Moore’s Law

Moore's Law graph showing the progression of transistor counts over time from Intel processors.
The first solid-state transistor
(Bardeen, Brattain & Shockley, 1947)
20th century quantum technology uses extensively the wave nature of particles, but does not make much use of quantum “weirdness,” the phenomena of superposition and entanglement.

Quantum information, an idea that is coming of age only in the 21st century, relies heavily on superposition and entanglement, enriching information science in astounding ways:

A quantum computer (if it could be made) would be more different from today’s digital computers than today’s computers are from the abacus.
Binary digit
“bit”
0, 1

Quantum bit
“qubit”
$|0\rangle$, $|1\rangle$, $|0\rangle + |1\rangle$

Physical realization of qubits $\rightarrow$ any 2 level system

2-level atom: spin-1/2: polarization:
$|g\rangle$, $|e\rangle$
$|\uparrow\rangle$, $|\downarrow\rangle$
$|H\rangle$, $|V\rangle$, $|H\rangle + |V\rangle$

All 2-level systems are created equal, but some are more equal than others!

Quantum communication $\rightarrow$ photons
Quantum storage $\rightarrow$ atoms, spins
Scaleable circuits $\rightarrow$ superconducting systems

“Quantum” phenomena
Superposition
Interference
Wave-particle duality
Intrinsic randomness in measurement
Entanglement
“Entanglement”, and the scaling that results, is the key to the power of quantum computing.

- **Classically**, information is stored in a bit register: a 3-bit register can store one number, from 0 - 7. \[101\]

- **Quantum Mechanically**, a register of 3 entangled qubits can store all of these numbers in superposition:

\[a|000\rangle + b|001\rangle + c|010\rangle + d|011\rangle + e|100\rangle + f|101\rangle + g|110\rangle + h|111\rangle\]

- **Result:**
  
  -- **Classical**: one N-bit number

  -- **Quantum**: \(2^N\) (all possible) N-bit numbers

- **N.B.**: A 300-qubit register can simultaneously store \(2^{300} \sim 10^{90}\) numbers
That's a BIG number

$10^{90} = 1,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,
000,000,000,000,000,000,000,
000,000,000,000,000,000,000,
000,000,000,000,000,000,000$

This is more than the total number of particles in the Universe!

Some important problems benefit from this exponential scaling, enabling solutions of otherwise insoluble problems.
Experimental demonstrations of quantum computation primitives:

- Qubit teleportation
- Quantum error correction
- Deutsch-Josza algorithm
- Grover search algorithm
- Toffoli gate
- Quantum Fourier transform
- Entangled state purification
- Dense coding
- Entanglement assisted detection
- eight-qubit W state
- six-qubit GHZ state
- arbitrary motional state superposition generation
Implementation of the Deutsch–Jozsa algorithm on an ion-trap quantum computer


Table 2 Expected and measured results of the complete Deutsch–Jozsa algorithm

<table>
<thead>
<tr>
<th></th>
<th>Constant</th>
<th>Balanced</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Case 1</td>
<td>Case 2</td>
</tr>
<tr>
<td>Expected $</td>
<td>\langle 1</td>
<td>a \rangle</td>
</tr>
<tr>
<td>Measured $</td>
<td>\langle 1</td>
<td>a \rangle</td>
</tr>
<tr>
<td>Expected $</td>
<td>\langle 1</td>
<td>w \rangle</td>
</tr>
<tr>
<td>Measured $</td>
<td>\langle 1</td>
<td>w \rangle</td>
</tr>
</tbody>
</table>

The numbers in brackets are statistical 1σ uncertainties.
Demonstration of two-qubit algorithms with a superconducting quantum processor


Vol 460 | 9 July 2009 | doi:10.1038/nature08121

Figure 1: Implementation of Deutsch–Jozsa algorithm.

Fidelity to the ideal output state is $F = 0.94, 0.95, 0.92,$ and $0.85$

Table 1 | Summary of algorithmic performance

<table>
<thead>
<tr>
<th>Element</th>
<th>Grover search oracle*</th>
<th>Deutsch–Jozsa function†</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f_{00}$</td>
<td>$f_{01}$</td>
</tr>
<tr>
<td>$\langle 0,0 \rangle$</td>
<td>Ideal</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Measured</td>
<td>0.81(1)</td>
</tr>
<tr>
<td>$\langle 0,1 \rangle$</td>
<td>Ideal</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Measured</td>
<td>0.066(7)</td>
</tr>
<tr>
<td>$\langle 1,0 \rangle$</td>
<td>Ideal</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Measured</td>
<td>0.08(1)</td>
</tr>
<tr>
<td>$\langle 1,1 \rangle$</td>
<td>Ideal</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Measured</td>
<td>0.05(2)</td>
</tr>
</tbody>
</table>

Fidelity of the reconstructed output states of the Grover and Deutsch–Jozsa algorithms to their ideal outputs. These results suggest that, if combined with single-shot readout, the two algorithms executed with this processor would give the correct answer with probability far exceeding the 50% success probability of the best classical algorithms limited to single calls of the oracle* or function.

* Uncertainties are based on 10 repetitions.
† Uncertainties are based on 8 repetitions.
A hard problem: factoring large integers:

For example, it is hard to factor 167,659.

But an elementary school student can easily multiply \( 389 \times 431 = 167,659 \).

This asymmetry in the difficulty of factoring vs. multiplying is the basis of public key encryption, on which everything from on-line transaction security to ensuring diplomatic secrecy depends.
‘Hard’ vs ‘easy’ problems
- judged on how they scale with input size $N = 2^n$
  ($n = \log_2 N$)
$t \sim \text{poly}(n) \rightarrow \text{easy}$
  $+, \times, /, N^3$, etc.

$t \sim \exp(n) \rightarrow \text{hard}$
  factoring

Factoring: find factor in the range $(1, \sqrt{N})$
Simple ‘brute force’ method: try each number $\rightarrow \sqrt{N}$ tries
  $t \sim 2^{n/2} \rightarrow \text{hard}$

Best known classical algorithm:
  $t \sim \exp\{\log N^{1/3} [\log(\log N)]^{2/3}\} \sim e^{N^{1/3}} \rightarrow \text{still hard}$
  (though not proven to be)

Best known quantum algorithm to factor an $n$-bit number:
  $t_{\text{Shor}} \sim 25 \ n^3 + O(n^2)$ operations, $5n + 4$ qubits
Quantum Computing’s “killer app.”

Shor’s algorithm enables one to factor numbers into their prime constituents MUCH faster:

<table>
<thead>
<tr>
<th>RSA digits</th>
<th>PC (1 GHz)</th>
<th>Blue Waters (1OPF)</th>
<th>Quantum Computer (1 GHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>129</td>
<td>4 months</td>
<td>1 sec</td>
<td>10 sec</td>
</tr>
</tbody>
</table>

The difficulty (impossibility) of factoring large numbers (and the ease of creating a large number from its factors) is the basis of public key encryption (which nearly everyone uses for secure transmission today).
Algebraic and Number Theoretic Algorithms

Algorithm: Factoring
Speedup: Superpolynomial

Description: Given an N-bit integer, find the prime factorization. The quantum algorithm of Peter Shor solves this in O*(n^3) time [82,125]. The fastest known classical algorithm for integer factorization is the general number field sieve, which is believed to run in time 2O((n/3)). The best rigorously proven upper bound on the classical complexity of factoring is O(2^(n/3+o(1))) [325]. Shor's factoring algorithm breaks RSA public-key encryption and the closely related quantum algorithms for discrete logarithm break the DSA and ECDSA digital signature schemes and the Diffie-Hellman key-exchange protocol. A quantum algorithm even faster than Shor's for the special case of factoring "semiprimes", which are widely used in cryptography, is given in [271]. There are proposed classical public-key cryptosystems not believed to be broken by quantum algorithms, e.g., [248]. At the core of Shor's factoring algorithm is order finding, which can be reduced to the Abelian hidden subgroup problem, which is solved using the quantum Fourier transform. A number of other problems are known to reduce to integer factorization including the membership problem for matrix groups over fields of odd order [251], and certain diophantine problems relevant to the synthesis of quantum circuits [254].

Algorithm: Discrete-log
Speedup: Superpolynomial

Description: We are given three N-bit numbers a, b, and N, with the promise that b=a^modN for some s. The task is to find s. As shown by Shor [82], this can be achieved on a quantum computer in poly(n) time. The fastest known classical algorithm requires time superpolynomial in n. By similar techniques as those in [82], quantum computers can solve the discrete logarithm problem on elliptic curves, thereby breaking elliptic curve cryptography [109]. The superpolynomial quantum speedup has also been extended to the discrete logarithm problem on semigroups [203, 204]. See also Abelian Hidden Subgroup.

