The Mathematical Work of Daniel Spielman

Michel X. Goemans and Jonathan A. Kelner

The *Notices* solicited the following article describing the work of Daniel Spielman, recipient of the 2010 Nevanlinna Prize. The International Mathematical Union also issued a news release about the prize, which appeared in the December 2010 issue of the *Notices*.

The Rolf Nevanlinna Prize is awarded once every four years at the International Congress of Mathematicians by the International Mathematical Union for outstanding contributions in mathematical aspects of information sciences. The 2010 recipient was Daniel Spielman, who was cited for "smoothed analysis of Linear Programming, algorithms for graph-based codes, and applications of graph theory to Numerical Computing".

In this article, we summarize some of Spielman's seminal contributions in these areas. Unfortunately, because of space constraints, we can barely scratch the surface and have to leave out many of his impressive results (and their interconnections) over the last two decades.

Error-Correcting Codes

Several of Spielman's important early contributions were in the design of error-correcting codes. Error-correcting codes are fundamental in our increasingly digital lives, and they are important mathematical tools in the theory of computing.

In a code of block length n and rate $r < 1$ over the alphabet $\{0,1\}$, a message in $\{0,1\}^{rn}$ is encoded into a codeword in $\{0,1\}^n$. To be able to correct errors in the transmission of a codeword, one would like the minimum distance d between two codewords to be large. A code is said to be *asymptotically good* if both the rate r and the relative distance d/n are bounded below by positive constants as n grows. The goal is to get not only asymptotically good codes with the best possible trade-off between the rate and the relative minimum distance but also codes for which both the encoding and the decoding can be done as fast as possible, ideally in linear time. In his Ph.D. work, Spielman and his advisor Michael Sipser proposed the first asymptotically good codes that allow linear-time decoding (for a survey, see [6] and references therein). Their codes are low-density parity check (LDPC) codes, introduced by Gallager half a century ago, in which a sparse bipartite graph links the bits of the codeword on one side to *parity checks* on the other side that constrain sets of bits to sum to 0 over $GF(2)$; these are *linear* codes. The Sipser and Spielman codes use a bipartite expander graph as the parity check graph. The expansion property not only implies a good bound on the minimum distance but also allows a simple decoding algorithm, in which one repeatedly flips codeword bits with more than half of their neighboring constraints violated. The Sipser and Spielman codes require quadratic time for encoding, as in any linear code. Subsequently, Spielman (see also [6]) provided a more intricate construction, still based on expanders, that yielded the first asymptotically good codes with linear-time encoding *and* decoding.

In later work, Spielman and collaborators designed highly practical LDPC codes for the erasure channel model, in which some of the bits are simply lost. Their codes [4] approach the maximum capacity possible according to Shannon's theory, and they can be encoded and decoded in linear time. The decoding is based on belief propagation, which sets a missing bit whenever it can be recovered unambiguously. Despite the simplicity of these

Michel X. Goemans is Leighton Family Professor of Applied Mathematics at the Massachusetts Institute of Technology. His email address is goemans@math.mit.edu.

Jonathan Kelner is the Kokusai Denshin Denwa Assistant Professor of Applied Mathematics at MIT and a member of the MIT Computer Science and Artificial Intelligence Laboratory. His email address is kelner@mit.edu.

algorithms, the design and analysis of these codes are mathematically quite involved. Some of this work has had considerable practical impact.

Smoothed Analysis of Algorithms

One of Daniel Spielman's most celebrated contributions is his introduction of a new notion for analyzing the efficiency of algorithms, the concept of smoothed analysis, and its powerful and highly technical illustration on the simplex algorithm for linear programming. This was done in joint work with Shang-Hua Teng, his longtime collaborator.

The traditional way to measure the efficiency of an algorithm for a computational problem (such as inverting a matrix, factoring a number, or computing a shortest path in a graph) is to measure its worst-case running time over all possible instances of a given input size. Efficient algorithms are considered to be those whose worst-case running times grow polynomially with the size of the input. But this worst-case, pessimistic measure does not always reflect the behavior of algorithms on typical instances that arise in practice.

