A New Map Traces the Limits of Computation

A major advance reveals deep connections between the classes of problems that computers can — and can’t — possibly do.

By John Pavlus

At first glance, the big news coming out of this summer’s conference on the theory of computing appeared to be something of a letdown. For more than 40 years, researchers had been trying to find a better way to compare two arbitrary strings of characters, such as the long strings of chemical letters within DNA molecules. The most widely used algorithm is slow and not all that clever: It proceeds step-by-step down the two lists, comparing values at each step. If a better method to calculate this “edit distance” could be found, researchers would be able to quickly compare full genomes or large data sets, and computer scientists would have a powerful new tool with which they could attempt to solve additional problems in the field.

Yet in a paper presented at the ACM Symposium on Theory of Computing, two researchers from the Massachusetts Institute of Technology put forth a mathematical proof that the current best algorithm was “optimal” — in other words, that finding a more efficient way to compute edit distance was mathematically impossible. The Boston Globe celebrated the hometown researchers’ achievement with a headline that read “For 40 Years, Computer Scientists Looked for a Solution That Doesn’t Exist.”
But researchers aren’t quite ready to record the time of death. One significant loophole remains. The impossibility result is only true if another, famously unproven statement called the strong exponential time hypothesis (SETH) is also true. Most computational complexity researchers assume that this is the case — including Piotr Indyk and Artūrs Bačkurs of MIT, who published the edit-distance finding — but SETH’s validity is still an open question. This makes the article about the edit-distance problem seem like a mathematical version of the legendary report of Mark Twain’s death: greatly exaggerated.

The media’s confusion about edit distance reflects a murkiness in the depths of complexity theory itself, where mathematicians and computer scientists attempt to map out what is and is not feasible to compute as though they were deep-sea explorers charting the bottom of an ocean trench. This algorithmic terrain is just as vast — and poorly understood — as the real seafloor, said Russell Impagliazzo, a complexity theorist who first formulated the exponential-time hypothesis with Ramamohan Paturi in 1999. “The analogy is a good one,” he said. “The oceans are where computational hardness is. What we’re attempting to do is use finer tools to measure the depth of the ocean in different places.”

According to Ryan Williams, a computational complexity theorist at Stanford University, an imprecise understanding of theoretical concepts like SETH may have real-world consequences. “If a funding agency read that [Boston Globe headline] and took it to heart, then I don’t see any reason why they’d ever fund work on edit distance again,” he said. “To me, that’s a little dangerous.” Williams rejects the conclusion that a better edit-distance algorithm is impossible, since he happens to believe that SETH is false. “My stance [on SETH] is a little controversial,” he admits, “but there isn’t a consensus. It is a hypothesis, and I don’t believe it’s true.”

SETH is more than a mere loophole in the edit-distance problem. It embodies a number of deep connections that tie together the hardest problems in computation. The ambiguity over its truth or falsity also reveals the basic practices of theoretical computer science, in which math and logic often marshal “strong evidence,” rather than proof, of how algorithms behave at a fundamental level.
Whether by assuming SETH’s validity or, in Williams’ case, trying to refute it, complexity theorists are using this arcane hypothesis to explore two different versions of our universe: one in which precise answers to tough problems stay forever buried like needles within a vast computational haystack, and one in which it may be possible to speed up the search for knowledge ever so slightly.

**How Hard Can It Be?**

Computational complexity theory is the study of problems. Specifically, it attempts to classify how “hard” they are — that is, how efficiently a solution can be computed under realistic conditions.

SETH is a hardness assumption about one of the central problems in theoretical computer science: Boolean satisfiability, which is abbreviated as SAT. On its face, SAT seems simple. If you have a formula containing variables that can be set as true or false rather than as number values, is it possible to set those variables in such a way that the formula outputs “true”? Translating SAT into plain language, though, reveals its metamathematical thorniness: Essentially, it asks if a generic problem (as modeled by a logical formula) is solvable at all.

As far as computer scientists know, the only general-purpose method to find the correct answer to a SAT problem is to try all possible settings of the variables one by one. The amount of time that this exhaustive or “brute-force” approach takes depends on how many variables there are in the formula. As the number of variables increases, the time it takes to search through all the possibilities increases exponentially. To complexity theorists and algorithm designers, this is bad. (Or, technically speaking, hard.)

SETH takes this situation from bad to worse. It implies that finding a better general-purpose algorithm for SAT — even one that only improves on brute-force searching by a small amount — is impossible.

The computational boundaries of SAT are important because SAT is mathematically equivalent to thousands of other problems related to search and optimization. If it were possible to find one efficient, general-purpose algorithm for any of these so-called “NP-complete” problems, all the rest of them would be instantly unlocked too.

This relationship between NP-complete problems is central to the “P versus NP” conjecture, the most famous unsolved problem in computer science, which seeks to define the limits of computation in mathematical terms. (The informal version: if P equals NP, we could quickly compute the true answer to almost any question we wanted, as long as we knew how to describe what we wanted to find and could easily recognize it once we saw it, much like a finished jigsaw puzzle. The vast majority of computer scientists believe that P does not equal NP.) The P versus NP problem also helps draw an informal line between tractable (“easy”) and intractable (“hard”) computational procedures.

SETH addresses an open question about the hardness of NP-complete problems under worst-case conditions: What happens as the number of variables in a SAT formula gets larger and larger? SETH’s answer is given in razor-sharp terms: You shall never do better than exhaustive search. According to Scott Aaronson, a computational complexity expert at MIT, “it’s like ‘P not equal to NP’ on turbochargers.”