Algorithm: Ellipsoid's Problem

Description: Given a positive nonsquare integer d, Ellipsoid's problem is x^2-dy^2=1. For any such d there are infinitely many pairs of integers (x,y) solving this equation. Let (x1,y1) be the pair that minimizes x+y/√d. If d is an N-bit integer (i.e. 0≤d=2^n−1), (x1,y1) may in general require exponentially many bits to write down. Thus it is in general impossible to find (x1,y1) in polynomial time. Let K=O((x1+y1)d). *K* uniquely identifies (x1,y1). As shown by Hallgren [49], given a N-bit number d, a quantum computer can find x+y/√d in poly(n) time. No polynomial time classical algorithm for this problem is known. Factoring reduces to this problem. This algorithm breaks the Buchman-Williams cryptosystem. See also Abelian hidden subgroup.

Algorithm: Principal Ideal

Description: Given a finite ring R, we are given an N-bit integer d and an invertible ideal I of the ring Z[d]. I is a principal ideal if there exists r in Z(d) such that I=Z[d]r, r can be exponentially large in d. Therefore r cannot in general be written down in polynomial time. However, r is uniquely determined. The task is to determine whether I is principal and if so find r. As shown by Hallgren, this can be done in polynomial time on a quantum computer [49]. A modified quantum algorithm for this problem using fewer qubits was given in [131]. Factoring reduces to solving Pell's equation, which reduces to the principal ideal problem. Thus the principal ideal problem is at least as hard as factoring and therefore is probably not in P. See also Abelian hidden subgroup.

Algorithm: Class Group

Description: Given a finite field F with characteristic p, let o=Fp^r be the ring of integers of the field. As shown by Hallgren [50], and independently by Schmidt and Vollmer [116], for any Q(θ) of fixed degree, a quantum computer can find in polynomial time a set of generators for Oθ given a description of θ. No polynomial time classical algorithms for this problem is known. Hallgren and collaborators subsequently discovered how to achieve polynomial scaling in the degree [213]. The algorithm relies on solving Abelian Hidden subgroup problems over the additive group of real numbers.

Algorithm: Group Class

Description: We are given a finite ring R, a finite field F, and r∈F. We must find x,y∈F such that ax+by=c. As shown in [111], quantum computers can solve this problem in O*(n^3) time whereas the best classical algorithm requires O*(n^8) time. The quantum algorithm of [111] is based on the quantum algorithms for discrete logarithms and searching.

Algorithm: Matrices Elements of Group Representations

Description: All representations of finite groups and compact linear groups can be expressed as unitary matrices given an orthonormal basis. The number field Q(θ) of fixed degree, a quantum computer can find in polynomial time a set of generators for Oθ given a description of θ. No polynomial time classical algorithms for this problem is known. Hallgren and collaborators subsequently discovered how to achieve polynomial scaling in the degree [213]. The algorithm relies on solving Abelian Hidden subgroup problems over the additive group of real numbers.

Algorithm: Gauss Sums

Description: Given a finite field F of characteristic p, let o=Fp^r be the ring of integers of the field. As shown by Hallgren [50], and independently by Schmidt and Vollmer [116], for any Q(θ) of fixed degree, a quantum computer can find in polynomial time a set of generators for Oθ given a description of θ. No polynomial time classical algorithms for this problem is known. Hallgren and collaborators subsequently discovered how to achieve polynomial scaling in the degree [213]. The algorithm relies on solving Abelian Hidden subgroup problems over the additive group of real numbers.

Algorithm: Solving Exponential Congruences

Description: We are given a,b,c∈F, a≠0. We must find x,y∈F such that ax+by=c. As shown in [111], quantum computers can solve this problem in O*(n^3) time whereas the best classical algorithm requires O*(n^8) time. The quantum algorithm of [111] is based on the quantum algorithms for discrete logarithms and searching.

Algorithm: Matrix Elements of Group Representations

Description: We are given a finite ring R, a finite field F, and r∈F. We must find x,y∈F such that ax+by=c. As shown in [111], quantum computers can solve this problem in O*(n^3) time whereas the best classical algorithm requires O*(n^8) time. The quantum algorithm of [111] is based on the quantum algorithms for discrete logarithms and searching.

Algorithm: Matrices Product Modules

Description: We are given a,b,c∈F, a≠0. We must find x,y∈F such that ax+by=c. As shown in [111], quantum computers can solve this problem in O*(n^3) time whereas the best classical algorithm requires O*(n^8) time. The quantum algorithm of [111] is based on the quantum algorithms for discrete logarithms and searching.

Algorithm: Simplified Polynomial

Description: Given a list of integers x1,…,xn, and a target integer s, the subset-sum problem is to determine whether the sum of any subset of the given integers adds up to s. This problem is NP-complete, and therefore is unlikely to be solvable by classical or quantum algorithms with polynomial worst-case complexity. In the hard instances the given integers are of order 2n. In [171], a quantum algorithm is given that solves this problem in time 2O(n/4), up to polynomial factors. This quantum algorithm works by applying a variant of Ambainis's quantum walk algorithm for element distinctness [17] speed up a classical algorithm for this problem due to Howgrave-Graham and Joux. The fastest known classical algorithm for subset-sum runs in time 2O(n1/4), up to polynomial factors.

pea: 3
Algorithm: Decoding

Description: Classical error correcting codes allow the detection and correction of bit-flips by storing data redundantly. Maximum-likelihood decoding for arbitrary linear codes is NP-complete in the worst case, but for structured codes or bounded error efficient decoding algorithms are known. Quantum algorithms have been formulated to speed up the decoding of convolutional codes [238] and simplex codes [239].
Algorithm: Constraint Satisfaction
Speedup: Polynomial
Description: Constraint satisfaction problems, many of which are NP-hard, are ubiquitous in computer science, a canonical example being 3-SAT. If one wishes to satisfy as many constraints as possible rather than all of them, these become combinatorial optimization problems. (See also entry on adiabatic algorithms.) The brute force solution to constraint satisfaction problems can be quadratically sped up using Grover's algorithm. However, most constant satisfaction problems are solvable by classical algorithms that (although still exponential-time) run more than quadratically faster than brute force checking of all possible solutions. Nevertheless, a polynomial quantum speedup over the fastest known classical algorithm for 3-SAT is given in [133], and polynomial quantum speedups for some other constraint satisfaction problems are given in [134]. A commonly used classical algorithm for constraint satisfaction is backtracking, and for some problems this algorithm is the fastest known. A general quantum speedup for backtracking algorithms is given in [264].

Oracular Algorithms
Algorithm: Searching
Speedup: Polynomial
Description: We are given an oracle with N allowed inputs. For one input w ("the winner") the corresponding output is 1, and for all other inputs the corresponding output is 0. The task is to find w. On a classical computer this requires Ω(N) queries. The quantum algorithm of Lov Grover achieves this using O(N−1/2) queries [48], which is optimal [216]. This has algorithm has subsequently been generalized to search in the presence of multiple "winners" [15], evaluate the sum of an arbitrary function [15,16,73], find the global minimum of an arbitrary function [35,75, 255], take advantage of alternative initial states [100] or nonuniform probabilistic priors [123], work with oracles whose runtime varies between inputs [138], approximate definite integrals [77], and converge to a fixed-point [208, 209]. The generalization of Grover's algorithm known as amplitude estimation [17] is now an important primitive in quantum algorithm design. Amplitude estimation forms the core of most known quantum algorithms related to collision finding and graph properties. One of the natural applications for Grover search is speeding up the solution to NP-complete problems such as 3-SAT. Doing so is nontrivial, because the best classical algorithm for 3-SAT is not quite a brute force search. Nevertheless, amplitude amplification enables a quadratic quantum speedup over the best classical 3-SAT algorithm, as shown in [133]. Quadratic speedups for other constraint satisfaction problems are obtained in [134]. For further examples of application of Grover search and amplitude amplification see [261, 262]. A problem closely related to, but harder than, Grover search, is spatial search, in which database queries are limited by some graph structure. On sufficiently well-connected graphs, O(√N) quantum query complexity is still achievable [274,275].

Algorithm: Abelian Hidden Subgroup
Speedup: Superpolynomial
Description: Let G be a finitely generated Abelian group, and let H be some subgroup of G such that G/H is finite. Let f be a function on G such that for any g1,g2∈G, f(g1)=f(g2) if and only if g1 and g2 are in the same coset of H. The task is to find H (i.e. find a set of generators for H) by making queries to f. This is solvable on a quantum computer using O(log|G|) queries, whereas classically Ω(|G|) are required. This algorithm was first formulated in full generality by Boneh and Lipton in [14]. However, proper attribution of this algorithm is difficult because, as described in chapter 5 of [76], it subsumes many historically important quantum algorithms as special cases, including Simon's algorithm [108], which was the inspiration for Shor's period finding algorithm, which forms the core of his factoring and discrete-log algorithms. The Abelian hidden subgroup algorithm is also at the core of the Pell's equation, principal ideal, unit group, and class group algorithms. In certain instances, the Abelian hidden subgroup problem can be solved using a single query rather than order log|G|, as shown in [30].

