DOI:10.1145/3460351

Advances in algorithms, machine learning, and hardware can help tackle many NP-hard problems once thought impossible.

BY LANCE FORTNOW

Fifty Years of P vs. NP and the Possibility of the Impossible

ON MAY 4, 1971, computer scientist/mathematician Steve Cook introduced the P vs. NP problem to the world in his paper, “The Complexity of Theorem Proving Procedures.” More than 50 years later, the world is still trying to solve it. In fact, I addressed the subject 12 years ago in a *Communications* article, “The Status of the P versus NP Problem.”¹³

The P vs. NP problem, and the theory behind it, has not changed dramatically since that 2009 article, but the world of computing most certainly has. The growth of cloud computing has helped to empower social networks, smartphones, the gig economy, fintech, spatial computing, online education, and, perhaps

most importantly, the rise of data science and machine learning. In 2009, the top 10 companies by market cap included a single Big Tech company: Microsoft. As of September 2020, the first seven are Apple, Microsoft, Amazon, Alphabet (Google), Alibaba, Facebook, and Tencent.³⁸ The number of computer science (CS) graduates in the U.S. more than tripled⁸ and does not come close to meeting demand.

Rather than simply revise or update the 2009 survey, I have chosen to view advances in computing, optimization, and machine learning through a P vs. NP lens. I look at how these advances bring us closer to a world in which $P = NP$, the limitations still presented by P vs. NP, and the new opportunities of study which have been created. In particular, I look at how we are heading toward a world I call “Optiland,” where we can almost miraculously gain many of the advantages of $P = NP$ while avoiding some of the disadvantages, such as breaking cryptography.

As an open mathematical problem, P vs. NP remains one of the most important; it is listed on the Clay Mathematical Institute’s Millennium Problems²¹ (the organization offers a million-dollar bounty for the solution). I close the article by describing some new theoretical computer science results that, while not getting us closer to solving the P vs. NP question, show us that thinking

» key insights

- The P vs. NP problem turned 50 in 2021 and its resolution remains far out of reach. Dramatic advances in algorithms and hardware have allowed us to tackle many NP-complete problems while making little progress breaking cryptographic systems.
- As we continue to make dramatic progress in machine learning and data-centric computing, P vs. NP can give us a valuable lens to view what is and is not possible for the future of learning.
- While P vs. NP began as a way to characterize difficult problems to solve computationally, we now view the problem as a way to chart the future possibilities for our field.

about P vs. NP still drives much of the important research in the area.

The P vs. NP Problem

Are there 300 Facebook users who are all friends with each other? How would you go about answering that question? Let's assume you work at Facebook. You have access to the entire Facebook graph and can see which users are friends. You now need to write an algorithm to find that large clique of friends. You could try all groups of 300, but there are far too many to search them all. You could try something smarter, perhaps starting with small groups and merging them into bigger

groups, but nothing you do seems to work. In fact, nobody knows of a significantly faster solution than to try all the groups, but neither do we know that no such solution exists.

This is basically the P vs. NP question. NP represents problems that have solutions you can check efficiently. If I tell you which 300 people might form a clique, you can check relatively quickly that the 44,850 pairs of users are all friends. *Clique* is an NP problem. P represents problems where you can find those solutions efficiently. We don't know whether the clique problem is in P. Perhaps, surprisingly, *Clique* has a property called NP-complete—that is,

we can efficiently solve the *Clique* problem quickly if and only if $P = NP$. Many other problems have this property, including *3-Coloring* (can a map be colored using only three colors so that no two neighboring countries have the same color?), *Traveling Salesman* (find the shortest route through a list of cities, visiting every city and returning to the starting place), and tens to hundreds of thousands of others.

Formally, P stands for “polynomial time,” the class of problems that one can solve in time bounded by a fixed polynomial in the length of the input. NP stands for “nondeterministic polynomial time,” where one can use a

nondeterministic machine that can magically choose the best answer. For the purposes of this survey, it is best to think of P and NP simply as efficiently computable and efficiently checkable.

For those who want a longer informal discussion on the importance of the P vs. NP problem, see the 2009 survey¹³ or the popular science book based on that survey.¹⁴ For a more technical introduction, the 1979 book by Michael Garey and David Johnson¹⁶ has held up surprisingly well and remains an invaluable reference for those who need to understand which problems are NP-complete.

Why Talk About It Now?

On that Tuesday afternoon in 1971, when Cook presented his paper to ACM Symposium on the Theory of Computing attendees at the Stouffer's Somerset Inn in Shaker Heights, OH, he proved that *Satisfiability* is NP-complete and *Tautology* is NP-hard.¹⁰ The theorems suggest that Tautology is a good candidate for an interesting set not in [P], and I feel it is worth spending considerable effort trying to prove this conjecture. Such a proof would represent a major breakthrough in complexity theory.

