from *SIAM News*, Volume 31, Number 8

Quantum Computation: A New Spin On Complexity Theory

By Barry A. Cipra

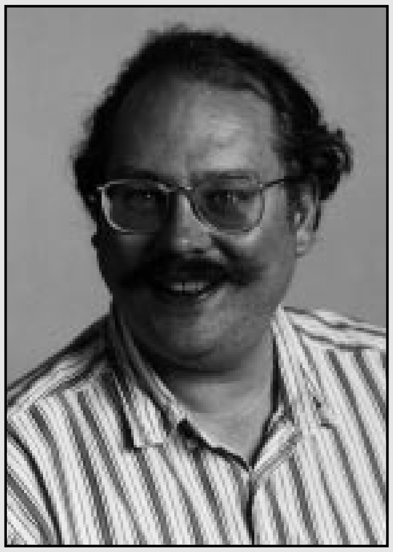
"Anyone who can contemplate quantum mechanics without getting dizzy hasn't properly understood it."

The pronouncement of Niels Bohr, one of the pioneers of quantum mechanics, still rings true. Despite decades of detailed development, the theory that underpins modern physics continues to conflict with common sense. While many have "mastered" quantum mechanics, few claim to understand it.

Could computer science change all that? Some researchers believe that a budding subject called quantum computation will finally force physicists to make sense of the unreal rules that seem to govern how the world really works.

At the same time, quantum computation could change the face of computer science. It's already clear that the principles of quantum mechanics can radically reduce the apparent complexity of certain problems, including one—the factorization of large numbers—that is currently used for cryptographic purposes. Theorists are scrambling to extend the notion of computer power to include the potential of these as-yet-unrealized machines.

Or make that *proto*-realized. Several groups have built rudimentary quantum computers. So far the most complicated task carried out has been an "efficient" search for one item in a four-item "database"—the computational equivalent of finding a needle in a three-straw haystack. Everyone agrees that substantial technological hurdles lie ahead, and some skeptics doubt that they'll all be cleared. But the prize is tantalizing, and the race alone should be informative.



Peter Shor of AT&T Laboratories received the 1998 Nevanlinna Award at the International Congress of Mathematicians, held in Berlin in August. Given every four years in recognition of outstanding work in computer science done by a researcher under the age of 40, the prestigious award recognizes Shor's work in quantum computing.

Before Shor's 1994 demonstration that "quantum computation is "tailor-made for factoring large numbers," Barry Cipra points out in the accompanying article, "there were no real applications on the horizon." Shor's algorithm, in Cipra's words, "reduces factoring to a polynomial-time snap." It was also Shor who, in 1995, showed that error correction is possible on a quantum machine.

"The theory of quantum computing," according to Caltech theoretical physicist John Preskill, "passed four major milestones in just a little over two years: exponential speedups, quantum error correction, good codes, and fault tolerance. And Shor led the way to every one of these developments."

Schrödinger's Calculating Cat

What *is* quantum computation? Roughly speaking, it is a way of exploiting the weird, but well-defined rules of quantum mechanics to carry out "impossible" calculations—impossible for classical computers, that is.

Conceptually, classical computers operate within the literal sense of the word "calculate": The computers move little "stones" around to count and keep track of things. The stones, of course, have gotten electronically small, and the procedures for moving them are extremely fast and complex, but none of that matters. What's important is that at each stage of a calculation, the stones are in a definite arrangement; pausing between steps, you can look at the stones, reposition any that you notice have gotten out of place, and, if you like, duplicate portions of the arrangement.

In a quantum computer, by contrast, the stones are usually *not* in a definite arrangement. Instead, at each step, they exist (if that's the right word) in a linear combination of many, many arrangements. The computation itself is a sequence of unitary transformations—i.e., rotations—within a high-dimensional vector space spanned by the classical arrangements. In effect, a quantum computer carries out a truly massively parallel computation on *all possible* inputs. But there's a catch: You can't peek at a quantum computation, because looking at quantum stones causes them to "collapse" into a definite arrangement—a decidedly nonunitary transformation.

