COMPUTATIONAL COMPLEXITY

Complexity Theory's 50-Year Journey to the Limits of Knowledge

By BEN BRUBAKER

August 17, 2023

How hard is it to prove that problems are hard to solve? Meta-complexity theorists have been asking questions like this for decades. A string of recent results has started to deliver answers.

Complexity theorists are confronting their most puzzling problem yet: complexity theory itself.

—

Tommy Parker for *Quanta Magazine*

I. Origins

In the first week of the fall semester in 2007, Marco Carmosino dragged himself to a math class required for all computer science majors at the University of Massachusetts, Amherst. Carmosino, a sophomore, was considering dropping out of college to design video games. Then the professor posed a simple question that would change the course of his life: How do you know math actually works?

"That made me sit up and pay attention," recalled Carmosino, now a theoretical computer scientist at IBM. He signed up for an optional seminar on the work of Kurt Gödel, whose dizzying self-referential arguments first exposed the limits of mathematical reasoning and created the foundation for all future work on the fundamental limits of computation. It was a lot to take in.

"I 100% did not understand," Carmosino said. "But I knew that I wanted to."

Today, even seasoned researchers find understanding in short supply when they confront the central open question in theoretical computer science, known as the P versus NP problem. In essence, that question asks whether many computational problems long considered extremely difficult can actually be solved easily (via a secret shortcut we haven't discovered yet), or whether, as most researchers suspect, they truly are hard. At stake is nothing less than the nature of what's knowable.

Despite decades of effort by researchers in the field of computational complexity theory — the study of such questions about the intrinsic difficulty of different problems — a resolution to the P versus NP question has remained elusive. And it's not even clear where a would-be proof should start.

"There's no road map," said Michael Sipser, a veteran complexity theorist at the Massachusetts Institute of Technology who spent years grappling with the problem in the 1980s. "It's like you're going into the wilderness."

It seems that proving that computational problems are hard to solve is itself a hard task. But why is it so hard? And just how hard is it? Carmosino and other researchers in the subfield of meta-complexity reformulate questions like this as computational problems, propelling the field forward by turning the lens of complexity theory back on itself.

"You might think, 'OK, that's kind of cool. Maybe the complexity theorists have gone crazy,'" said Rahul Ilango, a graduate student at MIT who has produced some of the most exciting recent results in the field.

By studying these inward-looking questions, researchers have learned that the hardness of proving computational hardness is intimately tied to fundamental questions that may at first seem unrelated. How hard is it to spot hidden patterns in apparently random data? And if truly hard problems do exist, how often are they hard?

"It's become clear that meta-complexity is close to the heart of things," said Scott Aaronson, a complexity theorist at the University of Texas, Austin.

This is the story of the long and winding trail that led researchers from the P versus NP problem to meta-complexity. It hasn't been an easy journey — the path is littered with false turns and roadblocks, and it loops back on itself again and again. Yet for meta-complexity researchers, that journey into an

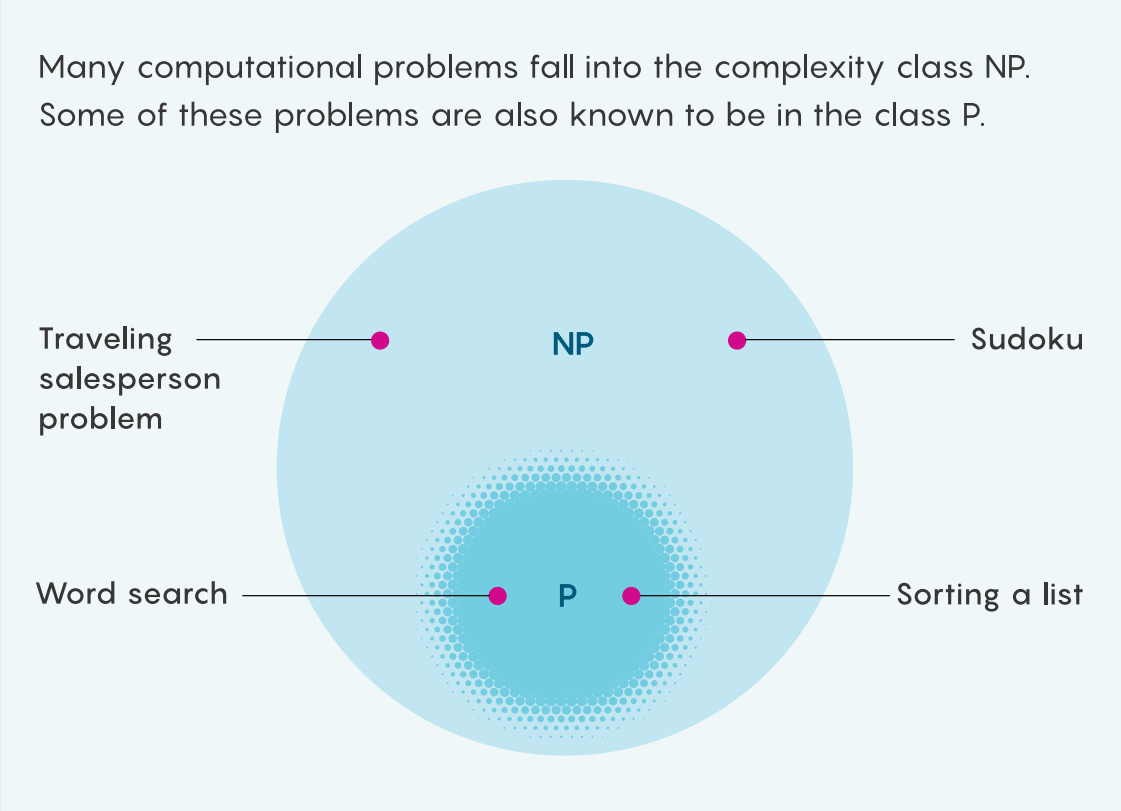
uncharted landscape is its own reward. Start asking seemingly simple questions, said Valentine Kabanets, a complexity theorist at Simon Fraser University in Canada, and "you have no idea where you're going to go."

Known Unknowns

The P versus NP problem owes its lackluster name to complexity theorists' habit of sorting computational problems into broad "complexity classes" with labels suggestive of Nasdaq ticker symbols. A computational problem is one that can in principle be solved by an algorithm — a precisely specified list of instructions. But not all algorithms are equally useful, and the variation among algorithms hints at fundamental differences between problems in different classes. The challenge for complexity theorists is to turn these hints into rigorous theorems about the relationships between complexity classes.

These relationships reflect immutable truths about computation that go far beyond any specific technology. "This is like discovering the laws of the universe," Kabanets said.

"P" and "NP" are the two most famous members of a growing menagerie of hundreds of complexity classes. Roughly speaking, P is the class of problems that can be solved easily, like alphabetizing a list. NP is the class of problems with easily checkable solutions, like sudoku puzzles. Since all easily solvable problems are also easy to check, problems in P are also in NP. But some NP problems seem hard to solve — you can't immediately intuit the solution to a sudoku puzzle without first trying out many possibilities. Could it be that this apparent hardness is just an illusion — that there's a single simple trick for solving every easily checkable problem?



—

Samuel Velasco/ *Quanta Magazine*

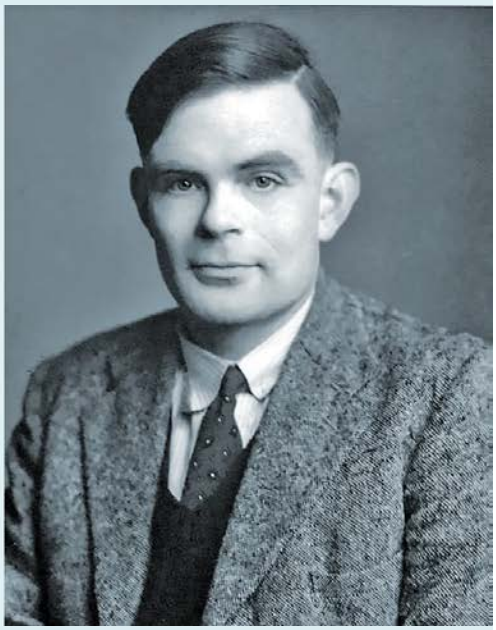
If so, then P = NP: The two classes are equivalent. If that's the case, there must be some algorithm that makes it trivial to solve enormous sudoku puzzles, optimize global shipping routes, break state-of-the-art encryption and automate the proofs of mathematical theorems. If P ≠ NP, then many computational problems that are solvable in principle will in practice remain forever beyond our grasp.

Researchers worried about the limits of formal mathematical reasoning long before the P versus NP problem was first articulated — indeed, long before the beginning of modern computer science. In 1921, struggling with the same question that would grab Carmosino's attention nearly a century later, the mathematician David Hilbert proposed a research program for grounding mathematics in absolute certainty. He hoped to start from a few simple assumptions, called axioms, and derive a unified mathematical theory that met three key criteria.

