THE LIMITS OF Quantum

By Scott Aaronson

Quantum computers would be exceptionally fast at a few specific tasks, but it appears that for most problems they would outclass today's computers only modestly. This realization may lead to a new fundamental physical principle

"Haggar Physicists Develop 'Quantum Slacks,' " read a headline in the satirical weekly the *Onion*. By exploiting a bizarre "Schrödinger's Pants" duality, the article explained, these non-Newtonian pants could paradoxically behave like formal wear and casual wear at the same time. *Onion* writers were apparently spoofing the breathless articles about quantum computing that have filled the popular science press for a decade.

A common mistake—see for instance the February 15, 2007, issue of the *Economist*—is to claim that, in principle, quantum computers could rapidly solve a particularly difficult set of mathematical challenges called NP-complete problems, which even the best existing computers cannot solve quickly (so far as anyone knows). Quantum computers would supposedly achieve this feat not by being formal and casual at the same time but by having hardware capable of processing every possible answer simultaneously.

If we really could build a magic computer capable of solving an NP-complete problem in a snap, the world would be a very different place: we could ask our magic computer to look for whatever patterns might exist in stock-market data or in recordings of the weather or brain activity. Unlike with today's computers, finding these patterns would be completely routine and require no detailed understanding of the subject of the problem. The magic computer could also automate mathematical creativ-

ILLUSTRATIONS BY DUŠAN PETRIČIĆ

ity. Given any holy grail of mathematics—such as Goldbach's conjecture or the Riemann hypothesis, both of which have resisted resolution for well over a century—we could simply ask our computer to search through all possible proofs and disproofs containing up to, say, a billion symbols. (If a proof were much longer than that, it is not clear that we would even want to read it.)

If quantum computers promised such godlike mathematical powers, maybe we should expect them on store shelves at about the same time as warp-drive generators and antigravity shields. But although we should not accept the usual hype, in my view it is equally misguided to dismiss quantum computing as science fiction. Instead we should find out what the limits of quantum computers are and what they could really do if we had them.

In the 26 years since physicist Richard Feynman first proposed the idea of quantum computing, computer scientists have made enormous progress in figuring out what problems quantum computers would be good for. According to our current understanding, they *would* provide dramatic speedups for a few specific problems—such as breaking the cryptographic codes that are widely used for monetary transactions on the Internet. For other problems, however—such as playing chess, scheduling airline flights and proving theorems—evidence now strongly suggests that quantum computers would suffer from many of the same algorithmic limitations as today's classical computers. These limitations are completely separate from the practical difficulties of building quantum computers, such as decoherence (unwanted interaction between a quantum computer and its environment, which introduces errors). In particular, the bounds on what it is mathematically possible to program a computer to do would apply even if physicists managed to build a quantum computer with no decoherence at all.

Hard, Harder, Hardest

How is it that a quantum computer could provide speedups for some problems, such as breaking codes, but not for others? Isn't a faster computer just a faster computer? The answer is no, and to explain why takes one straight to the intellectual core of computer science. For computer scientists, the crucial thing about a problem is how quickly the time needed to solve it grows as the problem size increases. The time is measured in the number of elementary steps required by the algorithm to reach a solution. For example, using the grade school method, we can multiply two n-digit numbers in an amount of time that grows like the number of digits squared, n^2 (an amount of time said to be "a polynomial in n"). But for factoring a number into primes, even the most advanced methods known take an amount of time that grows exponentially with the number of digits (in particular, like 2 to the cube root of n power). Thus, factoring seems intrinsically harder than multiplying—and when we get up to thousands of digits, this difference matters much more than the difference between a Commodore 64 and a supercomputer.

The kind of problems that computers can solve in a reasonable amount of time, even for large values of n, are those for which we have an algorithm that uses a number of steps that grows

KEY CONCEPTS

- Quantum computers would exploit the strange rules of quantum mechanics to process information in ways that are impossible on a standard computer.
- They would solve certain specific problems, such as factoring integers, dramatically faster than we know how to solve them with today's computers, but analysis suggests that for most problems quantum computers would surpass conventional ones only slightly.
- Exotic alterations to the known laws of physics *would* allow construction of computers that could solve large classes of hard problems efficiently. But those alterations seem implausible. In the real world, perhaps the impossibility of efficiently solving these problems should be taken as a basic physical principle.