A quantum bit, or qubit, is thus more than just a 0 or a 1; it is a *superposition* of the two possibilities. In general, a qubit has the form $a|0\rangle + b|1\rangle$, where a and b are complex numbers with $|a|^2 + |b|^2 = 1$. Whereas a classical two-bit state is one of the four combinations 00, 01, 10, and 11, a two-qubit state is a four-dimensional complex vector, $a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$, with $|a|^2 + |b|^2 + |c|^2 + |d|^2 = 1$. The basis vectors $|00\rangle, |01\rangle, |10\rangle,$ and $|11\rangle$ can be thought of as the observable states of the system—indicating whether, for example, the spins of two atoms in a molecule are aligned with or against a magnetic field. Similarly, an N-qubit state is a 2^N-dimensional vector, with basis vectors $|00\ldots0\rangle$ through $|11\ldots1\rangle$.

If you measure the spins of the aforementioned two atoms when the system is in the

How Complex Can You Get?

In classical computing, a decision problem is considered tractable if there is a polynomial-time probabilistic algorithm that provides the correct Yes/No answer at least, say, two-thirds of the time, on all inputs. (The 2/3 is somewhat arbitrary: As long as the probability of error is bounded away from 1/2, it is possible to boost confidence in the answer arbitrarily close to 1 by repeating the calculation a relatively small number of times.) Such problems belong to the class BPP: Bounded-error, Probabilistic, Polynomial-time.

In 1993, Ethan Bernstein and Umesh Vazirani of the University of California at Berkeley showed that the quantum analogue, BQP, fits between BPP and the class called P-SPACE, which, roughly speaking, consists of problems that can be solved with polynomial amounts of reusable memory (see Figure 1). However, the question "P = P-SPACE?" has the same status as its famous cousin, "P = NP?"—neither has been answered.

BQP picks up all the NP problems in BPP, and maybe a few more, such as factoring. But researchers have yet to find a quantum-mechanical algorithm for any of the NP-complete problems. It's not that a quantum computer couldn't calculate all $N!$ itineraries for a traveling salesman in polynomial time. That would be easy. The trick would be getting the machine to tell you the answer.

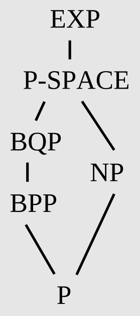


Figure 1. *Complexity Classes Made Simple. Problems that can be solved in polynomial time reside at the bottom of a hierarchy of complexity classes, with problems requiring exponential time sitting at the top. The potential power of quantum computers floats somewhere in between. The class NP, for now, sits off to the side.*

state $a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$, what you see is either with–with, with–against, against–with, or against–against, with probabilities $|a|^2, |b|^2, |c|^2, |d|^2$, respectively. This is the mysterious "collapse of the wave packet": After the measurement, the system is no longer in a superposition, but in one of the four classical states.

The weirdness of quantum mechanics resides in states that are impossible classically. In terms of spin, for example, the state $(|00\rangle + |11\rangle)/\sqrt{2}$ says that two atoms are simultaneously both aligned *with* a magnetic field and both aligned *against* the field. If you measure the spins, you'll see either both with or both against, with a 50:50 probability of each result. What's weird is what happens if you measure just *one* spin: Whichever way it collapses (with or against), it takes the other spin along, even if the two atoms are far apart. The two spins are "entangled" in a way that contradicts common sense.

Over the decades, physicists have explored the creation of entangled quantum states—by subjecting molecules in a magnetic field to pulses of radio-frequency fields, for example, or by shining carefully chosen frequencies of laser light on interacting ions. But the prospect of quantum computation, of being able to handle larger systems with greater precision, is driving researchers to refine their techniques. It's also led scientists to think deeply about the implications of quantum mechanics, in order to take full, algorithmic advantage of nature's operating system.

The Collapse Factor

The essence of quantum computation, then, is to "rotate" some initial vector in such a way that, when observed, the system collapses with high probability into a string of 0's and 1's that can be interpreted as the (correct) answer to a particular problem. On the practical side, the unitary transformations that carry out the rotations are limited to acting on one or two qubits at a time—it's hard to coordinate a simultaneous action on three quantum-mechanical objects. Fortunately, theorists have shown that every unitary transformation can be realized by a sequence of two-qubit entanglements.

