From *SIAM News*, Volume 38, Number 1, January/February 2005

Computer Scientist Finds Small-Memory Algorithm for Fundamental Graph Problem

By Sara Robinson

Important connections are often made serendipitously, by researchers attempting to prove something else. Such was the case in the 1970s when Leslie Valiant, now a professor of computer science at Harvard University, discovered that a type of graph defined by researchers in information theory had important implications for computer science, implications that continue to be explored to this day.

Since Valiant's insight, computer scientists have used "expander graphs," or "expanders" as they are known, to design communication networks and sorting algorithms, to create linear-time encodable and decodable error-correcting codes, and to solve problems in statistical physics. In pure mathematics, expanders have been used to resolve problems in measure theory, geometry and topology, and computational group theory.

Most recently, Omer Reingold, a computer scientist at the Weizmann Institute of Science, used expander techniques to resolve a longstanding open problem in computation. Reingold devised an algorithm that uses, within a constant factor, the smallest possible amount of computer memory to decide whether two vertices in a given undirected graph are connected, finding a path between them if they are.

Imagine that you are lost without a map in a city where you do not speak the language, and need to find the way back to your car. Your only resource is a tiny, pocket-sized notebook. The Reingold algorithm, described in a preliminary paper posted in November, would provide you with a systematic way to explore the city, street by street, until you found your car. What is more, the same sequence of lefts and rights would work in any other city.

The result comes at a time when enormous data sets are becoming commonplace and small-memory algorithms, though rare, are increasingly important, says Avi Wigderson, a professor of computer science at the Institute for Advanced Study in Princeton. Among small-memory algorithms, Reingold's result is central, Wigderson adds, because it implies the existence of small-memory algorithms for many other important problems, such as finding a minimum-weight spanning tree for a graph or determining whether a graph can be embedded in the plane.

The problem of determining whether two vertices in an undirected graph are connected, nicknamed "undirected connectivity," is a special case of the corresponding problem for directed graphs, for which a small-memory algorithm would have profound philosophical implications, says Madhu Sudan, a professor of computer science at MIT. A similarly space-efficient algorithm for directed connectivity would resolve the computer-memory analog of the P versus NP problem.

For Reingold, however, the problem's primary allure was its connection to an important thread of recent research: the tradeoff between two resources, memory and randomness. For many computational problems, there is an efficient (with respect to time or space) randomized algorithm, but no comparably efficient deterministic one. Undirected connectivity, for which a simple, randomized, log-space algorithm has existed since the 1970s, was such a problem—until now. The new algorithm, Reingold says, hints at a broader conclusion: that randomness cannot save computer memory. "I am optimistic that my algorithm will eventually lead to such a general result, and this is the main implication I am after," he says. (Understanding the tradeoff between time and randomness is thought to be a significantly harder problem.)

A Serendipitous Connection

During the summer of 1974, Valiant, then at Leeds University in England, was thinking about a fundamental problem in the emerging field of computational complexity: He was trying to show that certain simple functions, such as the product of two n-digit numbers, are not computable in a linear number of operations.

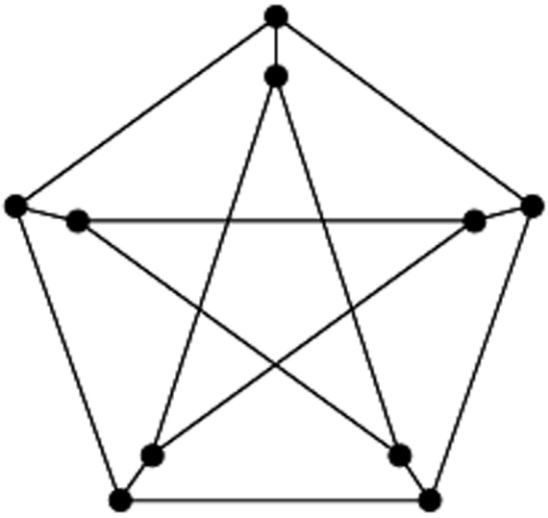
After writing down the computation as a circuit—a directed, acyclic graph with logic gates as its vertices, linking input digits on one side with output digits on the other—Valiant observed that a lot of information must be communicated from any k input digits to any k output digits. He captured this observation with a graph theoretic property: Given sets of input and output digits of size k, there are k vertex-disjoint paths linking them.

