Must "Hard Problems" Be Hard?

Computer scientists have new hope that they will be able to learn how hard problems really are

Mathematicians have a joke that the way to live forever is to choose a problem that is so important that you cannot die until you solve it. Then never work on it. In the field of computational complexity theory, the polynomial hierarchy problem was beginning to look as if it would be suitable for someone who wants to become immortal. It is difficult—so difficult that, as Ronald Graham of AT&T Bell Laboratories puts it, "many mathematicians have broken their pencils on it." It is important—perhaps the most important problem in theoretical computer science. And, until recently, it looked as if it was unlikely to be solved in the foreseeable future.

But now Andrew Yao of Stanford University has announced a difficult and complex proof that, although it does not directly solve the problem, may at least give mathematicians a foot in the door. It is, says Graham, "one of the most striking developments yet," in work on the problem. Although Yao is still in the process of writing up his 25 or so page result, already mathematicians from around the world have begun phoning him and leaving electronic messages of congratulations on his computer.

The problem is to decide just how hard it is to solve problems on computers. Researchers have classified groups of problems according to how much time they seem to require for their solutions and have generated what they call a "polynomical hierarchy" of classes of problems that seem to be increasingly difficult to solve. The questions are, Is this hierarchy real? Does it reflect the true nature of problems or does it merely reflect our own inadequacies in solving them? Could it be that even the hardest problems at the top of the hierarchy are actually easily solvable? Most computer scientists believe that the hierarchy is real, but proving it is quite a different matter. This is the problem in which Yao has made a dent.

At the bottom of the hierarchy are the problems that are easiest to solve. These are problems that can be solved in a number of steps that is only a polynomial function of the size of the problem. For example, a problem with n variables might be solved in n^2 steps. This is a good situation since the number of computations grows only slowly as the problem size gets larger. If n goes from 10 to 20, for example, the number of steps, n^2, only quadruples. Since, in computer science, the number of steps needed to solve a problem is proportional to the amount of time needed, the researchers coined the term polynomial time, or P, to describe this group of problems.

One example of a polynomial time problem, according to Graham, is that of pairing acquaintances. You have a group consisting of equal numbers of men and women, each of whom knows some of the others. Can you pair each man with a woman he knows? This problem occurs in other guises as a problem in graph theory and design of networks.

> Yao's proof is "very intricate and deep rather than just a clever trick."

After the polynomial time problems, the problems in the hierarchy get more and more complicated. All of them are suspected to be exponential time problems, meaning that the only general solutions for them require a number of computations that is an exponential function of the size of a problem. For example, a problem with n variables might require 2^n steps, which is an exponential function of n. In this case, if n increases from 10 to 20, the number of steps necessary to solve the problem increases 1000-fold. The exponential time problems take so long to solve that, in most cases, it is completely impossible to even think of grinding out their solutions on a computer and researchers have had to deal with special cases or approximations to their solutions instead.

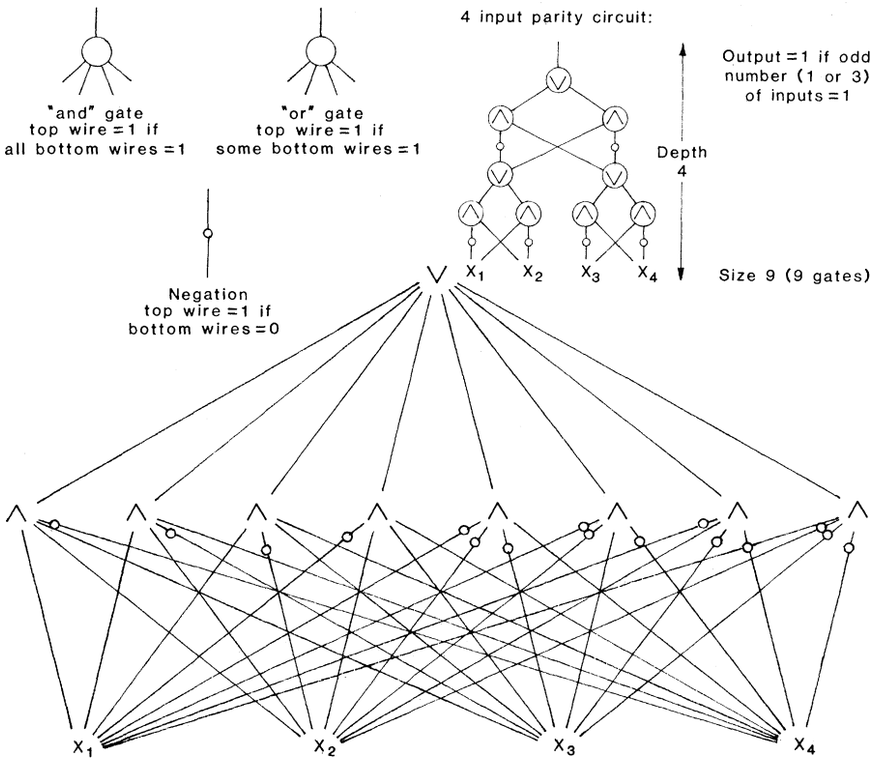
The second step in the hierarchy is the set of problems classified as NP, for nondeterministic polynomial time. These are problems that have no known polynomial time algorithms, but, given a possible solution, you could check it in polynomial time.

