

Fifty Years of P Versus NP and the Possibility of the Impossible

Lance Fortnow

lfortnow@iit.edu

Illinois Institute of Technology

Chicago, Illinois, USA

ABSTRACT

On May 4, 1971, Steve Cook presented the paper that announced the P versus NP problem to the world. The P versus NP problem has withstood the test of time but how we compute has greatly evolved. Through advances in algorithms, learning and hardware we can tackle many NP-hard problems thought impossible many years ago. We explore how thinking about P v NP now often leads to possibilities instead of barriers.

CCS CONCEPTS

• **Theory of computation** → **Complexity classes**; *Problems, reductions and completeness*; • **General and reference** → *Surveys and overviews*; • **Social and professional topics** → *History of computing theory*.

KEYWORDS

P versus NP, Computational Complexity, History of Computing

ACM Reference Format:

Lance Fortnow. 2021. Fifty Years of P Versus NP and the Possibility of the Impossible. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 8 pages.

1 INTRODUCTION

As we prepared to celebrate this fiftieth anniversary of the P versus NP question, Moshe Vardi asked me if I would like to update my 2009 Communications of the ACM article “The Status of the P versus NP Problem” [13]. Although it was a dozen years ago, the status remains “still open”.

The P vs NP problem and the theory behind it has not changed dramatically in the last 12 years but the world of computing most certainly has. We have seen the growth of cloud computing that has helped empower social networks, smart phones, the gig economy, fintech, spatial computing, online education and perhaps most importantly the rise of data science and machine learning. In 2009 the top ten companies by market cap included only one big tech company, Microsoft. As the end of 2020 the first seven are Microsoft, Apple, Amazon, Alphabet (Google), Alibaba, Facebook and Tencent [38]. The number of CS graduates in the United States more than tripled [8] and does not come close to meeting demand.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, July 2017, Washington, DC, USA

© 2021 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00

Instead of revising or updating the 2009 survey, we will view advances in computing, optimization and machine learning through a P v NP lens. In particular we see how these advances give us a taste of the world where $P = NP$, and while P v NP still gives us limits on what we can do the question also paints some new opportunities of study. In particular we will see how we are heading towards a world I call Optiland, where almost miraculously we can gain many of the advantages of $P = NP$ without some of the disadvantages such as breaking cryptography.

As a mathematical problem P v NP still remains one of the most important open problems, one of the Clay Mathematical Institute millennium problems [21] that offers a million dollar bounty for the solution. We end describing some new theoretical computer science results that, while not getting us closer to solving the P v NP question, show us that thinking about P v NP still drives much of the important research in the area.

2 THE P VERSUS NP PROBLEM

Are there 300 Facebook users who are all friends with each other? How would you go about finding the answer to that question?

Let's assume you work at Facebook and have access to the entire Facebook graph, who is friends with whom. You now need to write an algorithm to find that large clique of friends. You could try all groups of size 300, but there are far too many such groups 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 well. In fact nobody knows a significantly faster algorithm than trying all the groups but neither do we know that no such algorithm exists.

This is basically the P versus NP question. NP are problems that have solutions you can check efficiently. If I tell you which 300 people might form a clique, you can check that the 44,850 pairs of users are all friends relatively quickly. Clique is an NP problem. P are 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, i.e., we can efficiently solve the clique problem quickly if and only if $P = NP$. Many other problems have this property including, 3-coloring (can we color a map using only three colors so no two neighboring countries have the same color), traveling salesman (can we find a short route through a list of cities if we can visit in any order) 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, 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 versus NP problem see the 2009 survey [13] or the popular science book based on that survey [14]. For a more technical introduction, the 1979 book of Michael Garey and David Johnson [16] has held up surprisingly well and is still an invaluable reference for those who need to understand which problems are NP-complete.

3 WHY CELEBRATE NOW?

On the afternoon of Tuesday, May 4, 1971, in the Stouffer's Somerset Inn in Shaker Heights, Ohio, Steve Cook presented his ACM Symposium on the Theory of Computing paper proving 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 be a major breakthrough in complexity theory.

Steve Cook's paper thus introduced the P versus NP problem to the world.

Dating a mathematical concept is almost always fraught with challenges and there are many other possibilities of where to start the clock on P versus NP.

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 like P versus NP. The basics of efficient computation and nondeterminism were developed in the 1960s. The P versus NP question was formulated earlier than that, we just didn't know it.

