NETWORKS

Researchers Achieve 'Absurdly Fast' Algorithm for Network Flow

By ERICA KLARREICH

June 8, 2022

Computer scientists can now solve a decades-old problem in practically the time it takes to write it down.

12 |

2Dwork/Shutterstock

A team of computer scientists has come up with a dramatically faster algorithm for one of the oldest problems in computer science: maximum flow. The problem asks how much material can flow through a network from a source to a destination if the links in the network have capacity limits.

The new algorithm is "absurdly fast," said Daniel Spielman of Yale University. "I was actually inclined to believe ... algorithms this good for this problem would not exist."

Maximum flow has been studied since the 1950s, when it was formulated to study the Soviet railway system. "It's older than maybe the theory of computer science," said Edith Cohen of Google Research in Mountain View, California. The problem has many applications: internet data flow, airline scheduling and even matching job applicants to open positions. The new paper handles both maximum flow and a more general version of the problem in which you also want to minimize costs. Over the years, these two problems have inspired many of the biggest advances in algorithmic techniques. "They're almost why we have a field of algorithms," Spielman said.

The new algorithm solves these two problems in "almost linear" time, which means that the algorithm's runtime is roughly proportional to the amount of time it takes merely to write down the details of the network in the first place. No other algorithm for these problems comes close to running this fast for all possible networks. The result made him "jump up and down," said Satish Rao of the University of California, Berkeley. "It's amazing."

For now, it's primarily a theoretical advance, since the speed improvements kick in only for networks that are far larger than the ones we encounter in the real world, for which maximum flow problems can already be solved fairly quickly (at least, if they don't involve minimizing costs). But pieces of the new algorithm might see practical use within a year, predicted Richard Peng of the University of Waterloo in Canada, one of the algorithm's six creators. And in the coming years, researchers said, computer scientists will likely find ways to make it more practical and perhaps even slightly faster.

Even if improvements do come along, though, this new paper is the "slam dunk," said Aleksander Mądry of the Massachusetts Institute of Technology. It's "essentially the best possible," he said.

One Path at a Time

So many computer scientists have studied maximum flow that conference talks about past work look like the credits after a movie, Peng said. But most people date the first formal algorithm to 1956, when Lester Ford and Delbert Fulkerson solved maximum flow using what's called a "greedy" approach — one that, at every step, uses the objects that come most easily to hand.

To understand this approach, imagine a network of highways on which you'd like to send as many delivery trucks as possible from Los Angeles to New York City in a given amount of time. Ford and Fulkerson's algorithm starts by choosing one path from LA to New York and routing as many trucks along it as possible. This typically won't take advantage of the full capacity of all the roads on that particular path: If one segment of your path is a five-lane highway but it leads into a two-lane bridge, you can launch only two trucks at any moment.

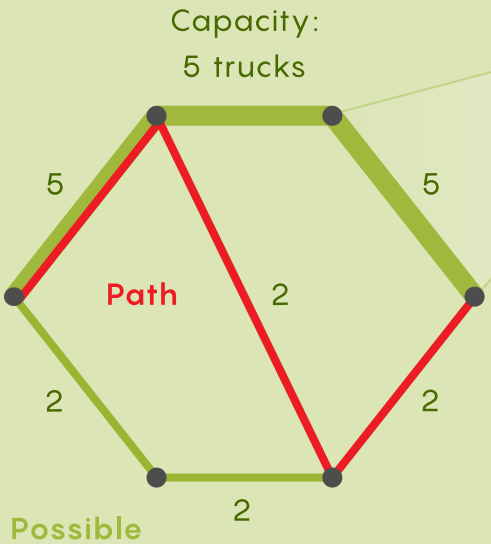
Next, the algorithm revises the capacities of these segments to reflect that you've now used some of their capacity: It subtracts the number of trucks you sent from the capacities of the segments, so the five-lane highway now only has a capacity of 3 and the two-lane bridge has a capacity of zero. At the same time, the algorithm adds 2 to the capacities of these highways in the reverse direction, so we can undo some of this flow later if we wish.

The algorithm then finds a new path from LA to New York that has room for some trucks, sends as many trucks along it as possible, and updates capacities again. After a finite number of these steps, there will be no path from LA to New York that can accept any more trucks, and then we're done — we just combine all the flows we've constructed, and we have the maximum possible flow through the network.

