IS QUANTUM SEARCH PRACTICAL?

Gauging a quantum algorithm's practical significance requires weighing it against the best conventional techniques applied to useful instances of the same problem. The authors show that several commonly suggested applications of Grover's quantum search algorithm fail to offer computational improvements over the best conventional algorithms.

Some researchers suggest achieving massive speedups in computing by exploiting quantum-mechanical effects such as superposition (quantum parallelism) and entanglement.[1] A quantum algorithm typically consists of applying quantum gates to quantum states, but because the input to the algorithm might be normal classical bits (or nonquantum), it only affects the selection of quantum gates. After all the gates are applied, quantum measurement is performed, producing the algorithm's nonquantum output. Deutsch's algorithm, for instance, solves a certain artificial problem in fewer steps than any classical (nonquantum) algorithm, and its relative speedup grows with the problem size. However, Charles Bennett and his colleagues[2] have shown that quantum algorithms are unlikely to solve NP-complete problems in polynomial time, although more modest speedups remain possible. Peter Shor designed a fast (polynomial-time) quantum algorithm for number factoring—a key problem in cryptography that isn't believed to be NP-complete. No classical polynomial-time algorithm for number factoring is known, and the security of the RSA code used on the Internet relies on this problem's difficulty. If a large and error-tolerant quantum computer were available today, running Shor's algorithm on it could compromise e-commerce.

Lov Grover's quantum search algorithm is also widely studied. It must compete with advanced classical search techniques in applications that use parallel processing[3] or exploit problem structure, often implicitly. Despite its promise, though, it is by no means clear whether, or how soon, quantum computing methods will offer better performance in useful applications.[4] Traditional complexity analysis of Grover's algorithm doesn't consider the complexity of processing the so-called oracle queries—that is, certain questions with yes or no answers. The query process is simply treated as a black box, thus making Grover's algorithm appealing because it needs fewer queries than classical search. However, with sufficiently time-consuming query processing, Grover's algorithm can become nearly as slow as a simple (exhaustive) classical search.

In this article, we identify requirements for Grover's algorithm to be useful in practice:

1. A search application S where classical methods don't provide sufficient scalability.
2. An instantiation $Q(S)$ of Grover's search for

GEORGE F. VIAMONTES, IGOR L. MARKOV, AND JOHN P. HAYES
University of Michigan

Grover's Quantum Search Algorithm

Grover's quantum algorithm searches for a subset of items in an unstructured set of N items.[1] The algorithm incorporates the search criteria in the form of a black-box predicate that can be evaluated on any items in the set. The complexity of this evaluation (query) varies depending on the search criteria. With conventional algorithms, searching an unstructured set of N items requires $\Omega(N)$ queries in the worst case. In the quantum domain, however, Grover's algorithm can perform unstructured search by making only $O(\sqrt{N})$ queries, a quadratic speedup over the classical case. This improvement is contingent on the assumption that the search predicate can be evaluated on a superposition of all database items. Additionally, converting classical search criteria to quantum circuits often entails a moderate overhead, and the quantum predicate's complexity can offset the reduction in the number of queries.

Figure A shows a high-level circuit representation of Grover's algorithm. (See related research[2,3] for more detailed circuits.) The algorithm's first step is to initialize $\log(N)$ qubits, each with a value of $|0\rangle$. These qubits are then placed into an equal superposition of all values from 0 to $N-1$ (encoded in binary) by applying one Hadamard gate on each input qubit. Because the superposition con-
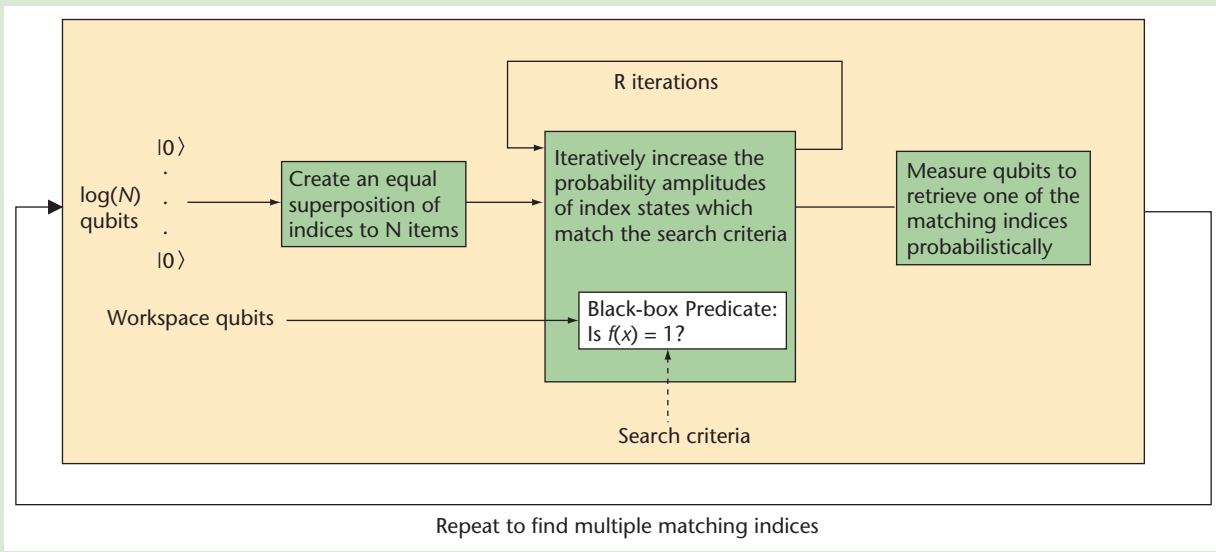
Figure A. A high-level circuit depiction of Grover's quantum search algorithm.

