum systems.[40–42] 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 has 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.[43]

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.[44] 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;[45] optimising the algorithm for particular applications such as quantum chemistry;[46] and exploring applications to new areas such as quantum field theory.[47]

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, then we build another system that 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.[40]

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.[48,49] The separation between quantum and

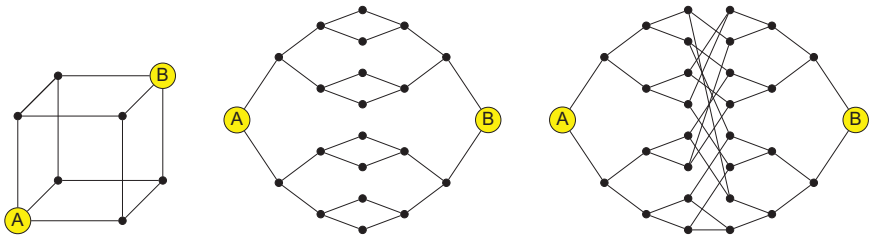

Figure 2. Three graphs for whose natural generalisations to N vertices a classical random walk requires exponentially more time than a quantum walk to reach the exit (B) from the entrance (A). However, on the first two graphs there exist efficient classical algorithms to find the exit which are not based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

classical mixing time can be quadratic, but no more than this[50] (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 that 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/poly(log N). However, any classical algorithm requires time of order $N^{1/6}$ to find the exit.[51] 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 $x_1,\ldots, x_N$ 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 $x_1,\ldots, x_N$. 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,[52] while it is known that for a wide class of boolean formulae, any randomised classical algorithm requires time of order $N^{0.753\cdots}$ in the worst case.[53] 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 that 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 $1/\sqrt{\delta}$.[54–56] This framework has been used to obtain quantum speedups for a variety of problems,[4] ranging from determining whether a list of integers are all distinct[54] to finding triangles in graphs.[57]

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 $\mathbf{A}$, and a vector $\mathbf{b}\in\mathbb{R}^N$, and are asked to output x such that $\mathbf{A}x=\mathbf{b}$. This problem can be solved in time polynomial in N by straightfoward 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[58] (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 $\mathbf{A}$, 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.

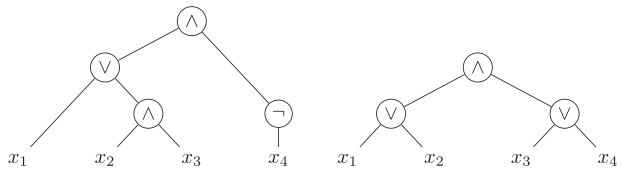

Figure 3. Two boolean formulae on 4 bits. For $x_1=1$, $x_2=x_3=x_4=0$, for example, the first formula evaluates to 1 and the second to 0. The second formula is an AND–OR tree.

The algorithm runs efficiently, assuming that the matrix $\mathbf{A}$ 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 $1\leqslant i\leqslant d$, 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.[58,59] If d and κ are small, then 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.[58]

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,[59] as well as many other simple global properties.[60]

The HHL algorithm is likely to find applications in settings where the matrix $\mathbf{A}$ and the vector $\mathbf{b}$ 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.[60] 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,[61,62] data fitting[63] and various tasks in machine learning.[64] 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.[65]

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

Table 3. Some proof-of-concept experimental implementations of quantum algorithms

Algorithm	*Technology*	*Problem solved*
Shor's algorithm	Bulk optics[96]	Factorisation of 21
Grover's algorithm	NMR[97]	Unstructured search, $N = 8$
Quantum annealing	D-Wave 2X[38]	Ising model on a 'Chimera' graph with 1097 vertices
HHL algorithm	Bulk optics,[98,99] NMR[100]	2×2 system of linear equations

Abbreviations: HHL, Harrow, Hassidim and Lloyd; NMR, nuclear magnetic resonance.
Table only includes some 'largest' problem instances solved thus far.

implementations of algorithms discussed here, focusing on the largest problem sizes considered thus far (although note that one has to be careful when using 'problem size' as a proxy for 'difficulty in solving on a quantum computer'.[66]).

