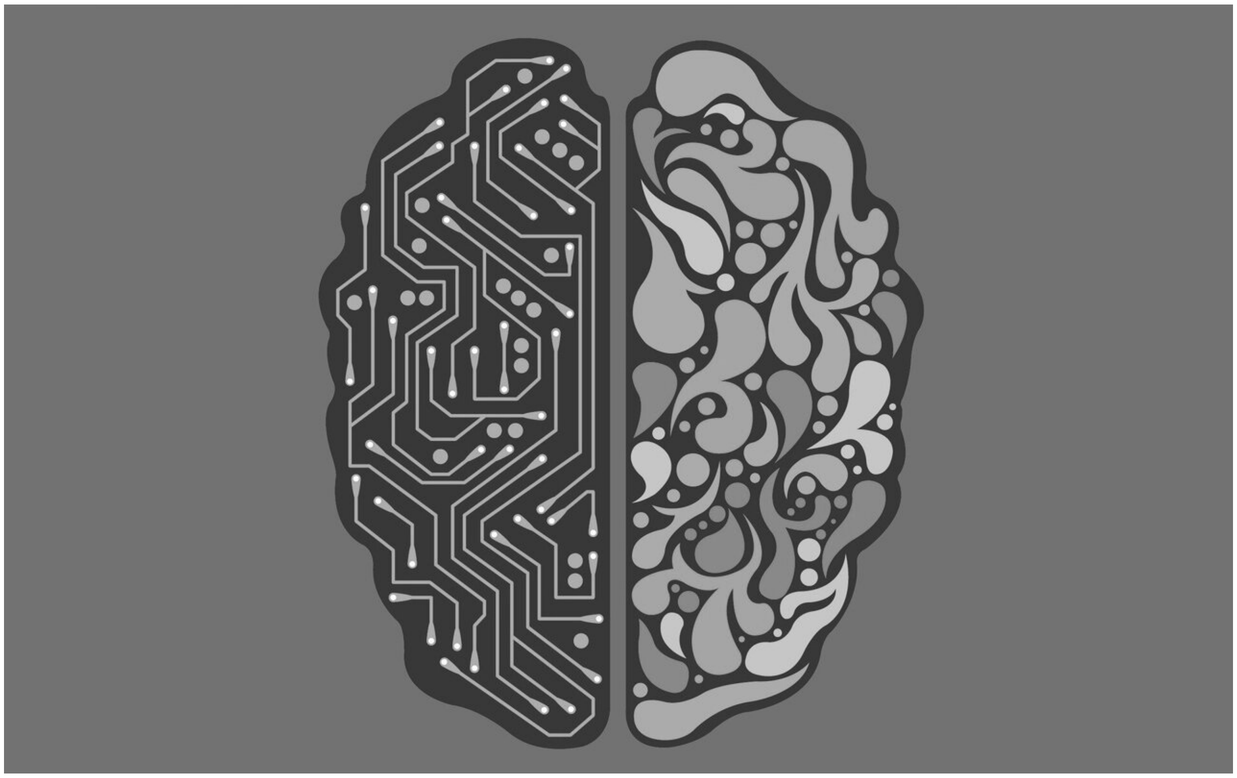
Limits to computing: A computer scientist explains why even in the age of AI, some problems are just too difficult

January 30 2023, by Jie Wang



Credit: Pixabay/CC0 Public Domain

Empowered by artificial intelligence technologies, computers today can engage in convincing conversations with people, compose songs, paint paintings, play chess and go, and diagnose diseases, to name just a few examples of their technological prowess.

These successes could be taken to indicate that computation has no limits. To see if that's the case, it's important to understand what makes a computer powerful.

There are two aspects to a computer's power: the number of operations its hardware can execute per second and the efficiency of the algorithms it runs. The hardware speed is limited by the laws of physics. Algorithms—basically sets of instructions—are written by humans and translated into a sequence of operations that computer hardware can execute. Even if a computer's speed could reach the physical limit, computational hurdles remain due to the limits of algorithms.

These hurdles include problems that are impossible for computers to solve and problems that are theoretically solvable but in practice are beyond the capabilities of even the most powerful versions of today's computers imaginable. Mathematicians and computer scientists attempt to determine whether a problem is solvable by trying them out on an imaginary machine.