S with an asymptotic worst-case runtime, which is less than that of any classical algorithm $C(S)$ for S.

3. A $Q(S)$ with an actual runtime for practical instances of S, which is less than that of any $C(S)$.

Yet, we also show that real-life database applications rarely satisfy these requirements. We also describe a simulation methodology for evaluating potential speedups of quantum computation for specific instances of S. We demonstrate that search problems often contain a great deal of structure, which suggests several directions for future research.

Quantum Search

Grover's quantum algorithm[1,5,6] searches an unstructured database to find M records that satisfy a given criterion. (See the "Grover's Quantum Search Algorithm" sidebar for a more detailed description of this algorithm and its implementation.) For any N-element database, it takes $\sim \sqrt{N/M}$ evaluations of the search criterion (queries to an oracle) on database elements, whereas classical algorithms provably need at least $\sim N$ evaluations for some inputs.

To search a (large) database used in a particular application S, Grover's algorithm must be supplied with two different kinds of inputs that depend on S:

- the database, including its read-access mechanism, and
- the search criteria,

each of which is specified by a black-box predicate

tains bit strings, they're thought of as indices to the N items rather than as the items themselves.

The next step is to iteratively increase the probability amplitudes of those indices in the superposition that match the search criteria. The key component of each iteration is querying the black-box predicate. We can view the predicate abstractly as a function $f(x)$ that returns 1 if the index x matches the search criteria and 0 otherwise. Assuming that the predicate can be evaluated on the superposition of indices, a single query then evaluates the predicate on all indices simultaneously. In conjunction with extra gates and qubits (or workspace qubits), the indices for which $f(x) = 1$ can be marked by rotating their phases by π radians. To capitalize on this distinction, additional gates are applied to increase the probability amplitudes of marked indices and decrease the probability amplitudes of unmarked indices. Mathematically, this transformation is a form of inversion about the mean. It can be illustrated on sample input $\{-1/2, 1/2, 1/2, 1/2\}$ with mean 1/4, where inversion about mean produces $\{1, 0, 0, 0\}$. Both vectors in the example have norm 1. In general, the inversion about mean is a unitary transformation. One of Grover's insights was that it can be implemented with fairly small quantum circuits.

In the case in which only one element out of N satisfies the search criterion, each iteration of Grover's algorithm increases the amplitude of this state by approximately $O(1/\sqrt{N})$. Therefore, approximately $\sqrt{N}$ iterations are required to maximize the probability that a quantum measurement will yield the sought index (bit string). For the more general case with M elements satisfying the search criteria, the optimal number of iterations[4] is

$$R = \left\lceil \frac{\pi \arcsin\left(\sqrt{M/N}\right)}{4} \right\rceil .$$

Grover's algorithm also exhibits a periodic behavior, and after the amplitudes of sought elements peak, they start decreasing.

In the final step, quantum measurement is applied to each of the $\log(N)$ qubits, producing a single bit string. The larger a bit string's amplitude, the more likely the bit string is to be observed.

When M items match the search criteria in a particular search problem, then Grover's algorithm produces one of them. Each item is equally likely to appear because the inversion about the mean process increases the probability amplitudes of matching items equally. If all such items must be found, Grover's algorithm might have to be repeated more than M times, potentially returning some items more than once. On the other hand, classical deterministic search techniques avoid such duplication and might be more suitable in applications in which M is a significant fraction of N.

References

1. L.K. Grover, "Quantum Mechanics Helps in Searching for a Needle in a Haystack," *Physical Rev. Letters*, vol. 79, no. 2, 1997, pp. 325–328.

2. M.A. Nielsen and I.L. Chuang, *Quantum Computation and Quantum Information*, Cambridge Univ. Press, 2000.

3. G.F. Viamontes, I.L. Markov, and J.P. Hayes, "More Efficient Gate-Level Simulation of Quantum Circuits," *Quantum Information Processing*, vol. 2, no. 5, 2003, pp. 347–380.

