Progress in quantum algorithms

Peter W. Shor
Dept. of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

September 14, 2005

Abstract

We discuss the progress (or lack of it) that has been made in discovering algorithms for computation on a quantum computer. Some possible reasons are given for the paucity of quantum algorithms so far discovered, and a short survey is given of the state of the field.

1 Introduction

It has now been ten years since I discovered the quantum factoring algorithm [27]. This discovery caused great excitement; although some quantum algorithms had previously been discovered, this was the first algorithm that gave a substantial speedup over a classical algorithm for a well-studied and interesting problem. Many people expected a succession of other interesting quantum algorithms to quickly follow. Lov Grover indeed discovered his quantum searching algorithm shortly thereafter [18], but the progress since has been disappointing, especially compared with the progress the rest of the field of quantum information processing has been making. Physicists have been proposing and experimenters have been exploring possible physical implementations of quantum computers at a pace I believe is faster than what anybody but the most optimistic people expected; these developments are covered in the rest of this issue. Quantum cryptography is coming of age, with several theoretical proofs of its security recently discovered, and commercial quantum cryptography systems now on the market. The field of quantum information theory and quantum computational complexity have both been quite active, with a succession of interesting and important theoretical results. Meanwhile, the development of quantum algorithms appears to have lagged behind, with what seem like barely any significant new algorithms having been discovered. We will speculate on why more quantum algorithms haven't been found, and survey the progress that has been made. This is an expansion and update of my paper [28] which also discusses this issue.

2 Thoughts on quantum algorithms

One thing I am often asked is why so few new quantum algorithms for solving classical problems have been discovered. It has not been for lack of effort; people have looked quite hard for new quantum algorithms. I can think of two reasons that quantum algorithms might be difficult to discover. The first is that there might really be only a few problems for which quantum computers can offer a substantial speed-up over classical computers; in the most

pessimistic scenario, we have already discovered most of the important algorithms. The second is that quantum computers operate in a manner so non-intuitive, and so different from classical computers, that all the experience of the last fifty years in discovering classical algorithms offers little insight into how to go about finding quantum algorithms, so that while efficient quantum algorithms for many more problems exist, they are very hard to find. It appears impossible to tell which of these two cases is the actuality.

Another thing that I am often asked is what kind of problems are susceptible to attack by a quantum computer. Unfortunately, even the classical analog of this question: *What kind of problems are can be solved in polynomial time by a digital computer?* does not have a satisfactory answer. Computer scientists have a plethora of techniques they can try to apply to a problem: linear programming, divide-and-conquer, dynamic programming, Monte Carlo methods, semidefinite programming, and so forth. However, deciding which of these methods is likely to work for a given problem, and how to apply it, remains more of an art than a science, and there is no good way known to characterize the class of problems having polynomial-time algorithms. Characterizing the class of problems having polynomial-time quantum algorithms appears equally, if not more, difficult, one of the main additional difficulties being that we have so far discovered very few algorithmic techniques.

One of the things that has made it difficult to find new quantum algorithms that perform better than classical algorithms is the remarkable job that computer scientists have done over the last fifty years in finding good classical algorithms for problems. For the most part, researchers have been looking for quantum algorithms that efficiently solve problems which are not known to be solvable classically in polynomial time. These would yield the most impressive advances, and are also very likely to be the first problems for which, when and if quantum computers are developed, the quantum algorithms will give a practical advantage in the real world. To find such a problem, if we make the assumption that quantum computers cannot solve NP-complete problems in faster than exponential time, we would need to find a problem which is neither in P nor is NP-hard. Remarkably, in part because of the success of the classical theory of algorithms, there are relatively few natural problems which fit this criterion.