Kurt Gödel wrote a letter [17] in 1956 to John von Neumann describing essentially the P versus NP problem. It is not clear that John von Neumann, then suffering from cancer, ever read the letter and the letter was not discovered and widely distributed until 1988.

The P v NP question didn't really become a phenomenon until Richard Karp published his 1972 paper [23] showing 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 paper based on his independent 1971 research that defined the P v NP problem [27]. By the time Levin's paper reached the west, the P versus NP problem had already established itself as the most important question in all of computing.

4 OPTILAND

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

- **Algorithmica** $P = NP$ or something "morally equivalent" like 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 not only can we not solve hard problems on average but we apparently do not get

any cryptographic advantage from the hardness of these problems.

- **Minicrypt** Cryptographic one-way functions exist but we do not have public-key cryptography.
- **Cryptomania** Public-key cryptography is possible, i.e. 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 v 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 the theory of P versus NP. 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 both in 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 $P=NP$?" of 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 take your smartphone, unlock it by having the phone scan your face, and ask it a question by talking and often get a reasonable answer, or have your question translated into a different language. You get alerts on your phone for weather and earthquakes, with far better predictions than we would have thought possible a dozen years ago.

Meanwhile cryptography has gone mostly unscathed beyond brute-force like attacks on small key lengths.

In the next couple of sections we show how recent advances in computing, optimization and learning are leading us to Optiland.

5 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 United Kingdom in the shortest distance possible? They took a list of 24,727 pubs and created the ultimate pub crawl, a walking trip that took 45,495,239 meters, approximately 28,269 miles, a bit longer than walking around the earth.

Cook had cheated a bit, eliminating a number of pubs to keep the size reasonable. After some press coverage in the UK [7] many complained about missing their favorite watering holes. So Cook and company went back to work now using a list of 49,687 pubs. After a year and a half of more effort they came up with a tour length of 63,739,687 meters, or about 39,606 miles (see Figure 1). One needs just a 40% longer walk to reach over 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 3 followed by 211,761 zeros. Of course Cook's computers don't search the whole set of

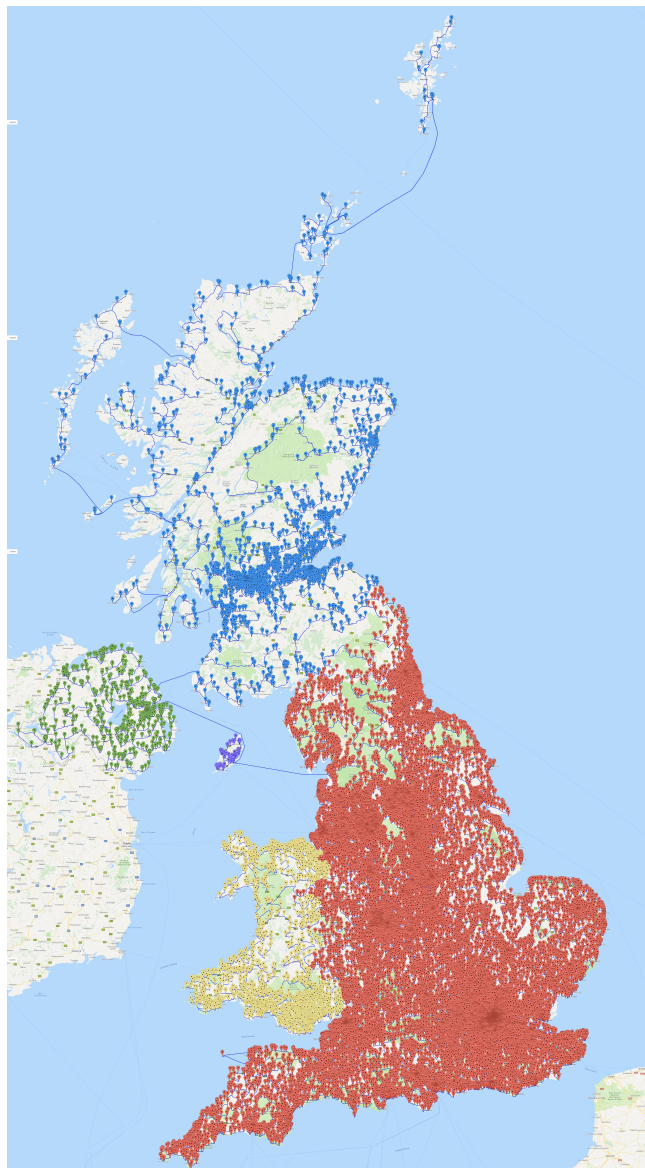


Figure 1: Shortest route through 49,687 UK pubs. Used by permission (<http://www.math.uwaterloo.ca/tsp/uk>)

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 aim to find the shortest tour through over 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 solving some hard mathematical problems.