4. M. Boyer et al., "Tight Bounds on Quantum Searching," *Fortschritte der Physik*, vol. 46, no. 4–5, 1998, pp. 493–505.

or oracle $p(x)$ that can be evaluated on any record x of the database. The algorithm then looks for an x such that $p(x) = 1$. In this context, x can be addressed by a k bit string, and the database can contain up to $N = 2^k$ records.

Classically, we can evaluate or query $p(\cdot)$ on one input at a time. In the quantum domain, however, if $p(\cdot)$ can be evaluated on either x or y, then it can also be evaluated on the superposition $(x + y)/\sqrt{2}$, with the result $(p(x) + p(y))/\sqrt{2}$. This quantum parallelism enables search with $\sqrt{N}$ queries.[5] If M elements satisfy the predicate, then $\sqrt{N/M}$ queries suffice.[6] The parallel evaluation of $p(\cdot)$ requires a superposition of multiple bit strings at the input, which can be achieved by starting in the $|00\ldots0\rangle$ state and applying the Hadamard gate H on every qubit. This, of course, requires that $p(\cdot)$ can interpret a bit string as a database record's in-

dex.

Researchers are aware of several variants of Grover's algorithm, including those based on quantum circuits and different forms of adiabatic evolution. However, as other work has discussed,[7] all are closely related to the original algorithm and have similar computational behavior.

For comparison, consider a classical deterministic algorithm for unstructured search, assuming that we aren't using parallel processing techniques.[3] It requires making many queries because an unsuccessful evaluation of $p(x)$ does not, in general, yield new information about records other than x. Therefore, we might need anywhere from 1 to N queries, depending on the input—$N/2$ on average. We must make queries until one of them is successful, and we can't take advantage of an unsuccessful query. Thus, every deterministic algorithm must visit the data-

base records one by one, in some order, and independently try up to 2^k database records until it finds a desired record. Randomized algorithms can pick records at random and have an edge over deterministic algorithms when many records satisfy $p(\cdot)$ because, for any input, approximately $N/(2M)$ queries suffice with very high probability. However, this improvement doesn't compare with the quadratic speedup Grover's algorithm offers.

Application Scalability

Although Grover's algorithm relies on quantum mechanics, it nevertheless solves a classical search problem and competes with advanced classical search techniques in existing and new applications. Existing applications include Web search engines, large databases for real-time processing of credit-card transactions, analysis of high-volume astronomic observations, and so forth. Such databases explicitly store numerous pieces of classical information (records). Another class of existing applications is illustrated by code breaking and Boolean satisfiability, where the input is a mathematical function $p(x)$ specified concisely by a formula, algorithm, or logic circuit. We seek the bits of x' such that $p(x') = 1$, which might represent a correct password or encryption key. The database of all possible values of x is implicit and doesn't require large amounts of memory.

Explicit databases in existing applications are often too large to fit in one computer's memory. They're distributed through the network and searched in parallel, and records can be quickly added, copied, and modified. Distributed storage also facilitates redundancy, backup, and crash recovery. The records in such databases correspond to physical objects (sensors, people, or Web pages), which tends to limit databases' typical growth rates. Explicit databases might temporarily experience exponential growth, as exemplified by the Web, but existing search infrastructures appear scalable enough for such applications, as Google's continuing success shows. Grover's algorithm, on the other hand, isn't well-suited to searching explicit databases of this kind because it demands a quantum superposition of all database records. Creating such a superposition, or using a superposition of indices in that capacity, seems to require localizing classical records in one place, which is impractical for the largest explicit databases.

Consequently, Grover's search algorithm seems confined to *implicit databases*, where it also faces serious competition from classical parallel methods.[3] This application class includes cryptographic problems, which are amenable to classical massively parallel computation. For instance, the DES Chal-
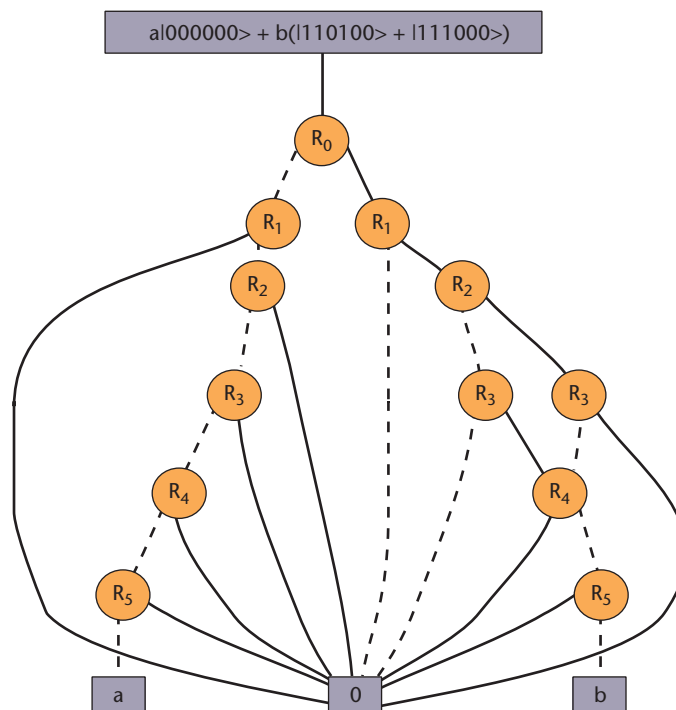