Hilbert's first condition, consistency, was the essential requirement that mathematics be free of contradictions: If two contradictory statements could be proved starting from the same axioms, the whole theory would be unsalvageable. But a theory could be free of contradiction and still limited in its reach. That was the motivation for Hilbert's second condition, completeness: the requirement that all mathematical statements be either provably true or provably false. His third criterion, decidability, demanded an unambiguous mechanical procedure for determining whether any mathematical statement was true or false. Addressing a conference in 1930, Hilbert declared: "Our slogan shall be 'We must know, we will know.'"

Just a year later, Gödel delivered the first blow to Hilbert's dream. He proved that a self-defeating statement like "this statement is unprovable" could be derived from any appropriate set of axioms. If such a statement is indeed unprovable, the theory is incomplete, but if it's provable, the theory is inconsistent — an even worse outcome. In the same paper, Gödel also proved that no mathematical theory could ever prove its own consistency.





In the 1920s, David Hilbert (left) wanted to put mathematics on firmer foundations. Kurt Gödel (center) and Alan Turing later showed that Hilbert's dream was impossible.

—

University of Göttingen (left); Kurt Gödel Papers, the Shelby White and Leon Levy Archives Center, Institute for Advanced Study; GL Archive/Alamy Stock Photo

Researchers still held out hope that a future theory of mathematics, though necessarily incomplete, might nonetheless be proved decidable. Perhaps they could develop procedures that would identify all provable statements while steering clear of vexing propositions like Gödel's. The trouble was that nobody knew how to reason about these hypothetical procedures.

Then in 1936, a 23-year-old graduate student named Alan Turing rephrased Hilbert's decidability condition in the then-unfamiliar language of computation and dealt it a fatal blow. Turing formulated a mathematical model — now known as the Turing machine — that could represent all possible algorithms, and showed that if Hilbert's procedure existed, it would fit within this model. He then used self-referential methods like Gödel's to prove the existence of undecidable statements — or, equivalently, uncomputable problems that no algorithm could solve.

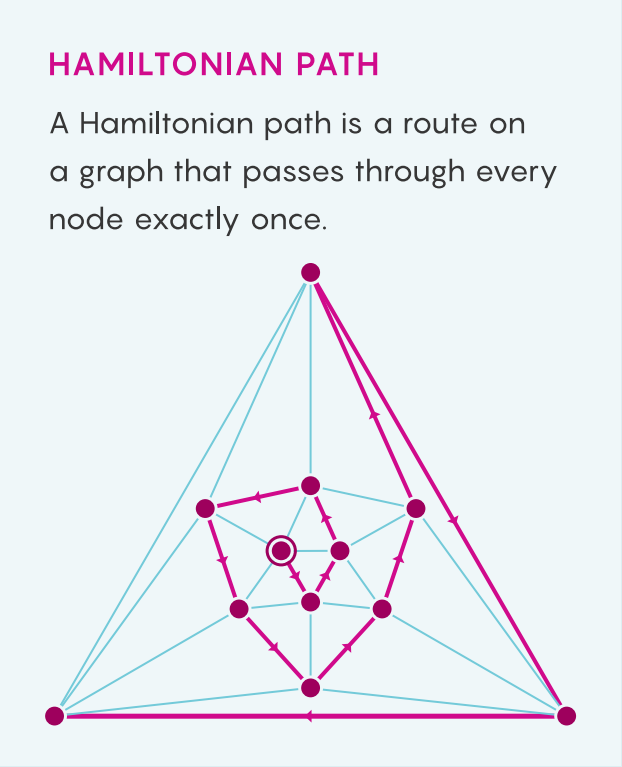
Hilbert's program lay in ruins: There would forever be fundamental limits on what could be proved and what could be computed. But as computers evolved from theoretical abstractions to real machines, researchers realized that Turing's simple distinction between solvable and unsolvable problems left many questions unanswered.

By the 1960s, computer scientists had developed fast algorithms to solve some problems, while for others the only known algorithms were excruciatingly slow. What if the question wasn't just whether problems are solvable, but how hard they are to solve?

"A rich theory emerges, and we don't know the answers anymore," Carmosino said.

Divergent Paths

To illustrate just how perplexing questions about hardness can be, let's consider a pair of closely related problems involving graphs. These are networks of points, called nodes, connected by lines, called edges. Computer scientists use them to model everything from quantum computation to the flow of traffic.



—

Samuel Velasco/ *Quanta Magazine*

Suppose you're given a graph and asked to find something called a Hamiltonian path — a route that passes through every node exactly once. This problem is clearly solvable in principle: There are only a finite number of possible paths, so if all else fails you can just check each one. That's fine if there are only a few nodes, but for even slightly larger graphs the number of possibilities spirals out of control, quickly rendering this simple algorithm useless.

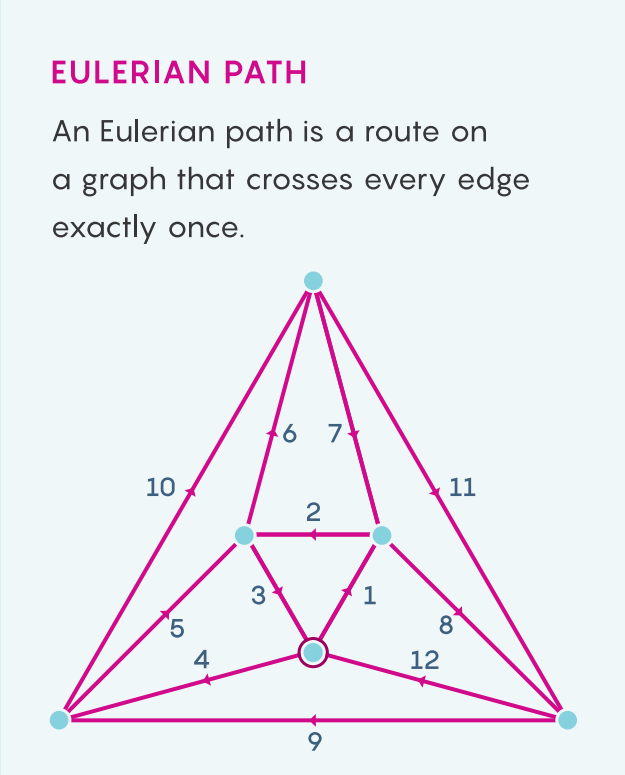
There are more sophisticated Hamiltonian path algorithms that put up a better fight, but the time that algorithms require to solve the problem invariably grows exponentially with the size of the graph. Graphs don't have to be very large before even the best algorithm researchers have discovered can't find a path "in any reasonable amount of time," said Russell Impagliazzo, a complexity theorist at the University of California, San Diego. "And by 'reasonable amount of time,' I mean 'before the universe ends.'"

The Hamiltonian path problem has another interesting property. If somebody claims to have found a Hamiltonian path on a particular graph, you can quickly check whether the solution is valid, even if the graph is very large. All you need to do is follow the path and tick off the nodes one by one, checking to make sure you haven't ticked off any node twice. If no nodes are missing at the end, then the path is Hamiltonian.

—

Paul Chaikin/ *Quanta Magazine*

The time required to run this solution-checking algorithm is proportional to the size of the graph. That puts it into the broader category of polynomial algorithms, whose run times increase as polynomial functions of the graph size. Polynomial growth is tame compared to exponential growth, so polynomial algorithms remain viable even on large graphs. "It's dramatically more efficient," Carmosino said.



—

Samuel Velasco/*Quanta Magazine*

The Hamiltonian path problem has a stark asymmetry to it: You can verify a correct solution using a fast polynomial algorithm, but to find a solution you'll need a slow exponential algorithm. That asymmetry may not seem surprising — it's easier to recognize an artistic masterpiece than to create one, easier to check a mathematical proof than to prove a new theorem — yet not all computational problems have this asymmetric character. In fact, a problem very similar to finding Hamiltonian paths behaves quite differently.

Suppose you're again given a graph, but now you're asked to find an "Eulerian path" that crosses every edge exactly once. Again, there's a polynomial algorithm for checking possible solutions, but this time there's also a polynomial algorithm for solving the problem. No asymmetry here. In complexity theory, some paths seem easier to find than others.