6 DATA SCIENCE AND MACHINE LEARNING

Any reader of CACM and most everyone else cannot dismiss the transformative effects of machine learning, particularly learning by neural nets. The notion of modeling computation by artificial neurons, basically objects that computes weighted thresholds functions, goes back to the work of Warren McCulloch and Walter Pitts in the 40's [28]. In the 90's, Yoshua Bengio, Geoffrey Hinton and Yan LeCun (see [26]) would develop 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 where it can accomplish many human-oriented tasks surprisingly well. ACM honored Bengio, Hinton and LeCun with the 2018 A. M. Turing Award for their work recognizing the incredible impact it has had in our society.

How does machine learning mesh with P v 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.

6.1 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 has 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 [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, 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, even do some coding. Though it has a long way to go, GPT-3 has drawn rave reviews as generating 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 do built-in learning: Generate a random example from a given context.

6.2 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, change or throw away the model and start again. After we have a strong model we can then make that expensive test in an physical reactor.

If $P = NP$, as we've mentioned above we could use a 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 hypothesis with a 95% confidence level, meaning that one out of twenty bad hypotheses might pass. Machine learning and data science tools can allow us to generate hypotheses at will run the risk of publishing results not grounded in truth.

Medical researchers, particularly those trying to tackle diseases like 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.

As I wrote this, on November 30, 2020, Google's DeepMind announced a new algorithm, AlphaFold, 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 or not DeepMind has actually "solved" protein folding and it is far too early to gauge its impact but it in the long run this could give us a new digital tool to study proteins, learn how they interact and how to design them to fight disease.

6.3 Beyond $P \neq NP$: 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, like 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 program. 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's 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 [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, the then world's champion but Go programs struggled against even strong amateurs.

Machine learning made dramatic improvements to computer game playing. While there is a lengthy history, let me jump to AlphaZero developed by Google's DeepMind [35] in 2017. 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 on both chess and go, two very different games that share little other than alternating moves and a fixed-size board. Recently DeepMind went even further with MuZero [33] that doesn't even get the full rules, just some representation of the board position, a list of legal moves and whether the position is a win, lose or draw.

Now we've gotten 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 like Chess and Go, machine learning can achieve success where $P \neq NP$ wouldn't be enough.

6.4 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 say 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 a different level of trust. It's okay if Netflix makes a bad movie recommendation, but less so if a self-driving car makes a mistake.
- **Fairness** Many examples abound of algorithms trained on data will learn intended or unintended biases in that data (see [30]). If you don't understand the program how do figure out the biases?
- **Security** If you use machine learning to monitor systems for security, you won't know what exploits still might exist, especially if your adversary is being adaptive. If you can understand the code you could spot and fix security leaks. Of course if the adversary had the code, they might find exploits.
- **Cause and Effect** Right now at best you can 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 say matching or traveling salesman but you would have no clue why that circuit works. On the other hand, the reasons 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.

6.5 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 specialized hardware. Machine learning does seem to have its limits. As we've seen above, machine learning will give us a taste of $P = NP$, it will never substitute for it.

Machine learning makes little progress on breaking cryptography which we will discuss more in Section 7.

Machine learning seems to fail learning simple arithmetic, summing up a large collection of numbers or multiplying large numbers for example. 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 not from the distribution in which it was trained. Anything from low-probability edge cases, face recognition from a race not well represented in the training data, or even adversarial attempt to force a different output by making a small change in the input, like changing a few pixels of a stop sign to make an algorithm to decide its 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 generalizing but without the experiment we can't perform we never will know.

As impressive as we've seen machine learning, we have not achieved anything close to Artificial General Intelligence, a term that can mean something like true comprehension of a topic or an artificial system that achieves true consciousness or self-awareness. Even defining these terms can be tricky, controversial, perhaps 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 $P = NP$.