Figure 1. A 5-qubit state-vector in the QuIDD data structure. Each decision variable R_i corresponds to a bit i in the binary encoding of indices in the vector. Dashed lines model zeroes assigned to the index bit, and solid lines model ones. Top-down paths represent the 32 entries of the state-vector; a path might capture multiple entries with equal values.

lenge II decryption problem was solved in one day by a custom set of parallel processors built by the Electronic Frontier Foundation for less than $250,000 and nearly 100,000 PCs on distributed.net, an Internet-wide distributed computing project similar to SETI@home but geared toward code cracking. (See http://www.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/ and http://www.distributed.net for details.)

Implicit search applications typically exhibit exponential scalability—for example, adding an extra bit to an encryption key doubles the key space. This can't be matched in principle by the linear scalability of classical parallel processing techniques such as by adding hardware. Therefore, we believe that these applications meet Requirement 1 in the introduction and thus are potential candidates for practical quantum search tasks.

Oracle Implementation

Although the oracle function $p(\cdot)$ in Grover's algorithm can be evaluated on multiple inputs simultaneously, the description of $p(\cdot)$ is usually left un-

Easy to Verify, Hard to Solve

Many practical computational problems exist in science and engineering that are difficult to solve even on a fast computer. For some of these problems, the best-known algorithms that find an exact solution require super-polynomially many computational steps in the problem size. However, verifying whether a given candidate solution solves the problem might require up to polynomially many steps. Problems with this feature are called NP-complete[1] and include

- *Traveling salesman problem (TSP)*. Given a list of cities connected by roads and a gasoline budget of k dollars, a route must be determined, if one exists, so that a salesman visits each city only once, returns to the starting city, and spends k or fewer dollars on gasoline. There's a different monetary cost for gasoline associated with traveling between different cities. Indeed, we can formulate TSP as a graph traversal problem (see Figure B).
- *Graph 3-coloring*. The input to this problem is three different colors and an arbitrary graph. The goal is to find an assignment of colors to al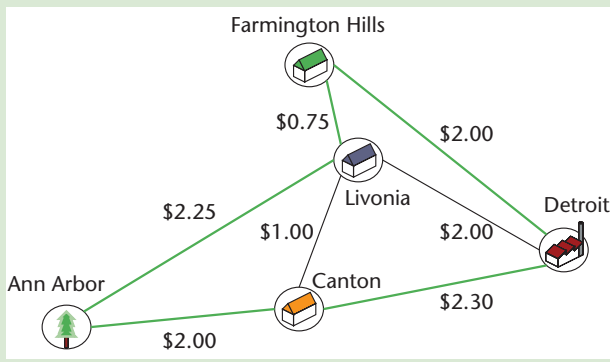l vertices, if one exists, so that no two vertices connected by an edge have the same color. This problem is NP-complete for graphs containing at least one vertex connected to four or more edges.[1]
- *Boolean 3-satisfiability (3-SAT)*. This problem requires finding an assignment of true/false values to variables of a Boolean function f in *conjunctive normal form* (CNF) such that f is true. In CNF, the function consists of the conjunction (logical AND) of clauses, and each clause is the disjunction (logical OR) of Boolean variables in complemented or uncomplemented form. In 3-SAT, each clause contains three variables, though the number of distinct variables and clauses might vary. To illustrate, given the CNF formula, $f = (a + b + c)(a + \bar{b} + d)(\bar{b} + \bar{c} + d)$, where + denotes disjunction and neighboring clauses imply conjunction, a solution is a, c, d = true; b = false.

Despite the difficulty of finding solutions to NP-complete problems, researchers have developed many heuristics and approximation algorithms exhibiting efficient performance in many useful cases.[2,3] Heuristics are generally algorithms that exhibit good empirical performance though their worst-case computational complexities might not be well understood. Approximation algorithms have well-defined computational complexities, but their solution quality and runtime might be worse than heuristics in specific cases. However, because we can map any NP-Complete problem to an instance of any other NP-complete problem with negligible overhead, flexibility exists in choosing different approaches.[1]

References

1. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 2000.
2. Z. Michalewicz and D.B. Fogel, *How to Solve It: Modern Heuristics*, Springer, 2000.
3. V.V. Vazirani, *Approximation Algorithms*, Springer, 2001.

Figure B. A TSP with Ann Arbor as the starting city. Given a gasoline budget of $9.30, a solution is in green.

specified.[1,5,6] To actually implement Grover's algorithm for a particular search problem, we must explicitly construct $p(\cdot)$. Several pitfalls are associated with this important step and are related to the complexity of $p(\cdot)$.