Dating a mathematical concept is almost always a challenge, and there are many other possible times where we can start the P vs. NP clock. The basic notions of algorithms and proofs date back to at least the ancient Greeks, but as far as we know they never considered a general problem such as P vs. NP. The basics of efficient computation and nondeterminism were developed in the 1960s. The P vs. NP question was formulated earlier than that, we just didn't know it.

Kurt Gödel wrote a letter¹⁷ in 1956 to John von Neumann that essentially described the P vs. NP problem. It is not clear if von Neumann, then suffering from cancer, ever read the letter, which was not discovered and widely distributed until 1988. The P vs. NP question didn't really become a phenomenon until Richard Karp published his 1972 paper²³ showing that a large number of well-known combinatorial problems were NP-complete, including Clique, 3-Coloring, and Traveling Salesman. In 1973, Leonid Levin, then in Russia, published a pa-

The P vs. NP problem, and the theory behind it, has not changed dramatically, but the world of computing most certainly has.

per based on his independent 1971 research that defined the P vs. NP problem.²⁷ By the time Levin's paper reached the west, P vs. NP had already established itself as computing's most important question.

Optiland

Russell Impagliazzo, in a classic 1995 paper,²⁰ described five worlds with varying degrees of possibilities for the P vs. NP problem:

- **Algorithmica:** $P = NP$ or something “morally equivalent,” such as fast probabilistic algorithms for NP.
- **Heuristica:** NP problems are hard in the worst case but easy on average.
- **Pessiland:** We can easily create hard NP problems, but not hard NP problems where we know the solution. This is the worst of all possible worlds, since we can neither solve hard problems on average nor do we get any apparent cryptographic advantage from the difficulty of these problems.
- **Minicrypt:** Cryptographic one-way functions exist, but we do not have public-key cryptography.
- **Cryptomania:** Public-key cryptography is possible—that is, two parties can exchange secret messages over open channels.

These worlds are purposely not formally defined but rather suggest the unknown possibilities given our knowledge of the P vs. NP problem. The general belief, though not universal, is that we live in Cryptomania.

Impagliazzo draws upon a “you can't have it all” from P vs. NP theory. You can either solve hard NP problems or have cryptography, but you can't have both (you can have neither). Perhaps, though, we are heading to a de facto *Optiland*. Advances in machine learning and optimization in both software and hardware are allowing us to make progress on problems long thought difficult or impossible—from voice recognition to protein folding—and yet, for the most part, our cryptographic protocols remain secure.

In a section called “What if $P=NP$?” from the 2009 survey,¹³ I wrote, “Learning becomes easy by using the principle of Occam's razor—we simply find the smallest program consistent with the data. Near-perfect vision recognition, language comprehension and

translation, and all other learning tasks become trivial. We will also have much better predictions of weather and earthquakes and other natural phenomenon.”

Today, you can use face-scanning to unlock your smartphone, talk to the device to ask it a question and often get a reasonable answer, or have your question translated into a different language. Your phone receives alerts about weather and other climatic events, with far better predictions than we would have thought possible just a dozen years ago. Meanwhile, cryptography has gone mostly unscathed beyond brute-force-like attacks on small key lengths. Now let’s look at how recent advances in computing, optimization, and learning are leading us to Optiland.

Solving Hard Problems

In 2016, Bill Cook (no relation to Steve) and his colleagues decided to tackle the following challenge:⁹ How do you visit every pub in the U.K. in the shortest distance possible? They made a list of 24,727 pubs and created the ultimate pub crawl, a walking trip that spanned 45,495,239 meters—approximately 28,269 miles—a bit longer than walking around the earth.

Cook had cheated a bit, eliminating some pubs to keep the size reasonable. After some press coverage in the U.K.,⁷ many complained about missing their favorite watering holes. Cook and company went back to work, building up the list to 49,687 pubs. The new tour length would be 63,739,687 meters, or about 39,606 miles (see Figure). One needs just a 40% longer walk to reach more than twice as many pubs. The pub crawl is just a traveling salesman problem, one of the most famous of the NP-complete problems. The number of possible tours through all the 49,687 pubs is roughly three followed by 211,761 zeros. Of course, Cook’s computers don’t search the whole set of tours but use a variety of optimization techniques. Even more impressive, the tour comes with a proof of optimality based on linear program duality.

Taking on a larger task, Cook and company aimed to find the shortest tour through more than two million stars where distances could be computed. Their tour of 28,884,456 parsecs is within a mere 683 parsecs of optimal.

Beyond Traveling Salesman, we have seen major advances in solving satisfiability and mixed-integer programming—a variation of linear programming where some, but not necessarily all, of the variables are required to be integers. Using highly refined heuristics, fast processors, specialized hardware, and distributed cloud computing, one can often solve problems that arise in practice with tens of thousands of variables and hundreds of thousands or even millions of constraints.

Faced with an NP problem to solve, one can often formulate the problem

as a satisfiability or mixed-integer programming question and throw it at one of the top solvers. These tools have been used successfully in verification and automated testing of circuits and code, computational biology, system security, product and packaging design, financial trading, and even to solve some difficult mathematical problems.