7 CRYPTOGRAPHY

While we have seen much progress in attacking NP problems, cryptography in its many forms, including one-way functions, secure hashes, and public-key cryptography seemed 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 follow 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 plaintext. Encryption powers blockchain and cryptocurrencies, meaning people trust cryptography enough to exchange money for bits.

Michael Kearns and Leslie Valiant [24] in 1994 showed that learning the smallest circuit, even learning the smallest bounded-layer neural net would could be used to factor number and break public-key cryptosystems. Machine learning algorithms have so far 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 and tend not to be the hardest cases, and more amenable to current technologies.

Quantum computing seems to threaten current public-key protocols that secure our Internet transactions. Shor's algorithm [34] 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 towards developing public-key cryptosystems that appear resistant to quantum attacks [31]. We will dwell more on quantum computing in Section 9.

Factoring is not known to be NP-complete and it is certainly possible 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 the future of quantum.

8 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 friction usually costs us energy to overcome but on the other hand 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 in many cases to eliminate this friction. Recent advances in computing show us that sometimes eliminating friction doesn't always have the best consequences.

Consider even our private selves. No one can read our minds, only see the actions that we take. Economists have a term "preference revelation" that tries 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 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 so much more. Moreover machine learning can process this information and make eerily accurate predictions of 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 US government deregulated airline pricing in 1978 but finding the best price for a route required making phone calls to several airline or working through a travel agent who didn't always have the incentive 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 and so airlines have put considerable effort into competing on this single dimension of price and have used computation to optimize pricing to 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 80's can now be tackled easily by Mathematica. I now have trouble creating homework and exam questions in introductory theory courses whose solutions cannot be found online. With GPT-3 and its successors, even essay and coding questions can be automatically generated. How do we even motivate students when they will have even complex questions answered just by asking them?

Stock trading used to happen in big pits with traders using hand signals to match prices. Now algorithmic trading algorithms adjust to new pricing automatically occasionally leading to "flash crashes".

Machine learning techniques have led to decision making systems for face recognition, matching social media content to users and judicial sentencing often at scale. These decision systems have

done some good but also have led to significant challenges leading to amplifying biases and political polarization [30]. No easy answers here.

This is just a few of many such stories. As computer scientists our goal is to make computation as efficient and simple as possible but we must keep in our minds the costs of reducing friction.

9 THE POWER OF QUANTUM COMPUTERS

As the limits of Moore's laws have become far more apparent, computer researchers have looked towards non-traditional computation models to make the next computational breakthroughs leading to a large growth in research and applications of quantum computing.

Major tech companies including Google, Microsoft and IBM have put considerable resources toward developing quantum computers, not to mention a raft of startups. The United States has launched a National Quantum Initiative and other countries, notably China, have followed suit.

In 2019, Google announced [1] they have used a quantum computer with 53 qubits to achieve "quantum supremacy", solving a computational task that current traditional computation cannot. While some have questioned that 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 algorithm [34] to find prime factors of numbers that we cannot factor by today's machines.

Often quantum computing gets described by 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, say, 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].

10 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 towards resolving the P versus NP problem, they still show how $P \neq NP$ continues to inspire great research.

10.1 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 discuss in Section 7, still requires exponential time to solve.

For another such problem, graph isomorphism, we have seen dramatic recent progress. The graph isomorphism problem asks whether two graphs are identical up to relabeling. Thinking in terms of Facebook, given two groups of 1000 people, can we map names from one group onto the other that preserve friendships?

Results related to interactive proofs in the 80's gave strong evidence that graph-isomorphism is not NP-complete [4] and even simple heuristics can generally solve graph isomorphism problems

quickly in practice. Nevertheless we still lack a polynomial-time algorithm for graph isomorphism that works for all instances.

László Babai had a breakthrough result in 2016 giving 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, for example 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^ϵ}) that we expect NP-complete problems will require.

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 gives a major theoretical result making dramatic progress on one of the most important problems between P and NP-complete.

10.2 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 twenty years prior. That lasted about one more year.

In 1987, Razborov [32] and Smolensky [36] 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 showed [39] 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 of applying satisfiability algorithms that do just slightly better than trying all assignments and by drawing in several complexity tools can achieve the lower bounds. Later Williams with his student Cody Murray strengthened [29] the result to show that nondeterministic quasipolynomial-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 $P \neq NP$, still remains far out of reach.

10.3 Complexity Strikes Back?

In a section entitled "A New Hope?" in the 2009 survey [13] we discuss a new geometric complexity theory approach to attacking the P versus NP problem 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 Panova [6] showed that this approach cannot succeed.