An imaginary computing machine

The modern notion of an algorithm, known as a Turing machine, was formulated in 1936 by British mathematician Alan Turing. It's an imaginary device that imitates how arithmetic calculations are carried out with a pencil on paper. The Turing machine is the template all computers today are based on.

To accommodate computations that would need more paper if done manually, the supply of imaginary paper in a Turing machine is assumed to be unlimited. This is equivalent to an imaginary limitless ribbon, or "tape," of squares, each of which is either blank or contains one symbol.

The machine is controlled by a finite set of rules and starts on an initial sequence of symbols on the tape. The operations the machine can carry out are moving to a neighboring square, erasing a symbol and writing a symbol on a blank square. The machine computes by carrying out a sequence of these operations. When the machine finishes, or "halts," the symbols remaining on the tape are the output or result.

Computing is often about decisions with yes or no answers. By analogy, a medical test (type of problem) checks if a patient's specimen (an instance of the problem) has a certain disease indicator (yes or no answer). The instance, represented in a Turing machine in digital form, is the initial sequence of symbols.

A problem is considered "solvable" if a Turing machine can be designed that halts for every instance whether positive or negative and correctly determines which answer the instance yields.

Not every problem can be solved

Many problems are solvable using a Turing machine and therefore can be solved on a computer, while many others are not. For example, the domino problem, a variation of the tiling problem formulated by Chinese American mathematician Hao Wang in 1961, is not solvable.

The task is to use a set of dominoes to cover an entire grid and, following the rules of most dominoes games, matching the number of pips on the ends of abutting dominoes. It turns out that there is no algorithm that can start with a set of dominoes and determine whether or not the set will completely cover the grid.

Keeping it reasonable

A number of solvable problems can be solved by algorithms that halt in a reasonable amount of time. These "polynomial-time algorithms" are efficient algorithms, meaning it's practical to use computers to solve instances of them.

Thousands of other solvable problems are not known to have polynomial-time algorithms, despite ongoing intensive efforts to find such algorithms. These include the Traveling Salesman Problem.

The Traveling Salesman Problem asks whether a set of points with some points directly connected, called a graph, has a path that starts from any point and goes through every other point exactly once, and comes back to the original point. Imagine that a salesman wants to find a route that passes all households in a neighborhood exactly once and returns to the starting point.

These problems, called NP-complete, were independently formulated and shown to exist in the early 1970s by two computer scientists, American Canadian Stephen Cook and Ukrainian American Leonid Levin. Cook, whose work came first, was awarded the 1982 Turing Award, the highest in computer science, for this work.

The cost of knowing exactly

The best-known algorithms for NP-complete problems are essentially searching for a solution from all possible answers. The Traveling Salesman Problem on a graph of a few hundred points would take years to run on a supercomputer. Such algorithms are inefficient, meaning there are no mathematical shortcuts.

Practical algorithms that address these problems in the real world can only offer approximations, though the approximations are improving. Whether there are efficient polynomial-time algorithms that can solve

NP-complete problems is among the seven millennium open problems posted by the Clay Mathematics Institute at the turn of the 21st century, each carrying a prize of US$1 million.

Beyond Turing

Could there be a new form of computation beyond Turing's framework? In 1982, American physicist Richard Feynman, a Nobel laureate, put forward the idea of computation based on quantum mechanics.

In 1995, Peter Shor, an American applied mathematician, presented a quantum algorithm to factor integers in polynomial time. Mathematicians believe that this is unsolvable by polynomial-time algorithms in Turing's framework. Factoring an integer means finding a smaller integer greater than 1 that can divide the integer. For example, the integer 688,826,081 is divisible by a smaller integer 25,253, because 688,826,081 = 25,253 x 27,277.

A major algorithm called the RSA algorithm, widely used in securing network communications, is based on the computational difficulty of factoring large integers. Shor's result suggests that quantum computing, should it become a reality, will change the landscape of cybersecurity.

Can a full-fledged quantum computer be built to factor integers and solve other problems? Some scientists believe it can be. Several groups of scientists around the world are working to build one, and some have already built small-scale quantum computers.

Nevertheless, like all novel technologies invented before, issues with quantum computation are almost certain to arise that would impose new limits.

This article is republished from The Conversation under a Creative