Data Science and Machine Learning

Any reader of *Communications* and most everyone else cannot dismiss the transformative effects of machine

Shortest route through 49,687 U.K. pubs. Used by permission. (<http://www.math.uwaterloo.ca/tsp/uk>).

learning, particularly learning by neural nets. The notion of modeling computation by artificial neurons—basically objects that compute weighted threshold functions—goes back to the work of Warren McCulloch and Walter Pitts in the 1940s.²⁸ In the 1990s, Yoshua Bengio, Geoffrey Hinton, and Yann LeCun²⁶ developed the basic algorithms that would power the learning of neural nets, a circuit of these neurons several layers deep. Faster and more distributed computing, specialized hardware, and enormous amounts of data helped propel machine learning to the point where it can accomplish many human-oriented tasks surprisingly well. ACM recognized the incredible impact the work of Bengio, Hinton, and LeCun has had in our society with the 2018 A.M. Turing Award.

How does machine learning mesh with P vs. NP? In this section, when we talk about $P = NP$, it will be in the very strong sense of all problems in NP having efficient algorithms in practice. Occam's razor states that "entities should not be multiplied without necessity" or, informally, that the simplest explanation is likely to be the right one. If $P = NP$, we can use this idea to create a strong learning algorithm: Find the smallest circuit consistent with the data. Even though we likely don't have $P = NP$, machine learning can approximate this approach, which led to its surprising power. Nevertheless, the neural net is unlikely to be the "smallest" possible circuit. A neural net trained by today's deep-learning techniques is typically fixed in structure with parameters that are only on the weights on the wires. To allow sufficient expressibility, there are often millions or more such weights. This limits the power of neural nets. They can do very well with face recognition, but they can't learn to multiply based on examples.

Universal distribution and GPT-3. Consider distributions on the infinite set of binary strings. You can't have a uniform distribution, but you could create distributions where every string of the same length has the same probability. However, some strings are simply more important than others. For example, the first million digits of π have more meaning than just a million digits generated at random. You

might want to put a higher probability on the more meaningful strings. There are many ways to do this, but in fact there is a universal distribution that gets close to any other computable distribution (see Kirchherr et al.²⁵) This distribution has great connections to learning—for example, any algorithm that learns with small error to this distribution will learn for all computable distributions. The catch is that this distribution is horribly non-computable even if $P = NP$. If $P = NP$, we still get something useful by creating an efficiently computable distribution universal to other efficiently computable distributions.

What do we get out of machine learning? Consider the Generative Pre-trained Transformer (GPT), particularly GPT-3 released in 2020.⁵ GPT-3 has 175 billion parameters trained on 410 billion tokens taken from as much of the written corpus as could be made available. It can answer questions, write essays given a prompt, and even do some coding. Though it has a long way to go, GPT-3 has drawn rave reviews for its ability to generate material that looks human-produced. One can view GPT-3 in some sense like a distribution, where we can look at the probability of outputs generated by the algorithm, a weak version of a universal distribution. If we restrict a universal distribution to have a given prefix, that provides a random sample prompted by that prefix. GPT-3 can also build on such prompts, handling a surprisingly wide range of domain knowledge without further training. As this line of research progresses, we will get closer to a universal metric from which one can perform built-in learning: Generate a random example from a given context.

Science and medicine. In science, we have made advances by doing large-scale simulations to understand, for example, exploring nuclear fusion reactions. Researchers can then apply a form of the scientific method: Create a hypothesis for a physical system; use that model to make a prediction; and then, instead of attempting to create an actual reaction, use an experimental simulation to test that prediction. If the answer is not as predicted, then change or throw away the model and start again.

After we have a strong model, we can then make that expensive test in a

physical reactor. If $P = NP$, we could, as mentioned above, use an Occam's Razor approach to create hypotheses—find the smallest circuits that are consistent with the data. Machine-learning techniques can work along these lines, automating the hypothesis creation. Given data—whether generated by simulations, experiments, or sensors—machine learning can create models that match the data. We can use these models to make predictions and then test those predictions as before.

While these techniques allow us to find hypotheses and models that might have been missed, they can also lead to false positives. We generally accept a hypothesis with a 95% confidence level, meaning that one out of 20 bad hypotheses might pass. Machine-learning and data science tools can allow us to generate hypotheses that will run the risk of publishing results not grounded in truth. Medical researchers, particularly those trying to tackle diseases such as cancer, often hit upon hard algorithmic barriers. Biological systems are incredibly complex structures. We know that our DNA forms a code that describes how our bodies are formed and the functions they perform, but we have only a very limited understanding on how these processes work.

On November 30, 2020, Google's DeepMind announced AlphaFold, a new algorithm that predicts the shape of a protein based on its amino acid sequence.²² AlphaFold's predictions nearly reach the accuracy of experimentally building the amino acid sequence and measuring the shape of the protein that forms. There is some controversy as to whether DeepMind has actually "solved" protein folding and it is far too early to gauge its impact, but in the long run this could give us a new digital tool to study proteins, understand how they interact, and learn how to design them to fight disease.