Algorithm: Non-Abelian Hidden Subgroup
Speedup: Superpolynomial
Description: Let G be a finitely generated group, and let H be some subgroup of G that has finitely many left cosets. Let f be a function on G such that for any g1,g2∈G, f(g1)=f(g2) if and only if g1 and g2 are in the same left coset of H. The task is to find H (i.e. find a set of generators for H) by making queries to f. This is solvable on a quantum computer using O(log|G|) queries, whereas classically Ω(|G|) are required [37,51]. However, this does not qualify as an efficient quantum algorithm because in general, it may take exponential time to process the quantum states obtained from these queries. Efficient quantum algorithms for the hidden subgroup problem are known for certain specific non-Abelian groups [81,55,72,53,9,22,56,71,57,43,44,28,126,207,273]. A slightly outdated survey is given in [69]. Of particular interest are the symmetric group and the dihedral group. A solution for the symmetric group would solve graph isomorphism. A solution for the dihedral group would solve certain lattice problems [78]. Despite much effort, no polynomial-time solution for these groups is known. However, Kuperberg [66] found a time 2O(logN√) algorithm for finding a hidden subgroup of the dihedral group DN. Regev subsequently improved this algorithm so that it uses not only subexponential time but also polynomial space [79]. A further improvement in the asymptotic scaling of the required number of qubits is obtained in [218].

Algorithm: Bernstein-Vazirani
Speedup: Polynomial Directly, Superpolynomial Recursively
Description: We are given an oracle whose input is n bits and whose output is one bit. Given input x ∈ {0,1}n, the output is x·h, where h is the "hidden" string of n bits, and · denotes the bitwise inner product modulo 2. The task is to find h. On a classical computer this requires n queries. As shown by Bernstein and Vazirani [11], this can be achieved on a quantum computer using a single query. Furthermore, one can construct recursive versions of this problem, called recursive Fourier sampling, such that quantum computers require exponentially fewer queries than classical computers [11]. See [256, 257] for related work on the ubiquity of quantum speedups from generic quantum circuits and [258, 270] for related work on a quantum query speedup for detecting correlations between the an oracle function and the Fourier transform of another.

Algorithm: Deutsch-Jozsa
Speedup: Exponential over P, none over BPP
Description: We are given an oracle whose input is n bits and whose output is one bit. We are promised that out of the 2n possible inputs, either all of them, none of them, or half of them yield output 1. The task is to distinguish the balanced case (half of all inputs yield output 1) from the constant case (all or none of the inputs yield output 1). It was shown by Deutsch [32] that for n=1, this can be solved on a quantum computer using one query, whereas any deterministic classical algorithm requires two. This was historically the first well-defined quantum algorithm achieving a speedup over classical computation. A related, more recent, pedagogical example is given in [259]. A single-query quantum algorithm for arbitrary n was developed by Deutsch and Jozsa in [33]. Although probabilistically easy to solve with O(1) queries, the Deutsch-Jozsa problem has exponential worst case deterministic query complexity classically.
Algorithm: Formula Evaluation

es, and G is the number of gates in the formula. References [164], [165], and [269] consider special cases of the NAND tree problem in which the number of NAND gates taking unequal inputs is limited. Some of these cases yield superpolynomial separation between quantum and classical query complexity.

Algorithm: Gradients, Structured Search, and Learning Polynomials

Speedup: Superpolynomial

Description: Suppose we are given a oracle for computing some smooth function f:R^d→R. The inputs and outputs to f are given to the oracle with finitely many bits of precision. The task is to estimate ∇f at some specified point x0∈R^d. As shown in [61], a quantum computer can achieve this using one query, whereas a classical computer needs at least d+1 queries. In [20], Bulger suggested potential applications for optimization problems. As shown in appendix D of [62], a quantum computer can use the gradient algorithm to find the minimum of a quadratic form in d dimensions using O(d) queries, whereas, as shown in [94], a classical computer needs at least dΩ(2^d) queries. Single query quantum algorithms for finding the minima of basins based on Hamming distance were given in [147, 148, 223]. The quantum algorithm of [62] can also extract all d^2 matrix elements of the quadratic form using O(d) queries, and more generally, all dn nth derivatives of a smooth function of d variables in O(dn−1) queries. As shown in [130, 146], quadratic forms and multilinear polynomials in d variables over a finite field may be extracted with a factor of d fewer quantum queries than are required classically.

Algorithm: Hidden Shift

Speedup: Superpolynomial

Description: We are given oracle access to some function f on ZN. We know that f(x) = g(x+s) where g is a known function and s is an unknown shift. The hidden shift problem is to find s. By reduction from Grover’s problem it is clear that at least N−v queries are necessary to solve hidden shift in general. However, certain special cases of the hidden shift problem are solvable on quantum computers using O(1) queries. In particular, van Dam et al. showed that this can be done if f is a multiplicative character of a finite ring or field [89]. The previously discovered shifted Legendre symbol algorithm [88, 86] is subsumed as a special case of this, because the Legendre symbol (xp) is a multiplicative character of Fp. No classical algorithm running in time O(polylog(N)) is known for these problems. Furthermore, the quantum algorithm for the shifted Legendre symbol problem would break a certain cryptographic pseudorandom generator given the ability to make quantum queries to the generator [89]. Roetteler has found exponential quantum speedups for finding hidden shifts of certain nonlinear Boolean functions [105, 130]. Building on this work, Gavinsky, Roetteler, and Roland have shown [142] that the hidden shift problem on random boolean functions f:Z^m→Z^2 has O(n) average case quantum complexity, whereas the classical query complexity is Ω(2^m/2). The results in [143], though they are phrased in terms of the hidden subgroup problem for the dihedral group, imply that the quantum query complexity of the hidden shift problem for an injective function on ZN is O(log n), whereas the classical query complexity is Θ(N−v). However, the best known quantum circuit complexity for injective hidden shift on ZN is O(2^ClogN), achieved by Kuperberg’s sieve algorithm [66].

Algorithm: Pattern matching

Speedup: Superpolynomial

Description: Given strings T of length n and P of length m < n, both from some finite alphabet, the pattern matching problem is to find an occurrence of P as a substring of T or to report that P is not a substring of T. More generally, T and P could be d-dimensional arrays rather than one-dimensional arrays (strings). Then, the pattern matching problem is to return the location of P as a n×m…×m block within the n×n×…×n array T or report that no such location exists. The Ω(N−v) query lower bound for unstructured search [216] implies that the worst-case quantum query complexity of this problem is Ω(n^d+m−v). A quantum algorithm achieving this, up to logarithmic factors, was obtained in [217]. This quantum algorithm works through the use of Grover’s algorithm together with a classical method called deterministic sampling. More recently, Montanaro showed that superpolynomial quantum speedup can be achieved on average case instances of pattern matching, provided that m is greater than logarithmic in n. Specifically, the quantum algorithm given in [215] solves average case pattern matching in O((n/m)d/2(O(d/2logm))) time. This quantum algorithm is constructed by generalizing Kuperberg’s quantum sieve algorithm [66] for dihedral hidden subgroup and hidden shift problems so that it can operate in d dimensions and accommodate small amounts of noise, and then classically reducing the pattern matching problem to this noisy d-dimensional version of hidden shift.

Algorithm: Linear Systems

Speedup: Superpolynomial

Description: We are given oracle access to an n×n matrix A and some description of a vector b. We wish to find some property of f(A)b for some efficiently computable function f. Suppose A is a Hermitian matrix with O(polylog n) nonzero entries in each row and condition number k. As shown in [104], a quantum computer can in O(k^2log n) time compute to polynomial precision various expectation values of operators with respect to the vector f(A)b (provided that a quantum state proportional to b is efficiently constructable). For certain functions, such as f(x)=1/x, this procedure can be extended to non-Hermitian and even non-square A. The runtime of this algorithm was subsequently improved to O(k^2log^3log n) in [138]. Exponentially improved scaling of runtime with precision was obtained in [263]. Extensions of this quantum algorithm have been applied to problems of estimating electromagnetic scattering crossections [249], solving linear differential equations [156], estimating electrical resistance of networks [210], least-squares curve-fitting [169], and machine learning [214, 222, 250, 251]. Some limitations of the quantum machine learning algorithms based on linear systems are nicely summarized in [246]. In [220] it was shown that quantum computers can invert well-conditioned matrices using only logarithmically many qubits, whereas polynomially many classical bits are required.