The precise meaning of the "high probability" of getting a correct answer depends on what you're asking for. If the result can easily be checked for accuracy—say you're factoring a large number—then essentially *any* probability bounded away from 0 is good enough, since the calculation can be repeated until it produces the right answer.

Factoring, in fact, was the breakthrough that brought quantum computation to the fore. A small number of researchers in the 1980s, including Paul Benioff of Argonne National Laboratory, David Deutsch of Oxford University, and Seth Lloyd of MIT, had laid the conceptual foundations for quantum computing. In a 1982 paper, Nobel laureate Richard Feynman had proposed quantum computation as a way to beat the exponential complexity that appears to stymie any attempt to simulate a large quantum-mechanical system. In the early 90s, Ethan Bernstein, now at Microsoft, and Umesh Vazirani of the University of California at Berkeley had positioned quantum computing within the hierarchy of complexity theory (see sidebar). But there were no real applications on the horizon; no one had found a way to capitalize on the potential parallelism of quantum mechanics.

Then, in 1994, Peter Shor, a mathematician at AT&T Bell Laboratories (now AT&T Laboratories), showed that quantum computation is tailor-made for factoring large numbers. Factorization is a notoriously difficult problem for classical computers: The best algorithms known today still require an amount of computation that grows exponentially with the number of digits in the number to be factored. Shor's algorithm reduces factoring to a polynomial-time snap.

The output of Shor's quantum algorithm is not directly a factor of an n-digit number N, but rather the *period* mod N of a randomly chosen number x—i.e., the smallest number r such that $x^r \equiv 1 \bmod N$. Well-known (classical) results in number theory assert that at least half the time, r will be even and $x^{r/2} - 1$ will contain a factor of N, which is then easily found with the Euclidean algorithm (which rapidly computes the greatest common divisor of two numbers).

More precisely, the quantum computation returns a random multiple of the inverse of the period. Repeating the computation a relatively small number of times will, with high probability, reveal the inverse period itself.

The actual quantum calculation is done in three steps. First, a vector is prepared in the superposition

$$\frac{1}{\sqrt{q}}\sum_{k=1}^{q}|k,0\rangle,$$

where q is a number with roughly twice as many digits as N and $|k,0\rangle$ is shorthand for the basis vector of 1's and 0's with k written base 2 followed by as many 0's as there are bits (i.e., base-2 digits) in N. The key number-theoretic step is a unitary transformation that rotates each basis vector of the form $|k,0\rangle$ to $|k, f(k)\rangle$, with $f(k) = x^k$ mod N. The final step is simply a Fourier transform, taking $|k,-\rangle$ to

$$\frac{1}{\sqrt{q}}\sum_{h=1}^{q}e^{2\pi ihk/q}|h,-\rangle.$$

This rotates the initial vector into the state

$$\frac{1}{q}\sum_{h=1}^{q}\sum_{k=1}^{q}e^{2\pi ihk/q}|h, f(k)\rangle.$$

The key is to think of each k in the inner sum as a multiple of r plus a remainder. This means the expression for the state can be rewritten as

$$\frac{1}{q}\sum_{h-1}^{q}\sum_{b=1}^{r}\sum_{a=0}^{[(q-b)/r]}e^{2\pi ih(ar+b)/q}|h, f(b)\rangle.$$

The innermost sum,

$$\sum_{a=0}^{[(q-b)/r]}e^{2\pi iahr/q},$$

tends to be large (in absolute value) when hr/q is close to an integer; otherwise, it tends to be comparatively small. But the probability of observing the system in a state corresponding to h is the square of this absolute value (divided by q^2). Therefore, if you take an observed value of h and round the fraction h/q to the nearest fraction with denominator less than N, the result is likely to be a multiple of $1/r$. (The devil, of course, is in the details, which can be found in Shor's paper, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, published last year in *SIAM Journal on Computing*; an update is scheduled to appear in the SIGEST section of *SIAM Review* in June 1999.)