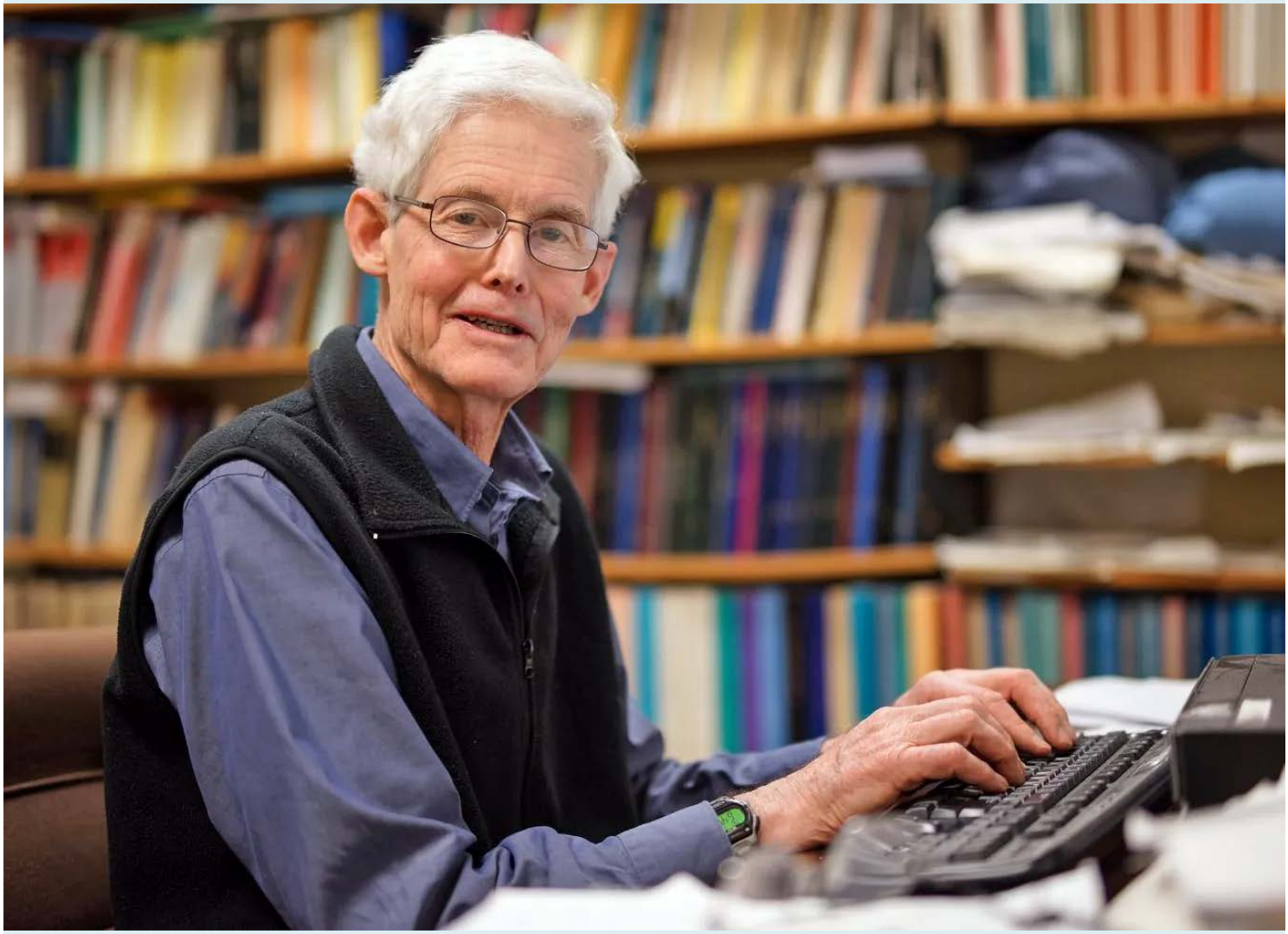
Both the Hamiltonian path problem and the Eulerian path problem are in the complexity class NP, defined to include all problems whose solutions can be checked by polynomial algorithms. The Eulerian path problem also falls into the class P because a polynomial algorithm can solve it, but to all appearances, that's not true for the Hamiltonian path problem. Why should these two graph problems, so superficially similar, differ so dramatically? That's the essence of the P versus NP problem.

Paul Chaikin/*Quanta Magazine*

Universally Hard

At first, complexity classes seemed like convenient categories for sorting problems that were similar but not directly related — nobody suspected that finding Hamiltonian paths had anything to do with other hard computational problems.

Then in 1971, within a year of relocating to the University of Toronto after being denied tenure in the United States, the complexity theorist Stephen Cook published an extraordinary result. He identified a particular NP problem with a strange feature: If there's a polynomial algorithm that can solve that problem, it can also solve every other problem in NP. Cook's "universal" problem, it seemed, was a lone column propping up the class of apparently hard problems, separating them from the easy problems below. Crack that one problem, and the rest of NP will come crashing down.



Stephen Cook formulated the P versus NP problem in the early 1970s, along with Leonid Levin and Richard Karp.

BBVA Foundation

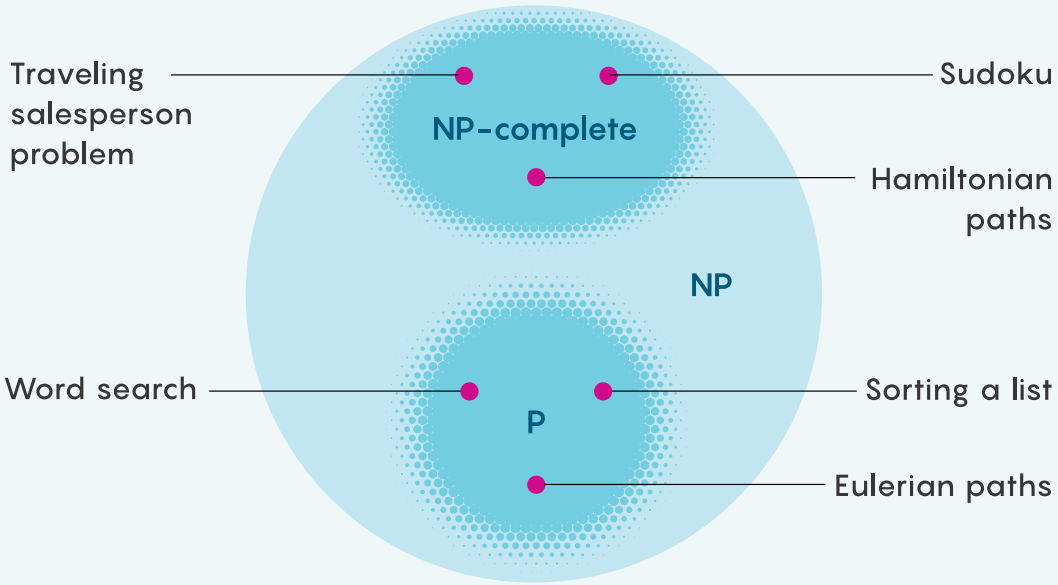
Cook suspected that there was no fast algorithm for his universal problem, and he said as much midway through the paper, adding, "I feel that it is worth spending considerable effort trying to prove this conjecture." "Considerable effort" would turn out to be an understatement.

Around the same time, a graduate student in the Soviet Union named Leonid Levin proved a similar result, except that he identified several different universal problems. In addition, the American complexity theorist Richard Karp proved that the universality property identified by Cook (and Levin, though Karp and Cook didn't know of Levin's work until years later) was itself all but universal. Nearly every NP problem without a known polynomial algorithm — that is, nearly every easily checkable problem that seemed hard — had the same property, which became known as NP-completeness.

This means all NP-complete problems — the Hamiltonian path problem, sudoku, and thousands of others — are in a precise sense equivalent. "You have all these different natural tasks, and they all magically turn out to be the same task," Ilango said. "And we still don't know whether that same task is possible or not."

Settling the difficulty of any NP-complete problem would be enough to resolve the P versus NP question. If P ≠ NP, the distinction between easy and hard problems is held up by thousands of columns that are all equally strong. If P = NP, the whole edifice is teetering on the brink of collapse, just waiting for a single piece to fall.

Nearly all seemingly hard NP problems are NP-complete: The difficulty of any one is equivalent to the P versus NP question.



—

Samuel Velasco/ *Quanta Magazine*

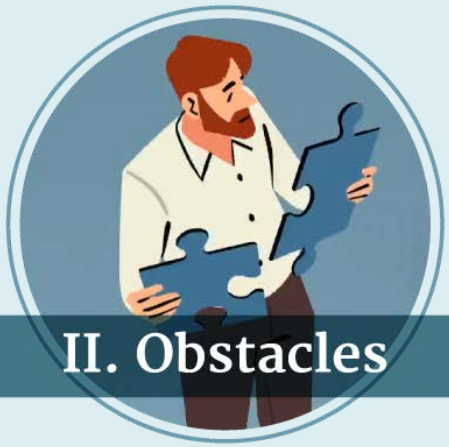
Cook, Levin and Karp had unified what seemed like many unrelated problems. Now all complexity theorists had to do was solve one problem: Does P = NP or not?

Fifty years later, the question remains unanswered. Kabanets likened reasoning about the limits of computation to surveying a vast territory without any sense of the big picture. A being of unlimited computational power could peer down from a mountaintop and take in the whole landscape at once, but mere mortals can't count on that kind of advantage. "Those of us at the bottom of the mountain can try to maybe jump up and down for a better view," he said.

Suppose that P = NP. To prove it, researchers would need to find a fast algorithm for an NP-complete problem, which might be hiding in some obscure corner of that vast landscape. There's no guarantee they'll find it any time soon: Complexity theorists have occasionally discovered ingenious algorithms for seemingly hard (though not NP-complete) problems only after decades of work.