It seemed to Valiant that graphs with so many disjoint paths might necessarily have many edges. If so, the circuits he was looking at would necessarily be superlinear in size, providing the lower bound he was seeking. Eventually, Valiant reduced his graph condition to a simpler one, and continued to ponder whether it is possible to have graphs with such properties.

Shortly afterward, during a trip to Paris, he happened upon a reference to a coding theory paper that addressed his graph question. Written a year earlier by Mark Pinsker, a Russian information theorist, the paper defined and showed the existence of linear-size graphs, which Pinsker called "concentrators," with exactly the simplified property described by Valiant. Concentrators stemmed from expanders, a more fundamental concept that Pinsker and L.A. Bassalygo had introduced in a paper on switching networks written in 1971. Like concentrators, expanders are seeming contradictions: They are simultaneously well connected and sparse.

From Pinsker's concentrator construction, Valiant was able to derive graphs with precisely the stronger graph property of his circuits, yet only a linear number of edges. Following Pinsker, Valiant called his graphs "superconcentrators."

Although Valiant was not able to resolve his question about functions (it remains open to this day), he demonstrated that his superconcentrators have many important applications in algorithm and network design. This work introduced the power of concentrators, superconcentrators, and expanders to the mathematics and computer science communities, where they have continued to play a crucial role in many areas of research, most recently in Reingold's result on undirected connectivity in a graph.



The Petersen graph is a good (albeit small) expander.

Undirected Connectivity in Log-Space: A 35-Year Quest

The time complexity of undirected connectivity was resolved in the 1970s. The standard graph-exploration algorithms—breadth-first search and depth-first search—take time linear in the number of vertices and edges, the best possible. These algorithms also use linear space, however.

Reingold's algorithm, given two vertices in an undirected graph, finds a path connecting them (if there is one) using an amount of computer memory logarithmic in the number of vertices. The algorithm places the problem in what computer scientists call "log-space": the collection of problems solvable with an amount of memory that is a constant multiple of the logarithm of the size of the problem. Because logarithmic memory is required just to remember the name of a single node of an n-vertex graph, the algorithm uses within a constant factor of the least amount of memory possible. It makes it possible to navigate a graph with the ability to remember only a constant number of vertices at a time.

Like the expander tools finally used to resolve it, the quest for a small-memory algorithm for the undirected connectivity problem dates to the early 1970s. The first effort came in 1970, when J. Savitch devised a $(\log n)^2$-memory algorithm that runs in superpolynomial time. Then, in 1979, Romas Aleliunas, Richard Karp, Richard Lipton, László Lovász, and Charles Rackoff made a leap forward by showing that a random walk, starting at any vertex of a connected, undirected graph, will visit all the vertices in a polynomial number of steps. Because their random walk algorithm needs to remember only the current vertex and the number of steps already taken, it yielded a probabilistic log-space algorithm for undirected connectivity. In effect, the quest had been redefined as a problem of derandomization.

Reingold's method relies on techniques altogether different from those of the previous efforts: Rather than paring down the input graph, his algorithm expands it.

The next major advance came in 1992, when Noam Nisan built on a critical insight of Steve Cook, best known for defining the notion of NP-completeness. In the 1970s, Cook conceived of the notion of a "universal traversal sequence" for any graph of a given size—a predetermined sequence of directions that guarantees a visit to every vertex in a connected component—and speculated that such a sequence could address the space complexity of undirected connectivity.

In their paper, Aleliunas et al. answered Cook's challenge by giving a counting argument that polynomial-length UTSs exist, although they were not able to construct one explicitly. Nisan, in his PhD thesis at the University of California, Berkeley, then showed how to generate a UTS of close to polynomial length using $O((\log n)^2)$ space. Shortly thereafter, Nisan, with Wigderson and Endre Szemeredi, built on Nisan's UTS work to reduce the space required to $O((\log n)^{3/2})$. They did this by iteratively creating new, much smaller graphs that captured the connectivity of the original. Wigderson and other colleagues were subsequently able to shave the exponent down even further, to 4/3.

In November, Reingold ended the quest by describing a way to generate a "universal exploration sequence" (similar to a UTS) of polynomial length using logarithmic memory. His method relies on techniques altogether different from those of the previous efforts: Rather than paring down the input graph, his algorithm expands it.