Quantum Search

In part because the difficulty of factoring underlies the popular RSA crypto-system (see *SIAM News*, July 1994, "The Magic Words Are Squeamish Ossifrage"), Shor's algorithm was hailed as quantum computation's "killer app": an application that justifies the costs of a new technology. However, it's not known for sure that factoring is exponentially difficult for classical computers. Much of the enthusiasm over Shor's quantum-mechanical breakthrough would likely evaporate were someone to find an equally fast classical algorithm (which, of course, would be astounding news).

More recently, Lov Grover, a computer scientist at Bell Laboratories, Lucent Technologies, has come up with another potential killer app, one that can't be outdone by a classical algorithm. Grover has found a way to locate a particular item in an N-item database with only $O(\sqrt{N})$ calculations.

That kind of efficiency is flat-out impossible for a classical computer. Given an "is-it-here?" function $f(k)$, which is guaranteed to be nonzero at precisely one value of k, between 1 and N, but which is otherwise arbitrary, classical computation has no recourse but to compute $f(k)$ for one value of k after another, so that the average search requires $N/2$ calculations. (If more is known about the function f, of course, the search can be made much more efficiently, but adding an "analyze f" subroutine to the algorithm is just asking for trouble.) It would seem to be obvious you can't do better than $N/2$. But quantum computers thrive on the nonobvious.

Grover's algorithm takes advantage of the linear-algebraic structure of quantum mechanics to achieve what he calls "amplitude amplification." The trick is to analyze how a certain pair of unitary transformations act on the two-dimensional subspace spanned by the "uniform" vector

$$u=\frac{1}{\sqrt{N}}\sum_{h=1}^{N}|h\rangle,$$

which is the beginning state for the computation, and the "target" vector $v=|t\rangle$, where t is the (unique) value for which $f(t)$ is nonzero.

The unitary transformations U_1 and U_2 are defined by their action on the standard basis vectors:

$$U_1|h\rangle = |h\rangle - \frac{2}{N}\sum_{k=1}^{N}|k\rangle = |h\rangle - \frac{2}{\sqrt{N}}u$$

and

$$U_2|h\rangle = \begin{cases} |h\rangle & \text{if } f(h)=0; \\ -|h\rangle & \text{if } f(h)\neq 0. \end{cases}$$

Grover shows that each transformation is an "easy" quantum computation: U_1 can be implemented by three elementary quantum operations, while U_2 can be implemented by a quantum circuit that does a single evaluation of $f(h)$. (Each transformation is obviously symmetric and easily seen to be its own inverse; hence, each is unitary.)

It is easy to see that the subspace spanned by u and v is invariant under both transformations. In fact, $U_1 u = -u$ while $U_1 v = v - \frac{2}{\sqrt{N}}u$, and $U_2 u = u - \frac{2}{\sqrt{N}}v$ while $U_2 v = -v$.

What's interesting is the 2×2 matrix corresponding to the *product* $U_1\, U_2$, written in terms of the *orthonormal* basis $e_1 = v$ and $e_2 = \left(\sqrt{N}u - v\right)\big/\sqrt{N-1}$:

$$U = \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix},$$

where $\cos\phi = 1 - 2/N$. Each application of the transformation U_1U_2 is, in effect, a glance at the database. The question is: After, say, k such glances, where is the state vector pointing—how much points toward $|t\rangle$ and how much points in the other random directions comprising u?

A straightforward bit of trigonometry reveals that, starting from the uniform state, written now as $u = \left(e_1 + \sqrt{N-1}e_2\right)\big/\sqrt{N}$, the e_1-coefficient after k iterations is $\sin\left(\left(k+\tfrac{1}{2}\right)\phi\right)$, which is clearly close to 1 when $\left(k+\tfrac{1}{2}\right)\phi \approx \pi/2$. Meanwhile, the e_2-coefficient, $\cos\left(\left(k+\tfrac{1}{2}\right)\phi\right)$, nearly vanishes. But because $\cos\phi = 1 - 2/N$ implies that $\phi \approx 2/\sqrt{N}$ (for large N), $k \approx (\pi/4)\sqrt{N}$ does the trick. Picking the "right" value for k—that is, knowing when to stop—is important: If you look at the database too many times, you can actually *reduce* the chance of finding what you're looking for!