Now suppose that P ≠ NP. Proving that seems even harder. Complexity theorists would have to establish that no fast algorithm could possibly exist, effectively anticipating and thwarting the best efforts of all future researchers.

Not knowing where to start is part of the problem. But it's not like researchers haven't tried. Over the decades they've attacked the problem from many angles and found the path blocked at every turn. "It's one of the most blatant truths in theoretical computer science," Carmosino said. "When you have a phenomenon that's that durable, you want some explanation."



II. Obstacles

By Carmosino's final year in college, his curiosity had led him from Gödel to a graduate course in complexity theory. He was surprised to realize he was spending more time on homework than on his passion project, a computer program that would learn the narrative structure of fairy tales and generate new ones.

"I thought, 'Wow, I need to take this seriously,'" Carmosino recalled. Before long, he was so absorbed in the subject that his mentor gently suggested he reconsider his post-graduation plans.

"He was like, 'You know, if you want to keep doing this, if you want to keep doing theoretical computer science and complexity theory, you can: It's called grad school,'" Carmosino said. After getting his master's, he moved to San Diego in 2012 to work toward a doctorate supervised by Impagliazzo.

Marco Carmosino's fascination with a major result from the 1990s inspired a breakthrough in meta-complexity 20 years later.

—

Kaitlin Abrahamson

Carmosino's main goal, at first, was to better understand a landmark paper from two decades earlier that had captured his imagination. That paper, by the complexity theorists Alexander Razborov and Steven Rudich, had shown that a certain "natural" strategy for proving P ≠ NP would almost certainly fail, because success would come at a steep cost — the complete breakdown of cryptography — that researchers regarded as very unlikely. Researchers interpreted Razborov and Rudich's result as a barrier to this popular approach to proving P ≠ NP.

This "natural proofs barrier" is just one of many known barriers to solving open problems in complexity theory. Each acts like a roadblock, warning that a seemingly promising path is actually a dead end. Together, these barriers indicate that any proof that resolves the P versus NP problem would have to be radically different than anything used in the past; that's why most researchers believe a solution remains far off. But at least barriers tell them where not to look.

"Complexity theory is both cursed and blessed with so many barriers," Ilango said.

By the time Carmosino encountered the natural proofs barrier, it was nearly 20 years old. But he suspected it had more lessons for researchers. That feeling would one day be vindicated when he and three colleagues proved a surprising result by examining the natural proofs barrier from the perspective of meta-complexity. Their proof was one of a few major results that sparked a new interest in meta-complexity, leading to a flurry of progress in the past several years.

But to follow the trail from the natural proofs barrier to meta-complexity, we'll have to jump back to where we left researchers in the 1970s, when they first began to tackle the P versus NP problem. What made it so hard to prove problems hard?

A Circuitous Path

At first, researchers tried to prove P ≠ NP — that is, prove that some NP problems aren't solvable by any possible polynomial algorithm — using variations on the techniques Turing had used to prove that some problems aren't solvable by any algorithm whatsoever. But they quickly discovered a proof that those methods wouldn't work — the first major barrier to resolving the P versus NP question. So they began to look for another approach, and they soon found one in the work of Turing's contemporary Claude Shannon.

In his master's thesis, Claude Shannon developed a theoretical model of computation based on electrical circuits.

—

Courtesy of MIT Museum

Shannon, who grew up in a small town in northern Michigan, seemed an unlikely figure to usher in the information age. Yet he exemplified the interdisciplinary nature of the emerging discipline of computer science, feeling equally at home in electrical engineering and mathematical logic. In his master's thesis, Shannon showed how circuits made of electromechanical switches could represent logical expressions involving Boolean variables — quantities that can take on only two values (such as true or false, or 1 and 0).

In these expressions, Boolean variables are linked together by the "logic gates" AND, OR and NOT. The elementary expression A AND B, for instance, is true when both A and B are true, and false otherwise; A OR B, on the other hand, is true if at least one of the two variables is true. A NOT gate is even simpler: It inverts the value of a single variable. With enough of these basic building blocks, you can perform any computation whatsoever.

"When you look at your computer, at the end of the day, what is it doing? It's running a circuit," Ilango said.

Shannon's work suggested a new way for theorists to think about the difficulty of computational problems, called "circuit complexity," even though the circuits in question are just mathematical abstractions. For a while, researchers thought this approach could be the way to resolve P versus NP, but eventually the trail ran up against the natural proofs barrier.

The building blocks of the Harvard Mark I computer, pictured in 1944, were electromechanical switches like those Shannon analyzed in his thesis.

—

RBM Vintage Images/Alamy Stock Photo

The circuit complexity framework requires rethinking the most basic concepts in Turing's model of computation. Here, instead of computational problems and the algorithms that solve them, researchers consider Boolean functions and the circuits that compute them. A Boolean function takes in Boolean variables — trues and falses, 1s and 0s — and outputs either true or false, 1 or 0. And like an algorithm, a circuit describes a procedure for producing an output given any input.

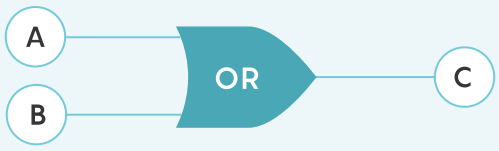
"My understanding is that people began working on circuit complexity because they decided that Turing machines were too complicated," said Ryan Williams, a complexity theorist at MIT. "We can study circuits gate by gate."

Just as there can be many algorithms for solving any given computational problem, some faster than others, many different circuits can compute any given Boolean function, some with fewer gates than others. Researchers define the circuit complexity of a function as the total number of gates in the smallest circuit that computes it. For a function with a fixed number of input variables, circuit complexity is also a fixed number — higher for some functions than for others.

Different Circuits, Same Results

The behavior of any Boolean function can be summarized in a truth table. But vastly different circuits can produce the same truth table.

A single OR gate outputs true if either input is true.

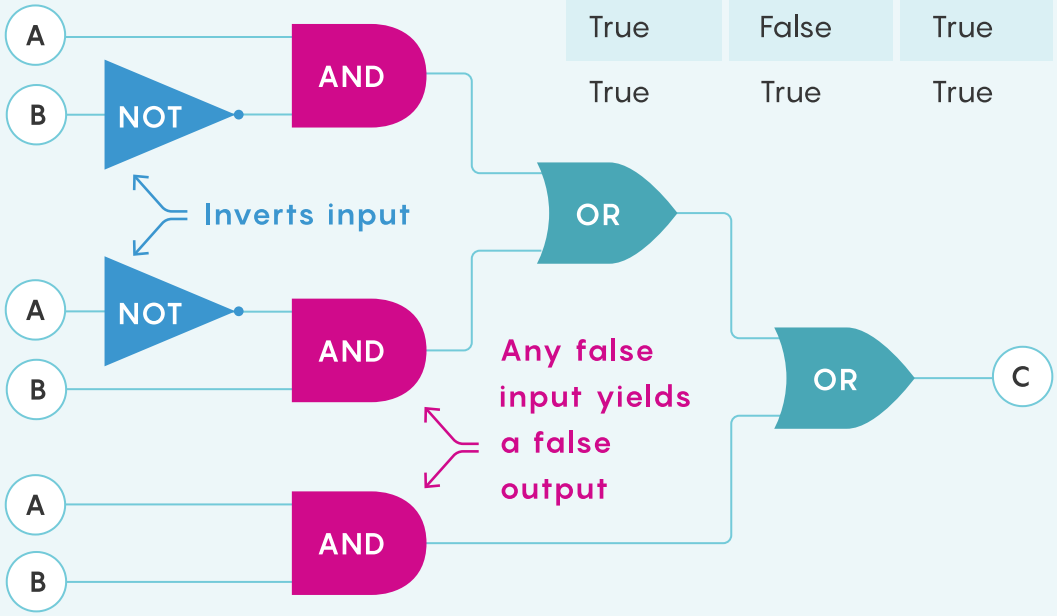


Truth table

If **A** is...	and **B** is...	**C** is...
False	False	False
False	True	True
True	False	True
True	True	True

A complex circuit can produce the same truth table.

If **A** is...	and **B** is...	**C** is...
False	False	False
False	True	True
True	False	True
True	True	True



Merrill Sherman/ *Quanta Magazine*

But in many cases, you can consider more complicated versions of the same function by increasing the number of input variables, just as you can make the Hamiltonian path problem harder by considering larger graphs. Researchers then consider the same question they ask when studying algorithm run times: Does the minimum number of gates needed to compute a Boolean function grow polynomially or exponentially as the number of input variables increases? Researchers call these two categories of functions "easy to compute" and "hard to compute," respectively.

An easy-to-compute Boolean function is similar to a computational problem in the class P — one that can be solved by an algorithm that runs in polynomial time. But there are also functions analogous to hard NP problems, where the best way researchers have discovered to compute progressively larger versions requires an exponentially increasing number of gates, yet the answer can be easily checked. If complexity theorists could prove that there really is no better way to compute such a function, that would imply $P \neq NP$.