I now give a brief digression on complexity theory. The complexity class P consists of those problems which can be solved using algorithms running in time bounded by a polynomial in the length of the input. The class of problems with probabilistic polynomial-time algorithms is called BPP, and the class with quantum probabilitistic polynomial-time algorithms is called BQP (quantum algorithms are in general inherently probabilistic, and so the class BQP should most fairly be compared with the class BPP rather than P). Polynomial running times are considered to be efficient by theoretical computer scientists. This isn't strictly true—nobody would call an algorithm that runs in n^{100} steps efficient in practice, where n is the length of the input, but this definition has proven to be a good compromise between theory and practice; it appears to be the case that most natural problems in P have algorithms with running time a relatively small power of n. The class NP consists of those problems for which a solution can be verified in polynomial time; this class contains P, and the containment is generally thought to be strict.

Computer scientists have identified a subclass of NP comprising the hardest problems in NP; these are called NP-complete problems [10, 22, 24], and a polynomial-time algorithm for any of these problems would imply a polynomial-time algorithm for all problems in NP, showing that P = NP. Remarkably, a large number of NP-complete problems have been identified [17]. When theoretical computer scientists consider a new problem, one of their first goals is to either show that it is NP-complete, or to find a polynomial-time algorithm for it. While these are mutually exclusive outcomes, it is not guaranteed that a problem in NP will either be NP-complete or in P; remarkably, however, the vast majority of problems

studied seem to fall in one of these two classes.

Why might we suspect that quantum computers cannot solve NP-complete problems? Let us consider the classical analog of that question: why do computer scientists believe that classical computers cannot solve NP-complete problems efficiently? This is the celebrated P vs. NP question. (See [9, 17, 26] for the history of this problem.) The class NP is the class of problems for which, once a solution has been found, it can be verified in polynomial time that it is indeed a solution. Mathematical speaking, NP is the set of languages for which there are polynomial length proofs that a string is in the language (although there are not necessarily short proofs that a string is *not* in the language). NP-complete problems are a subset of these NP problems which have the property that if any of these NP-complete problems is solvable by an efficient algorithm, then all NP problems are solvable by an efficient algorithm.

There are essentially two lines of argument for why P should be different from NP. The first, which in my opinion is not terribly convincing, is that nobody has yet found a polynomial-time algorithm for solving NP-complete problems. While such an algorithm would generate a complete upheaval of our understanding of computational complexity, similar revolutions have occurred, albiet infrequently, in other branches of mathematics and science. The second argument is barely more rigorous than the first. It relates NP completeness to the difficulty of finding mathematical proofs. If, for instance, a quadratic algorithm was discovered for solving an NP complete problem, then a mathematician could use this algorithm to mechanically check whether a conjectured theorem had a proof of length n using computation time of cn^2 steps for some constant c. Now, let us assume that the primes are in some sense quasi-randomly distributed, as is believed by many mathematicians (although many other quasi-randomly distributed objects could be used in this argument as well). It then seems that it should be very difficult to check the truth of a statement such as

There are 17 primes in arithmetic progression between integers a and b.

without testing a large fraction of the numbers between a and b for primality; here the relative sizes of a and b should be chosen so that the probability of the above statement is roughly $\frac{1}{2}$. On the other hand, if you are given 11 numbers, testing to see if these are indeed primes in arithmetic progression can be done in time polynomial in the length of b. This problem is in the class NP, which means that it can be efficiently translated into a 3SAT problem—a Boolean formula in conjuctive normal form with 3 variables per clause (this translation is essentially the proof of the NP-compleness result). However, this problem appears quite hard, and it is very likely not NP-complete (meaning the reverse translation cannot be done). If any NP-complete problem could be solved in polynomial time, then problems such as the above could be solved in polynomial time. Intuitively, it seems as though it would be very difficult to prove the non-existence of such an arithmetic progression of primes, especially if you believe the distribution of prime numbers is quasi-random. Thus, this is some intuitive evidence towards the conjecture that P $\neq$ NP.