Algorithm: Ordered Search

Speedup: Constant

Description: We are given oracle access to a list of N numbers in order from least to greatest. Given a number x, the task is to find out where in the list it would fit. Classically, the best possible algorithm is binary search which takes log2N queries. Farhi et al. showed that a quantum computer can achieve this using O(√n log2N) queries [39]. Currently, the best known deterministic quantum algorithm for this problem uses O(433 log2N) queries [103]. A lower bound of ln2πlog2N quantum queries has been proven for this problem [219, 24]. In [10], a randomized quantum algorithm is given whose expected query complexity is less than 13log2N.
Algorithm: Graph Properties in the Adjacency Matrix Model
Speedup: Polynomial
Description: Let G be a graph of n vertices. We are given access to an oracle, which gives a pair of integers in \{1,2,...,n\} tells us whether the corresponding vertices are connected by an edge. Building on previous work [35,52,36], Dürr et al. [34] show that the quantum query complexity of finding a minimum spanning tree of weighted graphs, and deciding connectivity for directed and undirected graphs have Θ(n3/2) quantum query complexity, and that finding lowest weight paths has O(nlogn) quantum query complexity. Suppose we are given oracle access to n2/3 quantum queries. A span-program-based quantum algorithm for detecting trees of a given size as minors in \(O(n)\) time is given in [204]. Span-program-based quantum algorithms for graph connectivity and bipartiteness matching the query complexity performance of the earlier quantum algorithms for these tasks are given in [227]. All of these problems are thought to have \(Ω(n^2)\) classical query complexity. A graph property is sparse if there exists a constant \(c\) such that every graph with the property has a ratio of edges to vertices at most \(c\). Childs and Kothari have shown that all sparse graph properties have query complexity \(Ω(n^2)\) if they cannot be characterized by a list of forbidden subgraphs and \(o(n^2)\) (little-o) if they can [140]. The former algorithm is based on Grover search, the latter on the quantum walk formalism of [141]. By Mader's theorem, sparse graph properties include all nontrivial minor-closed properties. These include planarity, being a forest, and not containing a path of given length. Another interesting computational problem is finding a subgraph H in a given graph G. The simplest case of this finding the triangle, that is, the clique of size three. The fastest known quantum algorithm for this finds a triangle in \(O(n^3/4)\) quantum queries [276], improving upon [175, 171, 70, 152, 21]. Classically, triangle finding requires \(Ω(n^2)\) queries [21]. More generally, a quantum computer can find an arbitrary subgraph of k vertices using \(O(n^{2−2/k−2})\) queries where \(t=(2k−d−3)/(kd+1)(t+2)\) and d and m are such that H has a vertex of degree d and m edges [153]. This improves upon the previous algorithm of [10]. In some cases, this quantum complexity is beaten by the quantum algorithm of [190], which finds H using \(Ω(n^{3−1/(5+1)})\) queries, provided G is sparse, where \(v(H)\) is the size of the minimal vertex cover of H. A quantum algorithm for finding constant-sized sub-hypergraphs over 3-uniform hypergraphs in \(O(n^{1.883})\) queries is given in [241].

Algorithm: Graph Properties in the Adjacency List Model
Speedup: Polynomial
Description: Let G be a graph of N vertices, M edges, and degree d. We are given access to an oracle which, when queried with the label of a vertex and \(j\in\{1,2,...,d\}\) outputs the \(j\)th neighbor of the vertex or null if the vertex has degree less than \(d\). Suppose we are given the promise that G is either bipartite or is far from bipartite in the sense that a constant fraction of the edges would need to be removed to achieve bipartiteness. Then, as shown in [144], the quantum complexity of deciding bipartiteness is \(O(N/3)\). Also in [144], it is shown that distinguishing expander graphs from graphs that are far from expanders has quantum complexity \(O(N/3)\) and \(Ω(N/4)\), whereas the classical complexity is \(Θ(N−ν)\). The key quantum algorithmic tool is Ambainis' algorithm for element distinctness. In [34], it is shown that finding a minimal spanning tree has quantum query complexity \(Θ(NM−ν)\), deciding graph connectivity has quantum query complexity \(Θ(N−ν)\) in the undirected case, and \(Θ(NM−ν)\) in the directed case, and computing the lowest weight path from a given source to all other vertices on a weighted graph has quantum query complexity \(Θ(NM−ν)\).

Algorithm: Welded Tree
Speedup: Superpolynomial
Description: Some computational problems can be phrased in terms of the query complexity of finding one's way through a maze. That is, there is some graph G to which one is given oracle access. When queried with the label of a given node, the oracle returns a list of the labels of all adjacent nodes. The task is, starting from some source node (i.e. its label), to find the label of a certain marked destination node. As shown by Childef et al. [26], quantum computers can exponentially outperform classical algorithms at this task for at least some graphs. Specifically, consider the graph obtained by joining together two depth-n binary trees by a random "weld" such that all nodes but the two roots have degree three. Starting from one root, a quantum computer can find the other root using poly(n) queries, whereas this is provably impossible using classical queries.

Algorithm: Collision Finding and Element Distinctness
Speedup: Polynomial
Description: Suppose we are given oracle access to a two to one function \(f\) on a domain of size N. The collision problem is to find a pair \(x, y \in \{1, 2, \ldots, N\}\) such that \(f(x) = f(y)\). The classical randomized query complexity of this problem is \(Θ(N−ν)\), whereas, as shown by Brassard et al., a quantum computer can achieve this using \(O(N/3)\) queries [18]. Removing the promise that \(f\) is two-to-one yields a problem called element distinctness, which has \(Θ(N)\) classical query complexity. Improving upon [21], Ambainis gives a quantum algorithm with query complexity of \(O(N^{2/3})\) for element distinctness, which is optimal [7]. The problem of deciding whether any \(k\)-fold collisions exist is called \(k\)-distinctness. Improving upon [7,154], the best quantum algorithms for \(k\)-distinctness have query complexity \(Θ(n^{3−4/4−k})\) [172, 173]. For \(k=2,3\) this is also the time-complexity, up to logarithmic factors, by [7]. Given two functions \(f\) and \(g\), each on a domain of size \(N\), a claw is a pair \(x, y\) such that \(f(x) = g(y)\). The algorithm of [7] solves claw-finding in \(O(N^{2/3})\) queries as a special case, improving on the classical \(O(N^{2/3})\) complexity. A related problem to element distinctness, is, given oracle access to a sequence, to estimate the \(k\)th frequency moment \(\sum_{i,j} x_i y_j\), where \(n\) is the number of times that \(j\) occurs in the sequence. An approximately quadratic speedup for this problem is obtained in [277]. See also graph collision.

Algorithm: Graph Collision
Speedup: Polynomial
Description: We are given an undirected graph of \(n\) vertices and oracle access to a labelling of the vertices by 1 and 0. The graph collision problem is, by querying this oracle, to decide whether there exists a pair of vertices, connected by an edge, both of which are labelled 1. One can embed Grover's unstructured search problem as an instance of graph collision by choosing the star graph, labelling the center 1, and labelling the remaining vertices by the database entries. Hence, this problem has quantum query complexity \(Ω(n)\) and classical query complexity \(Θ(n)\). In [70], Magniez, Nayak, and Szegedy gave a \(O(n/3)\) query algorithm for graph collision on general graphs. This remains the best upper bound on quantum query complexity for general graphs. However, stronger upper bounds have been obtained for several special classes of graphs. Specifically, the quantum query complexity on a graph \(G\) is \(Ω(n^{4/3})\) where \(l\) is the number of non-edges in \(G\) [161], \(O(n^{1/6})\) where \(a\) is the size of the largest independent set of \(G\) [172], \(O(n^{4/9−τ})\) where \(τ\) is the maximum total degree of any independent set of \(G\) [200], and \(O(n^{1/6})\) where \(t\) is the treewidth of \(G\) [201]. Furthermore, the quantum query complexity is \(Ω(n)\) with high probability for random graphs in which the presence or absence of an edge between each pair of vertices is chosen independently with fixed probability, (i.e. Erdős-Rényi graphs) [200]. See [201] for a summary of these results as well as new upper bounds for two additional classes of graphs that are too complicated to describe here.

Algorithm: Matrix Commutativity
Speedup: Polynomial
Description: We are given oracle access to \(k\) matrices, each of which are non-zero. Given integers \(i,j \in \{1, 2, \ldots, n\}\), and \(x \in \{1, 2, \ldots, k\}\) the oracle returns the \(ij\) matrix element of the \(x\)th matrix. The task is to decide whether all of these \(k\) matrices commute. As shown by Itakura [54], this can be achieved on a quantum computer using \(O(k^4n^9/5)\) queries, whereas classically this requires \(Ω(kn^2)\) queries.

Algorithm: Group Commutativity
Speedup: Polynomial
Description: We are given a list of \(k\) generators for a group \(G\) and access to a blackbox implementing group multiplication. By querying this blackbox we wish to determine whether the group is commutative. The best known classical algorithm is due to Pak and requires \(O(k)\) queries. Magniez and Nayak have shown that the quantum query complexity of this task is \(Θ(k^2/3)\) [139].