This was the strategy that most complexity theorists pursued in the 1980s. And the odds were on their side. Shannon had proved in 1949 that almost every Boolean truth table (which is just a long list of all possible inputs and outputs of a Boolean function of fixed size) has circuit complexity that's practically as high as possible. He used a stunningly simple argument: There are far fewer ways to combine a small number of gates than there are ways to combine many gates.

"There just aren't enough small circuits to go around," Aaronson said.

So complexity theorists found themselves in a curious situation. If nearly every truth table has high circuit complexity, then nearly every Boolean function must be hard to compute. Researchers just had to identify a single such function that was also known to be in the class NP. How hard could that be?

Crypto Bros

At first, progress was rapid. In 1981, Sipser and two collaborators proved that a certain Boolean function was definitely hard to compute if they used circuits with certain restrictions on how gates could be arranged.

"The fantasy was that you would be able to prove things about these restricted models, and then build on what you've learned to work with fewer and fewer restrictions," Sipser said.

Michael Sipser helped prove a milestone result with restricted circuit models in 1981, but progress eventually stalled.

—

Bryce Vickmark

In 1985, Razborov took the next big step. He'd just started graduate school in Moscow and joined the effort accidentally while tackling a problem in a different branch of mathematics, where it turned out that resolving the P versus NP problem was a prerequisite.

"I was simply lucky that I did not know how difficult this problem was," Razborov said. "Otherwise maybe I would not have even started."

Razborov analyzed circuits containing only AND and OR gates, and proved that a particular function was hard to compute using such circuits, no matter how gates were arranged — what's more, that function was known to be NP-complete. All researchers had to do to resolve P versus NP was extend Razborov's techniques to circuits with NOT gates.

"There was a sort of universal feeling that one more step, one more strike, and we will get it," Razborov said. But that's not what happened. Razborov himself proved that his method would fail if NOT gates were added to the mix, and nobody could find another way forward. As the years passed, he began to wonder why the trail had petered out.

In the United States, Rudich was pondering the same question. He and Impagliazzo were college classmates who had gone on to graduate school together. Their friendship had been sparked by a shared fascination with Gödel and Turing's self-referential proofs and their implications for the foundations of mathematics and computer science.

"Our joke was we were going to get a button that said 'self-reference,'" Impagliazzo said.

In 1994, Alexander Razborov (left) and Steven Rudich discovered the natural proofs barrier, which explained why previous attempts to prove P ≠ NP had failed.

—

Jean Lachat (left); Courtesy of Carnegie Mellon University

As graduate students, both Rudich and Impagliazzo worked on the complexity-theoretic foundations of cryptography, a subject that offers perhaps the best practical motivation for attempting to prove P ≠ NP. Cryptographers conceal secret messages by cloaking them in "pseudorandomness" — a message encrypted this way will look like a random jumble of numbers to any eavesdropper, but it can still be decoded by the intended recipient. But how can you be sure that a would-be eavesdropper will find it too difficult to break the code?

That's where complexity theory comes in. The encryption methods most widely used today are all based on seemingly hard NP problems — to decrypt the message, an attacker would need an as-yet-undiscovered fast algorithm for solving the problem. To establish that these methods are truly secure, one thing you'd need to do is prove that P ≠ NP. Without a proof, Sipser said, all you can do is "hope that whoever you're trying to keep the secret from is not a better mathematician than you are."

While fascinating in its own right, cryptography seemed far removed from the self-referential arguments that had first drawn Rudich and Impagliazzo into the field. But as Rudich struggled to understand why the circuit complexity approach had stalled, he began to realize that the two subjects weren't so far apart after all. The strategy researchers had adopted in their attempts to prove P ≠ NP had a self-defeating character reminiscent of Gödel's famous proposition "this statement is unprovable" — and cryptography could help explain why. In Russia, Razborov discovered a similar connection around the same time. These were the seeds of the natural proofs barrier.

The tension at the heart of the natural proofs barrier is that the task of distinguishing high-complexity functions from low-complexity ones is similar to the task of distinguishing true randomness from the pseudorandomness used to encrypt messages. We'd like to show that high-complexity functions are categorically different from low-complexity functions, to prove P ≠ NP. But we'd also like for pseudorandomness to be indistinguishable from randomness, to be confident in the security of cryptography. Maybe we can't have it both ways.

A Cruel Joke

In 1994, Razborov and Rudich realized that they had hit upon similar insights, and they began working together to combine their results. They first observed that all previous attempts to prove P ≠ NP using circuit complexity had adopted the same general strategy: Identify a special property of an NP-complete Boolean function, then prove that no easy-to-compute function could possibly share that property. That would show that the chosen NP-complete function was truly hard to compute, proving P ≠ NP.

Sipser, Razborov and others had used this same strategy successfully to prove their more limited results, and in every case, the special property that the researchers identified was shared by most Boolean functions. Razborov and Rudich coined the term "natural proof" to refer to this case where the property was widely shared, simply because there was no known alternative. If "unnatural" proofs were possible, they'd have to be very counterintuitive and deserving of the name.

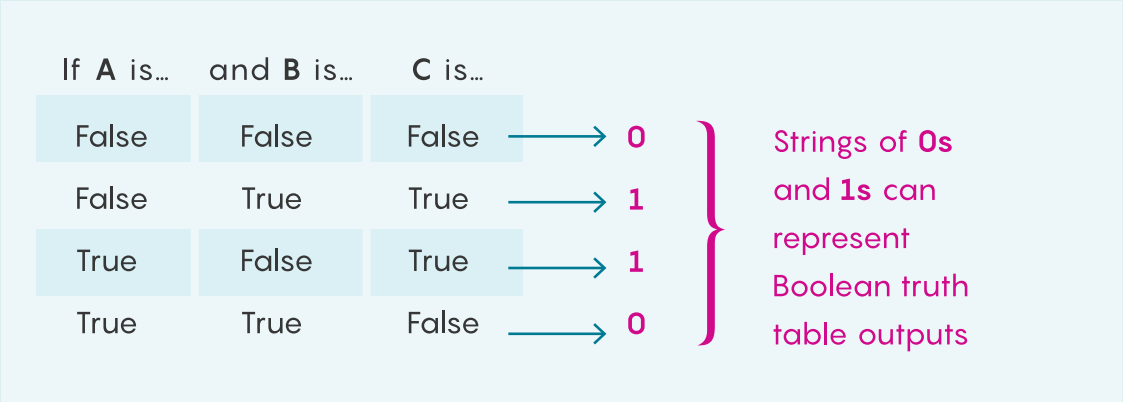
Razborov and Rudich then proved their main result: A natural proof of P ≠ NP would require a very comprehensive understanding of how easy-to-compute and hard-to-compute functions differ, and that knowledge could also fuel a fast algorithm for spotting easy-to-compute functions even if they're superficially complicated. If complexity theorists had succeeded in a natural proof of P ≠ NP, they

would have discovered a nearly infallible way to glance at an arbitrary truth table and determine whether the corresponding function had high or low circuit complexity — a much stronger and more general result than they had set out to prove.

"You almost can't help but get more than you bargained for," Carmosino said.

It's as if you tried to fact-check a specific statement, but every attempt turned into a blueprint for a general-purpose lie detector — it would seem too good to be true. For complexity theorists, the surprising power of natural proofs likewise made success seem less likely. But if such a proof had succeeded, the unexpected consequences would be bad news for cryptography, because of the connection between circuit complexity and pseudorandomness.

To understand this connection, imagine looking at the output column in the truth table of a Boolean function with many input variables and replacing every "true" with 1 and every "false" with 0:



Merrill Sherman/*Quanta Magazine*

If the Boolean function has high circuit complexity, that long list of outputs will in principle be indistinguishable from a truly random string of 0s and 1s — one obtained by repeatedly flipping a coin, say. But if the function has low circuit complexity, the string must have a simple, succinct description even if it looks complicated. That makes it very similar to the pseudorandom strings used in cryptography, whose succinct description is the secret message buried in that apparent randomness.

Pseudorandomness

A pseudorandom string of 0s and 1s looks similar to a truly random string, but it has a simple description.

RANDOM

Flip a coin 16 times:

1011100010011000

There's no way to describe this string more succinctly.

PSEUDORANDOM

Start with a simple pattern of alternating 1s and 0s:

1011001110001111

Then flip the bits in the second half:

1011001101110000

This string looks random but has a simple description.

—

Merrill Sherman/ *Quanta Magazine*

So Razborov and Rudich's result showed that any natural proof of P ≠ NP would also yield a fast algorithm that could distinguish pseudorandom strings containing hidden messages from truly random ones. Secure cryptography would be impossible — precisely the opposite of what researchers hoped to establish by proving P ≠ NP.

On the other hand, if secure cryptography is possible, then natural proofs aren't a viable strategy for proving P ≠ NP — a prerequisite for secure cryptography. That's the natural proofs barrier in a nutshell. It seemed as if complexity theorists were on the receiving end of a cruel joke.

"If you believe in hardness, then you should believe that it's hard to prove hardness," Kabanets said.

