Proceedings of the International Congress of Mathematicians
Hyderabad, India, 2010

The Work of Daniel A. Spielman

Gil Kalai

Dan Spielman has made groundbreaking contributions in theoretical computer science and mathematical programming and his work has profound connections to the study of polytopes and convex bodies, to error-correcting codes, expanders, and numerical analysis. Many of Spielman's achievements came with a beautiful collaboration spanned over two decades with Shang-Hua Teng. This paper describes some of Spielman's main achievements.

Section 1 describes smoothed analysis of algorithms, which is a new paradigm for the analysis of algorithms introduced by Spielman and Teng. Section 2 describes Spielman and Teng's explanation for the excellent practical performance of the simplex algorithm via smoothed analysis.

Spielman and Teng's theorem asserts that the simplex algorithm takes a polynomial number of steps for a random Gaussian perturbation of every linear programming problem.

Section 3 is devoted to Spielman's works on error-correcting codes and in particular his construction of linear-time encodable and decodable high-rate codes based on expander graphs. Section 4 describes other directions: spectral graph theory, sparsifiers, graph partitioning, numerical analysis, and linear equation solvers.

1. Smoothed analysis of algorithms

I will introduce the motivation for smoothed analysis by quoting Dan Spielman himself:

> *"Shang-Hua Teng and I introduced smoothed analysis to provide a means of explaining the practical success of algorithms and heuristics that have poor worst-case behavior and for which average-case analysis was unconvincing. The problem of explaining the success of heuristics that 'work in practice' has long plagued theoreticians. Many of these have poor worst-case complexity. While one may gain some insight into their performance by demonstrating that they have low average-case complexity, this analysis may be unconvincing as an average-case analysis is dominated by the performance of an algorithm on random inputs, and these may fail to resemble the inputs actually encountered in practice.*

Smoothed analysis is a hybrid of worst-case and average-case analyses that inherits advantages from both. The smoothed complexity of an algorithm is the maximum over its inputs of the expected running time of the algorithm under slight random perturbations of that input. The smoothed complexity is then measured as a function of both the input length and the magnitude of the perturbations.

If an algorithm has low smoothed complexity, then it should perform well on most inputs in every neighborhood of inputs. Smoothed analysis makes sense for algorithms whose inputs are subject to slight amounts of noise in their low-order digits, which is typically the case if they are derived from measurements of real-world phenomena.

The most important example of an algorithm with poor worst-case complexity and excellent practical behavior is the simplex algorithm for linear programming. We will discuss linear programming in the next section. Following Spielman and Teng, the smoothed complexity of quite a few other algorithms, of geometric, combinatorial, and numeric nature, were studied by various authors, see [48] for a survey. Some highlights are: k-means and clustering [5]; integer programming [7]; superstring approximation [30]; Smoothed formulas and graphs [26].

2. Why is the simplex algorithm so good?

2.1. Linear programming and the simplex algorithm. In 1950 George Dantzig (see [10]) introduced the *simplex algorithm* for solving linear programming problems. Linear programming and the simplex algorithm are among the most celebrated applications of mathematics. See [37, 53].

A linear programming problem is the problem of finding the maximum of a linear functional (called *a linear objective function*) on d variables subject to a system of n inequalities. The set of solutions to the inequalities is called the *feasible polyhedron* and the simplex algorithm consists of reaching the optimum by moving from one vertex to a neighboring vertex of the feasible polyhedron. The precise rule for this move is called *the pivot rule*. What we just described is sometimes called the second phase of the algorithm, and there is a first phase where some vertex of the feasible polyhedron is reached.

Understanding the complexity of linear programming and of the simplex algorithm is a major problem in mathematical programming and in theoretical computer science.

Early thoughts. The performance of the simplex algorithm is extremely good in practice. In the early days of linear programming it was believed that the common pivot rules reach the optimum in a number of steps that is polynomial or perhaps even close to linear in d and n. As we will see shortly, this belief turned out to be false.

A related conjecture by Hirsch asserts that for d-polytopes (bounded polyhedra) defined by n inequalities in d variables there is always a path of length at most $n-d$ between every two vertices. The Hirsch conjecture was recently disproved by Francisco Santos [36]. The known upper bound for diameter of graphs of polytopes is quasi-polynomial in d and n [22]. (Of course, an upper bounds for the diameter of the graph of the feasible polyhedron does not guarantee a similar bound for the number of pivots for an effective simplex type algorithm.)

The Klee-Minty example and worst-case behavior. Klee and Minty [24] found that one of the most common variants of the simplex algorithm is exponential in the worst case. In fact, the number of steps was quite close to the total number of vertices of the feasible polyhedron. Similar results for other pivot rules were subsequently found by several authors. No efficient pivot rules for linear programming is known which requires a polynomial number of pivots or even a sub-exponential number of pivot steps for every LP problem problem. There are randomized algorithms which requires in expectation a sub-exponential number of steps $\exp(K\sqrt{\log nd})$ [19, 34]. Even for randomized algorithms a polynomial number of steps is a distant goal.

***LP* ∈ *P*, the ellipsoid method and interior points methods.** What can explain the excellent practical performance? In 1979 Khachian [18] proved that $LP \in P$; namely, there is a polynomial time algorithm for linear programming. This had been a major open problem since the complexity classes P and NP were described in the late sixties, and the solution led to the discovery of polynomial algorithms for many other optimization problems [15]. Khachian's proof was based on Nemirovski and Shor's ellipsoid method, which is not practical. For a few years there was a feeling that there is a genuine tradeoff between being good in theory and being good in practice. This feeling was shattered with Karmarkar's 1984 interior point method [20] and subsequent theoretical and practical discoveries.