While the Bürgisser-Ikenmeyer-Panova result deals a blow to the GCT approach to separating P versus 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 versus NP problem anytime in the near future.

11 THE POSSIBILITY OF THE IMPOSSIBLE

As we reflect back through fifty years of P versus NP, we see the question having many different meanings. There is P versus NP the mathematical question, formally defined, stubbornly open and still with a million dollar bounty on its head. We've had times when we could see a way forward towards settling the P versus 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 versus NP problem. In some sense we are further away from solving the P versus NP problem than we ever were.

There's 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 a NP-complete optimization problem. Ultimately they go 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 would solve the problem either.

In those early days of P v NP, we saw NP-completeness as a barrier—these were problems that we just couldn't solve. As computers and algorithms evolved, we found would could make progress on many NP problems through a combination of heuristics, approximation and just brute-force computing. In the Garey and Johnson story, if I were the boss, I might not fire the employee but tell them to at least try mixed-integer programming, machine learning or just 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 v NP [14], I have a chapter entitled "A Beautiful World" imagining a world where a Czech mathematician proved $P = NP$ leading to a very efficient algorithm for all NP problems. A world with medical advances, virtual worlds indistinguishable from reality, learning algorithms that generate new works of art. While we do not and likely will not ever live in that ideal world, 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 truly on are way to nearly completely reversing the meaning of the P v NP problems. Instead of representing a barrier, thinking of P v NP opens door and shows us new directions, showing us the possibilities of the impossible.

ACKNOWLEDGMENTS

Thanks to Josh Grochow for helpful discussions on the GCT problem and for 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 survey is adapted from the author's blog [15].