Finding Maximum Flow

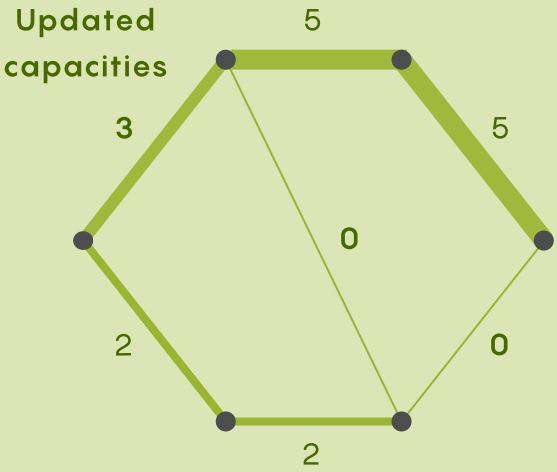
Imagine a delivery company hoping to send as many trucks as possible on a cross-country route. A maximum flow algorithm will determine how many can go at once.

The algorithm chooses a path from Los Angeles to New York.

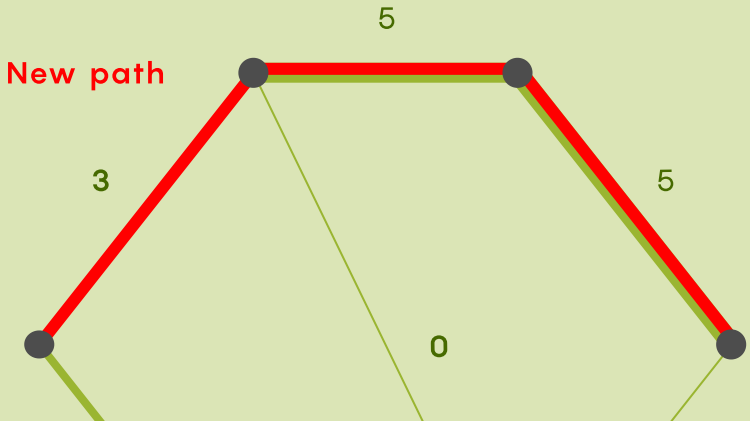


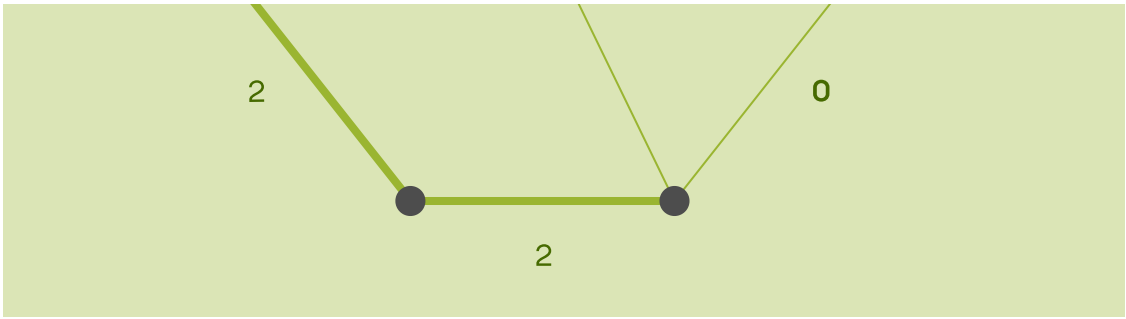
If a five-lane highway narrows to a two-lane bridge, only two trucks can be sent at a time.

The algorithm then takes those trucks into account and revises capacities.



Then it finds a new path, routes as many trucks along it as possible and updates capacities. Eventually it will have used every possible path.





Merrill Sherman/Quanta Magazine

Ford and Fulkerson's algorithm doesn't try to make smart choices along the way. If it chooses a path that cuts off other useful routes, that's just a problem it deals with later. In the decades that followed the algorithm's publication, researchers tried to speed up the runtime by having the algorithm make more judicious choices — such as always using the shortest available path or the one with the most available capacity.

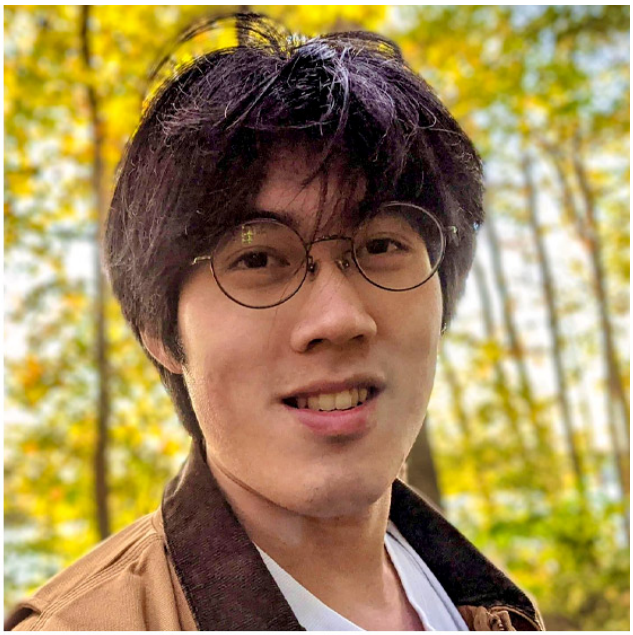
In 1970, Yefim Dinitz found an efficient way to use all the shortest paths in the network in a single step. This created an algorithm whose runtime, in networks with low capacities, was shown by Shimon Even and Robert Tarjan to be a multiple of $m^{1.5}$, where m is the number of links in the network. (The Ford-Fulkerson algorithm, by comparison, takes a multiple of m^2 steps in low-capacity networks.) Almost 30 years later, Rao and Andrew Goldberg, who is now at Amazon, produced similar results for networks with high capacities. But no one could beat the 1.5 exponent. "Coming into the 2000s ... this $m^{1.5}$ barrier was entrenched," said Sushant Sachdeva of the University of Toronto, one of the new paper's authors.