Expanders

Of the many equivalent definitions for expander graphs, the most intuitive is geometric: An (n,d,c)-expander, $c > 0$, is a d-regular graph on n vertices for which disconnecting any set of $k < n/2$ nodes from the rest requires severing ck edges. This definition, while it captures the notion of high connectivity, enforces sparseness only when the degree d is small relative to the number of nodes. In applications, what is typically needed is not a single graph, but an infinite family of expanders of fixed degree. As the number of vertices grows, the graphs become increasingly sparse.

The simplest way of proving the existence of such families is via a probabilistic construction originally given by Pinsker. Pinsker showed that almost every random graph is an expander, although he did not know how to construct one explicitly.

He posed that problem to Gregori A. Margulis, the Russian mathematician who, in 1973, gave the first explicit construction of a family of expanders. Margulis's construction, and most later ones, rely on an algebraic property of expanders: the relation between a graph's expansion properties and the ratio between the largest and second-largest eigenvalues of its adjacency matrix. A d-regular graph is a good expander if the magnitude of the second-largest eigenvalue of its adjacency matrix is strictly smaller than the degree. (And a family of graphs is an expander family if the second eigenvalues of all the graphs are uniformly bounded by $\lambda < d$.)

Expanders also have the useful probabilistic property that random walks on them converge rapidly to the uniform distribution, at a rate governed by the second eigenvalue. Computer scientists have devised approximation algorithms for combinatorial

properties of physical systems that rely on this feature of expanders.

Margulis's and subsequent constructions were quite technical, requiring high-powered theorems from algebra and number theory to prove the expansion properties. Wigderson found this dissatisfying: "For me, I always wanted an expander where I knew why it was expanding," he says.

Another Serendipitous Connection

In 1999, Wigderson and Reingold, working with Salil Vadhan, came up with a relatively simple combinatorial method for iteratively constructing a family of expanders. The basis of their method is an operation they call the "zigzag product": It takes two graphs, a large expander G and a much smaller one, H, and creates a new graph that is roughly the size of the larger graph, has degree determined only by the degree of the smaller graph, and has an expansion only slightly worse than either. "The expansion of the product graph follows from that of its components via an intuitive 'information flow' argument, and can be formalized by half a page of linear algebra," Wigderson explains.

Because the definition of expansion does not constrain the number of edges, it is easy to improve a graph's expansion properties by adding new edges. This method will not construct an expander family, however, because the degree will not remain constant. The key property of the zigzag product of G with H is that it retains most of the expansion qualities of G while reducing its degree. Thus, by alternately zigzagging and adding edges, it is possible to construct an explicit family of expanders.

In the 1999 paper, the researchers also show that a similar, simpler graph product called the "replacement product" can be used to generate an expander family with slightly weaker expansion properties. In the replacement product of G and H, each vertex of the larger graph G is replaced by a copy of the smaller graph H. The number of vertices of H is chosen to match the degree of G, so that each vertex in each copy of H can be connected to an edge in G. The resulting graph then behaves locally like H and globally like G and thus inherits the expansion properties of both.

A year ago, Reingold had a key insight—that it might be possible to reduce the computer memory needed to solve undirected connectivity by adding more paths to the input graph, making it easier to explore. Indeed, if he could somehow turn the graph into an expander, he would be done: Because expander graphs of size n have the property that their diameter—the longest distance between any two vertices—is $O(\log(n))$, a procedure that enumerates all possible paths of length $O(\log(n))$ from a starting point of a graph and explores them one after another would be guaranteed to find all vertices lying within that connected component.

But it wasn't until last summer, while visiting a colleague at UC Berkeley, that Reingold saw how he could make this approach work. The essence of his algorithm is this: Given an input graph G and vertices S and T, it alternately adds edges and applies the zigzag or replacement product, turning the connected component of S into a constant-degree expander, G'. The tricky parts are to show that an arbitrary graph can be transformed into an expander using the graph products, and to carry out this transformation using only a logarithmic amount of space. (The transformation does not require storing a full description of G' because each bit of it is computable in log-space.)

Reingold's construction is deceptively simple, Wigderson says; it took five years and a leap of insight to see the connection to expanders and the zigzag product. "I can tell you why I didn't find the algorithm," he says. "All the previous algorithms went by shrinking the graph; [Reingold's] does the opposite."

Postscript: In what may be a classic example of great work/unfortunate timing, Vladimir Trifonov, a student at the University of Texas, recently posted a paper giving a $O((\log n)(\log\log n))$-space algorithm for undirected connectivity, by a different method. If correct, the algorithm would surpass all previous efforts, except Reingold's.

Sara Robinson is a freelance writer based in Los Angeles, California.