Into the Metaverse

That connection between the implications of the P ≠ NP conjecture and the difficulty of proving it was intriguing, but tricky to pin down. For one thing, the natural proofs barrier only blocked off one approach to proving P ≠ NP. For another, it linked the difficulty of proving P ≠ NP not to P ≠ NP itself, but to the existence of secure cryptography — a closely related but not quite equivalent problem. To truly understand the connection, researchers would have to get comfortable with meta-complexity.

"There's this intuition that 'oh, because P ≠ NP, it is difficult to prove that P ≠ NP,'" Williams said. "But in order to even make sense of this intuition, you've got to start thinking about the task of proving something like P ≠ NP as a computational problem."

That's what Kabanets did as a graduate student. He'd grown up in Ukraine, and he finished his undergraduate studies in computer science two years after the fall of the Soviet Union. In the turmoil that followed, he had few opportunities to pursue the theoretical topics that interested him most.

As a graduate student, Valentine Kabanets wrote an influential paper about a quintessential meta-complexity problem that he dubbed the minimum circuit size problem (MCSP).

—

Antonina Kolokolova

"I wanted to do something more academic," Kabanets recalled. "And I was also curious to see the world." He moved to Canada for graduate school, and that's where he learned about the natural proofs barrier. Like Carmosino, Kabanets was smitten with the result. "It seemed very profound that you have this connection," he said.

In 2000, toward the end of his time in grad school, he found that the natural proofs barrier kept coming up in his conversations with Jin-Yi Cai, a complexity theorist who was visiting Toronto on sabbatical at the time. They began to see the barrier not as a roadblock but as an invitation — an opportunity to investigate precisely how hard it was to prove problems hard. The paper in which they laid out this new perspective would become one of the most influential early works in the nascent field of meta-complexity.

Kabanets and Cai's paper highlighted a computational problem implicit in Razborov and Rudich's formulation of the natural proofs barrier: Given the truth table of a Boolean function, determine whether it has high or low circuit complexity. They dubbed this the minimum circuit size problem, or MCSP.

MCSP is a quintessential meta-complexity problem: a computational problem whose subject is not graph theory or another external topic, but complexity theory itself. Indeed, it's like a quantitative version of the question that drove complexity theorists to tackle P versus NP using the circuit complexity approach in the 1980s: Which Boolean functions are hard to compute, and which are easy?

"If we came up with an MCSP algorithm, that would be like a way of automating what we're doing in complexity theory," Impagliazzo said. "It should at least give us a tremendous insight into how to do our job better."

Complexity theorists don't worry about this magical algorithm putting them out of work — they don't think it exists at all, because Razborov and Rudich showed that any such algorithm for distinguishing high-complexity truth tables from low-complexity ones would make cryptography impossible. That means MCSP is likely a hard computational problem. But how hard is it? Is it NP-complete, like the Hamiltonian path problem and nearly every other problem that researchers struggled with in the 1960s?

For problems in NP, "how hard is it?" is usually easy enough to answer, but MCSP seemed to be a strange outlier. "We have very few 'floating around' problems which have not been connected to this island of NP-complete problems, even though they seem to be hard," Kabanets said.

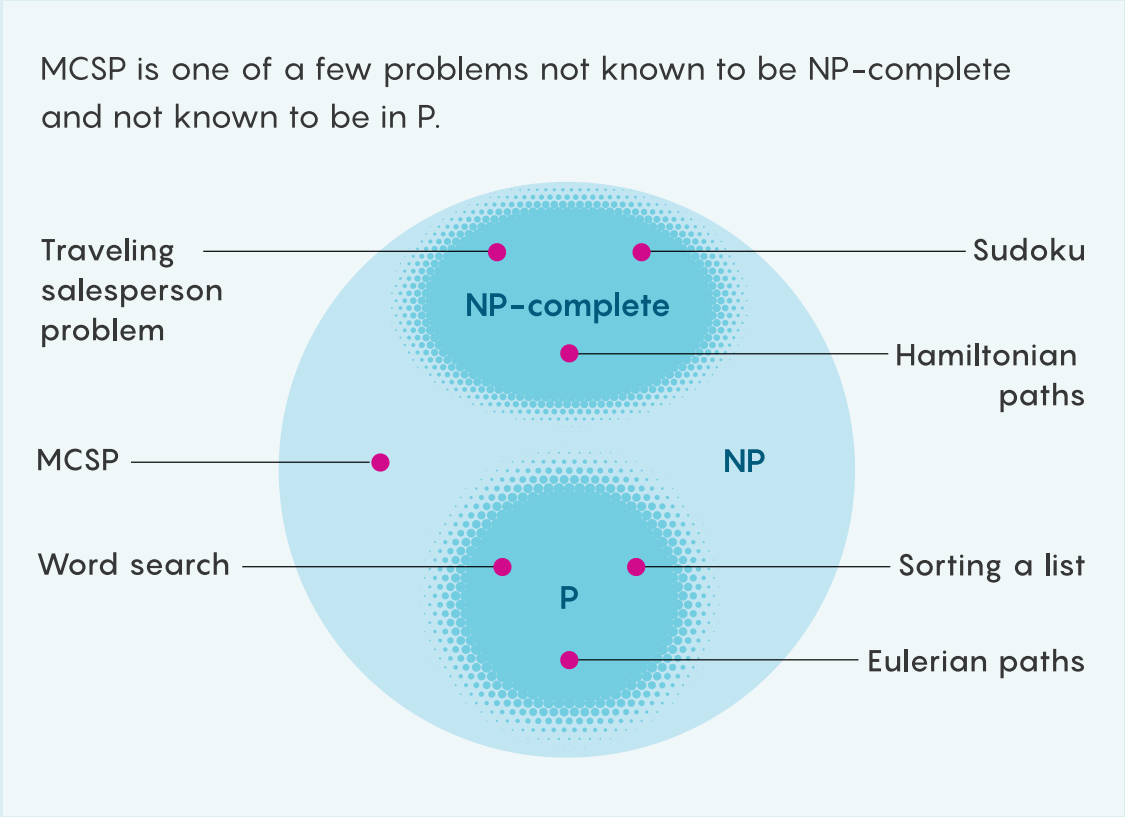
Kabanets knew that he and Cai weren't the first to consider the problem they had dubbed MCSP. Soviet mathematicians had studied a very similar problem beginning in the 1950s, in an early attempt to understand the intrinsic difficulty of different computational problems. Leonid Levin had wrestled with it while developing what would become the theory of NP-completeness in the late 1960s, but he couldn't prove it NP-complete, and he published his seminal paper without it.

After that, the problem attracted little attention for 30 years, until Kabanets and Cai noted its connection to the natural proofs barrier. Kabanets didn't expect to settle the question himself — instead he wanted to explore why it had been so hard to prove that this seemingly hard problem about computational hardness was actually hard.

"It is, in a sense, meta-meta-complexity," said Rahul Santhanam, a complexity theorist at the University of Oxford.

But was it hardness all the way down, or was there at least some way to understand why researchers hadn't succeeded in proving that MCSP was NP-complete? Kabanets discovered that, yes, there was a reason: The difficulty of understanding circuit complexity acts like a barrier to any known strategy for proving the NP-completeness of MCSP — a problem that's itself about the difficulty of understanding circuit complexity. The twisted, self-defeating logic of the natural proofs barrier seemed inescapable.

It's also possible that MCSP is not NP-complete, but that too seems unlikely — certain simpler variants of the problem are already known to be NP-complete.



—

Samuel Velasco/ *Quanta Magazine*

“We just don’t have a nice place to put it that directly relates it to all the other problems we study,” Impagliazzo said.

Kabanets had illuminated the strange behavior of MCSP, but he didn’t know how to make further progress. Meta-complexity research slowed to a trickle. It would flourish again 16 years later, when researchers discovered a surprising connection to another fundamental question: How hard is it to solve problems if you only care about getting the right answer most of the time?

War of the Worlds

For everyday problems, answers that work most of the time are often good enough. We plan our commutes for typical traffic patterns, for instance, not for worst-case scenarios.

Most complexity theorists are harder to satisfy: They’re only content to declare a problem easy if they can find a fast algorithm that gets the right answer on every possible input. That standard approach classifies problems according to what researchers call “worst-case” complexity. But there’s also a theory of “average-case” complexity, in which problems are considered easy if there’s a fast algorithm that gets the right answer on most inputs.

The distinction matters to cryptographers. Imagine a computational problem that’s easy to solve for nearly every input, except for a few stubborn cases where the best algorithm fails. Worst-case complexity theory considers that a hard problem, yet for cryptography it’s useless: If only some of your messages are hard to decipher, what’s the point?