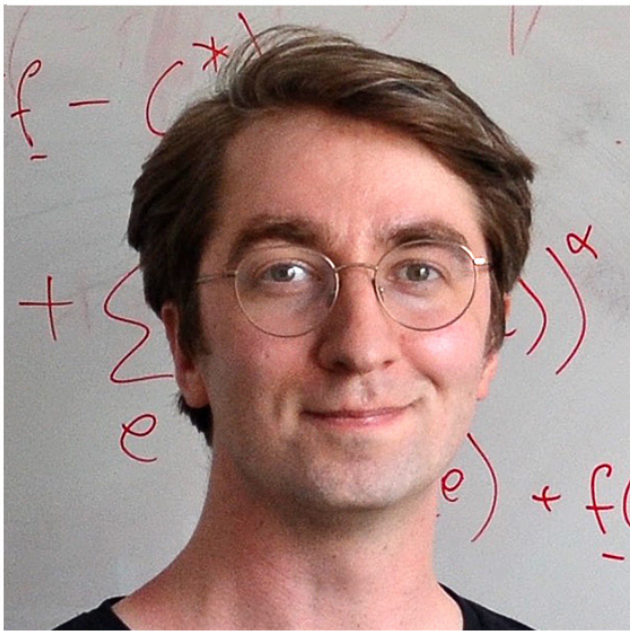
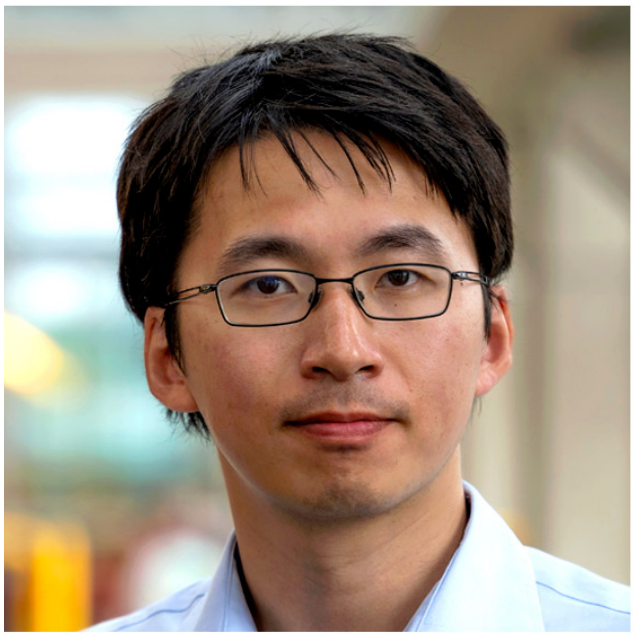
To make progress, computer scientists would have to take an entirely different approach.

From Combinations to Calculus

So far, all these algorithms employed combinatorial approaches, which look for some type of structure at each step and use that structure to update the flow. But in 2003, Spielman and Shang-Hua Teng of the University of Southern California opened the door to a radically different "optimization" approach, in which you use techniques from calculus to guide you toward an optimal solution.

Spielman and Teng developed a fast optimization algorithm that solves not the maximum flow problem, but the closely related problem of finding the lowest-energy electrical flow through a network of wires that each have a given resistance. Spielman and Teng's advance was "a key moment," said Sachdeva.


The team that created a super-fast algorithm that can determine a network's maximum flow and minimum cost: (clockwise from top left) Yang Liu, Li Chen, Rasmus Kyng, Maximilian Probst Gutenberg, Richard Peng, and Sushant Sachdeva.

—

(clockwise from top left) Courtesy of Yang Liu; Eva Huang; Courtesy of Rasmus Kyng; Courtesy of Maximilian Probst; Joe Petrik; Pushkarini Agharkar

Researchers soon started exploring how to apply this advance to the maximum flow problem. The idea is to imagine our highway network as a network of wires and to turn up the resistance on the highways that don't have much available capacity, to discourage electrons from running through them. Because of Spielman and Teng, we can quickly calculate the flow of electricity through these wires, and it will have the rough attributes of the maximum flow of vehicles through the highways. (They won't be exactly the same since the electrical flow problem doesn't feature any hard limits on the capacities of the wires.)