The first problem is that to query $p(\cdot)$ using quantum parallelism, we must implement $p(\cdot)$ in quantum hardware. This hardware can take various logical and physical forms.[1] If a quantum implementation of $p(\cdot)$ is derived from classical hardware design techniques, the classical and quantum implementations' circuit size might be similar. Circuit size is estimated by the number of logic operations (gates) used, and computation time by the circuit's maximum depth. However, if these numbers for an N-item database scale much worse than $\sqrt{N}$, then both classical and quantum searching will be dominated by the evaluation of $p(\cdot)$, diminishing the relative value of the quantum speedup on the log-scale.

A more subtle problem is the complexity of designing hardware implementations of $p(\cdot)$. Even if a given $p(\cdot)$ can theoretically be implemented without undermining the relative speedup of Grover's algorithm, there might not be a practical way to find compact classical or quantum implementations in a reasonable amount of time. In classical electronic design automation (EDA), finding small logic circuits is an enormously difficult computational and engineering task that requires synergies

between circuit designers and expensive design software. Automatic synthesis of small quantum circuits appears considerably harder because some formulations allow gates with a function that depends on continuous parameters,[8] rendering discrete methods irrelevant.

Although Grover's algorithm satisfies Requirement 2 in principle, satisfying it by a significant margin on the log-scale might be difficult in many cases because a small-circuit implementation of the oracle-function $p(\cdot)$ might not exist or might require an unreasonable effort to find.

Classical Simulation

As in the case of Grover's algorithm, quantum computation is often represented in the quantum-circuit formalism, which can be described mathematically using linear algebra and modeled on a classical computer.[1] Qubits are the fundamental units of information; k-qubit quantum states can be represented by 2^k-dimensional vectors, and square matrices of various sizes can represent gates. The parallel composition of gates corresponds to the tensor (Kronecker) product and serial composition to the ordinary matrix product. Like classical circuits, quantum circuits can be conveniently simulated by computer software for analysis or design purposes. A quantum circuit can be simulated naively by a sequence of $2^k \times 2^k$ matrices applied sequentially to a state vector. This reduces quantum simulation to standard linear-algebraic operations with exponentially sized matrices and vectors. Measurement is simulated similarly. Because Grover's algorithm only requires k qubits for a database of $N = 2^k$ records, a naive classical simulation of Grover's algorithm would be exponentially worse than an actual quantum circuit that implemented the algorithm.

The linear-algebraic formalism doesn't differentiate between structured and unstructured data. However, the state vectors and gate matrices that appear in typical quantum simulations are anything but unstructured. In particular, they compress well when simulated using the Quantum Information Decision Diagram (QuIDD) data structure.[9] A QuIDD is a directed acyclic graph with one source and multiple sinks, where each sink is labeled with a complex number. Matrix and vector elements are modeled by directed paths in the graph, as Figure 1 illustrates. Linear-algebraic operations can then be implemented by graph algorithms in terms of compressed data representations. Using data compression might substantially reduce simulation runtime for specific applications, especially those dealing with nonrandom data and circuits. This suggests a
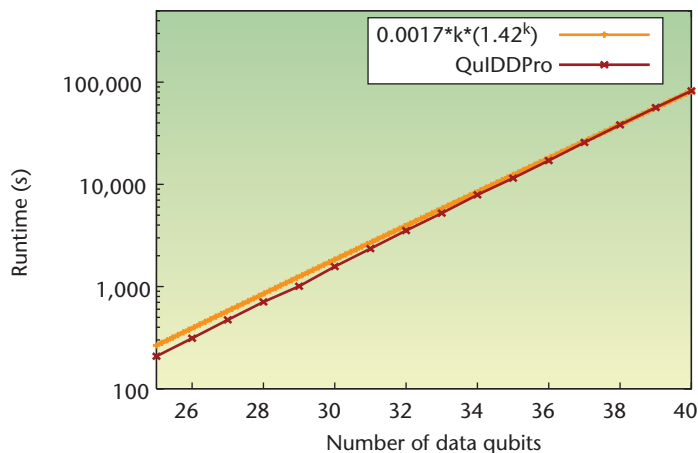

Figure 2. For *k* qubits, an actual quantum computer implementing Grover's algorithm must perform at least $ck(\sqrt{2})^k$ steps where *c* is a constant. For the type of predicates Grover originally considered,[5] the empirical runtime of QuIDDPro simulation fits well to this formula (plotted on a log-log scale), and the value of *c* is small.

test-by-simulation approach to identify violations of Requirement 3. Indeed, researchers have proposed polynomial-time simulation techniques for circuits with restricted gate types[10,11] and slightly entangled quantum computation.[12] However, they haven't applied these results to quantum search.