Could the use of a quantum computer help solve such problems in NP? In this new question, we now have lost the mathematical intuition that proofs can be much harder to discover than to check. The symmetry between checking and discovering the proof is now gone: we are allowed a quantum computer to discover these proofs, but only permitted a digital computer to check them. Although the argument is not as convincing, it still does not seem likely that quantum computers can solve NP complete problems in less than exponential time. There is more evidence in this direction, in that there is a proof that a quantum computer cannot search a space of size N in less than $O(\sqrt{N})$ time. [7]. This result shows that a quantum algorithm for solving NP-complete problems in sub-exponential time will have to use the structure of these problems, and this result can also be used to find an oracle with respect to which NP is not contained in BQP.

If quantum computers cannot indeed solve NP complete problems, then where should we be looking for problems to speed up using quantum algorithms? The obvious place to look is in problems neither known to be in P or to be NP-complete. There are only a few problems in this class. Those handful of these which appear to be related to periodicity, and thus possibly susceptible to attack using quantum Fourier transforms, have received substantial study from the quantum algorithms community. These include the two problems of graph isomorphism and of finding a short vector in a geometrical lattice. The problem of graph isomorphism is: given two graphs, is there a permutation of the nodes which renders them identical? The problem of finding a short vector in a lattice is: given a lattice in d dimensions—i.e., the integer combinations of a set of d independent basis vectors—is it possible to efficiently find a vector that is not much longer than the shortest vector in this lattice? This problem becomes hard for large d. Finding a vector with length within a constant factor of the length of the shortest vector is NP-hard, while the best classical polynomial-time algorithms known can only find a vector having length within a factor that is exponential in the dimension d. While neither of these can be solved efficiently by a quantum algorithm yet, the study of the lattice problem from a quantum point of view has led to a purely classical result that puts this problem (with certain parameters) in the complexity class NP $\cap$ co-NP [2].

If we moderate our goals somewhat, and look also for quantum algorithms that speed problems up by a polynomial factor, then we have not only all the problems in P to consider, but also the NP-complete problems. Grover's algorithm can be applied to speed up the algorithms for many of these problems by a quadratic factor, and it is conceivable that some of them can be sped up by a larger factor. In my paper [28], I suggested looking at trying to speed up the solutions of problems in P by quantum algorithms, and I still believe this is a good source of research problems.

3 Progress in quantum algorithms

Despite the general lack of progress that appears to have been made on quantum algorithms, there have been a number of results which I consider to represent incremental progress which may eventually lead to new quantum algorithms. In my talks, I generally classify known quantum algorithms into three classes: those using periodicity finding, those using variants of Grover search, and those using quantum computers to simulate quantum mechanics. There has been progress in all three areas of this classification, and a couple of new algorithmic techniques have been proposed which appear promising, although they have so far not resulted in any breakthroughs in the discovery of new algorithms. I will now describe some of this progress; I will not attempt to be comprehensive, but merely to give pointers to some papers which I think show the potential for substantial progress.

We first treat the progress in quantum algorithms that use the Fourier transform, the tool that let us perform periodicity finding. It did not take long after the papers [27, 29] to realize that a natural generalization of the factoring and discrete logarithm algorithms was to the abelian hidden subgroup problem: the problem of finding a subgroup of an abelian group which is hidden in the values of a function. Fourier transforms on abelian groups could be used to find periodicity and solve this problem in much the same way that the Fourier transform on the cyclic group was used to factor and find discrete logarithms (see, e.g., [25]). Hallgren [21] has recently shown that the Fourier transform can also be used to find the periodicity of functions with irrational periods, and that this is useful in solving certain number theory problems such as finding solutions to Pell's equation and finding class groups of number fields. There are other properties of the Fourier transform which can be used for purposes other than finding periodicity. For instance, shifts of a periodic function transform

nicely under the Fourier transform, and this fact can be used to solve certain hidden shift problem [14]. The Fourier transform can also be defined over non-abelian groups. It is not known how to compute this efficiently for all these groups, but it can be computed for some of them, such as the symmetric group [6] and the dihedral group [15]. Kuperberg has recently been able to give an algorithm solving the hidden subgroup problem over the dihedral group in subexponential time by working directly with the representations returned by the Fouier transform [23]. These results indicate at least that the Fourier transform can be used in ways that are more versatile and powerful than those previously considered.