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[67], 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[68] to implement general digital quantum simulation; we defer to survey papers[40,42,69,70] 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 that are beyond the reach of current classical simulation techniques.[71]

One application of digital quantum simulation which is currently the object of intensive study is quantum chemistry.[46,72,73] Classical techniques for molecular simulation are currently limited to molecules with 50–70 spin orbitals.[72] As each spin orbital corresponds to a qubit in 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,[73] but was rapidly improved via detailed analysis.[72]

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.[74] 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.[74] 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.[75] Although Boson Sampling was not originally designed with practical applications in mind, subsequent work has explored connections to molecular vibrations and vibronic spectra.[76,77]

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.[78]

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 Quantum Merlin–Arthur (QMA)-complete.[79,80] Problems in the class QMA are those which can be efficiently solved by a quantum computer given access to a quantum 'certificate'. We 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. Classically, if a problem is proven NP-complete, then this is considered as 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-state energies is hard, and for which the problem is easy.[29,81] 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.[82] Ideas from quantum computing have also been used to prove new limitations on classical data structures, codes and formulae.[83]

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.[84] 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.[85] This is one reason behind the success of quantum algorithms in cryptography: the existence of hidden problem structure that 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 Dam (see http://dabacon.org/pontiff/?p=1291) provides some justification for this. It is known that any quantum circuit can be approximated using only Toffoli and Hadamard quantum gates.[86] 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;[87] connections between quantum circuits and spin models;[88] the Elitzur–Vaidman quantum bomb tester;[89] and directly solving the semidefinite programming problem characterising quantum query complexity.[90,91]

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.

ACKNOWLEDGEMENTS

This work was supported by the UK EPSRC under Early Career Fellowship EP/L021005/1.

COMPETING INTERESTS

The author declares no conflict of interest.