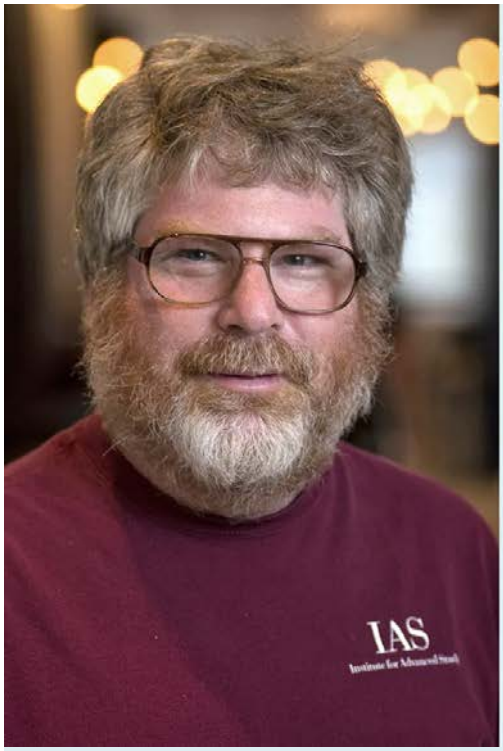
It was actually Levin who initiated the rigorous study of average-case complexity, a decade after his pioneering work on NP-completeness. In the intervening years, he had run afoul of Soviet authorities — he was an irreverent troublemaker who would occasionally undermine patriotic activities in his Communist party youth group. In 1972, he was denied his doctorate for explicitly political reasons.

“To be successful in the Soviet Union as a young researcher, you could not be very opinionated, and it’s hard to imagine Leonid not being opinionated,” Impagliazzo said.

Levin emigrated to the United States in 1978, and in the mid-1980s he turned his attention to average-case complexity. He began working with others to further develop the theory, including Impagliazzo, a graduate student at the time. But even as they made progress, Impagliazzo found that researchers often talked past each other. He wanted to get everyone on the same page, and it didn’t help that Levin’s papers were famously succinct — the one that initiated the field of average-case complexity was less than two pages long.

“I was going to do a translation of Leonid’s work into more accessible technical terms,” Impagliazzo said. He decided to start with a short, playful overview of the big picture before diving into the math. “That kind of took over the paper, and it’s the only part anyone remembers anyway.”

The paper, published in 1995, became an instant classic. Impagliazzo coined whimsical names for five worlds distinguished by different degrees of computational hardness and different cryptographic capabilities. We live in one of these worlds, but we don’t know which.



Leonid Levin (right) initiated the study of average-case complexity in the mid-1980s. Russell Impagliazzo later made the subject more accessible in an iconic paper about the five computational worlds we may live in.

—

Courtesy of Simons Foundation (left); Cydney Scott/Boston University Photography

Ever since Impagliazzo's paper appeared, researchers have dreamed of eliminating parts of his miniature multiverse — narrowing the space of possibilities by proving that some of the worlds aren't possible after all. Two worlds are especially tempting targets: those where cryptography is impossible even though P ≠ NP.

In one of these worlds, called Heuristica, all NP-complete problems are easy to solve on most inputs, but fast algorithms occasionally make mistakes, so these problems are still considered hard by the standards of worst-case complexity theory. This is the world in which cryptography is impossible because almost every code is easily cracked. In the other world, called Pessiland, cryptography is impossible for a different reason: Every problem is hard in the average-case sense, but encrypting a message makes it illegible even for the intended recipient.

These two worlds turn out to be closely tied to meta-complexity problems — in particular, the fate of Heuristica is linked to the long-standing question of whether MCSP is NP-complete. The question that fascinated Kabanets and stumped Levin so long ago is no mere curiosity: There's a whole world at stake.

To rule out Heuristica, researchers would have to collapse the distinction between worst-case and average-case complexity — that is, they'd have to prove that any hypothetical algorithm that solves an NP-complete problem correctly on most inputs can actually solve it in all cases. This kind of connection, called a worst-case to average-case reduction, is known to exist for certain problems, but none of them are NP-complete, so those results don't imply anything more general. Eliminating Heuristica would take cryptographers halfway to realizing the dream of secure encryption based on the single assumption that P ≠ NP.

But destroying a world is no small feat. In 2003, two complexity theorists showed that existing approaches to proving worst-case to average-case reductions for known NP-complete problems would imply outlandish consequences, suggesting that such proofs probably aren't possible.

Researchers would have to find another approach, and they now think MCSP might be just the problem they need. But that wouldn't become clear for over a decade. The first glimpse of the connection emerged from Marco Carmosino's persistent fascination with the natural proofs barrier.

III. Opportunities

Carmosino first encountered meta-complexity research as a graduate student through a 2013 paper by Kabanets and four other researchers, which further developed the approach to the natural proofs barrier that Kabanets had pioneered more than a decade earlier. It only bolstered his conviction that there was still more to learn from Razborov and Rudich's classic paper.

"I was obsessed with that paper at the time," Carmosino said. "Nothing has changed."

The obsession finally bore fruit during a visit to a semester-long workshop at the University of California, Berkeley, where he spent most of his time talking to Impagliazzo, Kabanets and Antonina Kolokolova, a complexity theorist at Memorial University of Newfoundland who'd collaborated with Kabanets on the 2013 paper. Carmosino had worked with the three of them once before, and that successful collaboration gave him the confidence to pepper them with questions about the topic that fascinated him the most.

"He was bugging people in a good way," Kabanets recalled.

At first, Carmosino had new ideas for proving NP-completeness for the version of MCSP that had appeared in Razborov and Rudich's paper on the natural proofs barrier. But those ideas didn't pan out. Instead, an off-the-cuff remark by Impagliazzo made the four researchers realize that the natural proofs barrier could yield more powerful algorithms than anybody had realized — there was a secret map etched into the roadblock.

In 2016, Antonina Kolokolova worked with Carmosino, Impagliazzo and Kabanets to prove a surprising connection between MCSP and learning that drew new attention to meta-complexity.

—

Colette Philips

In a 2016 paper, the four researchers proved that a certain kind of average-case MCSP algorithm could be used to construct a worst-case algorithm for identifying patterns hidden in random-looking strings of digits — a task that computer scientists refer to as "learning." It's a striking result because learning intuitively seems harder than the binary classification task — high complexity or low complexity — performed by an MCSP algorithm. And, surprisingly, it linked the worst-case complexity of one task to the average-case complexity of the other.

"It wasn't obvious that such a connection would exist at all," Impagliazzo said.

A fast algorithm for MCSP is purely hypothetical for general Boolean circuits: It can't exist unless MCSP turns out to be an easy computational problem, despite all evidence to the contrary, and that means the learning algorithm implied by the four researchers' paper is equally hypothetical.

But for some simpler versions of MCSP — distinguishing high-complexity truth tables from low-complexity ones when there are specific restrictions on the circuits — fast algorithms have been known for many years. Carmosino, Impagliazzo, Kabanets and Kolokolova's paper showed that these algorithms could be transformed into learning algorithms that were likewise restricted but still more powerful than any that researchers had previously understood at such a rigorous theoretical level.

"Somehow their self-referential flavor enables you to do things that seemingly you can't do with more standard problems," Ilango said.

The result grabbed the attention of complexity theorists working on other topics. It was also a preview of further connections between meta-complexity and average-case complexity that would emerge over the coming years.

Most of all, it was a testament to how far researchers can get by asking simple questions about barriers that at first seem only to obstruct their progress.

"This kind of duality is a theme throughout at least the last 30 or 40 years of complexity," Impagliazzo said. "The obstacles are often the opportunities."

Partial Credit

Progress has only accelerated in the years since Carmosino and his colleagues published their paper.

100 Years of Hardness

1921 Hilbert proposes his program for the foundations of mathematics.

1931 Gödel proves his incompleteness theorem.

1936 Turing proves that uncomputable problems exist.

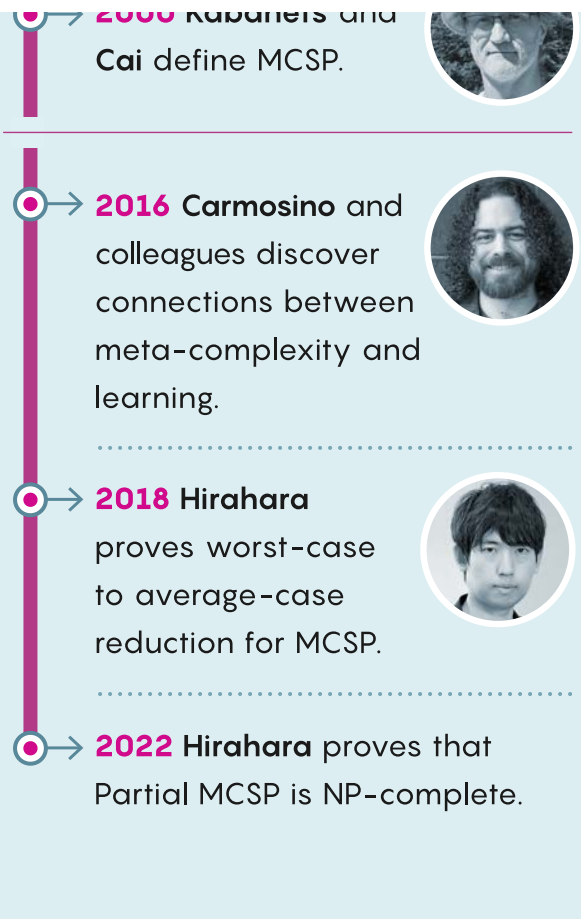
1971–1973 **Cook**, **Levin**, and Karp formulate the P versus NP problem.