—The Editors

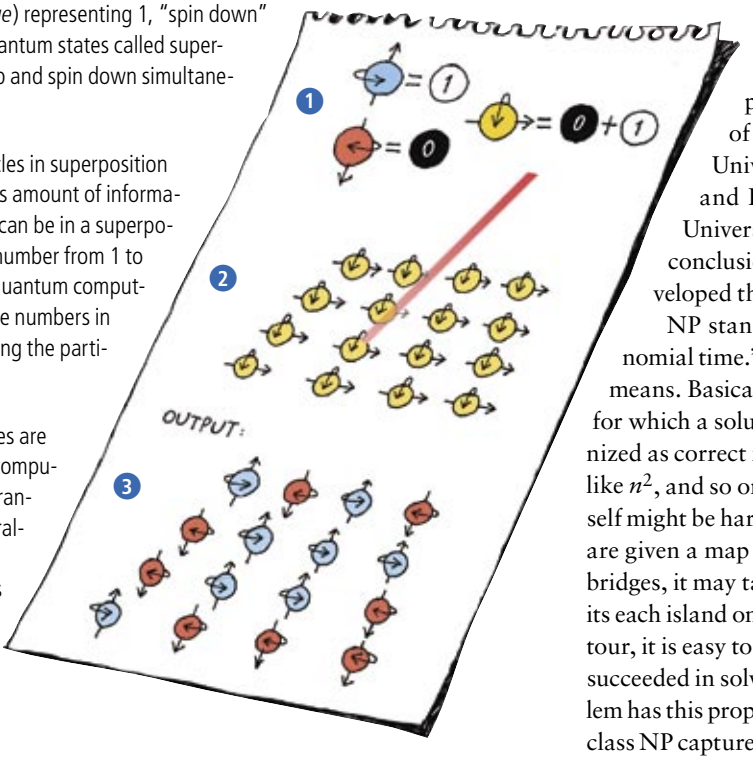
Quantum Computing 101

Physicists are hotly pursuing the construction of quantum computers, which would harness the quirks of quantum mechanics to perform certain computations more efficiently than a conventional computer.

1 The fundamental feature of a quantum computer is that it uses qubits instead of bits. A qubit may be a particle such as an electron, with "spin up" (*blue*) representing 1, "spin down" (*red*) representing 0, and quantum states called superpositions that involve spin up and spin down simultaneously (*yellow*).

2 A small number of particles in superposition states can carry an enormous amount of information: a mere 1,000 particles can be in a superposition that represents every number from 1 to $2^{1,000}$ (about 10^{300}), and a quantum computer would manipulate all those numbers in parallel, for instance, by hitting the particles with laser pulses.

3 When the particles' states are measured at the end of the computation, however, all but one random version of the 10^{300} parallel states vanish. Clever manipulation of the particles could nonetheless solve certain problems very rapidly, such as factoring a large number.



A good quantum computer algorithm ensures that **computational paths leading to a wrong answer cancel out** and that paths leading to a correct answer reinforce.

as n raised to a fixed power, such as n, n^2 or $n^{2.5}$. Computer scientists call such an algorithm efficient, and problems that can be solved by an efficient algorithm are said to be in the complexity class P, which stands for "polynomial time."

A simple example of a problem in P is: Given a road map, is every town reachable from every other town? P also contains some problems whose efficient solutions are not so obvious, such as: Given a whole number, is it prime (like 13) or composite (like 12)? Given a list of which men and women are willing to marry one another, is it possible to pair everyone off with a willing partner?

But now suppose you are given the dimensions of various boxes and you want a way to pack them in your trunk. Or suppose that you are given a map and you want to color each country red, blue or green so that no two neighboring countries are colored the same. Or that you are given a list of islands connected by bridges and you want a tour that visits each island exactly once. Although algorithms that are somewhat better than trying every possible solution are known for these problems, no algorithm is known that is fundamentally better. Every known algorithm will take an amount of time that increases exponentially with the problem size.

It turns out that the three problems I just listed have a very interesting property: they are all the "same" problem, in the sense that an efficient algorithm for any one of them would imply efficient algorithms for all the others. Stephen A. Cook of the University of Toronto, Richard Karp of the University of California, Berkeley, and Leonid Levin, now at Boston University, arrived at this remarkable conclusion in the 1970s, when they developed the theory of NP-completeness.