REFERENCES

1. Jordan, S. The quantum algorithm zoo. Available at http://math.nist.gov/quantum/zoo/.
2. Childs, A. & van Dam, W. Quantum algorithms for algebraic problems. *Rev. Mod. Phys.* **82**, 1–52 (2010).
3. Mosca, M. in *Computational Complexity* 2303–2333 (Springer, 2012).
4. Santha, M. Quantum walk based search algorithms. in *Theory Appl. Model. Comput.* **4978**, 31–46 (2008).
5. Bacon, D. & van Dam, W. Recent progress in quantum algorithms. *Commun. ACM* **53**, 84–93 (2010).
6. Ladd, T. *et al.* Quantum computing. *Nature* **464**, 45–53 (2010).
7. Fowler, A., Mariantoni, M., Martinis, J. & Cleland, A. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A* **86**, 032324 (2012).
8. Buhrman, H., Cleve, R., Massar, S. & de Wolf, R. Non-locality and communication complexity. *Rev. Mod. Phys.* **82**, 665–698 (2010).
9. Wilde, M. *Quantum Information Theory* (Cambridge Univ. Press, 2013).
10. Papadimitriou, C. *Computational Complexity* (Addison-Wesley, 1994).
11. Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **26**, 1484–1509 (1997).
12. Buhler, J. Jr., H. W. L. & Pomerance, C. Factoring integers with the number field sieve in *The Development Of The Number Field Sieve* Vol. 1554, 50–94 (Springer, 1993).
13. Rivest, R., Shamir, A. & Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21**, 120–126 (1978).
14. Kleinjung, T. *et al.* in *Advances in Cryptology—CRYPTO 2010* Vol. 1554, 333–350 (2010).
15. Boneh, D. & Lipton, R. in *Advances in Cryptology—CRYPT0' 95*, 424–437 (1995).
16. Brassard, G. & Høyer, P. in *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems* 12–23 (Ramat Gan, Israel, 1997).
17. Regev, O. Quantum computation and lattice problems. *SIAM J. Comput.* **33**, 738–760 (2004).
18. Grover, L. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.* **79**, 325–328 (1997).
19. Boyer, M., Brassard, G., Høyer, P. & Tapp, A. Tight bounds on quantum searching. *Fortschr. Phys.* **46**, 493–505 (1998).
20. Prasad, M., Biere, A. & Gupta, A. A survey of recent advances in SAT-based formal verification. *Int. J. Softw. Tool. Technol. Transf.* **7**, 156–173 (2005).
21. Williams, R. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.* **42**, 1218–1244 (2013).
22. Brassard, G., Høyer, P., Mosca, M. & Tapp, A. Quantum amplitude amplification and estimation. in *Quantum Computation and Information* Vol. 305 of AMS Contemporary Mathematics Series (eds Lomonaco, S. J. & Brandt, H. E.), 53–74 (2002).
23. Schöning, U. in *Proceedings of 40th Annual Symposium on Foundations of Computer Science*, 410–414 (Washington DC, USA, 1999).
24. Farhi, E., Goldstone, J. & Gutmann, S. A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem. Preprint at arXiv:1412.6062 (2014).
25. Barak, B. *et al.* Beating the random assignment on constraint satisfaction problems of bounded degree. Preprint at arXiv:1505.03424 (2015).
26. Dürr, C. & Høyer, P. A quantum algorithm for finding the minimum. Preprint at quant-ph/9607014 (1996).
27. Dürr, C., Heiligman, M., Høyer, P. & Mhalla, M. in *Proceedings of 31st International Conference on Automata, Languages and Programming (ICALP'04)*, 481–493 (Turku, Finland, 2004).
28. Ramesh, H. & Vinay, V. String matching in $\tilde{O}(\sqrt{n}+\sqrt{m})$ quantum time. *J. Discrete Algorithms* **1**, 103–110 (2003).
29. Montanaro, A. Quantum pattern matching fast on average. *Algorithmica* 1–24 (2015).