We've found that QuIDDs enable a classical computer to simulate a useful class of quantum circuits using time and memory that scale polynomially with the number of qubits.[9] All the components of Grover's algorithm, except for the application-dependent oracle, fall into this class. We've also proven that a QuIDD-based simulation of Grover's algorithm requires time and memory resources that are polynomial in the size of the oracle $p(\cdot)$ function represented as a QuIDD.[9] Thus, if a particular $p(\cdot)$ for some search problem can be represented as a QuIDD using polynomial time and memory resources (including conversion of an original specification into a QuIDD), then classical simulation of Grover's algorithm performs the search nearly as fast as an ideal quantum circuit. If a practical implementation of an oracle function $p(\cdot)$ is known, it's straightforward to represent it in QuIDD form because all relevant gate operations are defined for QuIDDs. Once $p(\cdot)$ is captured by a QuIDD, a QuIDD-based simulation requires $\sqrt{N/M}$ queries, just like an actual quantum computer.[9] If the size of the QuIDD for $p(\cdot)$ scales polynomially for k-qubit instances of the search problem, then Grover's algorithm offers no speedup for the given search problem.

We implemented a generic QuIDD-based sim-

ulator called QuIDDPro in the C++ programming language.[9] We can use this simulator in practice to perform the test by the simulation we just described. Figure 2 gives runtime results for QuIDDPro simulating Grover's algorithm for the search problem considered in Grover's seminal paper.[5] The oracle function for this search problem returns $p(x) = 1$ for one item in the database. In all such cases, the QuIDD for $p(\cdot)$ has only k nodes, and QuIDD-based simulation is empirically as fast as an actual quantum computer. Memory usage is only a few megabytes and grows linearly with the number of qubits.[9] This particular search problem fails to meet Requirement 3, so it doesn't benefit by being implemented on a quantum computer.

Predicates exist that don't compress well in QuIDD form, so they require super-polynomial time and memory resources. However, some of these predicates might also require a super-polynomial number of quantum gates. This might cause the evaluation of $p(x)$ to dominate the quantum search's runtime and undermine the speedup of Grover's algorithm over classical search.

Problem-Specific Algorithms

As we discussed earlier, comparisons between quantum and classical search algorithms often implicitly make strong assumptions—namely, the unrestricted use of black-box predicates and the misconception that no quantum circuit can be simulated efficiently on any inputs. These assumptions overestimate the potential speedup quantum search offers. Another, and perhaps more serious, oversight in popular analysis concerns the structure present in particular search problems. Therefore, we must compare Grover's algorithm against highly tuned classical algorithms specialized to a given search problem rather than against generic exhaustive search.

Boolean 3-satisfiability (3-SAT) and graph 3-coloring have been suggested as possible applications of quantum search because polynomial-time algorithms aren't known for these NP-complete problems and are unlikely to be found. (See the "Easy to Verify, Hard to Solve" sidebar for fuller definitions.) However, classical algorithms[13,14] exist that solve these two problems in less than $\sim poly(k)1.34^k$ and $\sim poly(k)1.37^k$ steps, respectively (k is the number of variables), thereby outperforming quantum techniques that require at least $\sim poly(k)1.41^k$ steps. (The algorithm Uwe Schöning[13] analyzed is a simplified version of the well-known WalkSAT program. It's a type of randomized local search in which variable assignments are changed one at a time to statistically decrease the number of unsat-

isfied clauses. Randomization in the algorithm facilitates hill climbing and an extremely fast move selection mechanism.) The algorithms that Uwe Schöning[13] and David Eppstein[14] describe exploit subtle structure in problem formulations, so they have the potential for further improvement. No such improvement is possible for Grover's algorithm unless additional assumptions are made.[15,7]

Of course, many practical NP-complete search problems have upper bounds that far exceed $\sim poly(k)1.41^k$—for example, k-satisfiability and k-coloring, with $k \geq 4$. However, known classical algorithms often finish much faster on certain inputs, application-derived and artificial, structured, or unstructured.[16] Indeed, we can now solve randomly generated hard-to-satisfy instances of Boolean satisfiability with a million variables in one day using a single-processor PC. Grover's algorithm is mainly sensitive to the number of solutions, but not to the solutions themselves or to the input features (such as symmetries) that classical algorithms sometimes exploit.

The Euclidean traveling salesman problem (TSP) and many other geometric optimization problems have defied fast, exact algorithms so far, but we can often solve them using polynomial-time approximation schemes that trade-off accuracy for runtime.[17] Their geometric structure lets us solve these problems to a given precision $\varepsilon > 0$ in polynomial time. Additionally, specific large instances of such problems have been solved optimally in the past—for example, the TSP for over 10,000 cities. Opportunistic algorithms and heuristics that work well only on some inputs are useful in practice, but comparable quantum heuristics are poorly understood.