NP stands for "*n*ondeterministic *p*olynomial time." Do not worry about what that means. Basically, NP is the class of problems for which a solution, once found, can be recognized as correct in polynomial time (something like n^2, and so on)—even though the solution itself might be hard to find. As an example, if you are given a map with thousands of islands and bridges, it may take years to find a tour that visits each island once. Yet if someone shows you a tour, it is easy to check whether that person has succeeded in solving the problem. When a problem has this property, we say that it is in NP. The class NP captures a huge number of problems of practical interest. Note that all the P problems are also NP problems, or to put it another way, the class P is contained within the class NP. If you can solve a problem quickly you can also verify the solution quickly.

NP-complete problems are in essence the hardest of the NP problems. They are the ones with the property found by Cook, Karp and Levin: If an efficient algorithm for any one of them were found, it could be adapted to solve all the other NP problems as well.

An efficient algorithm for an NP-complete problem would mean that computer scientists' present picture of the classes P, NP and NP-complete was utterly wrong, because it would mean that every NP problem (including all the NP-complete ones) was actually a P problem. In other words, the class P would equal the class NP, which is written P = NP.

Does such an algorithm exist? Is P equal to NP? That is literally a million-dollar question—it carries a $1,000,000 reward from the Clay Math Institute in Cambridge, Mass.—and it has

played cameo roles on at least three TV shows (*The Simpsons*, *Futurama* and *NUMB3RS*).

In the half a century since the problem was recognized, no one has found an efficient algorithm for an NP-complete problem. Consequently, computer scientists today almost universally believe P does not equal NP, or P ≠ NP, even if we are not yet smart enough to understand why this is or to prove it as a theorem.

What the Quantum Can Do

If we grant that P ≠ NP, then only one hope remains for solving NP-complete problems in polynomial time: namely, to broaden what we mean by "computer." At first sight, quantum mechanics would appear to provide just the kind of resources needed. Quantum mechanics makes it possible to store and manipulate a vast amount of information in the states of a relatively small number of particles. To see how this comes about, imagine that we have 1,000 particles and that each particle, when measured, can be found to be either spinning up or spinning down. For our purposes, what it means for a particle to spin up or down is irrelevant; all that matters is that there is some property of the particle that has one of two values when measured.

To describe the quantum state of this collection of particles, one must specify a number for every possible result of measuring the particles. These numbers are called the amplitudes of the possible outcomes and relate to each outcome's probability, but unlike probabilities, quantum amplitudes can be positive or negative (in fact, they are complex numbers). For example, an amplitude is needed for the possibility that all 1,000 particles will be found spinning up, another amplitude for the possibility of finding that the first 500 particles are spinning up and that the remaining 500 are spinning down, and so on. There are $2^{1,000}$ possible outcomes, or about 10^{300}, so that is how many numbers are needed—more than there are atoms in the visible universe! The technical terminology for this situation is that the 1,000 particles are in a superposition of those 10^{300} states.

Put another way, we can store 10^{300} numbers on our 1,000 particles simultaneously. Then, by performing various operations on the particles and on some auxiliary ones—perhaps hitting them with a sequence of laser pulses or radio waves—we can carry out an algorithm that transforms all 10^{300} numbers (each one a potential solution) at the same time. If at the end of doing that we could read out the particles' final quantum state accurately, we really would have a magic computer: it would be able to check 10^{300} possible solutions to a problem, and at the end we could quickly discern the right one.

Unfortunately, there is a catch. When the particles are measured (as is necessary to read out their final state), the rules of quantum mechanics dictate that the measurement will pick out just one of the 10^{300} possibilities at random and that all the others will then disappear. (To go back to the quantum slacks developed at Haggar, if you tried to wear them you would find yourself in either formal or casual attire, not both.) We would seem to be no better off than if we used a classical computer and tried out one randomly chosen possible solution—in either case, we end up knowing about only one such possible solution.

Happily, we still have tricks we can play to wring some advantage out of the quantum particles. Amplitudes can cancel out when positive ones combine with negative ones, a phenomenon

The "**killer app**" **for quantum computers** will most likely be simulating quantum physics.

The Good News

If a large, ideal quantum computer would face most of the same limitations as our present-day classical computers do, should the physicists working on the extraordinarily hard task of building even rudimentary quantum computers pack up and go home? I believe the answer is no, for four reasons.