30. Farhi, E., Goldstone, J., Gutmann, S. & Sipser, M. Quantum computation by adiabatic evolution. *Tech. Rep.*, MIT-CTP-2936, MIT (2000).
31. Farhi, E. *et al.* A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science* **292**, 472–475 (2001).
32. van Dam, W., Mosca, M. & Vazirani, U. in *Proceedings of 42nd Annual Symposium on Foundations of Computer Science*, 279–287 (IEEE, 2001).
33. Farhi, E., Goldstone, J., Gutmann, S. & Nagaj, D. How to make the quantum adiabatic algorithm fail. *Int. J. Quantum Inform.* **6**, 503 (2008).
34. Farhi, E. *et al.* Quantum adiabatic algorithms, small gaps, and different paths. *Quantum Inf. Comput.* **11**, 0181–0214 (2011).
35. Choi, V. Different adiabatic quantum optimization algorithms for the NP-complete exact cover and 3SAT problems. *Quantum Inf. Comput.* **11**, 0638–0648 (2011).
36. Johnson, M. *et al.* Quantum annealing with manufactured spins. *Nature* **473**, 194–198 (2011).
37. McGeoch, C. & Wang, C. in *Proceedings of ACM International Conference on Computing Frontiers (CF'13)*, 1–23 (Ischia, Italy, 2013).
38. King, J., Yarkoni, S., Nevisi, M., Hilton, J. & McGeoch, C. Benchmarking a quantum annealing processor with the time-to-target metric. Preprint at arXiv:1508.05087 (2015).
39. Rønnow, T. *et al.* Defining and detecting quantum speedup. *Science* **345**, 420–424 (2014).
40. Bulata, I. & Nori, F. Quantum simulators. *Science* **326**, 108–111 (2009).
41. Brown, K., Munro, W. & Kendon, V. Using quantum computers for quantum simulation. *Entropy* **12**, 2268–2307 (2010).
42. Georgescu, I., Ashhab, S. & Nori, F. Quantum simulation. *Rev. Mod. Phys.* **86**, 153 (2014).
43. Feynman, R. Simulating physics with computers. *Int. J. Theor. Phys.* **21**, 467–488 (1982).
44. Lloyd, S. Universal quantum simulators. *Science* **273**, 1073–1078 (1996).
45. Berry, D., Childs, A. & Kothari, R. Hamiltonian simulation with nearly optimal dependence on all parameters. Preprint at arXiv:1501.01715 (2015).
46. Hastings, M., Wecker, D., Bauer, B. & Troyer, M. Improving quantum algorithms for quantum chemistry. *Quantum Inf. Comput.* **15**, 1–21 (2015).
47. Jordan, S., Lee, K. & Preskill, J. Quantum algorithms for quantum field theories. *Science* **336**, 1130–1133 (2012).
48. Childs, A., Farhi, E. & Gutmann, S. An example of the difference between quantum and classical random walks. *Quantum Inform. Process.* **1**, 35–43 (2002).
49. Kempe, J. Quantum random walks hit exponentially faster. *Probab. Theory Rel. Fields* **133**, 215–235 (2005).
50. Aharonov, D., Ambainis, A., Kempe, J. & Vazirani, U. Quantum walks on graphs. in *Proceedings of 33rd Annual ACM Symposium on Theory of Computing*, 50–59 (Heraklion, Crete, Greece, 2001).
51. Childs, A. *et al.* Exponential algorithmic speedup by a quantum walk. in *Proceedings of 35th Annual ACM Symposium on Theory of Computing*, 59–68 (San Diego, CA, USA, 2003).
52. Ambainis, A., Childs, A., Reichardt, B., Špalek, R. & Zhang, S. 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**, 2513–2530 (2010).
53. Santha, M. On the Monte Carlo boolean decision tree complexity of read-once formulae. *Random. Struct. Algorithms* **6**, 75–87 (1995).
54. Ambainis, A., Schulman, L. J., Ta-Shma, A., Vazirani, U. & Wigderson, A. The quantum communication complexity of sampling. *SIAM J. Comput.* **32**, 1570–1585 (2003).