Beyond P vs. NP: chess and go. NP is like solving a puzzle. Sudoku, on an arbitrarily sized board, is NP-complete to solve from a given initial setting of numbers in some of the squares. But what about games with two players who take alternate turns, such as chess and go, when we ask about who wins from a given initial setting of the pieces? Even if we have $P = NP$, it wouldn't necessarily give us a perfect chess pro-

gram. You would have to ask if there is a move for white such that for every move of black, there is a move for white such that for every move of black ... white wins. You just can't do all those alternations of white and black on $P = NP$ alone. Games like these tend to be what is called *PSPACE-hard*, hard for computation that uses a reasonable amount of memory without any limit on time. Chess and go could even be harder depending on the precise formulation of the rules (see Demaine and Hearn.¹¹)

This doesn't mean you can't get a good chess program if $P = NP$. You could find an efficient computer program of one size that beats all efficient programs of slightly smaller sizes, if that's possible. Meanwhile, even without $P = NP$, computers have gotten very strong at chess and go. In 1997, IBM's Deep Blue defeated Gary Kasparov, chess world champion at the time, but go programs struggled against even strong amateurs. Machine learning has made dramatic improvements to computer game playing. While there is a lengthy history, let me jump to AlphaZero, developed in 2017 by Google's DeepMind.³⁵

AlphaZero uses a technique known as Monte Carlo tree search (MCTS) that randomly makes moves for both players to determine the best course of action. AlphaZero uses deep learning to predict the best distributions for the game positions to optimize the chances to win using MCTS. While AlphaZero is not the first program to use MCTS, it does not have any built-in strategy or access to a previous game database. AlphaZero assumes nothing more than the rules of the game. This allows AlphaZero to excel at both chess and go, two very different games that share little other than alternating moves and a fixed-size board. DeepMind recently went even further with MuZero,³³ which doesn't even get the full rules, just some representation of board position, a list of legal moves, and whether the position is a win, lose, or draw. Now we've come to the point that pure machine learning easily beats any human or other algorithm in chess or go. Human intervention only gets in the way. For games such as chess and go, machine learning can achieve success where $P = NP$ wouldn't be enough.

Machine learning may not do well when faced with tasks that are not from the distribution in which it was trained.

Explainable AI. Many machine-learning algorithms seem to work very well but we don't know why. If you look at a neural net trained for voice recognition, it's often very hard to understand why it makes the choices it makes. Why should we care? Here are a few of several reasons.

► **Trust:** How do we know that the neural net is acting correctly? Beyond checking input/output pairs we can't do any other analysis. Different applications have different levels of trust. It's okay if Netflix makes a bad movie recommendation, but less so if a self-driving car recommends a wrong turn.

► **Fairness:** Many examples abound in which algorithms trained on data learn the intended and unintended biases in that data (see O'Neil³⁰). If you don't understand the program, how do you figure out the biases?

► **Security:** If you use machine learning to monitor security systems, you won't know what exploits still exist, especially if your adversary is being adaptive. If you can understand the code, you could spot and fix security leaks. Of course, if adversaries have the code, they might find exploits.

► **Cause and effect:** Right now, you can, at best, check that a machine-learning algorithm only correlates with the kind of output you desire. Understanding the code might help us understand the causality in the data, leading to better science and medicine.

Would we get a better scenario if $P = NP$? If you had a quick algorithm for NP-complete problems, you could use it to find the smallest possible circuit for matching or Traveling Salesman, but you would not know why that circuit works. On the other hand, the reason you might want an explainable algorithm is so you can understand its properties, but we could use $P = NP$ to derive those properties directly. Whole conferences have cropped up studying explainable AI, such as the ACM Conference on Fairness, Accountability, and Trust.

Limits of machine learning. While machine learning has shown many surprising results in the last decade, these systems are far from perfect and, in most applications, can still be bested by humans. We will continue to improve machine-learning capability through new and optimized algorithms, data collection, and special-

ized hardware. Machine learning does seem to have its limits. As we've seen above, machine learning will give us a taste of $P = NP$, but it will never substitute for it. Machine learning makes little progress on breaking cryptography, which we will see later in the article.

Machine learning seems to fail learning simple arithmetic—for example, summing up a large collection of numbers or multiplying large numbers. One could imagine combining machine learning with symbolic mathematical tools. While we've seen some impressive advances in theorem provers,¹⁹ we sit a long way from my dream task of taking one of my research papers, with its informal proofs, and having an AI system fill in the details and verify the proof.

Again, $P = NP$ would make these tasks easy or at least tractable. Machine learning may not do well when faced with tasks that are not from the distribution in which it was trained. That could be low-probability edge cases, such as face recognition from a race not well represented in the training data, or even an adversarial attempt to force a different output by making a small change in the input—for example, changing a few pixels of a stop sign to force an algorithm to interpret it as a speed limit sign.¹² Deep neural-net algorithms can have millions of parameters, so they may not generalize well off distribution. If $P = NP$, one can produce minimum-sized models that would hopefully do a better job of generalizing, but without the experiment we can't perform, we will never know.