A compelling example of this is the case of linear programming, in which one aims to maximize a linear function subject to linear constraints: $\max\{\langle c, x\rangle : Ax \le b, x \in \mathbb{R}^d\}$ with the inputs $c \in \mathbb{R}^d$, $A \in \mathbb{R}^{m\times d}$, and $b \in \mathbb{R}^m$. This problem has numerous industrial applications. In the late 1940s, George Dantzig proposed the simplex algorithm for linear programming, which has been cited as one of the "top ten algorithms of the 20th century" (Dongarra and Sullivan). In the nondegenerate case, the simplex algorithm can be viewed as starting from a vertex of the polyhedron $P = \{x \in \mathbb{R}^d : Ax \le b\}$ and repeatedly moving to an adjacent vertex (connected by an edge, or face of dimension 1) while improving the value $\langle c, x\rangle$. However, no pivot rule (dictating which adjacent vertex is chosen next) is known for which the worst-case number of operations is polynomial in the size of the input. Even worse, for almost every known pivot rule, there exist instances for which the number of steps grows exponentially. Whether there exist polynomial-time algorithms for linear programming was a long-standing open question until the discovery of the ellipsoid algorithm (by Nemirovski and Shor, and Khachian) in the late 1970s and interior-point algorithms (first by Karmarkar) in the mid-1980s. Still, the simplex algorithm is the most often used algorithm for linear programming, as it performs extremely well on typical instances that arise in practice. This almost paradoxical disparity between its exponential worst-case behavior and its fast typical behavior called for an explanation. In the 1980s, researchers considered probabilistic analyses of the simplex method under various probabilistic models, but results in one model do not necessarily carry over to other probabilistic models and do not necessarily shed any light on instances that occur in practice.

Spielman and Teng [7] instead proposed *smoothed analysis*, a blend of worst-case and probabilistic analyses, which marvelously explains the typical behavior of the simplex algorithm. In smoothed analysis, one measures the maximum over all instances of a certain size of the expected running time of an algorithm under small random perturbations of the input. In the setting of the simplex algorithm for linear programming, Spielman and Teng consider any linear program given by $c \in \mathbb{R}^d$, $A \in \mathbb{R}^{m\times d}$, and $b \in \mathbb{R}^m$ and randomly perturb A and b by independently adding to each entry a Gaussian random variable of variance σ^2 times the maximum entry of A and b. Using an intricate technical argument, they prove that the *expected* number of steps of the simplex algorithm with the so-called shadow-vertex pivot rule is polynomial in the dimensions of A and $1/\sigma$. The shadow-vertex pivot rule essentially corresponds to walking along the projection of the polyhedron onto a 2-dimensional linear subspace containing c. One step of the proof is to bound the expected number of vertices along a random 2-dimensional projection by a polynomial in n, d, and $1/\sigma$. For this purpose, they prove that the angle at a vertex of the projection is unlikely to be too flat, and this is formalized and proved by showing that a random Gaussian perturbation of any given matrix is sufficiently well conditioned, a fundamental notion in numerical linear algebra. Their work has motivated further developments, including better estimates on the condition number of randomly perturbed matrices, other probabilistic models for the perturbations, and smoothed analyses for other algorithms and problems (see the Smoothed Analysis page in [5]). Smoothed analysis also suggested the first randomized polynomial-time simplex algorithm for linear programming by Daniel Spielman and the second author [3]. This could be a step toward finding a strongly (not depending on the size of the numbers involved) polynomial-time algorithm for linear programming.

Fast Algorithms for Numerical Linear Algebra and for Graph Problems

In recent years, Spielman and his collaborators have ignited what appears to be an incipient revolution in the theory of graph algorithms. By developing and exploiting deep connections between graph theory and computational linear algebra, they have introduced a new set of tools that have the potential to transform both fields. This work has already resulted in faster algorithms for several

fundamental problems in both linear algebra and graph theory.

Fast Algorithms for Laplacian Linear Systems

With any graph G, one can associate a matrix L_G known as the *graph Laplacian*. Much of Spielman's recent work has been in the realm of spectral graph theory, which studies the rich interplay between the combinatorial properties of G and the linear algebraic properties of L_G.

Motivated by scientific computing, Spielman's work in this field began with the search for a faster algorithm to solve linear systems of the form $L_G x = b$. While the best known algorithms for solving general $n \times n$ linear systems require time $O(n^{2.378})$, Spielman and Teng showed that this class of linear systems can be solved much more quickly. In a technical tour de force, they gave an algorithm [8] that approximately solves such linear systems to any given accuracy in an amount of time that is *nearly linear* in the number of edges of G. This allows one to solve sparse diagonally dominant linear systems, which are ubiquitous in scientific computing.

However, its impact has extended far beyond solving linear systems. Their algorithm works by finding simpler graphs whose Laplacians are algebraically similar to L_G and using the solutions to linear systems involving them to speed up an iterative solver. To find these graphs, they introduce two algorithmic tools that have found broader applicability and led to substantial follow-up work: spectral sparsification and local clustering.