REFERENCES

- [1] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (2019), 505–510. <https://doi.org/10.1038/s41586-019-1666-5>
- [2] László Babai. 2016. *Graph Isomorphism in Quasipolynomial Time [Extended Abstract]*. Association for Computing Machinery, New York, NY, USA, 684–697. <https://doi.org/10.1145/2897518.2897542>
- [3] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. 1997. Strengths and Weaknesses of Quantum Computing. *SIAM J. Comput.* 26, 5 (1997), 1510–1523.
- [4] R. Boppana, J. Håstad, and S. Zachos. 1987. Does co-NP have short interactive proofs? *Inform. Process. Lett.* 25, 2 (1987), 127–132.
- [5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. [arXiv:2005.14165 \[cs.CL\]](https://arxiv.org/abs/2005.14165)
- [6] Peter Bürgisser, Christian Ikenmeyer, and Greta Panova. 2019. No occurrence obstructions in geometric complexity theory. *Journal of the American Mathematical Society* 32, 1 (2019), 163–193. <https://doi.org/10.1090/jams/908>
- [7] Will Coldwell. 2016. World's longest pub crawl: Maths team plots route between 25,000 UK boozers. *The Guardian* (21 Oct. 2016). <https://www.theguardian.com/travel/2016/oct/21/worlds-longest-pub-crawl-maths-team-plots-route-between-every-pub-in-uk>
- [8] Computing Research Association. 2020. CRA Taulbee Survey. <https://cra.org/resources/taulbee-survey/> Online.
- [9] Bill Cook. 2020. Traveling Salesman Problem. <http://www.math.uwaterloo.ca/tsj/uk>
- [10] S. Cook. 1971. The Complexity of Theorem-Proving Procedures. In *Proceedings of the 3rd ACM Symposium on the Theory of Computing*. ACM, New York, 151–158.
- [11] Erik D. Demaine and Robert A. Hearn. 2009. Playing Games with Algorithms: Algorithmic Combinatorial Game Theory. In *Games of No Chance 3*, Michael H. Albert and Richard J. Nowakowski (Eds.). Mathematical Sciences Research Institute Publications, Vol. 56. Cambridge University Press, 3–56. http://erikdemaine.org/papers/AlgGameTheory_GONC3/
- [12] Kevin Eykholt, Ivan Evtimov, Earlene Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust Physical-World Attacks on Deep Learning Visual Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [13] L. Fortnow. 2009. The Status of the P versus NP problem. *Commun. ACM* 52, 9 (Sept. 2009), 78–86. <https://doi.org/10.1145/1562164.1562186> Review Article.
- [14] L. Fortnow. 2013. *The Golden Ticket: P, NP and the search for the impossible*. Princeton University Press, Princeton. <https://goldenticket.fortnow.com>
- [15] L. Fortnow and W. Gasarch. 2002-. Computational Complexity. <https://blog.computationalcomplexity.org> Weblog.
- [16] M. Garey and D. Johnson. 1979. *Computers and Intractability. A Guide to the theory of NP-completeness*. W. H. Freeman and Company, New York.
- [17] K. Gödel. 1956. Letter to John von Neumann. <https://www2.karlin.mff.cuni.cz/~krajicek/goedel-letter.pdf>
- [18] L. Grover. 1996. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the 28th ACM Symposium on the Theory of Computing*. ACM, New York, 212–219.
- [19] Kevin Hartnett. 2020. Building the Mathematical Library of the Future. *Quanta Magazine* (1 Oct. 2020). <https://www.quantamagazine.org/building-the-mathematical-library-of-the-future-20201001>
- [20] R. Impagliazzo. 1995. A personal view of average-case complexity theory. In *Proceedings of the 10th Annual Conference on Structure in Complexity Theory*. IEEE Computer Society Press, 134–147. <https://doi.org/10.1109/SCT.1995.514853>
- [21] Arthur Jaffe. 2006. The Millennium Grand Challenge in Mathematics. *Notices of the AMS* 53, 6 (June/July 2006), 652–660. <http://www.ams.org/notices/200606/fea-jaffe.pdf>
- [22] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Kathryn Tunyasuvunakool, Olaf Ronneberger, Russ Bates, Augustin Židek, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Anna Potapenko, Andrew J Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Martin Steinegger, Michalina Pacholska, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. 2020. High Accuracy Protein Structure Prediction Using Deep Learning. In *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book) (CASP)*. 22–24. https://predictioncenter.org/casp14/doc/CASP14_Abstracts.pdf
- [23] R. Karp. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. Miller and J. Thatcher (Eds.). Plenum Press, New York, 85–103.
- [24] Michael Kearns and Leslie Valiant. 1994. Cryptographic Limitations on Learning Boolean Formulae and Finite Automata. *J. ACM* 41, 1 (Jan. 1994), 67–95. <https://doi.org/10.1145/174644.174647>
- [25] Walter Kirchherr, Ming Li, and Paul Vitányi. 1997. The miraculous universal distribution. *The Mathematical Intelligencer* 19, 4 (01 Sep 1997), 7–15. <https://doi.org/10.1007/BF03024407>
- [26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (01 May 2015), 436–444. <https://doi.org/10.1038/nature14539>
- [27] L. Levin. 1973. Universal'nyye perebornye zadachi (Universal search problems: in Russian). *Problemy Peredachi Informatsii* 9, 3 (1973), 265–266. Corrected English translation in [37].
- [28] Warren S. McCulloch and Walter Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* 5, 4 (01 Dec 1943), 115–133. <https://doi.org/10.1007/BF02478259>
- [29] Cody Murray and Ryan Williams. 2018. Circuit Lower Bounds for Nondeterministic Quasi-Polytime: An Easy Witness Lemma for NP and NQP. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing* (Los Angeles, CA, USA) (*STOC 2018*). Association for Computing Machinery, New York, NY, USA, 890–901. <https://doi.org/10.1145/3188745.3188910>
- [30] Cathy O'Neil. 2016. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Crown, New York.
- [31] Chris Peikert. 2014. Lattice Cryptography for the Internet. In *Post-Quantum Cryptography*, Michele Mosca (Ed.). Springer International Publishing, Cham, 197–219.
- [32] A. Razborov. 1987. 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] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. 2020. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature* 588, 7839 (01 Dec 2020), 604–609. <https://doi.org/10.1038/s41586-020-03051-4>
- [34] P. Shor. 1997. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* 26, 5 (1997), 1484–1509.
- [35] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhruv Kumar, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2018. 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] R. Smolensky. 1987. Algebraic Methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the 19th ACM Symposium on the Theory of Computing*. ACM, New York, 77–82.
- [37] R. Trakhtenbrot. 1984. 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. 2021. 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 [Online; accessed 24-September-2021].
- [39] Ryan Williams. 2014. Nonuniform ACC Circuit Lower Bounds. *J. ACM* 61, 1, Article 2 (Jan. 2014), 32 pages. <https://doi.org/10.1145/2559903>