- If quantum computers ever become a reality, the "killer app" for them will most likely not be code breaking but rather something so obvious it is rarely even mentioned: simulating quantum physics. This is a fundamental problem for chemistry, nanotechnology and other fields, important enough that Nobel Prizes have been awarded even for partial progress.
- As transistors in microchips approach the atomic scale, ideas from quantum computing are likely to become relevant for classical computing as well.
- Quantum computing experiments focus attention directly on the most mystifying features of quantum mechanics—and I hope that the less we can sweep those puzzles under the rug, the more we will be forced to understand them.
- Quantum computing can be seen as the most stringent test to which quantum mechanics itself has ever been subjected. In my opinion, the most exciting possible outcome of quantum computing research would be to discover a fundamental reason why quantum computers are *not* possible. Such a failure would overturn our current picture of the physical world, whereas success would merely confirm it. —***S.A.***

What Classical Computers Can and Cannot Do

Computer scientists categorize problems according to how many computational steps it would take to solve a large example of the problem using the best algorithm known. The problems are grouped into broad, overlapping classes based on their difficulty. Three of the most important classes are listed below. Contrary to myth, quantum computers are not known to be able to solve efficiently the very hard class called NP-complete problems.

P PROBLEMS: Ones computers can solve efficiently, in polynomial time

Example: Given a road map showing n towns, can you get from any town to every other town? For a large value of n, the number of steps a computer needs to solve this problem increases in proportion to n^2, a polynomial. Because polynomials increase relatively slowly as n increases, computers can solve even very large P problems within a reasonable length of time.

NP PROBLEMS: Ones whose solutions are easy to verify

Example: You know an n-digit number is the product of two large prime numbers, and you want to find those prime factors. If you are given the factors, you can verify that they are the answer in polynomial time by multiplying them.

Every P problem is also an NP problem, so the class NP contains the class P within it. The factoring problem is in NP but conjectured to be outside of P, because no known algorithm for a standard computer can solve it in only a polynomial number of steps. Instead the number of steps increases exponentially as n gets bigger.

NP-COMPLETE PROBLEMS: An efficient solution to one would provide an efficient solution to all NP challenges

Example: Given a map, can you color it using only three colors so that no neighboring countries are the same color? If you had an algorithm to solve this problem, you could adapt the algorithm to solve any other NP problem (such as the factoring problem above or determining if you can pack n boxes of various sizes into a trunk of a certain size) in about the same number of steps. In that sense, NP-complete problems are the hardest of the NP problems. No known algorithm can solve an NP-complete problem efficiently.

called destructive interference. So a good quantum computer algorithm would ensure that computational paths leading to a wrong answer would cancel out in this way. It would also ensure that the paths leading to a correct answer would all have amplitudes with the same sign—which yields constructive interference and thereby boosts the probability of finding them when the particles are measured at the end.

For which computational problems can we choreograph this sort of interference, using fewer steps than it would take to solve the problem classically?

In 1994 Peter Shor, now at the Massachusetts Institute of Technology, found the first example of a quantum algorithm that could dramatically speed up the solution of a practical problem. In particular, Shor showed how a quantum computer could factor an n-digit number using a number of steps that increases only as about n^2—in other words, in polynomial time. As mentioned earlier, the best algorithm known for classical computers uses a number of steps that increases exponentially.

Black Boxes

So at least for factoring, one really can get an exponential speedup over known classical algorithms by using quantum methods. But despite a widespread misconception to the contrary, the factoring problem is neither known nor believed to be NP-complete. To create his algorithm, Shor exploited certain mathematical properties of composite numbers and their factors that are particularly well suited to producing the kind of constructive and destructive interference that a quantum computer can thrive on. The NP-complete problems do not seem to share those special properties. To this day, researchers have found only a few other quantum algorithms that appear to provide a speedup from exponential to polynomial time for a problem.

The question thus remains unanswered: Is there an efficient quantum algorithm to solve NP-complete problems? Despite much trying, no such algorithm has been found—though not surprisingly, computer scientists cannot prove that it does not exist. After all, we cannot even prove that there is no polynomial-time classical algorithm to solve NP-complete problems.

What we *can* say is that a quantum algorithm capable of solving NP-complete problems efficiently would, like Shor's algorithm, have to exploit the problems' structure, but in a way that is far beyond present-day techniques. One cannot

achieve an exponential speedup by treating the problems as structureless "black boxes," consisting of an exponential number of solutions to be tested in parallel. Some speedup can nonetheless be wrung out of this black box approach, and computer scientists have determined just how good—and how limited—that speedup is. The algorithm that produces the speedup is the second major quantum algorithm.