55. Szegedy, M. in *Proceedings of 45th Annual Symposium on Foundations of Computer Science*, 32–41 (Rome, Italy, 2004).
56. Magniez, F., Nayak, A., Roland, J. & Santha, M. Search via quantum walk. *SIAM J. Comput.* **40**, 142–164 (2011).
57. Le Gall, F. in *Proceedings of 55th Annual Symposium on Foundations of Computer Science*, 216–225 (Philadelphia, USA, 2014).
58. Harrow, A., Hassidim, A. & Lloyd, S. Quantum algorithm for solving linear systems of equations. *Phys. Rev. Lett.* **15**, 150502 (2009).
59. Ambainis, A. in *Proceedings of 29th Annual Symposium on Theoretical Aspects of Computer Science*, 636–647 (Paris, France, 2012).
60. Clader, B., Jacobs, B. & Sprouse, C. Preconditioned quantum linear system algorithm. *Phys. Rev. Lett.* **110**, 250504 (2013).
61. Leyton, S. & Osborne, T. A quantum algorithm to solve nonlinear differential equations. Preprint at arXiv:0812.4423 (2008).
62. Berry, D. High-order quantum algorithm for solving linear differential equations. *J. Phys. A Math. Gen.* **47**, 105301 (2014).
63. Wiebe, N., Braun, D. & Lloyd, S. Quantum algorithm for data fitting. *Phys. Rev. Lett.* **109**, 050505 (2012).
64. Lloyd, S., Mohseni, M. & Rebentrost, P. Quantum algorithms for supervised and unsupervised machine learning. Preprint at arXiv:1307.0411 (2013) .
65. Aaronson, S. Quantum machine learning algorithms: Read the fine print. *Nat. Phys.* **11**, 291–293 (2015).
66. Smolin, J., Smith, G. & Vargo, A. Oversimplifying quantum factoring. *Nature* **499**, 163–165 (2013).
67. Somaroo, S., Tseng, C., Havel, T., Laflamme, R. & Cory, D. Quantum simulations on a quantum computer. *Phys. Rev. Lett.* **82**, 5381 (1999).
68. Lanyon, B. *et al.* Universal digital quantum simulations with trapped ions. *Science* **334**, 57–61 (2011).
69. Blatt, R. & Roos, C. Quantum simulations with trapped ions. *Nat. Phys.* **8**, 277–284 (2012).
70. Aspuru-Guzik, A. & Walther, P. Photonic quantum simulators. *Nat. Phys.* **8**, 285–291 (2012).
71. Trotzky, S. *et al.* Probing the relaxation towards equilibrium in an isolated strongly correlated one-dimensional Bose gas. *Nat. Phys.* **8**, 325–330 (2012).
72. Poulin, D. *et al.* The Trotter step size required for accurate quantum simulation of quantum chemistry. *Quantum Inf. Comput.* **15**, 361–384 (2014).
73. Wecker, D., Bauer, B., Clark, B., Hastings, M. & Troyer, M. Gate count estimates for performing quantum chemistry on small quantum computers. *Phys. Rev. A* **90**, 022305 (2014).
74. Aaronson, S. & Arkhipov, A. The computational complexity of linear optics. *Theory Comput.* **9**, 143–252 (2013).
75. Ralph, T. Quantum computation: Boson sampling on a chip. *Nat. Photon.* **7**, 514–515 (2013).
76. Huh, J., Guerreschi, G., Peropadre, B., McClean, J. & Aspuru-Guzik, A. Boson sampling for molecular vibronic spectra. *Nat. Photon.* **9**, 615–620 (2015).
77. Martín-Lopéz, E. *et al.* Simulating molecular vibrations with photons (2015). in preparation.
78. Higgins, B., Berry, D., Bartlett, S., Wiseman, H. & Pryde, G. Entanglement-free Heisenberg-limited phase estimation. *Nature* **450**, 396–396 (2007).
79. Kitaev, A. Y., Shen, A. H. & Vyalyi, M. N. *Classical and Quantum Computation* Vol. 47. (AMS, 2002).
80. Bookatz, A. QMA-complete problems. *Quantum Inform. Comput.* **14**, 361–383 (2014).
81. Schuch, N. & Verstraete, F. Computational complexity of interacting electrons and fundamental limitations of Density Functional Theory. *Nat. Phys.* **5**, 732–735 (2009).
82. Cleve, R., van Dam, W., Nielsen, M. & Tapp, A. in *Procedings of the 1st NASA International Conference on Quantum Computing and Quantum Communications*, 61–74 (Palm Springs, CA, USA, 1998).
83. Drucker, A. & de Wolf, R. Quantum proofs for classical theorems. *Theory Comput. Grad. Surv.* **2**, 1–54 (2011).
84. Zalka, C. Grover's quantum searching algorithm is optimal. *Phys. Rev. A* **60**, 2746–2751 (1999).
85. Beals, R., Buhrman, H., Cleve, R., Mosca, M. & de Wolf, R. Quantum lower bounds by polynomials. *J. ACM* **48**, 778–797 (2001).
86. Shi, Y. Both Toffoli and controlled-NOT need little help to do universal quantum computing. *Quantum Inf. Comput.* **3**, 84–92 (2003).
87. Freedman, M., Larsen, M. & Wang, Z. A modular functor which is universal for quantum computation. *Commun. Math. Phys.* **227**, 605–622 (2002).
88. De las Cuevas, G., Dür, W., van den Nest, M. & Martin-Delgado, M. Quantum algorithms for classical lattice models. *New J. Phys.* **13**, 093021 (2011).
89. Lin, C. & Lin, H. in *Proceedings of 30th Annual IEEE Conference on Computational Complexity*, 537-566 (Portland, OR, USA, 2015).
90. Reichardt, B. in *Proceedings of 50th Annual Symposium on Foundations of Computer Science*, 544–551 (Atlanta, GA, USA, 2009).
91. Belovs, A. Quantum algorithms for learning symmetric juntas via adversary bound. in *Proceedings of 29th Annual IEEE Conference on Computational Complexity*, 22–31 (Vancouver, Canada, 2014).
92. Proos, J. & Zalka, C. Shor's discrete logarithm quantum algorithm for elliptic curves. *Quantum Inform. Comput.* **3**, 317–344 (2003).
93. Hallgren, S. Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem. *J. ACM* **54**, 4:1–4:19 (2007).
94. Kuperberg, G. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput.* **35**, 170–188 (2005).
95. Regev, O. A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space. Preprint at quant-ph/0406151 (2004).
96. Martín-López, E. *et al.* Experimental realisation of Shor's quantum factoring algorithm using qubit recycling. *Nat. Photon.* **6**, 773–776 (2012).
97. Vandersypen, L. *et al.* Implementation of a three-quantum-bit search algorithm. *Appl. Phys. Lett.* **76**, 646–648 (2000).
98. Cai, X.-D. *et al.* Experimental quantum computing to solve systems of linear equations. *Phys. Rev. Lett.* **110**, 230501 (2013).
99. Barz, S. *et al.* Solving systems of linear equations on a quantum computer. *Sci. Rep.* **4**, 115 (2014).
100. Pan, J. *et al.* Experimental realization of quantum algorithm for solving linear systems of equations. *Phys. Rev. A* **89**, 022313 (2014).