1980s Researchers attempt to prove P ≠ NP using circuit complexity.

1994 Razborov and **Rudich** discover the natural proofs barrier.

1995 Impagliazzo describes the five possible worlds of average-case complexity

2009 Kabanets and



Samuel Velasco/ *Quanta Magazine*

"New things are happening," Kolokolova said. "There are lots of really, really bright junior researchers."

Ilango is one of these young researchers — in his first three years of graduate school, he's attacked the daunting open problem of proving MCSP NP-complete using a two-pronged strategy: proving NP-completeness for simpler versions of MCSP, as circuit complexity researchers did when attacking P versus NP in the 1980s, while also proving NP-completeness for more complicated versions, which intuitively seem harder and thus are perhaps easier to prove hard.

Ilango credits his interest in meta-complexity to Eric Allender, a complexity theorist at Rutgers University and one of the few researchers who continued working on meta-complexity in the 2000s and early 2010s. "His enthusiasm was infectious," Ilango said.

Another young researcher inspired by Allender is Shuichi Hirahara, now a professor at the National Institute of Informatics in Tokyo. While still a graduate student in 2018, Hirahara revealed the true extent of the relationship between meta-complexity and average-case complexity that Carmosino and his co-authors had discovered. Those four researchers had found a connection between the average-case complexity of one problem — MCSP — and the worst-case complexity of another — Boolean learning. Hirahara developed their techniques further to derive a worst-case to average-case reduction for MCSP. His result implies that a hypothetical average-case MCSP algorithm like the one Carmosino and his colleagues had considered would actually be powerful enough to solve a slightly different version of MCSP without making any mistakes.

Hirahara's result is exciting because many researchers suspect that MCSP is NP-complete, unlike all other problems for which worst-case to average-case reductions are known. If they can extend Hirahara's results to cover all average-case algorithms and then prove that MCSP is NP-complete, that would prove we don't live in Heuristica.

"That would really be an earth-shattering result," Santhanam said.

Proving that MCSP is NP-complete may seem like a tall order — after all, the question has been open for over 50 years. But after a breakthrough last year by Hirahara, researchers are now much closer than anyone would have expected a few years ago.

Hirahara proved NP-completeness for a variant of the problem called partial MCSP, in which you ignore certain entries in each truth table. His proof built on methods developed by Ilango to show that partial MCSP was equivalent to a seemingly unrelated problem involving a cryptographic technique called secret sharing. This is a way to divide an encrypted message among many people so that it can only be decoded if a certain fraction of them work together.

For any real application in cryptography, you'd want to know that fraction in advance, but with the help of extra cryptographic tricks, you can construct a frustrating scenario in which it's hard just to figure out how many people need to cooperate. Hirahara found a way to prove that this contrived cryptographic problem was NP-complete and then showed that the proof implied the NP-completeness of partial MCSP as well.

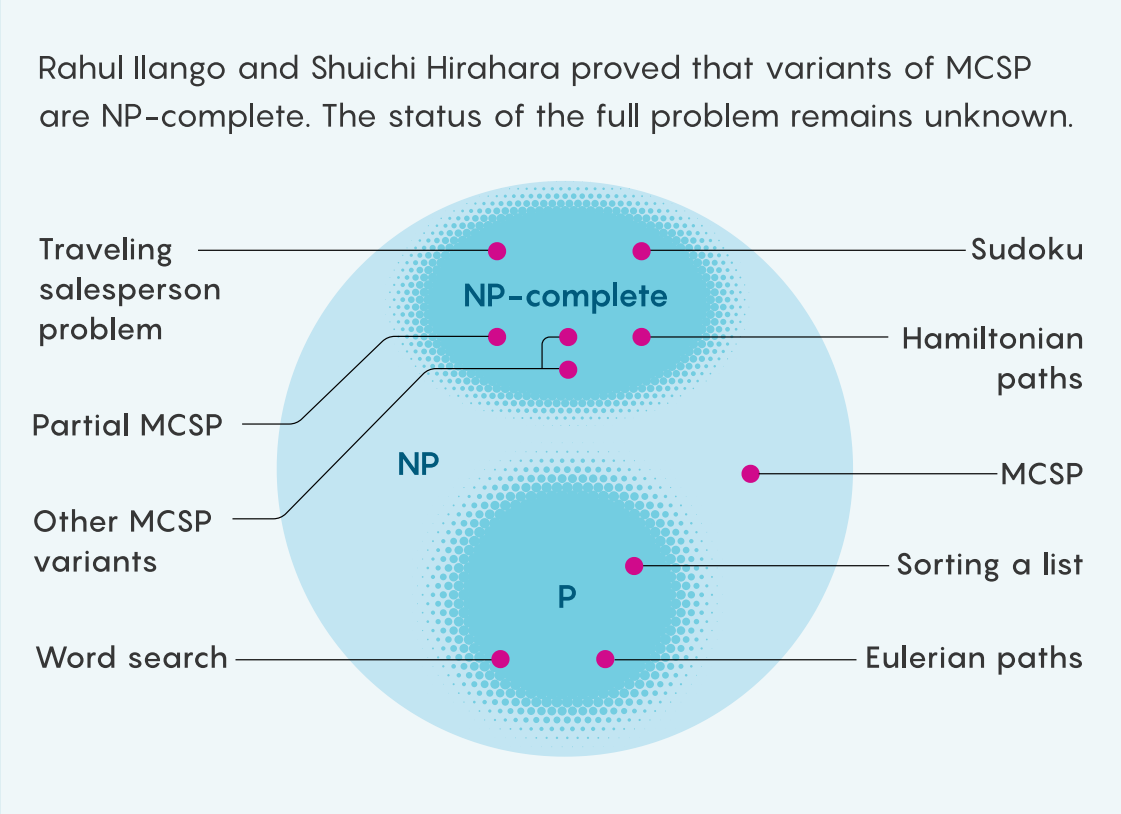
Rahul Ilango (left) and Shuichi Hirahara recently developed new cryptographic techniques to prove that variants of MCSP are NP-complete.

—

Jennifer Krupa (left); Takuma Imamura

This result energized researchers in meta-complexity even more than Hirahara's earlier work, and other researchers also took notice — the complexity theorist and blogger Lance Fortnow dubbed it the result of the year. That's because tackling such "partial function" versions of computational problems has been a key intermediate step in other NP-completeness proofs.

"It's amazing work," Williams said. "Everyone thought that these partial problems were roughly the same difficulty as the full problem."



—

Samuel Velasco/ *Quanta Magazine*

Impediments remain to proving NP-completeness for the full version of MCSP. But none are the sort of barriers that suggest an entirely new toolkit is needed — it may just be a matter of finding the right way to combine known techniques. A proof would finally settle the status of one of the few problems that have resisted classification for as long as complexity theory has existed. Over email, Levin wrote: "It would humble me showing I was stupid for not having been able to see it :-)."

The Missing Pieces

MCSP isn't even the only meta-complexity problem that's spurred a major breakthrough. In 2020, the Cornell Tech cryptographer Rafael Pass and his graduate student Yanyi Liu discovered a connection between a different meta-complexity problem and a fundamental cryptographic protocol that defines the boundary between Heuristica and Pessiland, the worst of Impagliazzo's worlds (where NP-complete problems are hard in the average-case sense but cryptography is still impossible). That makes the problem they studied a prime candidate for an assault on Pessiland, and their more recent work indicates that it could work against Heuristica as well.

"Different pieces of the puzzle are missing," Pass said. "To me it's just magical that these fields are so intimately connected."

Hirahara cautions that challenges still await researchers intent on culling the worlds Impagliazzo conjured up 30 years ago. "I'd like to say that at some point Heuristica and Pessiland will be ruled out, but I'm not sure how close we are," he said.

Many researchers expect that the biggest difficulty will be bridging a seemingly innocuous gap between two different models of average-case complexity. Cryptographers usually study average-case algorithms that make mistakes in both directions, occasionally mislabeling random strings as pseudorandom and vice versa. Hirahara's worst-case to average-case reductions, meanwhile, work for average-case algorithms that only make the first type of error. Subtle distinctions like this can make a world of difference in complexity theory. But despite this hurdle and many others, Allender can't help but sound a note of guarded optimism.

"I try not to let myself be too much of a believer because there's a pretty well-established track record of nothing working," he said. "But we're seeing a lot of really exciting developments — ways to overcome things that looked like barriers."

If there's one lesson researchers have learned from their struggles to answer the P versus NP question — or even just understand it — it's that complexity theory is itself complex. But that challenge is precisely what makes the quest so rewarding.

"It's actually kind of great that it's so hard," Carmosino said. "I'm never going to be bored."

Editor's note: Scott Aaronson is a member of Quanta Magazine*'s advisory board.*