Algorithm: Hidden Nonlinear Structures
Speedup: Superpolynomial
Description: Any Abelian group \(G\) can be visualized as a lattice. A subgroup \(H\) of \(G\) is a sublattice, and the cosets of \(H\) are all the shifts of that sublattice. The Abelian hidden subgroup problem is normally solved by obtaining superposition over a random coset of the hidden subgroup, and then taking the Fourier transform so as to sample from the dual lattice. Rather than generalizing to non-Abelian groups (see non-Abelian hidden subgroup), one can instead generalize to the problem of determining the hidden coset of the hidden coset. In the case where \(H\) is a coset of a normal subgroup \(N\) of \(G\), and \(K\) is the kernel of the quotient map, the hidden subgroup problem can also be solved by a quantum computer, by obtaining superposition over a random coset of \(K\). If \(H\) is not a coset of any subgroup of \(G\), then the problem is known as the non-Abelian hidden subgroup problem. The best known classical algorithm is due to Pak and requires \(O(k^4)\) queries. Magniez and Nayak have shown that the quantum query complexity of this task is \(Θ(k^2)\) [139].
Algorithm: Center of Radial Function
Speedup: Polynomial
Description: We are given an oracle that evaluates a function \( f \) from \( \mathbb{R}^d \) to some arbitrary set \( S \), where \( f \) is spherically symmetric. We wish to locate the center of symmetry, up to some precision. (For simplicity, let the precision be fixed.) In \cite{110}, Liu gives a quantum algorithm, based on a curvelet transform, that solves this problem using a constant number of quantum queries independent of \( d \). This constitutes a polynomial speedup over the classical lower bound, which is \( \Omega(d) \) queries. The algorithm works when the function \( f \) fluctuates on sufficiently small scales, e.g., when the level sets of \( f \) are sufficiently thin spherical shells. The quantum algorithm is shown to work in an idealized continuous model, and nonrigorous arguments suggest that discretization effects should be small.

Algorithm: Group Order and Membership
Speedup: Superpolynomial
Description: Suppose we are given oracle access to the (integer) entries of an \( n \times m \) matrix \( A \). We wish to determine the rank of the matrix. Classically this requires order \( nm \) queries. Building on \cite{149}, Belovs \cite{150} gives a quantum algorithm that can use fewer queries given a promise that the rank of the matrix is at least \( r \). Specifically, Belovs' algorithm uses \( O(n^{r+1}) \) queries to identify all of the counterfeit coins. We can introduce an oracle such that given a pair of subsets of the coins of equal cardinality, \( T \), if \( A \) has at most \( k \) nonzero entries in any row or column then \( T=O(\sqrt{n+m}) \). If \( A \) has at most \( k \) nonzero entries in any row or column then \( T=O(\sqrt{n+m}) \). (To achieve the corresponding query complexity in the \( k \)-sparse case, the oracle must take a column index as input, and provide a list of the nonzero matrix elements from that column as output.) As an important special case, one can use these quantum algorithms for the problem of determining whether a square matrix is singular, which is sometimes referred to as the determinant problem. For general \( A \) the quantum query complexity of the determinant problem is no lower than the classical query complexity \cite{151}. However, \cite{151} does not rule out a quantum speedup given a promise on \( A \), such as sparseness or lack of small singular values.
Approximation of general partition functions is given in [122]. A method based on quantum walks, achieving polynomial speedup for evaluating partition functions is given in [265].

For each for the minimum number of queries to f. As shown by Freedman [42, 41], et al., finding a certain additive approximation to the Jones polynomial of the plat closure of a braid at \( e^{i\pi/5} \) is a BQP-complete problem. This result was reformulated and extended to \( e^{i\pi/3} \). Algorithm: Knot Invariants

Many techniques for quantum simulation have been developed for general classes of Hamiltonians [1,99, 145], and quantum field theory [107,166,228,229, 230]. The exponential complexity of the search with wildcards problem is to identify a hidden n-bit string \( x \) by making queries to an oracle \( f \). Given \( S \subseteq \{1,2,...,n\} \) and \( y \in \{0,1\}^S \), returns one if the substring of \( x \) specified by \( S \) is equal to \( y \), and returns zero otherwise. Classically, this problem has query complexity \( \Theta(n) \).

Various

Algorithm: Learning Machine

On \( \mathbb{R} \) is specified, for example as an explicit list. Our task is to find a size-k subset of \( D \) satisfying \( P \), i.e. \( (x_1,f(x_1)),...,(x_k,f(x_k)) \notin P \), or reject if none exists. As usual, we wish to do this with the minimum number of queries to \( f \). As shown in [167], the quantum query complexity of this problem is \( \Theta(n^2) \). Interestingly, this quadratic speedup is achieved not through amplitude amplification or quantum walks, but rather through use of the so-called Pretty Good Measurement. The paper [167] also gives a quantum speedup for the related problem of combinatorial group testing. This result and subsequent faster quantum algorithms for group testing are discussed in the entry on Junta Testing and Group Testing.

Algorithm: Network Flows

In [161], the best known upper bounds on classical complexity of the network flow and matching problems are complicated to state because different classical algorithms are preferable in different parameter regimes. However, in certain regimes, the above quantum algorithms beat all known classical algorithms. (See [160] for details.)

Algorithm: Electrical Resistance

We are given oracle access to a weighted graph of \( n \) vertices and maximum degree \( d \) whose edge weights are to be interpreted as electrical resistances. Our task is to compute the resistance between a chosen pair of vertices. Wang gave two quantum algorithms in [210] for this task, that run in time \( \text{poly}(\log n,1/d,1/\epsilon) \), where \( \epsilon \) is the expansion of the graph, and the answer is to be given to within a factor of \( 1+\epsilon \). Known classical algorithms for this problem are polynomial in \( n \) rather than \( \log n \). One of Wang's algorithms is based on a novel use of quantum walks. The other is based on the quantum algorithm of [104] for solving linear systems of equations.

Algorithm: Machine Learning Varies

Description: Mach learning encompasses a wide variety of computational problems and can be attacked by a wide variety of algorithmic techniques. This entry summarizes quantum algorithmic techniques for improved machine learning. Many of the quantum algorithms here are cross-

Algorithm: Quantum Simulation

Description: We are given oracle access to a weighted graph of \( n \) vertices and maximum degree \( d \) whose edge weights are to be interpreted as electrical resistances. Our task is to compute the resistance between a chosen pair of vertices. Wang gave two quantum algorithms in [210] for this task, that run in time \( \text{poly}(\log n,1/d,1/\epsilon) \), where \( \epsilon \) is the expansion of the graph, and the answer is to be given to within a factor of \( 1+\epsilon \). Known classical algorithms for this problem are polynomial in \( n \) rather than \( \log n \). One of Wang's algorithms is based on a novel use of quantum walks. The other is based on the quantum algorithm of [104] for solving linear systems of equations.

Algorithm: Electrical Resistance

We are given oracle access to a weighted graph of \( n \) vertices and maximum degree \( d \) whose edge weights are to be interpreted as electrical resistances. Our task is to compute the resistance between a chosen pair of vertices. Wang gave two quantum algorithms in [210] for this task, that run in time \( \text{poly}(\log n,1/d,1/\epsilon) \), where \( \epsilon \) is the expansion of the graph, and the answer is to be given to within a factor of \( 1+\epsilon \). Known classical algorithms for this problem are polynomial in \( n \) rather than \( \log n \). One of Wang's algorithms is based on a novel use of quantum walks. The other is based on the quantum algorithm of [104] for solving linear systems of equations.

Algorithm: Machine Learning Varies

Description: Mach learning encompasses a wide variety of computational problems and can be attacked by a wide variety of algorithmic techniques. This entry summarizes quantum algorithmic techniques for improved machine learning. Many of the quantum algorithms here are cross-

Algorithm: Quantum Simulation

Description: We are given oracle access to a weighted graph of \( n \) vertices and maximum degree \( d \) whose edge weights are to be interpreted as electrical resistances. Our task is to compute the resistance between a chosen pair of vertices. Wang gave two quantum algorithms in [210] for this task, that run in time \( \text{poly}(\log n,1/d,1/\epsilon) \), where \( \epsilon \) is the expansion of the graph, and the answer is to be given to within a factor of \( 1+\epsilon \). Known classical algorithms for this problem are polynomial in \( n \) rather than \( \log n \). One of Wang's algorithms is based on a novel use of quantum walks. The other is based on the quantum algorithm of [104] for solving linear systems of equations.