As impressive as machine learning is, we have not achieved anything close to artificial general intelligence, a term that refer to something like true comprehension of a topic or to an artificial system that achieves true consciousness or self-awareness. Defining these terms can be tricky, controversial, or even impossible. Personally, I've never seen a formal definition of consciousness that captures my intuitive notion of the concept. I suspect we will never achieve artificial general intelligence in the strong sense, even if $P = NP$.

Cryptography

While we have seen much progress in attacking NP problems, cryptography in its many forms, including one-way functions, secure hashes, and public-

I suspect we will never achieve artificial general intelligence in the strong sense, even if $P = NP$.

key cryptography, seems to have survived intact. An efficient algorithm for NP, were it to exist, would break all cryptosystems save those that are information-theoretically safe, such as one-time pads and some based on quantum physics. We have seen many successful cybersecurity attacks, but usually they stem from bad implementations, weak random number generators, or human error, but rarely if ever from breaking the cryptography.

Most CPU chips now have AES built in, so once we've used public-key cryptography to set up a private key, we can send encrypted data as easily as plain text. Encryption powers blockchain and cryptocurrencies, meaning people trust cryptography enough to exchange money for bits. Michael Kearns and Leslie Valiant²⁴ showed in 1994 that learning the smallest circuit, even learning the smallest bounded-layer neural net, could be used to factor numbers and break public-key cryptosystems. So far, machine-learning algorithms have not been successfully used to break cryptographic protocols nor are they ever expected to.

Why does encryption do so well when we've made progress on many other NP problems? In cryptography, we can choose the problem, specifically designed to be hard to compute and well tested by the community. Other NP problems generally come to us from applications or nature. They tend to not be the hardest cases and are more amenable to current technologies.

Quantum computing seems to threaten current public-key protocols that secure our Internet transactions. Shor's algorithm³⁴ can factor numbers and other related number-theory computations. This concern can be tempered in a few ways. Despite some impressive advances in quantum computing, we are still decades if not centuries away from developing quantum machines that can handle enough entangled bits to implement Shor's algorithm on a scale that can break today's codes. Also, researchers have made good progress toward developing public-key cryptosystems that appear resistant to quantum attacks.³¹ We will dwell more on quantum computing later in this article.

Factoring is not known to be NP-complete, and it is certainly possible that a

mathematical breakthrough could lead to efficient algorithms even if we don't have large-scale quantum computers. Having multiple approaches to public-key systems may come in handy no matter your view of quantum's future.

Complexity as Friction

What advantages can we get from computational hardness? Cryptography comes to mind. But perhaps the universe made computation difficult for a reason, not unlike friction. In the physical world, overcoming friction usually comes at the cost of energy, but we can't walk without it. In the computational world, complexity can often slow progress, but if it didn't exist, we could have many other problems. $P = NP$ would allow us to, in many cases, eliminate this friction.

Recent advances in computing show us that eliminating friction can sometimes have negative consequences. For instance, no one can read our minds, only see the actions that we take. Economists have a term, "preference revelation," which attempts to determine our desires based on our actions. For most of history, the lack of data and computing power made this at best a highly imprecise art.

Today, we've collected a considerable amount of information about people from their web searches, their photos and videos, the purchases they make, the places they visit (virtual and real), their social media activity, and much more. Moreover, machine learning can process this information and make eerily accurate predictions about people's behavior. Computers often know more about us than we know about ourselves.

We have the technological capability to wear glasses that would allow you to learn the name, interests and hobbies, and even the political persuasion of the person you are looking at. Complexity no longer affords us privacy. We need to preserve privacy with laws and corporate responsibility.

Computational friction can go beyond privacy. The U.S. government deregulated airline pricing in 1978 but finding the best price for a route required making phone calls to several airlines or working through a travel agent, who wasn't always incentivized to find the lowest price. Airlines worked

on reputation, some for great service and others for lower prices. Today, we can easily find the cheapest airline flights, so airlines have put considerable effort into competing on this single dimension of price and have used computation to optimize pricing and fill their planes, at the expense of the whole flying experience.

Friction helped clamp down on cheating by students. Calculus questions I had to answer as a college student in the 1980s can now be tackled easily by Mathematica. For my introductory theory courses, I have trouble creating homework and exam questions whose answers and solutions cannot be found online. With GPT-3 and its successors, even essay and coding questions can be automatically generated. How do we motivate students when GPT and the like can answer even their most complex questions?

Stock trading used to happen in big pits, where traders used hand signals to match prices. Now, trading algorithms automatically adjust to new pricing, occasionally leading to "flash crashes." Machine-learning techniques have led to decision-making systems or face recognition, matching social media content to users and judicial sentencing often at scale. These decision systems have done some good but have also led to significant challenges, such as amplifying biases and political polarization.³⁰ There are no easy answers here.