Can life get any better than this? Apparently not: Christof Zalka, a theoretical physicist at Los Alamos National Laboratory, has shown that Grover's algorithm is optimal. Asymptotically speaking, you're never going to beat $(\pi/4)\sqrt{N}$.

The (decidedly nonasymptotic) case $N = 4$, which corresponds to just two qubits, is particularly interesting: $\cos\phi = 1/2$ implies that $\phi = \pi/3$, from which it follows that $\sin\left(\left(1+\tfrac{1}{2}\right)\phi\right) = 1$. Thus, the first application of U rotates u exactly to $|t\rangle$. In other words, if you've misplaced your glasses in one of four locations, all it takes to find them is a *single* quantum-mechanical glance.

Decoding Decoherence

That computation has actually been done. Earlier this year, Neil Gershenfeld of MIT and Isaac Chuang of the IBM Almaden Research Center in San Jose, California, built a tiny, two-qubit quantum computer and ran Grover's algorithm. Their approach, which they describe in an article in *Scientific American* (June 1998), uses principles of nuclear magnetic resonance (NMR) to manipulate the quantum state of molecules in a liquid sample. Gershenfeld likes to call this approach "computing in a coffee cup" (although the "coffee" they used was chloroform, a rather extreme form of decaf).

Compared with the capabilities of even the earliest classical computers, however, searching a four-item database is rather unimpressive. What's the holdup with quantum computers?

The problem is known as decoherence. For quantum computation to work, the state vector has to point extremely close to its ideal direction. In particular, while two qubits are being entangled in one step, the rest of the qubits need to stay put. A quantum computer is a little like a daycare worker who can concentrate on only one or two toddlers at a time.

The no-peeking clause in nature's quantum-mechanical contract makes matters even more difficult. Classical computers have their own imperfections, which can ruin a calculation, but researchers have perfected error-correcting techniques that effectively eliminate the mistakes these machines tend to make. However, classical error-correcting codes rely on the sensible assumption that it's OK to look at what you've got and make copies of it. The archetypal error-correcting routine is majority vote: Record a bit of information three times and use the value that occurs at least twice. But quantum mechanics doesn't allow the cloning of qubits. If you look at a qubit, you see either a 0 or a 1; either way, you've destroyed the superposition.

A "true" duplication of a quantum state $a|0\rangle + b|1\rangle$ is the two-qubit state $a^2|00\rangle + ab|01\rangle + ab|10\rangle + b^2|11\rangle$—if one qubit is observed, the other collapses back to the original state. It's certainly possible to rotate, say, $a|00\rangle + b|10\rangle$ into this state for any specific choice of a and b. But there's no single, unitary transformation that will do so for *all* choices of a and b. In other words, you can duplicate quantum information by redoing (on a separate machine) the computation that produced it, but you can't duplicate a quantum state you know nothing about.

This would seem to prove that error correction can't be done in a quantum computer. But once again, intuition is outwitted by quantum mechanics. Error correction *is* possible in a quantum machine. In fact, much of the classical theory of error-correcting codes seems tailor-made for quantum computation.

The breakthrough again came from Shor, who discovered a nine-qubit analogue of the three-bit, majority-vote approach. Shor's algorithm begins by converting the original into three triples of qubits. If the original qubit is a $|0\rangle$, each triple has the form $\left(|000\rangle+|111\rangle\right)/\sqrt{2}$; if it's a $|1\rangle$, each triple is $\left(|000\rangle-|111\rangle\right)/\sqrt{2}$. In effect, the quantum information has been converted from a 0/1 to a +/– form, and the "standard" basis $|000\rangle$ through $|111\rangle$ has been replaced by the basis $|000\rangle\pm|111\rangle$, $|100\rangle\pm|011\rangle, |010\rangle\pm|101\rangle, |001\rangle\pm|110\rangle$.