**The Upside of Impossible**

Paradoxically, it’s SETH’s sharpness about what cannot be done that makes it so useful to
complexity researchers. By assuming that certain problems are computationally intractable under precise constraints, researchers can make airtight inferences about the properties of other problems, even ones that look unrelated at first. This technique, combined with another called reduction (which can translate one question into the mathematical language of another), is a powerful way for complexity theorists to examine the features of problems. According to Impagliazzo, SETH’s precision compared to that of other hardness conjectures (such as P not equal to NP) is a bit like the difference between a scalpel and a club. “We’re trying to [use SETH to] form more delicate connections between problems,” he said.

SETH speaks directly about the hardness of NP-complete problems, but some surprising reductions have connected it to important problems in the complexity class P — the territory of so-called easy or efficiently solvable problems. One such P-class problem is edit distance, which computes the lowest number of operations (or edits) required to transform one sequence of symbols into another. For instance, the edit distance between book and back is 2, because one can be turned into the other with two edits: Swap the first o for an a, and the second o for a c.

Indyk and Bačkurs were able to prove a connection between the complexity of edit distance and that of k-SAT, a version of SAT that researchers often use in reductions. K-SAT is “the canonical NP-complete problem,” Aaronson said, which meant that Indyk could use SETH, and its pessimistic assumptions about k-SAT’s hardness, to make inferences about the hardness of the edit-distance problem.

The result was striking because edit distance, while theoretically an easy problem in the complexity class P, would take perhaps 1,000 years to run when applied to real-world tasks like comparing genomes, where the number of symbols is in the billions (as opposed to book and back). Discovering a more efficient algorithm for edit distance would have major implications for bioinformatics, which currently relies on approximations and shortcuts to deal with edit distance. But if SETH is true — which Indyk and Bačkurs’ proof assumes — then there is no hope of ever finding a substantially better algorithm.

The key word, of course, is “if.” Indyk readily concedes that their result is not an unconditional impossibility proof, which is “the holy grail of theoretical computer science,” he said. “Unfortunately, we are very, very far from proving anything like that. As a result, we do the next best thing.”

Indyk also wryly admits that he was “on the receiving end of several tweets” regarding the Globe’s overstatement of his and Bačkurs’ achievement. “A more accurate way of phrasing it would be that [our result] is strong evidence that the edit-distance problem doesn’t have a more efficient algorithm than the one we already have. But people might vary in their interpretation of that evidence.”

Ryan Williams certainly interprets it differently. “It’s a remarkable connection they made, but I have a different interpretation,” he said. He flips the problem around: “If I want to refute SETH, I just have to solve edit distance faster.” And not even by a margin that would make a practical dent in how genomes get sequenced. If Williams or anyone else can prove the existence of an edit-distance algorithm that runs even moderately faster than normal, SETH is history.

And while Williams is one of the only experts trying to refute SETH, it’s not a heretical position to take. “I think it’s entirely possible,” Aaronson said. Williams is making progress: His latest research refutes another hardness assumption closely related to SETH. (He is preparing the work for publication.) If refuting SETH is scaling Everest, this latest result is like arriving at base camp.

Even though falsifying SETH “could be the result of the decade,” in Aaronson’s words, to hear
Williams tell it, SETH’s truth or falsity is not the point. “It’s almost like the truth value isn’t so relevant to me while I’m working,” he said. What he means is that the scalpel of SETH is double-edged: Most researchers like to prove results by assuming that SETH is true, but Williams gets more leverage by assuming it is false. “For me it seems to be a good working hypothesis,” he said. “As long as I believe it’s false, I seem to be able to make lots of progress.”

Williams’ attempts at disproving SETH have borne considerable fruit. For example, in October he will present a new algorithm for solving the “nearest neighbors” problem. The advance grew out of a failed attempt to disprove SETH. Two years ago, he took a tactic that he had used to try and refute SETH and applied it to the “all-pairs shortest paths” problem, a classic optimization task “taught in every undergraduate computer science curriculum,” he said. His new algorithm improved on computational strategies that hadn’t significantly changed since the 1960s. And before that, another abortive approach led Williams to derive a breakthrough proof in a related domain of computer science called circuit complexity. Lance Fortnow, a complexity theorist and chairman of the Georgia Institute of Technology’s School of Computer Science, called Williams’ proof “the best progress in circuit lower bounds in nearly a quarter century.”

The Map and the Territory

In addition to these peripheral benefits, attacking SETH head-on helps researchers like Williams make progress in one of the central tasks of theoretical computer science: mapping the territory. Just as we know more about the surface of the moon than we do about the depths of our own oceans, algorithms are all around us, and yet they seem to defy researchers’ efforts to understand their properties. “In general, I think we underestimate the power of algorithms and overestimate our own ability to find them,” Williams said. Whether SETH is true or false, what matters is the ability to use it as a tool to map what Williams calls the topography of computational complexity.

Indyk agrees. While he didn’t prove that edit distance is impossible to solve more efficiently, he did prove that this theoretically tractable problem is fundamentally connected to the intrinsic hardness of NP-complete problems. The work is like discovering a mysterious isthmus connecting two landmasses that had previously been thought to be oceans apart. Why does this strange connection exist? What does it tell us about the contours of the mathematical coastline that defines hard problems?

“P versus NP and SETH are ultimately asking about the same thing, just quantifying it differently,” Aaronson said. “We want to know, How much better can we do than blindly searching for the answers to these very hard computational problems? Is there a quicker, cleverer way to mathematical truth, or not? How close can we get?” The difference between solving the mysteries of SETH and those of P versus NP, Aaronson adds, may be significant in degree, but not in kind. “What would be the implications of discovering one extraterrestrial civilization versus a thousand?” he mused. “One finding is more staggering than the other, but they’re both monumental.”

This article was reprinted on Wired.com.