Algorithm: Electrical Resistance

We are given oracle access to a weighted graph of \( n \) vertices and maximum degree \( d \) whose edge weights are to be interpreted as electrical resistances. Our task is to compute the resistance between a chosen pair of vertices. Wang gave two quantum algorithms in [210] for this task, that run in time \( \text{poly}(\log n,1/d,1/\epsilon) \), where \( \epsilon \) is the expansion of the graph, and the answer is to be given to within a factor of \( 1+\epsilon \). Known classical algorithms for this problem are polynomial in \( n \) rather than \( \log n \). One of Wang's algorithms is based on a novel use of quantum walks. The other is based on the quantum algorithm of [104] for solving linear systems of equations.

Algorithm: Machine Learning Varies

Description: Mach learning encompasses a wide variety of computational problems and can be attacked by a wide variety of algorithmic techniques. This entry summarizes quantum algorithmic techniques for improved machine learning. Many of the quantum algorithms here are cross-

Algorithm: Quantum Simulation

Description: We are given oracle access to a weighted graph of \( n \) vertices and maximum degree \( d \) whose edge weights are to be interpreted as electrical resistances. Our task is to compute the resistance between a chosen pair of vertices. Wang gave two quantum algorithms in [210] for this task, that run in time \( \text{poly}(\log n,1/d,1/\epsilon) \), where \( \epsilon \) is the expansion of the graph, and the answer is to be given to within a factor of \( 1+\epsilon \). Known classical algorithms for this problem are polynomial in \( n \) rather than \( \log n \). One of Wang's algorithms is based on a novel use of quantum walks. The other is based on the quantum algorithm of [104] for solving linear systems of equations.
Algorithm: Adiabatic Algorithms
Speedup: Unknown
Description: In adiabatic quantum computation one starts with an initial Hamiltonian whose ground state is easy to prepare, and slowly varies the Hamiltonian to one whose ground state encodes the solution to some computational problem. By the adiabatic theorem, the system will track the instantaneous ground state provided the variation of the Hamiltonian is sufficiently slow. The runtime of an adiabatic algorithm scales at worst as $1/\gamma^2$ where $\gamma$ is the minimum eigenvalue gap between the ground state and the first excited state [186]. If the Hamiltonian is varied sufficiently smoothly, one can improve this to $O(1/\gamma^2)$ [247]. Adiabatic quantum computation was first proposed by Farhi et al. as a method for solving NP-complete combinatorial optimization problems [96, 186]. Adiabatic quantum algorithms are sometimes referred to as quantum annealing. Adiabatic quantum computation with non-stoquastic Hamiltonians is probably no more powerful than classical computation. The asymptotic runtime of adiabatic optimization algorithms is notoriously difficult to analyze, but some progress has been achieved [179, 180, 181, 182, 187, 188, 189, 190, 191, 226]. (Also relevant is an earlier literature on quantum annealing, which originally referred to a classical optimization algorithm that works by simulating a quantum process, much as simulated annealing is a classical optimization algorithm that works by simulating a thermal process. See e.g. [199, 198].) Adiabatic quantum computers can perform a process somewhat analogous to Grover search in $O(N^{1/4})$ time [98]. Adiabatic quantum algorithms achieving quadratic speedup for a more general class of problems are constructed in [184] by adapting techniques from [85]. Adiabatic quantum algorithms have been proposed for several specific problems, including PageRank [179], machine learning [192, 195], and graph problems [193, 194]. Some quantum simulation algorithms also use adiabatic state preparation.

Algorithm: Quantum Approximate Optimization
Speedup: Superpolynomial
Description: For many combinatorial optimization problems, finding the exact optimal solution is NP-complete. There are also hardness-of-approximation results proving that finding an approximation with sufficiently small error bound is NP-complete. For certain problems there is a gap between the best error bound achieved by a polynomial-time classical approximation algorithm and the error bound proven to be NP-hard. In this regime there is potential for exponential speedup by quantum computation. In [242] a new quantum algorithmic technique was proposed for finding approximate solutions to combinatorial optimization problems. In [243] it was subsequently shown that this algorithm solves a combinatorial optimization problem called Max E3LIN2 with a better approximation ratio than any polynomial-time classical algorithm known at the time. However, an efficient classical algorithm achieving an even better approximation ratio (in fact, the approximation ratio saturating the limit set by hardness-of-approximation) was subsequently discovered [260]. Presently, the power of quantum approximate optimization algorithms relative to classical algorithms remains unclear and is an active area of research.

Algorithm: Zeta Functions
Speedup: Superpolynomial
Description: Let $f(x,y)$ be a degree-d polynomial over a finite field $F_p$. Let $N$ be the degree of projective solutions to $f(x,y) = 0$ over the extension field $F_{p^r}$. The zeta function for $f$ is defined to be $Z(T) = \exp(\sum_{r \geq 1} N^r T^r)$ over $F_{p^r}$. Remarkably, $Z(T)$ always has the form $Z(T) = ZF_{p^r}Q(T)$ where $Q(T)$ is a polynomial of degree $2g$ and $\frac{\log(2)}{2d-1}(2d-1)$ is called the genus of $f$. Given $Z(T)$ one can easily compute the number of zeros of $f$ over any extension field $F_{p^r}$. One can similarly define the zeta function when the original field over which $f$ is defined does not have prime order. As shown by Kedlaya [64], quantum computers can determine the zeta function of a genus $g$ curve over a finite field $F_{p^r}$ in $\text{poly}(\log p, g, T)$ time. The fastest known classical algorithms are all exponential in either $\log(p)$ or $g$. In a different, but somewhat related context, van Dam has conjectured that due to a connection between the zeros of Riemann zeta functions and the eigenvalues of certain quantum operators, quantum computers might be able to efficiently approximate the number of solutions to equations over finite fields [87].

Algorithm: Weight Enumerators
Speedup: Superpolynomial
Description: Let $C$ be a code on $n$ bits, i.e. a subset of $Z_2^n$. The weight enumerator of $C$ is $SC(x,y) = \sum_{c \subseteq C} x^{n-c}y^c$ where $c$ denotes the Hamming weight of $c$. Weight enumerators have many uses in the study of classical codes. If $C$ is a linear code, it can be defined by $C = \{c | A \cdot c = 0\}$ where $A$ is a matrix over $Z_2$. In this case $SC(x,y) = \sum_{c \subseteq C} x^{n-c}y^c$ is a polynomial over $Z_2$ such that the degree of $SC(x,y)$ is the rank of $A$. Let $I_{r}$ be the lower triangular matrix resulting from setting all entries above the diagonal in $A$ to zero. Let $l_{r}$ be positive integers. Given the promise that $SC(A \cdot w, 1) = I_{r}$ for some small $r$, one can easily compute the number of zeros of $f$ over any extension field $F_{p^r}$. One can similarly define the zeta function when the original field over which $f$ is defined does not have prime order. As shown by Kedlaya [64], quantum computers can determine the zeta function of a genus $g$ curve over a finite field $F_{p^r}$ in $\text{poly}(\log p, g, T)$ time. The fastest known classical algorithms are all exponential in either $\log(p)$ or $g$. In a different, but somewhat related context, van Dam has conjectured that due to a connection between the zeros of Riemann zeta functions and the eigenvalues of certain quantum operators, quantum computers might be able to efficiently approximate the number of solutions to equations over finite fields [87].

Algorithm: Simulated Annealing
Speedup: Superpolynomial
Description: In simulated annealing, one has a series of Markov chains defined by stochastic matrices $M_1, M_2, \ldots, M_n$. These are slowly varying in the sense that their limiting distributions $\pi_1, \pi_2, \ldots, \pi_m$ satisfy $\gamma T + \eta_1 \eta_2 T = O(1)$ for some small $\eta$. These distributions can often be thought of as thermal distributions at successively lower temperatures. If $\gamma$ can be easily computed, then by applying this series of Markov chains one can sample from $\pi_m$. Typically, one wishes for $\gamma$ to be a distribution over good solutions to some optimization problem. Let $\delta$ be the gap between the largest and second largest eigenvalues of $M$. Let $\delta_{\text{min}}$. The run time of this classical algorithm is proportional to $1/\delta$. Building upon results of Szegedy [135, 85], Somma et al. have shown [84, 177] that quantum computers can sample from $\pi_m$ with a runtime proportional to $1/\delta^2$. Additional methods by which classical Markov chain Monte Carlo algorithms can be sped up using quantum walks are given in [265].

Algorithm: String Rewriting
Speedup: Superpolynomial
Description: String rewriting is a fairly general model of computation. String rewriting systems (sometimes called grammars) are specified by a list of rules by which certain substrings are allowed to be replaced by certain other substrings. For example, context free grammars are equivalent to the pushdown automata. In [59], Janzing and Wocjan showed that a certain string rewriting problem is PromiseBQP-complete. Thus quantum computers can solve it in polynomial time, but classical computers probably cannot. Given three strings $s_1, s_2, s_3$, and a set of rewriting rules satisfying certain premises, the problem is to find a certain approximation to the difference between the number of ways of obtaining $s_3$ from $s_1$ and the number of ways of obtaining $s_3$ from $s_2$. Similarly, certain problems of approximating the difference in number of paths between pairs of vertices in a graph, and difference in transition probabilities between pairs of states in a random walk are also BQP-complete [58].