These are just a few of many such possible scenarios. Our goal, as computer scientists, is to make computation as efficient and simple as possible, but we must keep the costs of reducing friction on our minds.

The Power of Quantum Computers

As the limits of Moore's law have become more apparent, computer researchers have looked toward non-traditional computation models to make the next breakthroughs, leading to large growth in the research and application of quantum computing. Major tech companies, such as Google, Microsoft, and IBM—not to mention a raft of startups—have thrown considerable resources at developing quantum computers. The U.S. has launched a National Quantum Initiative and other countries, notably China, have followed suit.

In 2019, Google announced¹ it used a quantum computer with 53 qubits to achieve "quantum supremacy," solving a computational task that current traditional computation cannot. While some have questioned this claim, we certainly sit at the precipice of a new era in quantum computing. Nevertheless, we remain far away from having the tens of thousands of quantum bits required to run Peter Shor's algorithm³⁴ to find prime factors of numbers that today's machines cannot factor. Often, quantum computing gets described as the number of states represented by the bits—for example, the 2^{53} states of a 53-qubit machine. This might suggest that we could use quantum computing to solve NP-complete problems by creating enough states to, for instance, check all the potential cliques in a graph. Unfortunately, there are limits to how a quantum algorithm can manipulate these states, and all evidence suggests that quantum computers cannot solve NP-complete problems,³ beyond a quadratic improvement given by Grover's algorithm.¹⁸

Complexity Updates

Since the 2009 survey, we have seen several major advances in our understanding of the power of efficient computation. While these results do not make significant progress toward resolving P vs. NP , they still show how it continues to inspire great research.

Graph isomorphism. Some NP problems resist characterization as either in P (efficiently solvable) or NP-complete (as hard as the Clique problem). The most famous, integer factoring, which we discussed previously, still requires exponential time to solve. For another such problem, graph isomorphism, we have recently seen dramatic progress. Graph isomorphism asks whether two graphs are identical up to relabeling. Thinking in terms of Facebook, given two groups of 1,000 people, can we map names from one group onto the other in a way that preserves friendships?

Results related to interactive proofs in the 1980s offered strong evidence that graph isomorphism is not NP-complete,⁴ and even simple heuristics can generally solve such problems quickly in practice. Nevertheless, we still lack a polynomial-time algorithm

for graph isomorphism that works in all instances. László Babai achieved a breakthrough result in 2016, presenting a quasipolynomial-time algorithm for graph isomorphism.² The problems in P run in polynomial-time—that is, n^k for some constant k , where n is the size of the input, such as the number of people in each group. A quasipolynomial-time algorithm runs in time $n^{(\log n)^k}$, a bit worse than polynomial time but considerably better than the exponential time (2^n) that we expect NP-complete problems will need.

Babai's proof is a tour-de-force masterpiece combining combinatorics and group theory. Although getting the algorithm to run in polynomial-time would require several new breakthroughs, Babai provides a major theoretical result, making dramatic progress on one of the most important problems between P and NP-complete.

Circuits. If NP does not have small circuits over a complete basis (AND, OR, NOT) then $P \neq NP$. While there were significant circuit complexity results in the 1980s, none get close to showing $P \neq NP$. The 2009 survey remarked that there were no major results in circuit complexity in the 20 years prior. That lasted about one more year. In 1987, Razborov³² and Smolensky³⁶ showed the impossibility of computing the majority function with constant-depth circuits of AND, OR, NOT, and Mod_p gates for some fixed prime p . We could prove little, though, for circuits with Mod_6 gates. Even showing that NEXP, an exponential-time version of NP, could not be computed by small, constant-depth circuits of AND, OR, NOT, and Mod_6 gates remained open for decades. Constant-depth circuits are believed to be computationally weak. The lack of results reflects the paltry progress we have had in showing the limits of computation models.

In 2010, Ryan Williams showed³⁹ that NEXP indeed didn't have such small constant-depth circuits with Mod_6 or any other Mod gate. He had created a new technique, applying satisfiability algorithms that do just slightly better than trying all assignments and drawing in several complexity tools to achieve the lower bounds. Later, Williams and his student Cody Murray strengthened²⁹ the result to show that nondeterministic quasipoly-

All evidence suggests that quantum computers cannot solve NP-complete problems, beyond a quadratic improvement given by Grover's algorithm.

nomial-time doesn't have small constant-depth circuits with Mod_m gates for any fixed m . Nevertheless, showing that NP does not have small circuits of arbitrary depth—which is what you would need to show $P \neq NP$ —remains far out of reach.

Complexity strikes back? In a section of the 2009 survey titled, “A New Hope?”¹³ we discussed a new geometric-complexity-theory approach to attacking P vs. NP based on algebraic geometry and representation theory developed by Ketan Mulmuley and Milind Sohoni. In short, Mulmuley and Sohoni sought to create high-dimension polygons capturing the power of a problem in an algebraic version of NP and show that it had different properties than any such polygon corresponding to an algebraic property of P. One of their conjectures considered the property that the polygons contained a certain representation-theoretic object. In 2016, Peter Bürgisser, Christian Ikenmeyer, and Greta Panova⁶ showed that this approach cannot succeed.