Hard cryptography problems have also been mentioned as potential applications of Grover's algorithm.[3] They include code breaking (particularly, the Data Encryption Standard [DES] and Advanced Encryption Standard [AES] cryptosystems) and reversing cryptographically secure hash functions (Message Digest Algorithm [MD5] and Secure Hash Algorithm [SHA-1]). Indeed, these can be cast as unstructured search algorithms, but cryptographers have identified the structure in all such applications. In fact, using clever classical methods, cryptographers have recently cracked the hash functions MD5, MD4, HAVAL-128, and RIPEMD.[18] Other new classical attacks have also been developed that seriously challenge the security of SHA-0 and SHA-1.[19] The task of breaking DES has been reduced to Boolean 3-satisfiability[20] and clearly doesn't require a naive enumeration of keys. Essential algebraic structure[21] has been identified in AES (see http://csrc.nist.gov/Crypto

Toolkit/aes/aesfact.html), which has recently replaced DES as the US encryption standard.

Although quantum computing has dramatically advanced during the past decade, researchers have yet to demonstrate its potential applications at full scale. Such demonstrations are likely to require breakthroughs in physics, computer science, and engineering.[4] It's important to understand current roadblocks to achieving practical speedups with quantum algorithms. Our hope is that the work here will temper unreasonable expectations of quantum speedups and encourage further study of ways to improve quantum search.

In the near term, quantum computers running Grover's algorithm are unlikely to be competitive with the best classical computers in practical applications. Adding to the arguments of previous researchers[3] regarding classical parallelism, we've pointed out that recent work on solving problems such as Boolean satisfiability and direct simulation of quantum circuits offers opportunities to exploit subtle domain-specific structures, even on a single classical processor. Despite their exponential worst-case runtime, some of these algorithms are always faster than Grover's search.

Some interesting open research issues that deserve attention include

- search applications in which classical methods don't offer sufficient scalability;
- algorithms for near-optimal synthesis of quantum oracle circuits;
- quantum heuristics that finish faster or produce better solutions on practical inputs; and
- quantum algorithms that exploit the structure of useful search problems.

Recent work on variants of Grover's search accounts for problem structure and seems particularly promising. Building on earlier proof-of-concept results, Jérémie Roland and Nicolas Cerf[22] compare the fastest known classical algorithms for 3-satisfiability[13] to their quantum search algorithm cognizant of the 3-literal limitation. Their analysis shows that the quantum algorithm has a smaller asymptotic expected runtime, averaged over multiple satisfiability instances with a particular clause-to-variable ratio (at the phase-transition), which are known to be the most difficult to solve on average. Similar comparisons for worst-case asymptotic runtime remain an attractive goal for future research.

Several considerations in our work aren't restricted to Grover's search and apply to other potential applications of quantum computing. The *graph automorphism* problem, for example, seeks symmetries of a given graph and is sometimes suggested as a candidate for polynomial-time quantum algorithms.[4] Classically, this problem appears to require more than polynomial time in the worst case but is unlikely to be NP-complete, just like number factoring. However, graph automorphism is provably easy for random graphs and can also be solved quickly in many practical cases—for example, in the context of microprocessor verification[23]—making existing algorithms and software strong competitors of potential future quantum algorithms.

Daniel Curtis and David Meyer have proposed a novel quantum algorithm application, finding template matches in photographs,[24] but it relies on the quantum Fourier transform rather than Grover's search. Proposals of this kind also deserve careful evaluation from the computer science and engineering perspective and must be compared to state-of-the-art classical methods on realistic inputs. **CiSE**

References

1. M.A. Nielsen and I.L. Chuang, *Quantum Computation and Quantum Information*, Cambridge Univ. Press 2000.

2. C. Bennett et al., "Strengths and Weaknesses of Quantum Computing," *SIAM J. Computing*, vol. 26, no. 5, 1997, pp. 1510–1523.

3. C. Zalka, "Using Grover's Quantum Algorithm for Searching Actual Databases," *Physical Rev. A*, vol. 62, no. 5, 2000, pp. 052305-1–052305-4.

4. J. Preskill, "Quantum Computing: Pro and Con," *Proc. Royal Soc. London A*, vol. 454, no. 1969, 1998, pp. 469–486.

5. L.K. Grover, "Quantum Mechanics Helps in Searching for a Needle in a Haystack," *Physical Rev. Letters*, vol. 79, no. 2, 1997, pp. 325–328.

6. M. Boyer et al., "Tight Bounds on Quantum Searching," *Fortschritte der Physik*, vol. 46, no. 4–5, 1998, pp. 493–505.

7. J. Roland and N.J. Cerf, "Quantum-Circuit Model of Hamiltonian Search Algorithms," *Physical Rev. A*, vol. 68, no. 6, 2003, pp. 062311-1–062311-6.

8. V.V. Shende, I.L. Markov, and S.S. Bullock, "Smaller Two-Qubit Circuits for Quantum Communication and Computation," *Proc. Design Automation and Test in Europe* (DATE), CS Press, 2004, pp. 980–985.