Algorithm: Matrix Powers
Speedup: Superpolynomial
Description: Quantum computers have an exponential advantage in approximating matrix powers of powers of exponentially large sparse matrices. Suppose we have an $N \times N$ symmetric matrix $A$ such that there are at most $\text{polylog}(N)$ nonzero entries in each row, and given a row index, the set of nonzero entries can be efficiently computed. The task is, for any $1 \leq i \leq N$, and any $m \text{ polylogarithmic in } N$, to approximate $(Am)_{ii}$ the $i$th diagonal matrix element of $Am$. The approximation is additive to within $\text{polylog}$ of the true element. As shown by Janzing and Wocjan, this problem is PromiseBQP-complete, as is the corresponding problem for off-diagonal matrix elements [60]. Thus, quantum computers can solve it in polynomial time, but classical computers cannot.

Algorithm: Stoquastic Ising
Speedup: Superpolynomial
Description: In adiabatic quantum computation one starts with an initial Hamiltonian whose ground state is easy to prepare, and slowly varies the Hamiltonian to one whose ground state encodes the solution to some computational problem. By the adiabatic theorem, the system will track the instantaneous ground state provided the variation of the Hamiltonian is sufficiently slow. The runtime of an adiabatic algorithm scales at worst as $1/\gamma^2$ where $\gamma$ is the minimum eigenvalue gap between the ground state and the first excited state [186]. If the Hamiltonian is varied sufficiently smoothly, one can improve this to $O(1/\gamma^2)$ [247]. Adiabatic quantum computation was first proposed by Farhi et al. as a method for solving NP-complete combinatorial optimization problems [96, 186]. Adiabatic quantum algorithms are sometimes referred to as quantum annealing. Adiabatic quantum computation with non-stoquastic Hamiltonians is probably no more powerful than classical computation. The asymptotic runtime of adiabatic optimization algorithms is notoriously difficult to analyze, but some progress has been achieved [179, 180, 181, 182, 187, 188, 189, 190, 191, 226]. (Also relevant is an earlier literature on quantum annealing, which originally referred to a classical optimization algorithm that works by simulating a quantum process, much as simulated annealing is a classical optimization algorithm that works by simulating a thermal process. See e.g. [199, 198].) Adiabatic quantum computers can perform a process somewhat analogous to Grover search in $O(N^{1/4})$ time [98]. Adiabatic quantum algorithms achieving quadratic speedup for a more general class of problems are constructed in [184] by adapting techniques from [85]. Adiabatic quantum algorithms have been proposed for several specific problems, including PageRank [179], machine learning [192, 195], and graph problems [193, 194]. Some quantum simulation algorithms also use adiabatic state preparation.
1. A scalable physical system with well-characterized qubits
2. Ability to initialize the state of the qubits to a simply fiducial state, e.g., $|0\ 0\ 0\ 0\ 0\>$
3. Long relevant decoherence times $\gg$ gate operation times
   {+ absolute speed requirement}
4. ‘Universal’ set of quantum gates {+ w/ error correction methods}
5. A qubit-specific measurement capability
6. The ability to interconvert stationary and flying qubits
7. The ability to faithfully transmit flying qubits between specified locations
Living with Errors/Decoherence
- originally thought to be the death of QC
- ~4 strategies:
  - **Quantum Error Correction (concatenation):** use multiple qubits to encode, measure ancillas to find and correct errors
  - **Dynamical decoupling:** rapidly swap between qubit states (e.g., 0 and 1) factor out errors
  - **Decoherence-free subspaces:** If several qubits are subject to identical environmental noise, certain states are inherently robust to this
  - **Topological encoding:** immune to perturbation
\begin{align*}
|0\rangle & \rightarrow |0\rangle \equiv |000\rangle, \\
|1\rangle & \rightarrow |1\rangle = |111\rangle.
\end{align*}

\begin{align*}
 a |0\rangle + b |1\rangle & \rightarrow a |0\rangle + b |1\rangle = a |000\rangle + b |111\rangle. \\
 a |100\rangle + b |011\rangle
\end{align*}
Concatenation
Requirements for Fault-Tolerance

1. Low gate error rates.
2. Ability to perform operations in parallel.
3. A way of remaining in, or returning to, the computational Hilbert space.
4. A source of fresh initialized qubits during the computation.
5. Benign error scaling: error rates that do not increase as the computer gets larger, and no large-scale correlated errors.

These slides borrowed* from Daniel Gottesman (Perimeter Institute, Waterloo)

*Borrowed = stolen
Additional Desiderata

1. Ability to perform gates between distant qubits.
2. Fast and reliable measurement and classical computation.
3. Little or no error correlation (unless the registers are linked by a gate).
5. High parallelism.
6. An ample supply of extra qubits.
7. Even lower error rates.

These slides borrowed* from Daniel Gottesman (Perimeter Institute, Waterloo)

*Borrowed = stolen
Threshold Values

Computed threshold value depends on error-correcting code, fault-tolerant circuitry, analysis technique. Assume for now that all additional desiderata are satisfied.

- Concatenated 7-qubit code, standard circuitry:
  - Threshold $\sim 10^{-3}$ (various simulations)
  - Threshold $\sim 3 \times 10^{-5}$ (proof: Aliferis, Gottesman, Preskill, quant-ph/0504218; also Reichardt, quant-ph/0509203)

- Best known code: 25-qubit Bacon-Shor code
  - Threshold $\sim 2 \times 10^{-4}$ (proof: Aliferis, Cross, quant-ph/0610063)

These slides borrowed* from Daniel Gottesman (Perimeter Institute, Waterloo)
*Borrowed = stolen
Ancilla Factories

Best methods trade extra ancilla qubits for error rate: Ancilla factories create complex ancilla states to substitute for most gates on the data. Errors on ancillas are less serious, since bad ancillas can be discarded safely (Steane, quant-ph/9611027).

Extreme case: Create all states using error-detecting codes, ensuring a low basic error rate but very high overheads (e.g. $10^6$ or more physical qubits per logical qubit) -- Knill, quant-ph/0404104, Reichardt, quant-ph/0406025.

- Simulations: threshold $\sim 1\%$ or higher.
- Provable threshold $\sim 10^{-3}$?

These slides borrowed* from Daniel Gottesman (Perimeter Institute, Waterloo)

*Borrowed = stolen
Dynamical Decoupling

Dynamical suppression of decoherence in two-state quantum systems

Lorenza Viola* and Seth Lloyd†
ad’Arbeloff Laboratory for Information Systems and Technology, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139
(Received 19 March 1998)

The dynamics of a decohering two-level system driven by a suitable control Hamiltonian is studied. The control procedure is implemented as a sequence of radio-frequency pulses that repetitively flip the state of the system, a technique that can be termed quantum “bang-bang” control after its classical analog. Decoherence introduced by the system’s interaction with a quantum environment is shown to be washed out completely in the limit of continuous flipping and greatly suppressed provided the interval between the pulses is made comparable to the correlation time of the environment. The model suggests a strategy to fight against decoherence that complements existing quantum error-correction techniques. [S1050-2947(98)07109-1]

\[ |0\rangle + |1\rangle \rightarrow e^{i\varphi} |0\rangle + e^{i\chi} |1\rangle \Rightarrow e^{i\varphi} |1\rangle + e^{i\chi} |0\rangle \]

\[ \rightarrow e^{i\varphi} e^{i\chi'} |1\rangle + e^{i\chi} e^{i\varphi'} |0\rangle \Rightarrow e^{i\varphi} e^{i\chi'} |0\rangle + e^{i\chi} e^{i\varphi'} |1\rangle \]

\[ = e^{i\varphi} e^{i\chi'} \left( |0\rangle + e^{i(\chi-\chi')} e^{i(\varphi'-\varphi')} |1\rangle \right) \approx |0\rangle + |1\rangle \]
Dynamical Decoupling

Dephasing time of GaAs electron-spin qubits coupled to a nuclear bath exceeding 200 µs

Hendrik Bluhm¹†, Sandra Foletti¹†, Izhar Neder¹, Mark Rudner¹, Diana Mahalu², Vladimir Umansky² and Amir Yacoby¹*  

Qubits, the quantum mechanical bits required for quantum computing, must retain their quantum states for times long enough to allow the information contained in them to be processed. In many types of electron-spin qubits, the primary source of information loss is decoherence due to the interaction with nuclear spins of the host lattice. For electrons in gate-defined GaAs quantum dots, spin-echo measurements have revealed coherence times of about 1 µs at magnetic fields below 100 mT (refs 1,2). Here, we show that coherence in such devices can survive much longer, and provide a detailed understanding of the measured nuclear-spin-induced decoherence. At fields above a few hundred millitesla, the coherence time measured using a single-pulse spin echo is 30 µs. At lower fields, the echo first collapses, but then revives at times determined by the relative Larmor precession of different nuclear species. This behaviour was recently predicted³,⁴, and can, as we show, be quantitatively accounted for by a semiclassical model for the dynamics of electron and nuclear spins. Using a multiple-pulse Carr-Purcell-Meiboom-Gill echo sequence, the decoherence time can be extended to more than 200 µs, an improvement by two orders of magnitude compared with previous measurements¹²⁵.
Decoherence-Free Subspaces for Quantum Computation

D. A. Lidar, I. L. Chuang, and K. B. Whaley

Decoherence in quantum computers is formulated within the semigroup approach. The error generators are identified with the generators of a Lie algebra. This allows for a comprehensive description which includes as a special case the frequently assumed spin-boson model. A generic condition is presented for errorless quantum computation: decoherence-free subspaces are spanned by those states which are annihilated by all the generators. It is shown that these subspaces are stable to perturbations and, moreover, that universal quantum computation is possible within them.