Spectral Sparsification

The goal of sparsification is to approximate a dense graph G by a sparser graph H. In a landmark paper, Benczur and Karger showed that any graph G can be approximated in nearly linear time by a weighted graph H with $O(n \log n/\epsilon^2)$ so that the weight of every cut in H is preserved up to a factor of at most $1 + \epsilon$. This allows one to approximately solve any problem whose solution depends only on the weights of cuts by operating on H instead of G. Since H has many fewer edges when G is dense, this is usually much faster.

For their solver, Spielman and Teng introduced the notion of spectral sparsification, which requires that L_G and L_H be approximately the same as quadratic forms. This is a strictly stronger requirement that implies that H approximately preserves the weights of cuts as well. They then showed how to compute such sparsifiers with $O(n \log^{O(1)} n/\epsilon^2)$ edges in nearly linear time [9].

In later work, Batson, Spielman, and Srivastava [1] have shown that one can eliminate the polylogarithmic factors and compute spectral sparsifiers with just $O(n/\epsilon^2)$ edges. This was not previously known even for the weaker cut-based notion of sparsification. In addition to being a surprising result in graph theory, their techniques have led to a new simpler proof of an important theorem by Bourgain and Tzafriri in functional analysis and a new approach to the long-open Kadison-Singer conjecture.

Local Clustering

A key part of Spielman and Teng's construction of sparsifiers was an algorithm for *local clustering* [10]. The goal is to find a well-connected cluster of vertices that contains a given vertex v in some graph G so that the running time grows with the size of the cluster and depends only very weakly on the size of G. This can be a vast improvement over global algorithms when one is looking for a small cluster inside a huge ambient graph.

Assuming that v is sufficiently well contained in a cluster, Spielman and Teng provide an algorithm for finding this cluster using the connection between graph conductance and random walks. This has sparked an active area of research into improving and applying these techniques.

Electrical Flows and Graph Algorithms

In constructing their solver, Spielman and Teng use graph theory to speed up linear algebra. In more recent work, Spielman and his collaborators have shown how to exploit this connection in the other direction, using the linear system solver to obtain faster algorithms for several fundamental graph problems, most notably for approximating maximum flows in undirected graphs [2]. To do this, they model the edges of a graph as resistors, and they study the electrical currents and potentials that result when one imposes voltages or injects current into various parts of the graph. It turns out that this can be computed by solving a Laplacian linear system, so it can be done using the Spielman-Teng solver in nearly linear time. Since electrical flows encode complex global information about a circuit, this provides a powerful new tool to probe the structure of a graph.

Summary

We were able to give only a glimpse of Daniel Spielman's mathematical work in the theory of computing, but we hope we were able to convey that his numerous mathematical insights have led to groundbreaking results in the design and analysis of algorithms for error-correcting codes, linear programming, graph algorithms, and numerical linear algebra. We refer the interested reader to his website [5] for additional results, pointers, references, and mathematical gems.

References

[1] Joshua Batson, Daniel A. Spielman, and N. Srivastava, Twice-Ramanujan sparsifiers, arXiv:0808.0163, 2008.

[2] Paul Christiano, Jonathan A. Kelner, Aleksander Madry, Daniel A. Spielman, and Shang-Hua Teng, Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs, arXiv:1010.2921.

[3] Jonathan A. Kelner and Daniel A. Spielman, A randomized polynomial-time simplex algorithm for linear programming, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, 2006.

[4] Michael G. Luby, Michael Mitzenmacher, M. Amin Shokrollahi, and Daniel A. Spielman, Efficient erasure correcting codes; Improved low-density parity-check codes using irregular graphs, *IEEE Transactions on Information Theory*, **47**(2), 569–584; 585–598. 2001.

[5] Daniel A. Spielman, `http://www.cs.yale.edu/homes/spielman/`.

[6] Daniel A. Spielman, Constructing error-correcting codes from expanders, in: *Emerging Applications of Number Theory*, IMA Volume 109 (Dennis A. Hejhal, Joel Friedman, Martin C. Gutzwiller, and Andrew M. Odlyzko, eds.), 591–600, 1999.

[7] Daniel A. Spielman and Shang-Hua Teng, Smoothed analysis: Why the simplex algorithm usually takes polynomial time, *Journal of the ACM* **51** (3), 385–463, 2004.

[8] ———, Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems, arXiv:cs/0607105, 2006.

[9] ———, Spectral sparsification of graphs, arXiv:0808.4134, 2008.

[10] ———, A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning, arXiv:0809.3232, 2008.