The black box approach can be illustrated by pretending that you are searching for the solution to a difficult problem and that the only operation you know how to perform is to guess a solution and see if it works. Let us say there are S possible solutions, where S grows exponentially as the problem size n increases. You might get lucky and guess the solution on your first try, but in the worst case you will need S tries, and on average you will need $S/2$.

Now suppose you can ask about all the possible solutions in quantum superposition. In 1996 Lov Grover of Bell Laboratories developed an algorithm to find the correct solution in such a scenario using only about $\sqrt{S}$ steps. A speedup from $S/2$ to $\sqrt{S}$ is a useful advance for some problems—if there are a million possible solutions, you will need around 1,000 steps instead of 500,000. But the square root does not transform an exponential time into a polynomial time; it just produces a smaller exponential. And Grover's algorithm is as good as it gets for this kind of black box searching: in 1994 researchers had shown that any black box quantum algorithm needs at least $\sqrt{S}$ steps.

Over the past decade, researchers have shown that similar modest speedups are the limit for many other problems besides searching a list, such as counting ballots in an election, finding the shortest route on a map, and playing games of strategy such as chess or Go. One problem that presented particular difficulty was the so-called collision problem, the problem of finding two items that are identical, or that "collide," in a long list. If there were a fast quantum algorithm to solve this problem, many of the basic building blocks of secure electronic commerce would be useless in a world with quantum computers.

Searching a list for an item is like looking for a needle in a haystack, whereas searching for a collision is like looking for two identical pieces of hay, which provides the problem with a kind of structure that a quantum computer could potentially exploit. Nevertheless, I showed in 2002 that within the black box model, any quantum algorithm needs exponential time to solve the collision problem.

Admittedly, these black box limitations do

[THE AUTHOR]

Scott Aaronson is an assistant professor of electrical engineering and computer science at the Massachusetts Institute of Technology. A high school dropout, he went on to receive a bachelor's degree from Cornell University and a Ph.D. in computer science from the University of California, Berkeley, advised by Umesh Vazirani. Outside of research, Aaronson is best known for his widely read blog (www.scottaaronson.com/blog), as well as for creating the Complexity Zoo (www.complexityzoo.com), an online encyclopedia of more than 400 complexity classes.

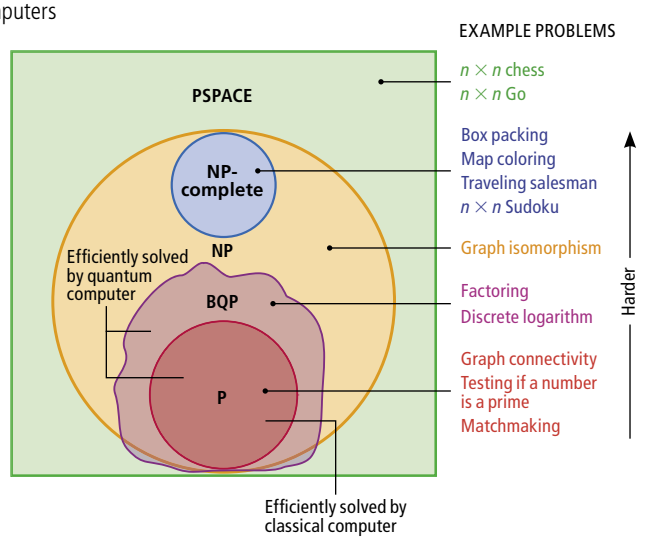
Where Quantum Computers Fit In

The map at the right depicts how the class of problems that quantum computers would solve efficiently (BQP) might relate to other fundamental classes of computational problems. (The irregular border signifies that BQP does not seem to fit neatly with the other classes.)

The BQP class (the letters stand for *b*ounded-error, *q*uantum, *p*olynomial time) includes all the P problems and also a few other NP problems, such as factoring and the so-called discrete logarithm problem. Most other NP and all NP-complete problems are believed to be outside BQP, meaning that even a quantum computer would require more than a polynomial number of steps to solve them.

In addition, BQP might protrude beyond NP, meaning that quantum computers could solve certain problems faster than classical computers could even check the answer. (Recall that a conventional computer can efficiently verify the answer of an NP problem but can efficiently solve only the P problems.) To date, however, no convincing example of such a problem is known.