Then, having produced this initial flow, you readjust the capacities just as in the combinatorial algorithms like Ford and Fulkerson's. Next you reset the resistance of each wire to reflect these changed capacities and solve this new electrical problem, and so on.

Unlike the combinatorial approaches, which change one chunk of the network at a time, Spielman and Teng's optimization approach takes in the entire sweep of the network at once. That makes each step more powerful, so you need fewer total steps to reach the maximum. In 2008, Samuel Daitch of Google and Spielman used this approach to essentially match the previous $m^{1.5}$ bound for maximum flow. Then in 2013, Mądry pushed the approach further to break through the $m^{1.5}$ barrier for low-capacity networks, speeding up the runtime to a multiple of about $m^{1.43}$. "That was a shocker," Rao said.

Over the following years, researchers extended this approach even further, but they got stuck at $m^{1.33}$. They started to wonder if this exponent was a fundamental limit.

The fastest conceivable runtime for a maximum flow algorithm would just be a multiple of m (that is, $m^{1.0}$), since it takes some multiple of m steps just to write down a network. This is referred to as linear time. But to many researchers, such a blazingly fast algorithm seemed unthinkable.

"There was no good reason to believe we could do that," Spielman said.

Reduce, Reuse, Recycle

But the authors of the new paper felt there was more juice to squeeze out of Daitch and Spielman's approach. Mądry had used it to reduce the number of steps needed to solve a maximum flow problem, but that reduction came at a price: At each step, the entire network had to be rewritten and its electrical flow solved from scratch.

This method treated Spielman and Teng's algorithm as a black box — it didn't matter what the algorithm was doing internally. The six researchers decided instead to dig into the guts of the algorithm and tailor its various components to the maximum flow problem. These components, they suspected, might even allow them to solve the harder "minimum cost" problem, in which you're looking for the cheapest way to route a given amount of material. Computer scientists have long known that any minimum cost algorithm can solve the maximum flow problem as well.

As with other optimization approaches, the researchers came up with a notion of the "energy" of a flow — a function that considers the links' costs and capacities. Shifting flow from an expensive, low-capacity link to a cheap, high-capacity link lowers the total energy of the system.

To figure out how to quickly move a flow toward a low-energy state, the researchers merged this optimization approach with the older combinatorial viewpoint. The latter, which changes only part of the network at a time, offers the potential of reusing some of your work from previous steps.

At each step the algorithm looks for a cycle — a path that starts and ends at the same point — that can reduce energy. The basic idea is simple: Imagine that you've created a flow that routes trucks from Chicago to New York along a toll road, but then you discover there's a cheaper freeway available. Adding the cycle that starts in New York, runs to Chicago along the expensive road and comes back along the cheaper route effectively undoes the expensive path and replaces it with the cheaper one, reducing the total cost of the flow.

This approach uses many more steps than the electrical approach, so at first glance it "sounds like regressing," said Valerie King of the University of Victoria in Canada. To compensate, at each step the algorithm must find a good cycle incredibly quickly — faster than it takes just to inspect the entire network.

That's where the inner workings of Spielman and Teng's algorithm come in. Their algorithm provides a novel way to use a "low-stretch spanning tree" — a sort of internal backbone that captures many of the network's most salient features. Given such a tree, there's always at least one good cycle you can build by adding a single link from outside the tree. So having a low-stretch spanning tree drastically reduces the number of cycles you need to consider.

Even then, for the algorithm to run quickly, the team couldn't afford to build a brand new spanning tree at every step. Instead, they had to ensure that each new cycle caused only minor ripple effects in the spanning trees, so they could reuse most of their previous computations. Achieving this level of control was "the core difficulty," said Yang Liu, a graduate student at Stanford University who is one of the paper's authors.

Eventually, the researchers created an algorithm that runs in "almost linear" time, meaning that as you look at larger and larger networks, its runtime approaches some multiple of m. It's a "tour de force," Mądry said.

The algorithm uses many ideas from Spielman and Teng's work, but it puts them together into something completely new. Looking at the algorithm is like encountering a car if you've only seen horse-drawn carriages, Spielman said. "I see it's got a place to sit, I see it's got wheels, I see it's got something that makes it move. But they've replaced the horse with an engine."

The team's analysis is long and complicated, but other researchers will soon dive in to simplify things, Rao predicted. "My feeling is that it will all be made cleaner and better over the next few years," he said.

Once the algorithm is streamlined, computer scientists will likely start using it as a subroutine in algorithms solving different problems, Spielman said. "Now that we know we can do this really quickly, people will find all sorts of applications for it that they just weren't thinking of before."

This dizzying speedup to the maximum flow problem has Spielman wondering about the other central problems in the theory of algorithms. "What else could we do?"