One of the best known NP problems is the notorious traveling salesman problem. A salesman has to plan a route that takes him to, say, 60 cities, passing through each one only once but only has enough time to go, say, 5000 miles. Is there a route that allows this? "It is certainly easy to check that a given route works, but finding it yourself is certainly hard," says Michael Sipser of Massachusetts Institute of Technology. Yet this problem is an immensely important one and plagues numerous corporations, including airlines that must plan routes and telephone companies that must decide the best order to drill holes during the fabrication of printed circuit boards.

One way to solve the traveling salesman problem is to try all possible routes, looking for the shortest, and, in fact, although this method is cumbersome and far from clever, no one has ever found a significantly better way that will work for all possible traveling salesman problems and that will always give the shortest route as its answer. But, if there are 60 cities, this method would take more than 10^{79} steps, and the number of steps increases astronomically as the number of cities is increased. In order to get answers to practical traveling salesman problems, computer scientists have devised methods that are faster than the brute force one but that are not guaranteed always to find the best route. What they promise instead is to give a route that is not too bad.

Another well-known NP problem is the Hamiltonian circuit problem. You are given a graph, which is a set of points, some of which are connected by lines, and are asked to decide, Is there a circuit connecting the points so that each point is in the circuit and the circuit goes through each point only once? It is like the traveling salesman problem, only you are not asked to find the shortest route—all you are asked is whether a route can even be drawn.

As hard as the NP problems are, they are only the beginning. In the hierarchy classification, computer scientists distinguish them because they all have the form, "There exists . . ." and the problem is to answer yes or no. In the case of the traveling salesman problems, this is formulated as "There exists a 5000 mile route." The Hamiltonian circuit problem is of the form, "There exists a Hamiltonian circuit."



Building Boolean circuits

The "and" and "or" gates along with the negation make up the circuit. With these, you can build a 4 input parity circuit with depth 4. But if you squeeze the depth of this circuit down to 2, its width blows up. [Source: Michael Sipser]

The next step up the hierarchy is the set of problems that are formulated, "For all . . . there exists . . . ." "For example," says Sipser, "the lines of a graph might represent roads in the northeast. One-half might be impassable because of snow. Is there guaranteed to be a Hamiltonian circuit?" Or, "For all ways of pulling out half the edges of a graph, is there a Hamiltonian circuit?"

The fourth step up is problems of the form, "For all . . . there exists . . . for all . . . there exists . . . ." For all ways that you can find to pull out one-half of the edges out of a graph there exist one half of these that I can choose to put back so that for all ways you can find of pulling out one-fourth of these there exists a Hamiltonian circuit made up of the remainder. The fifth step is problems of the form "For all . . . there exists . . . for all . . . there exists . . . for all . . . there exists . . ." The hierarchy goes up in this way and at the top of it are problems that mathematicians call "polynomial space."