Computer scientists do know that BQP cannot extend outside the class known as PSPACE, which also contains all the NP problems. PSPACE problems are those that a conventional computer can solve using only a polynomial amount of memory but possibly requiring an exponential number of steps.



JASON DORFMAN (*Aaronson*)

ZONES OF THOUGHT

Unlike the real world, in which computational limits are believed to be the same everywhere, the galaxy in Vernor Vinge's 1992 science-fiction novel *A Fire Upon the Deep* is divided into concentric "zones of thought" having different inherent computational and technological limits.

In the Unthinking Depths, nearest the galactic core, even simple automation fails and IQs plummet.

The Slow Zone contains Earth and is as limited as we know it.

In the Beyond, nearly sentient nanotechnology factories construct wonders such as anti-gravity fabrics, and hypercomputation enables faster-than-light travel.

The Transcend is populated by dangerous, godlike über-intelligences having technologies and thought processes unfathomable to lower beings.

not rule out the possibility that efficient quantum algorithms for NP-complete or even harder problems are waiting to be discovered. If such algorithms existed, however, they would have to exploit the problems' structure in ways that are unlike anything we have seen, in much the same way that efficient classical algorithms for the same problems would have to. Quantum magic by itself is not going to do the job. Based on this insight, many computer scientists now conjecture not only that $P \neq NP$ but also that quantum computers cannot solve NP-complete problems in polynomial time.

Magical Theories

Everything we know is consistent with the possibility that quantum computers are the end of the line—that is, that they are the most general kind of computer compatible with the laws of physics. But physicists do not yet have a final theory of physics, so one cannot rule out the possibility that someday a future theory might reveal a physical means to solve NP-complete problems efficiently. As you would expect, people speculate about yet more powerful kinds of computers, some of which would make quantum computers look as pedestrian as vending machines. All of them, however, would rely on speculative changes to the laws of physics.

One of the central features of quantum mechanics is a mathematical property called linearity. In 1998 Daniel S. Abrams and Seth Lloyd, both then at M.I.T., showed that if a small nonlinear term is added to the equations of quantum mechanics, quantum computers would be able to solve NP-complete problems efficiently. Before you get too excited, you should realize that if such a nonlinear term existed, then one could also violate Heisenberg's uncertainty principle and send signals faster than the speed of light. As Abrams and Lloyd pointed out, perhaps the best interpretation of these results is that they help to explain why quantum mechanics is linear.

Another speculative type of machine would achieve extravagant computational abilities by cramming an infinite number of steps into a finite time. Unfortunately, according to physicists' current understanding, time seems to degenerate into a sea of quantum fluctuations—something like a foam instead of a uniform smooth line—on the scale of 10^{-43} second (the Planck time), which would seem to make this kind of machine impossible.

If time cannot be sliced with arbitrary thinness, then perhaps another way to solve NP-complete problems efficiently would be to exploit time travel. Physicists studying the issue talk not about time machines but about closed timelike curves (CTCs). In essence a CTC is a route through space and time that matter or energy could travel along to meet up with itself in the past, forming a closed loop. Current physical theory is inconclusive on whether CTCs can exist, but that need not stop us from asking what the consequences would be for computer science if they did exist.

It seems obvious how one could use a CTC to speed up a computation: program your computer to take however long it needs to solve the problem and then send the answer back in time to yourself at a point before the computer started. Alas, this simple idea does not work, because it ignores the famous grandfather paradox, where you go back in time to kill your own grandfather (but then you are never born, so you never go back in time, and so your grandfather lives to have children after all, and later you *are* born, but then …). In our setting, what would happen if you turned off the computer after you received its answer from the future?

In 1991 physicist David Deutsch of the University of Oxford defined a model of computation with CTCs that avoids this difficulty. In Deutsch's model, nature will ensure that as events unfold along the circular timeline that makes up the CTC, no paradoxes ever arise, a fact that can be exploited to program a computer that loops

Über-Computers from Exotic Physics?

Although quantum computers seem unlikely to solve NP-complete problems quickly, certain other extraordinary, speculative physical processes would allow construction of computers with that ability and much more. Time travel, for instance, would make it possible to efficiently solve any PSPACE problem, including those harder than NP-complete ones—such as how to play the perfect game of chess on any size board, including those larger than the standard 8×8 version. Employing time travel to solve problems would not be as simple as having a computer finish a long computation in the far future and send the answer back to itself in the present, but that kind of loop in time would be exploited. Just one problem: the speculative processes defy the known laws of physics.