While the Bürgisser-Ikenmeyer-Panova result deals a blow to the GCT approach to separating P vs. NP, it does not count it out. One could still potentially create polygons that differ based on the number of these representation-theoretic objects. Nevertheless, we shouldn't expect the GCT approach to settle the P vs. NP problem anytime in the near future.

The Possibility of the Impossible

As we reflect on P vs. NP, we see the question having many different meanings. There is P vs. NP the mathematical question—formally defined, stubbornly open, and still with a million-dollar bounty on its head. There were times when we could see a way forward toward settling P vs. NP through tools of computability theory, circuits, proofs, and algebraic geometry. At the moment, we don't have a strong way forward to solving the P vs. NP problem. In some sense, we are further from solving it than we ever were.

There are also the NP problems we just want or need to solve. In the classic 1976 text, *Computers and Intractability: A Guide to the Theory of NP-Completeness*,¹⁶ Garey and Johnson give an example of a hapless employee asked to solve an NP-complete optimization

problem. Ultimately, the employee goes to the boss and says, “I can’t find an efficient algorithm, but neither can all these famous people,” indicating that the boss shouldn’t fire the employee since no other hire could solve the problem.

In those early days of P vs. NP, we saw NP-completeness as a barrier—these were problems that we just couldn’t solve. As computers and algorithms evolved, we found we could make progress on many NP problems through a combination of heuristics, approximation, and brute-force computing. In the Garey and Johnson story, if I were the boss, I might not fire the employee but advise trying mixed-integer programming, machine learning, or a brute-force search. We are well past the time that NP-complete means impossible. It just means there is likely no algorithm that will always work and scale.

In my 2013 book on P vs. NP,¹⁴ I have a chapter titled, “A Beautiful World,” where I imagine an idealized world in which a Czech mathematician proves $P = NP$, leading to a very efficient algorithm for all NP problems. While we do not and likely will not ever live in this ideal world—with medical advances, virtual worlds indistinguishable from reality, and learning algorithms that generate new works of art—the wonderful (and not so wonderful) consequences of $P = NP$ no longer seem out of reach, but rather an eventual consequence of our further advances in computing.

We are truly on our way to nearly completely reversing the meaning of the P vs. NP problems. Instead of representing a barrier, think of P vs. NP opening doors, presenting us with new directions, and showing us the possibility of the impossible.

Acknowledgments

Thanks to Josh Grochow for helpful discussions on the GCT problem and to Bill Cook for allowing us to use the picture in Figure 1. I also thank Josh and the anonymous reviewers for helping to improve the exhibition. Some material in this article is adapted from the author’s blog.¹⁵

References

1. Arute, F., Arya, K., Babbus, R., Bacon, D., Bardin, J.C., Barends, R., Biswas, R., Boixo, S., Brandao, F.G.S.L., Buell, D.A., et al. Quantum supremacy using a