Shor then assumes that one of the nine qubits decoheres. This means that it interacts with an unknown outside environment $|e\rangle$ in an unknown, but quantum-mechanical way, i.e., $|e\rangle|0\rangle$ goes to some state $|a\rangle|0\rangle+|b\rangle|1\rangle$ and $|e\rangle|1\rangle$ goes to $|c\rangle|0\rangle+|d\rangle|1\rangle$, where $|a\rangle, |b\rangle, |c\rangle,$ and $|d\rangle$ are further states of the environment. The upshot is that for this qubit's triple (with, say, the error occurring in the first qubit), $|e\rangle\left(|000\rangle\pm|111\rangle\right)$ goes to

$$\begin{aligned} &\left(|a\rangle+|d\rangle\right)\left(|000\rangle\pm|111\rangle\right)\\ +\ &\left(|a\rangle-|d\rangle\right)\left(|000\rangle\mp|111\rangle\right)\\ +\ &\left(|b\rangle+|c\rangle\right)\left(|100\rangle\pm|011\rangle\right)\\ +\ &\left(|b\rangle-|c\rangle\right)\left(|100\rangle\mp|011\rangle\right). \end{aligned}$$

What's crucial is that the information-bearing +/– sign stick strictly to the description of the triple. Therefore, even though the environment is in an unknown state, we know it's in the *same* unknown state regardless of the sign. We can repair a damaged triple, then, by applying a unitary transformation that takes, for example, the vector $\left(|000\rangle+|111\rangle\right)\left(|100\rangle-|011\rangle\right)\left(|000\rangle+|111\rangle\right)\quad|00000\ldots\rangle$ to $\left(|000\rangle+|111\rangle\right)\left(|000\rangle+|111\rangle\right)\quad\left(|000\rangle+|111\rangle\right)|garbage\rangle$. The "garbage" bits provide a record of the error, where it occurred and whether it was a bit flip, a sign mistake, or (as in the example) both, but that's all.

Shor's demonstration, which appeared in *Physical Review A* in 1995, corrected the notion that quantum error correction couldn't be done. Within months of the announcement, the subject of quantum error-correcting codes exploded, with almost an overabundance of quantum codes. Shor and Rob Calderbank, also of AT&T Labs, and, independently, Andrew Steane of Oxford University found a family of classical error-correcting codes that have quantum counterparts. In a 1997 PhD dissertation at Caltech, Daniel Gottesman, a physicist now at Los Alamos National Laboratory, developed a mathematical formalism for quantum error correction, further expanding the classical-to-quantum conversion.

Quantum Doubts

Despite the rapid progress of the last few years, the future of quantum computing remains uncertain. The difficulty of building devices that sustain and manipulate coherent quantum states has limited researchers to "computers" with just a handful of qubits. The NMR approach, for example, has an "output" problem: As you try to increase the number of qubits by working with larger molecules, the ratio of noise to signal also increases, until you've essentially got nothing but noise.

Indeed, the difficulties may be insurmountable. That's the position taken by Rolf Landauer, a physicist at IBM's T.J. Watson Research Center in Yorktown Heights, New York. The vision of a large quantum system adhering to a carefully orchestrated score—even one that corrects for the occasional wrong note of, say, an errant oboe—may make for a lovely theory, he argues, but it ignores aspects of quantum mechanics that limit the precision with which a quantum system can be controlled.

Landauer should know: He pioneered the study of physical limits to computation, and his work has been fundamental to what progress has been made in quantum computing. His objections, quantum enthusiasts say, have helped clarify the pitfalls into which quantum computation is prone to fall. They're just hoping they can prove him wrong.

But what if quantum mechanics itself is wrong? In particular, quantum computation is based on the linearity of quantum mechanics. What if physicists discover that quantum mechanics is only an approximation to a deeper, *nonlinear* theory?

Shockingly, that might be the best news of all. Seth Lloyd and Daniel Abrams, also of MIT, have shown that if small nonlinearities occur during the evolution of a quantum state in time—and judging by the success quantum mechanics has enjoyed so far, any nonlinearity must be small—they can be exploited in the form of nonlinear quantum logic gates to solve NP-complete problems in (quantum) polynomial time. In other words, what seems like the most important open problem in theoretical computer science could be rendered moot by an advance in physics. Of course, the problem of building a quantum NP-complete problem cruncher would still loom large, and Landauer's objections might seem optimistic by comparison. But talk about your killer apps!

Barry A. Cipra is a mathematician and writer based in Northfield, Minnesota.