Richard Karp of the University of California at Berkeley says you can think of the polynomial hierarchy as games with alternating moves. Polynomial time problems are like solitaire. There is only one player and the outcome is fixed. The third level of the hierarchy is like a game where you and your opponent each have one move. The game is short but it still may be hard to tell who will win. For the fourth level, the total number of moves in the game is three. Once again, the question is, Who will win the game? "Polynomial space can also be seen as a game," says Karp, "but it is a game with no fixed number of moves. It is a game like chess or Go and the problem is to decide who will win after the first move has been made by the first player."

The question that plagues computer scientists is to determine whether the polynomial hierarchy is real or whether the classes actually are all the same, which would mean that all the problems in the hierarchy are really solvable quickly, in polynomial time. And, if there is at least some true distinction between problems in the hierarchy, is the entire hierarchy distinct or does some of it collapse? Assuming there is a hierarchy, where does it stop? Finally, is polynomial space different from the rest of the hierarchy?

These questions are of immense practical importance because they essentially ask whether problems that arise daily have straightforward, although as yet undiscovered, solutions or whether they are as difficult as they appear. If the polynomial hierarchy is real, the best that can be hoped for is approximate solutions to these very difficult problems.

For awhile, it looked to many researchers as if the way to get at the polynomial hierarchy problem is with mathematical logic. The idea was to look at the logical structure of the problems by getting outside them, into what is really another world. Suppose, the logicians reasoned, there were an oracle—an all-knowing creature that could answer one particular type of question for you. For example, the oracle might have a list of every possible graph with a Hamiltonian circuit. Then if you wanted to know if your graph has a Hamiltonian circuit, all you would have to do is ask the oracle if your graph is on his list.

But the oracle work had a curious outcome. What happened was that two groups of researchers, John Gill of Stanford and Robert Solovay of the University of California at Berkeley and, independently, Theodore Baker, now at Florida State University, and Alan Selman of Iowa State University showed that there exist oracles that make the class of problems called NP, the second class of the hierarchy, collapse into the first class, P. On the other hand, there are other oracles that make P and NP completely different. This result, says Juris Hartmanis of Cornell University, "came as a great shock."

Still, it seemed to computer scientists that if the polynomial hierarchy really is a hierarchy, then there should be an oracle that can separate the whole string of classes of problems. But they were stymied. They were able to show that there are oracles that can separate the fourth class of problems from P and NP but no one could extend the separation any farther.

Then, a few years ago, Sipser, working with Merrick Furst and James Saxe of Carnegie-Mellon University, found another way to get at the polynomial hierarchy problem. They discovered, says Sipser, that "the problem can essentially be phrased as a question about Boolean circuits." These circuits, familiar to computer hardware designers, are networks of gates that compute "ands" and "ors." Computer information is stored as bits, which are 0's and 1's. To work with these bits, the computer combines them with "and" and "or" functions. "Once you have these circuits, you can compute all kinds of things," Sipser says. "Now imagine that you are making networks to tell you how to solve problems in the class P and problems in the class NP. If you can prove that the network has to be really big for NP but not for P, then you have proved that P does not equal NP."

However, no one knows how to prove

that these Boolean circuits have to be really big for problems in NP and for problems farther up the polynomial hierarchy. So what Sipser and his colleagues did was to look at restricted versions of the network. Suppose that the networks can grow as wide as you want but that their depth is limited. Then, Sipser asked, Can you prove that there is a difference between classes in the polynomial hierarchy? The three investigators looked at the parity function, which simply computes whether there is an even or odd number of ones in a string of bits and showed that, if they limit the depth of the circuit, the width of it blows up so that it is bigger than any polynomial. At about the same time, Miklós Ajtai of Hungary, who is now at IBM in San Jose, proved a similar result.

But this finding, impressive as it was, was not enough to show that the polynomial hierarchy is distinct in this restricted circumstance. This result was finally obtained by Yao, who now has shown that the width of the Boolean circuit for the parity function increases even faster than Sipser, Furst, and Saxe and Ajtai found. It grows exponentially, which is incomparably faster than polynomial growth. And, as a consequence and extension of his result, he showed that there exists an oracle for which the polynomial hierarchy is distinct.