- programmable superconducting processor. *Nature* 574, 7779 (2019), 505–510. <https://doi.org/10.1038/s41586-019-1666-5>
2. Babai, L. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing* (2016), 684–697. <https://doi.org/10.1145/2897518.2897542>
3. Bennett, C., Bernstein, E., Brassard, G., and Vazirani, U. Strengths and weaknesses of quantum computing. *SIAM J. Comput.* 26, 5 (1997), 1510–1523.
4. Boppana, R., Håstad, J., and Zachos, S. Does co-NP have short interactive proofs? *Information Processing Letters* 25, 2 (1987), 127–132.
5. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners (2020). arXiv:2005.14165 [cs.CL]
6. Bürgisser, P., Ikenmeyer, C., and Panova, G. No occurrence obstructions in geometric complexity theory. *J. of the American Mathematical Society* 32, 1 (2019), 163–193. <https://doi.org/10.1090/jams/908>
7. Coldwell, W. World’s longest pub crawl: Maths team plots route between 25,000 UK boozers. *The Guardian* (Oct. 21, 2016). <https://www.theguardian.com/travel/2016/oct/21/worlds-longest-pub-crawl-maths-team-plots-route-between-every-pub-in-uk>
8. CRA Taulbee Survey. Computing Research Association (2020). <https://cra.org/resources/taulbee-survey>
9. Cook, B. Traveling salesman problem (2020). <http://www.math.uwaterloo.ca/~tsp/uk>
10. Cook, S. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on the Theory of Computing* (1971), 151–158.
11. Demaine, E.D. and Hearn, R.A. Playing games with algorithms: Algorithmic combinatorial game theory. *Games of No Chance 3: Mathematical Sciences Research Institute Publications, Vol. 56*. Michael H. Albert and Richard J. Nowakowski (Eds.), Cambridge University Press (2009), 3–56. http://erikdemaine.org/papers/AlgGameTheory_GONC3/
12. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition* (2018).
13. Fortnow, L. The status of the P versus NP problem. *Commun. ACM* 52, 9 (Sept. 2009), 78–86. <https://doi.org/10.1145/1562164.1562186>
14. Fortnow, L. *The Golden Ticket: P, NP and the Search for the Impossible*. Princeton University Press, Princeton, (2013). <https://goldenticket.fortnow.com>
15. Fortnow, L. and Gasarch, W. Computational Complexity. <https://blog.computationalcomplexity.org>
16. Garey, M. and Johnson, D. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, (1979).
17. Gödel, K. Letter to John von Neumann. (1956). <https://www2.karlin.mff.cuni.cz/~krajicek/goedel-letter.pdf>
18. Grover, L. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th ACM Symposium on the Theory of Computing* (1996), 212–219.
19. Hartnett, K. Building the mathematical library of the future. *Quanta Magazine* (Oct. 1, 2020). <https://www.quantamagazine.org/building-the-mathematical-library-of-the-future-20201001/>
20. Impagliazzo, R. A personal view of average-case complexity theory. In *Proceedings of the 10th Annual Conference on Structure in Complexity Theory*. IEEE Computer Society Press (1995), 134–147. <https://doi.org/10.1109/SCT.1995.514853>
21. Jaffe, A. The Millennium Grand Challenge in Mathematics. *Notices of the AMS* 53, 6 (June/July 2006), 652–660. <http://www.ams.org/notices/200606/fejaffe.pdf>
22. Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Tunyasuvunakool, K., Ronneberger, O., Bates, R., Židek, A., Bridgland, A., Meyer, C., Kohl, S.A.A., Potapenko, A., Ballard, A.J., Cowie, A., Romero-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Steinegger, M., Pacholska, M., Silver, D., Vinyals, O., Senior, A.W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. High accuracy protein structure prediction using deep learning. In *14th Critical Assessment of Techniques for Protein Structure Prediction* (Abstract Book) (2020), 22–24. https://predictioncenter.org/casp14/doc/CASP14_Abstracts.pdf
23. Karp, R. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. Miller and J. Thatcher (Eds.). Plenum Press, New York, (1972), 85–103.
24. Kearns, M. and Valiant, L. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM* 41, 1 (Jan. 1994), 67–95. <https://doi.org/10.1145/174644.174647>
25. Kirchherr, W., Li, M., and Vitányi, P. The miraculous universal distribution. *The Mathematical Intelligencer* 19, 4 (Sep. 1, 1997), 7–15. <https://doi.org/10.1007/BF03024407>
26. LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature* 521, 7553 (May 1, 2015), 436–444. <https://doi.org/10.1038/nature14539>
27. Levin, L. Universal’ny’ie pereborny’ie zadachi (Universal search problems: in Russian). *Problemy Peredachi Informatsii* 9, 3 (1973), 265–266. Corrected English translation in.³⁷
28. McCulloch, W.S. and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics* 5, 4 (Dec. 1, 1943), 115–133. <https://doi.org/10.1007/BF02478259>
29. Murray, C. and Williams, R. Circuit lower bounds for nondeterministic quasi polynomial: An easy witness Lemma for NP and NQP. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing* (2018), 890–901. <https://doi.org/10.1145/3188745.3188910>
30. O’Neil, C. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Crown (2016), New York.
31. Peikert, C. Lattice cryptography for the Internet. In *Post-Quantum Cryptography*, Michele Mosca (Ed.). Springer International Publishing, Cham (2014), 197–219.
32. Razborov, A. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR* 41, 4 (1987), 333–338.
33. Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Laurent Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., and Silver, D. Mastering Atari, go, chess and shogi by planning with a learned model. *Nature* 588, 7839 (Dec. 1, 2020), 604–609. <https://doi.org/10.1038/s41586-020-03051-4>
34. Shor, P. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* 26, 5 (1997), 1484–1509.
35. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* 362, 6419 (2018), 1140–1144. <https://doi.org/10.1126/science.aar6404>
36. Smolensky, R. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the 19th ACM Symposium on the Theory of Computing* (1987), 77–82.
37. Trakhtenbrot, R. A survey of Russian approaches to Perebor (brute-force search) algorithms. *Annals of the History of Computing* 6, 4 (1984), 384–400.
38. Wikipedia contributors. List of public corporations by market capitalization. *Wikipedia, the Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=List_of_public_corporations_by_market_capitalization&oldid=1045945999
39. Williams, R. Nonuniform ACC circuit lower bounds. *Journal of the ACM* 61, 1, Article 2 (Jan. 2014). <https://doi.org/10.1145/2559903>

Lance Fortnow (lfortnow@iit.edu) is a professor and dean of the College of Computing at Illinois Institute of Technology, Chicago, IL, USA.

Copyright held by author(s)/owner(s).
Publication rights licensed to ACM.

Watch the author discuss this work in the exclusive *Communications* video. <https://cacm.acm.org/videos/fifty-years-of-p-vs-np>