A New Physical Principle?

Because implausible kinds of physics (such as time travel) seem necessary for constructing a computer able to solve NP-complete problems quickly, I predict that scientists might one day adopt a new principle: "NP-complete problems are hard." That is, solving those problems efficiently is impossible on any device that could be built in the real world, whatever the final laws of physics turn out to be. The principle implies that time travel is impossible, because such travel would enable creation of über-computers that could solve NP-complete problems efficiently. Further, if a proposed theory were shown to permit such computers, that theory could be ruled out. Application of the principle would be analogous to applying the laws of thermodynamics to conclude that perpetual-motion machines are impossible (the laws of thermodynamics forbid them) and to deduce previously unknown features of physical processes. —***S.A.***

around inside the CTC to solve hard problems.

Indeed, by using a CTC, we could efficiently solve not only NP problems but even problems in an apparently larger class called PSPACE. PSPACE is the class of problems that could be solved on a conventional computer using a polynomial amount of memory but possibly taking an exponential amount of time. In effect, a CTC would make time and space interchangeable as computational resources. (I did not have to mention the polynomial memory constraint until now, because for P and NP problems it makes no difference if the computer has access to more than polynomial memory.) Recently John Watrous of the University of Waterloo in Ontario and I showed that using a quantum computer in a CTC instead of a conventional one does not enable anything beyond PSPACE to be efficiently solved. In other words, if CTCs exist, then quantum computers are no more powerful than classical ones.

Computational Kryptonite

Physicists do not know if future theories will permit any of these extraordinary machines. Yet without denying our ignorance, we can view that ignorance from a different perspective. Instead of starting from physical theories and then asking about their computational implications, we could start by assuming that NP-complete problems are hard and then study the consequences of that assumption for physics. For instance, if CTCs would let us solve NP-complete problems efficiently, then by starting from the assumption that NP-complete problems are intractable, we could conclude that CTCs cannot exist.

To some, such an approach will seem overly dogmatic. To me, it is no different from assuming the second law of thermodynamics or the impossibility of faster-than-light communication—two earlier limitations on technology that over time earned the status of physical principles. Yes, the second law might be experimentally falsified tomorrow—but until that happens, physicists find it vastly more useful to assume it is correct and then use that assumption for studying everything from car engines to black holes. I predict that the hardness of NP-complete problems will someday be seen the same way: as a fundamental principle that describes part of the essential nature of our universe. There is no way of telling what theoretical enlightenment or what practical consequences might come from future application of this kind of fundamental principle.

In the meantime, we know not to expect magic from quantum computers. To some, the apparent limitations of quantum computers might come as a letdown. One can, however, give those same limitations a more optimistic spin. They mean that although certain cryptographic codes could be broken in a world with quantum computers, other codes would probably remain secure. They increase our confidence that quantum computing will be possible at all—because the more a proposed technology sounds like a science-fiction caricature, the more skeptical we should be. (Who would you be more inclined to believe: the salesperson offering a device that produces unlimited free energy from the quantum vacuum or the one offering a refrigerator that is more efficient than last year's model?) And last, such limitations ensure that computer scientists will continue to have their work cut out for them in designing new quantum algorithms. Like Achilles without his heel or Superman without kryptonite, a computer without any limitations would get boring pretty quickly. ■

More about nonlinear quantum mechanics, hypercomputing, use of time travel, and another scheme called anthropic computing can be found at **www.SciAm.com/ontheweb**

MORE TO EXPLORE

Quantum Computation and Quantum Information. Michael A. Nielsen and Isaac L. Chuang. Cambridge University Press, 2000.

NP-Complete Problems and Physical Reality. Scott Aaronson in *ACM SIGACT News,* Complexity Theory Column, Vol. 36, No. 1, pages 30–52; March 2005. Available at **www.scottaaronson.com/papers/npcomplete.pdf**

Quantum Computer Science: An Introduction. N. David Mermin. Cambridge University Press, 2007.

Shor, I'll Do It. (An explanation of Shor's algorithm for the layperson.) Scott Aaronson. Available at **www.scottaaronson.com/blog/?p=208**

Quantum Computing since Democritus. Lecture notes from course PHYS771, University of Waterloo, Fall 2006. Available at **www.scottaaronson.com/democritus/**