The second class of quantum algorithms I discuss are the generalization of Grover's algorithm for searching a set of N things in time $O(\sqrt{N})$ [18]. Many of these are covered in the survey [20]. The most important is probably that of amplitude amplification [8, 19], which lets one amplify the probability of success of a quantum algorithm which has only a small probability of success with efficiency quadratically better than would be possible classically. Recently, there have been a number of algorithms discovered that combine the techniques of Grover's algorithm with quantum walks to perform certain tasks faster than one can do classically [3, 4, 5]. It was first shown that quantum random walks could be used to solve some problems faster than classical algorithms could in [12]; these problems, however, appeared fairly artificial. Ambainis combined these with random walk techniques to give a near-optimal time quantum algorithm for testing whether two elements in a database are distinct [3]. These techniques have also been used to show that certain graphs with locality can be searched quickly by a quantum computer, where the computer program has the option either going from a vertex to a neighboring vertex or testing that vertex to see whether it is the "goal" vertex [5]. A survey of these results appears in [4].

The third class of quantum algorithms are those simulating quantum mechanics. There are two recent papers showing that the simulation of quantum mechanical processes can be used to solve certain classical problems faster than it is known how to do classically. One of these [11] uses the fact that a certain observable in topological quantum field theories has its expectation value equal to the value of the Jones polynomial evaluated at certain points. While the variance of this observable is too large for a quantum computer to compute the value of this Jones polynomial exactly in polynomial time, it can be approximated by a quantum computer much more efficiently than it is known how to do using a classical computer. Another paper along these lines shows how to approximate zeros of certain finite field zeta functions by a quantum computer [13]. Inspired by the spectral approach to the Riemann hypothesis, which attempts to relate the zeros of the Riemann zeta function to the eigenvalues of a (currently unknown) chaotic quantum system, van Dam shows that the zeros of certain finite field zeta functions are given by the eigenvalues of a quantum circuit, and that this fact can be used to approximate them more efficiently on a quantum computer than it is known how to do classically.

Finally, I want to mention adiabatic quantum computation. This heuristic attempts to find the ground state of a Hamiltonian by tracking its evolution by a quantum computer as the Hamiltonian is evolved from one whose ground state is known to one whose ground state is the desired result of the quantum computation [16]. The adiabatic theorem says that this approach is efficient if there exists a spectral gap of size at least reciprocal polynomial for all the intermediate Hamiltonians in the evolution. Although this approach has not yet been shown to yield an algorithm for an interesting problem, it does appear to me to have promise. It has been shown recently that all polynomial time quantum computations can be translated so they can be solved by this adiabatic method [1].

References

[1] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd and O. Regev, "Adiabatic quantum computation is equivalent to standard quantum computation," arXiv quant-ph/0405098.

[2] D. Aharonov and O. Regev, "Lattice problems in NP ∩ co-NP," manuscript in preparation, available at www.tau.ac.il/ odedr/.

[3] Andris Ambainis, "Quantum walk algorithm for element distinctness," quant-ph/0311001.

[4] Andris Ambainis, "Quantum walks and their algorithmic applications," quant-ph/0403120.

[5] Andris Ambainis, J. Kempe, and A. Rivosh, "Coins make quantum walks faster," quant-ph/0402107.

[6] R. Beals, "Quantum computation of Fourier transforms over symmetric groups," in *Proc. 29th Annual ACM Symposium on Theory of Computing,* (1997), pp. 48–53.