9. G.F. Viamontes, I.L. Markov, and J.P. Hayes, "More Efficient Gate-Level Simulation of Quantum Circuits," *Quantum Information Processing*, vol. 2, no. 5, 2003, pp. 347–380.

10. S. Aaronson and D. Gottesman, "Improved Simulation of Stabi-

lizer Circuits," *Physical Rev. A*, vol. 70, no. 5, 2004, pp. 052328-1–052328-14.

11. L.G. Valiant, "Quantum Circuits That Can Be Simulated Classically in Polynomial Time," *SIAM J. Computing*, vol. 31, no. 4, 2002, pp. 1229–1254.

12. G. Vidal, "Efficient Classical Simulation of Slightly Entangled Quantum Computations," *Physical Rev. Letters*, vol. 91, no. 14, 2003, pp. 147902-1– 147902-4.

13. U. Schöning, "A Probabilistic Algorithm for k-SAT and Constraint Satisfaction Problems," *Proc. IEEE Symp. Foundation of Comp. Science*, ACM Press, 1999, p. 410.

14. D. Eppstein, "Improved Algorithms for 3-Coloring, 3-Edge-Coloring, and Constraint Satisfaction," *Proc. Symp. Discrete Algorithms*, ACM/SIAM Press, 2001, pp. 329–337.

15. C. Zalka, "Grover's Quantum Searching Algorithm Is Optimal," *Physical Rev. A*, vol. 60, no. 4, 1999, pp. 2746–2751.

16. H. Kautz and B. Selman, "Ten Challenges Redux: Recent Progress in Propositional Reasoning and Search," *Proc. Principles and Practice of Constraint Programming* (CP), LNCS 2833, Springer, 2003, pp. 1–18.

17. S. Arora, "Polynomial-Time Approximation Schemes for Euclidean Traveling Salesman and Other Geometric Problems," *J. ACM*, vol. 45, no. 5, 1998, pp. 753–782.

18. X. Wang et al., "Collisions for Hash Functions MD4, MD5, HAVAL-128, and RIPEMD," Cryptology ePrint Archive, report no. 199, Aug. 2004, pp. 1–4.

19. E. Biham and R. Chen, "Near-Collisions of SHA-0," *Proc. Int'l Cryptology Conf.* (CRYPTO), LNCS 3152, Springer, 2004, pp. 290–305.

20. L. Marraro and F. Massacci, "Towards the Formal Verification of Ciphers: Logical Cryptanalysis of DES," *Proc. Workshop on Formal Methods and Security Protocols* (FMSP), 1999.

21. S.P. Murphy and M.J.B. Robshaw, "Essential Algebraic Structure within the AES," *Proc. Int'l Cryptology Conf.* (CRYPTO), LNCS 2442, Springer, 2002, pp. 1–16.

22. J. Roland and N.J. Cerf, "Adiabatic Quantum Search Algorithm for Structured Problems," *Physical Rev. A*, vol. 68, no. 6, 2003, pp. 062312-1–062312-7.

23. P.T. Darga et al., "Exploiting Structure in Symmetry Generation for CNF," *Proc. Design Automation Conf.* (DAC), ACM Press, 2004, pp. 530–534.

24. D. Curtis and D. A. Meyer, "Towards Quantum Template Matching," *Proc. SPIE*, vol. 5161 (Quantum Communications and Quantum Imaging), SPIE Press, 2004, pp. 134–141.

George F. Viamontes is a PhD candidate in the Electrical Engineering and Computer Science Department at the University of Michigan. He is working to develop algorithms that exploit implicit problem structure to simulate quantum mechanical phenomena efficiently. He has an MS in computer science and engineering from the University of Michigan. He is a member of the IEEE and ACM.

Igor L. Markov is an assistant professor in the Electrical Engineering and Computer Science Department at the University of Michigan. His interests are in quantum computing and combinatorial optimization applied to the design and verification of integrated circuits. He received his PhD in computer science from the University of California, Los Angeles. He is an associate editor of the ACM SIGDA newsletter. He also received the 2004 IEEE CAS Donald O. Pederson paper-of-the-year award and the ACM SIGDA Outstanding New Faculty Award. He is a member of the IEEE and ACM.

John P. Hayes is a professor in the Electrical Engineering and Computer Science Department at the University of Michigan, where he holds the Claude E. Shannon Chair of Engineering Science. His research interests include computer hardware design, quantum computing; computer-aided design, testing, and verification of digital systems; VLSI design; and fault-tolerant and embedded computer systems. He is the author of *Computer Architecture and Organization 3rd ed.* (McGraw-Hill, 1998), *Layout Minimization for CMOS Cells* (Kluwer, 1992), and *Introduction to Digital Logic Design* (Addison-Wesley, 1993). He received the University of Michigan's Distinguished Faculty Award in 1999 and the Humboldt Research Award in 2004. He received his PhD from the University of Illinois, Urbana-Champaign. He is a fellow of the IEEE and ACM.