It is, says Yao, "a difficult proof." Sipser describes it as "very intricate and deep rather than just a clever trick." But Yao, Sipser, and others think the implications of the proof go far beyond the immediate discovery that the polynomial hierarchy is separable in the strange world of oracles. In fact, Yao remarks, his work is "a positive step" toward proving that the polynomial hierarchy may be distinct without oracles, although mathematicians find the oracle approach controversial. Some are enthusiastic whereas others say that the results have no obvious connection to the real world.

The true importance of Yao's proof, according to even the oracle skeptics, is that it shows that a combinatorial approach, using Boolean circuits, might be used to get at the polynomial hierarchy problem.

Graham explains, "To many people, Andy's result provides real evidence that there's a lot to be learned by looking at sophisticated combinatorial arguments. People could never really get their hands on circuits until now. I expect that this will give people renewed impetus to push this approach. There is a lot more hope now that the circuits will give you something."

If the circuits really are to solve the polynomial hierarchy problem, and without an oracle, then researchers will have to extend their results to unrestricted circuits, those that can grow to any depth necessary. How feasible is this? Sipser, for one, suspects that it can be done. "I have some ideas on how to go to unrestricted circuits," he says. "Let's just say that I am confident enough to invest my time on it."

And, finally, there is a small but not insignificant practical application of this highly abstract research. Certain kinds of circuits, called programmable logic arrays, are easy to put on chips and, Sipser points out, "they have intrinsic depth limits." Computer engineers had noticed that there are certain kinds of operations that simply could not be put on these circuits, a key one being the parity function and another being multiplication. "Everyone knew that you can't do parity or multiplication on a programmable logic array, but no one could prove it," Sipser remarks. "Now we know why these things are impossible."—GINA KOLATA

Synchrotron Radiation Takes Over at Orsay

Once a small, piggyback operation, the LURE laboratory now has exclusive use of all Orsay accelerators and is building a new one

Those used to a "small science" style of research now often have to travel to large centralized research facilities in order to do forefront experiments. This article is one of a series in which Science *looks at such centers in Europe.*

Orsay. The rags to riches tale of the clerk who rises to become president of the company applies at least in part to the Laboratory for the Use of Electromagnetic Radiation (LURE in French) here at the University of Paris-South. As of 1 January, LURE assumed full responsibility (hardware, personnel, and administration) for the accelerators formerly operated for high energy physicists by Orsay's Linear Accelerator Laboratory (LAL) and now dedicates them to the production of ultraviolet and x-ray synchrotron light.

Now renamed the Laboratory André Lagarigue (after its founder), LAL has by no means gone out of business. Its high energy physicists, as is the custom nowadays in Europe, carry out their elementary particle experiments at the large laboratories CERN near Geneva and DESY in Hamburg. At the same time, from its meager start in 1972 as a single research group siphoning off ultraviolet synchrotron light from the ACO electron-positron storage ring at LAL, LURE has grown into a major research center serving over 400 scientists.

In addition to 130 former LAL engineers, technicians, and support staff inherited as part of the restructuring, LURE is gaining a 2-billion-electron-volt (GeV) electron linear accelerator (the one from which LAL got its name) and a 1.85-GeV electron-positron storage ring named DCI, which serves as the x-ray source for 11 instruments. The laboratory already had taken control in 1975 of the smaller 547-million-electron-volt (MeV) ACO, which generates ultraviolet light for a dozen experimental stations.

At 20 years of age, ACO is ancient as storage rings go and, of course, was not designed with synchrotron radiation in mind anyway. LURE is nearing the end of the civil construction phase of a project to build Super ACO, an 800-MeV machine that should be ready for experimenters in a little under 2 years. Super ACO will compare favorably with dedicated ultraviolet synchrotron sources in Berlin and at Brookhaven National Laboratory (*Science*, 15 March, p. 1323), but it will also be quite innovative.

Most notably, Super ACO will be the first synchrotron facility to rely on a special kind of magnet called an undulator as the primary source of radiation, which will make for an extremely bright light. As frosting on the cake, it will also store a beam of positrons rather than electrons in order to enhance the stability and lifetime of the beam and thereby allow longer uninterrupted periods of bright light for researchers.