[7] C. H. Bennett, E. Bernstein, G. Brassard and U. Vazirani, "Strengths and weakness of quantum computing. *SIAM J. Comput.* **26,** pp. 1510–1523 (1997).

[8] G. Brassard, P. Höyer, M. Mosca and A. Tapp, "Quantum amplitude amplification and estimation," *AMS Conterporary Math Series* **305** pp. 53–74 *Quantum Computation and Information*, Amer. Math. Soc. (2002).

[9] S. Cook, "The P versus NP problem," at http://www.claymath.org/millennium/.

[10] S. Cook, "The complexity of theorem proving procedures," in *Proc. of the 3rd Annual ACM Symposium on Theory of Computing,* pp. 151–158, ACM Press, New York (1971).

[11] M. Bordewich, M. Freedman, L. Lovász and D. Welsh, "Approximate counting and quantum computation," available at http://research.microsoft.com/research/theory/freedman/.

[12] A. M. Childs, R. E. Cleve, E. Deotto, E.Farhi, S. Gutmann, and D. A. Spielman, "Exponential algorithmic speedup by quantum walk," *Proc. 35th ACM Symposium on Theory of Computing*, ACM Press (2003), 59-68.

[13] W. van Dam, "Quantum computing and zeroes of zeta functions," arXiv quant-ph/0405081.

[14] W. van Dam, S. Hallgren and L. Ip, "Quantum algorithms for some hidden shift problems," *Proc. ACM-SIAM Symposium on Discrete Algorithms,* 489-498 (2003).

[15] M. Ettinger and E. Höyer, "On quantum algorithms for non-commutative hidden subgroups," arXiv quant-ph/9807029.

[16] E. Farhi, J. Goldstone, S. Gutman and M. Sipser, "Quantum computation by adiabatic evolution," arXiv quant-ph/0001106.

[17] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W. H. Freeman and Company (1979).

[18] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack, *Phys. Rev. Lett.* **78**, pp. 325–328 (1997).

[19] L. K. Grover, "Quantum computers can search rapdily by using almost any transformation," *Phys. Rev. Lett.* **80** 4329–4332 (1998). needle in a haystack, *Phys. Rev. Lett.* **78**, pp. 325–328 (1997).

[20] L. K. Grover and A. M. Sengupta, "From coupled pendulums to quantum search," in *Mathematics of Quantum Computation,* R. K. Brylinski and G. Chen, Eds. Chapman&Hall/CRC, Boca Raton, FL, pp. 119–134.

[21] S. Hallgren, "Polynomial-time algorithms for Pell's equation and the principal ideal problem," in *Proc. 34th Annual ACM Symposium on Theory of Computing,* pp. 653–658, ACM Press (2002).

[22] R. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations,* (R. Miller, J. Thatcher eds.), Plenum, NY (1972) pp. 85-103.

[23] G. Kuperberg, "A subexponential-time quantum algorithm for the dihedral hidden subgroup problem," arXiv: quant-ph/0302112.

[24] L. A. Levin, *Problems of Information Transmission* "Universal search problems," **9**(3), pp. 265–266 (1973) [Russian].

[25] M. Mosca and A. Ekert,."The hidden subgroup problem and eigenvalue estimation on a quantum computer," *Proceedings of the 1st NASA International Conference on Quantum Computing and Quantum Communication*, Palm Springs, USA, *Lecture Notes in Computer Science* **1509** (1999); arXiv quant-ph/9903071.

[26] M. Sipser, "The history and status of the P versus NP question," *Proc. 24th ACM Symposium on the Theory of Computing,* pp. 603-619 (1992).

[27] P. W. Shor, "Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer,"

[28] P. W. Shor, "Why haven't more quantum algorithms been found?" J. ACM **50**, 87–90 (2003). *Siam J. Comput.* **26,** pp. 1484–1509 (1997).

[29] D. R. Simon, "On the power of quantum computation *Siam J. Comput.* **26,** pp. 1474–1483 (1997).