Concatenating Decoherence-Free Subspaces with Quantum Error Correcting Codes

D. A. Lidar, D. Bacon, and K. B. Whaley

An operator sum representation is derived for a decoherence-free subspace (DFS) and used to (i) show that DFS’s are the class of quantum error correcting codes (QECC’s) with fixed, unitary recovery operators and (ii) find explicit representations for the Kraus operators of collective decoherence. We demonstrate how this can be used to construct a concatenated DFS-QECC code which protects against collective decoherence perturbed by independent decoherence. The code yields an error threshold which depends only on the perturbing independent decoherence rate.

Universal Fault-Tolerant Quantum Computation on Decoherence-Free Subspaces

D. Bacon, J. Kempe, D. A. Lidar, and K. B. Whaley

A general scheme to perform universal, fault-tolerant quantum computation within decoherence-free subspaces (DFSs) is presented. At most two-qubit interactions are required, and the system remains within the DFS throughout the entire implementation of a quantum gate. We show explicitly how to perform universal computation on clusters of the four-qubit DFS encoding one logical qubit each under spatially symmetric (collective) decoherence. Our results have immediate relevance to quantum computer implementations in which quantum logic is implemented through exchange interactions, such as the recently proposed spin-spin coupled quantum dot arrays and donor-atom arrays.
A Decoherence-Free Quantum Memory Using Trapped Ions

D. Kielpinski, V. Meyer, M. A. Rowe, C. A. Sackett, W. M. Itano, C. Monroe, D. J. Wineland

We demonstrate a decoherence-free quantum memory of one qubit. By encoding the qubit into the decoherence-free subspace (DFS) of a pair of trapped $^9$Be$^+$ ions, we protect the qubit from environment-induced dephasing that limits the storage time of a qubit composed of a single ion. We measured the storage time under ambient conditions and under interaction with an engineered noisy environment and observed that encoding into the DFS increases the storage time by up to an order of magnitude. The encoding reversibly transfers an arbitrary qubit stored in a single ion to the DFS of two ions.

The DFS realized here is spanned by $|\psi_-\rangle = (|\downarrow\uparrow\rangle - i |\uparrow\downarrow\rangle)/\sqrt{2}$ and $|\psi_+\rangle = (|\downarrow\uparrow\rangle + i |\uparrow\downarrow\rangle)/\sqrt{2}$, which form the basis states for our logical qubit. These states are clearly invariant under collective dephasing; the transformation $|\uparrow\rangle \rightarrow e^{i\xi} |\uparrow\rangle$, applied simultaneously to both ions, leaves any superposition of $|\psi_-\rangle$ and $|\psi_+\rangle$ invariant. Such collective dephasing is expected to be a major source of qubit decoherence for quantum information processing using trapped atoms. The encoding method dem-
Fig. 1. Decay of the DFS-encoded state (circles) and the test state (crosses) under engineered dephasing noise. The noise is applied for a fraction of the delay time of $\sim 25 \, \mu s$ between encoding and decoding. Coherence data are normalized to their values for zero applied noise. The fit lines are exponential decay curves. The test data are predicted to decay exponentially for white noise, so we exclude the point with zero applied noise from the fit. The DFS data are fit for comparison. The decay rate of the test state is $0.18 \pm 0.01 \, \mu s^{-1}$, whereas the decay rate of the DFS state is $0.0035 \pm 0.0050 \, \mu s^{-1}$, consistent with zero decay.

Fig. 2. Decay of the DFS-encoded state (circles) and the test state (crosses) under ambient decoherence. We vary the delay time between encoding and decoding to give the ambient noise a variable time to act. Coherence data are normalized to their values for zero applied noise. The fit lines are exponential decay curves for purposes of comparison and are not theoretical predictions. The decay rate of the test state is $(7.9 \pm 1.5) \times 10^{-3} \, \mu s^{-1}$, whereas the decay rate of the DFS state is $(2.2 \pm 0.3) \times 10^{-3} \, \mu s^{-1}$. Because the coherence time of the DFS-encoded state is much longer than that of the test state, we see that the chief source of ambient decoherence is collective dephasing.
'True' DFS $\rightarrow$ singlets (look same in all bases)

$k = 2: |0\rangle = |01\rangle - |10\rangle \Rightarrow \text{no "1" state}$

$k = 3: \text{no singlet state}$

$k = 4: |0\rangle = \frac{1}{2} \left[ |0110\rangle + |1001\rangle - |0011\rangle - |1100\rangle \right]$  

$|1\rangle = \frac{1}{2\sqrt{3}} \left[ 2|1010\rangle + 2|0101\rangle - |0011\rangle - |1100\rangle - |0110\rangle - |1001\rangle \right]$  

Can you perform logic on these and not leave the DFS?  Yes!
Current state-of-the art...

Many systems being investigated: ions, atoms, photons, superconductors, ...

At present the ions are 'winning', with 14 qubits

14-Qubit Entanglement: Creation and Coherence

Thomas Monz,1 Philipp Schindler,1 Julio T. Barreiro,1 Michael Chwalla,1 Daniel Nigg,1 William A. Coish,2,3 Maximilian Harlander,1 Wolfgang Hänsel,4 Markus Hennrich,1,* and Rainer Blatt1,4

1Institut für Experimentalphysik, Universität Innsbruck, Technikerstr. 25, A-6020 Innsbruck, Austria
2Institute for Quantum Computing and Department of Physics and Astronomy, University of Waterloo, Waterloo, ON, N2L 3G1, Canada
3Department of Physics, McGill University, Montreal, Quebec, Canada H3A 2T8
4Institut für Quantenoptik und Quanteninformation, Österreichische Akademie der Wissenschaften, Otto-Hitimir-Platz 1, A-6020 Innsbruck, Austria

(Received 30 September 2010; published 31 March 2011)

We report the creation of Greenberger-Horne-Zeilinger states with up to 14 qubits. By investigating the coherence of up to 8 ions over time, we observe a decay proportional to the square of the number of qubits. The observed decay agrees with a theoretical model which assumes a system affected by correlated, Gaussian phase noise. This model holds for the majority of current experimental systems developed towards quantum computation and quantum metrology.
So, what is the “winning” technology going to be? I don’t think that any living mortal has an answer to this question, and at this point it may be counterproductive even to ask it. Even though we have lived with quantum mechanics for a century, our study of quantum effects in complex artificial systems like those we have in mind for quantum computing is in its infancy. No one can see how or whether all the requirements above can be fulfilled, or whether there are new tradeoffs, not envisioned in our present theoretical discussions but suggested by further experiments, that might take our investigations in an entirely new path.

D. Divincenzo

Now multiple paradigms:
- circuit model
- cluster-state approach (‘one-way q. computing’)
- adiabatic quantum computing
- topological quantum computing
Surface codes: Towards practical large-scale quantum computation

Austin G. Fowler
Centre for Quantum Computation and Communication Technology, School of Physics, The University of Melbourne, Victoria 3010, Australia

Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland
Department of Physics, University of California, Santa Barbara, California 93106-9530, USA
and California Nanosystems Institute, University of California, Santa Barbara, California 93106-9530, USA
(Received 2 August 2012; published 18 September 2012)
(b) Estimate

- $d=3$
- $d=7$
- $d=11$
- $d=25$
- $d=55$

Logical $X$ error rate $P_L$

Per-step error rate $p$

Number of qubits $n_q$

Single step error rate $p/p_{th}$

$P_L = 10^{-20}$

$P_L = 10^{-10}$

$P_L = 10^{-5}$