Average case complexity. We come now to developments that are most closely related to Spielman and Teng's work. Borgwardt [8] and Smale [38] pioneered the study of average case complexity for linear programming. It turns out that a certain pivot rule first introduced by Gass and Saaty called the *shadow boundary rule* is most amenable to average-case study. Borgwardt was able to show polynomial average-case behavior for a certain model that exhibits rotational symmetry. In the mid-80s, three groups of researchers [1, 2, 52] were able to prove *quadratic* upper bound for the simplex algorithm for very general random models that exhibit certain sign invariance.

2.2. Smoothed analysis of the simplex algorithm.

We start with a linear programming (LP) problem:

$$\mathbf{max} < \mathbf{c}, \mathbf{x} >, \quad \mathbf{x} \in \mathbf{R}^d$$

$$\textbf{subject to } \mathbf{Ax} \leq \mathbf{b}$$

Here, $\mathbf{A}$ is an n by d matrix, $\mathbf{b}$ is a column vector of length n, and $\mathbf{c}$ is a column vector of length d.

Spielman and Teng considered a Gaussian perturbation of the matrix A where a Gaussian random variable with variance σ is added independently to each entry of the matrix A.

Theorem 2.1 (Spielman and Teng [43]). *For the shadow-boundary pivot rule, the average number of pivot steps required for a random Gaussian perturbation of variance σ of an arbitrary LP problem is polynomial in d, n, and σ^{-1}.*

Spielman and Teng's proof [43] is truly a tour de force. It relies on a very delicate analysis of random perturbations of convex polytopes.

Let me mention two ingredients of the proof: The shadow-boundary method can be described geometrically as follows. Consider an orthogonal projection of the feasible polytope to two dimensions such that the starting vertex and the optimal vertex are mapped to vertices of the projection. The walk performed by the algorithm (in phase I) is a pre-image of walking along the boundary in the projection. An important step in the proof is to show that the number of vertices[1] in the projection is only polynomial in d, n, and σ^{-1}.

A crucial part of the proof is explaining what a random perturbed polytope looks like. In particular, it is required to prove that the angles at vertices of these polytopes are not "flat." Every vertex corresponds to a solution of d linear equations in d variables and the angle at the vertex is described by a certain algebraic parameter of the linear system called the *condition number*.

The study of condition numbers of random matrices is crucial to Spielman and Teng's original results as well as to many subsequent developments.

2.3. Further developments.

Smoothed analysis of interior point methods. For interior point methods there is also an unexplained exponential gap between the practical and proven number of iterations. Renegar [35] proved an upper bound of $O(\sqrt{n}L)$ for the number of iterations required for an interior point method to reach the optimum. Here, L is the number of bits in the binary description of the problem. His analysis strongly relies on the notion of condition number we men tioned above. (In fact, Renegar's result gives the upper bound $O(\sqrt{n}R)$ where R is the logarithm of the condition number.) Practical experience tells us that in real-life problems the number of iterations is logarithmic in n. Can smoothed analysis explain this as well?

Dunagan, Spielman, and Teng [11] proved that the expectation of R, the log of the condition number of any appropriately scaled linear program subject to a Gaussian perturbation of variance σ^2 is at most $O(\log nd/\sigma)$ with high probability. This may offer some explanation for the fast convergence practically observed by various interior point methods.

Improvements, extensions, and simplifications. Spielman and Teng themselves, Vershynin [55], Tao and Vu [51], and others over the years have introduced significant improvements to the polynomials estimates, simplifications of various

[1]The whole difficulty of linear programming consists in on the fact that the number of vertices of the feasible polytope can be exponential in d.

parts of the original proof, and extensions, to more general classes of perturbations. The best-known bound for the polynomial in the theorem that was proved by Vershynin is $\max(d^5 \log^2 n, d^9 \log^4 d, d^3\sigma^{-4})$.

The framework of smoothed analysis can be rendered more and more convincing by restricting the families of perturbations to more closely model the noise one would actually expect in a particular problem domain. Tao and Vu [51] were able to replace the Gaussian noise by various more general and arguably more realistic types of noise.

Towards a strongly polynomial algorithm for linear programming. One of the outstanding open problems in computational complexity is that of finding a "strongly polynomial algorithm for linear programming" [32, 39]. This roughly means an algorithm that requires a polynomial number of arithmetic operations in terms of d and n which do not depend on L, the number of bits required to present the inequalities. The number of arithmetic operations required by the simplex algorithm depends exponentially on d but does not depend on L. One can hope that a strongly polynomial algorithm for linear programming will be achieved by some clever pivot rule for the simplex algorithm. The most significant result in this direction is by Tardos [50] and takes an entirely different route. She proved that the polynomial algorithms that are in general not strongly polynomial are strongly polynomial for a large family of linear programming problems that arise in combinatorial optimization.

Based on the smoothed analysis ideas, Kelner and Spielman [23] found a randomized polynomial-time simplex algorithm for linear programming. Finding such an algorithm was a goal for a long time and the result may be a step towards a strongly polynomial simplex algorithm.

3. Linear-time decodable and encodable high rate codes

3.1. Codes and expanders.

Codes. The construction of error-correcting codes [54] is also among the most celebrated applications of mathematics. Error-correcting codes are eminent in today's technology from satellite communications to computer memories.

A binary code C is simply a set of 0-1 vectors of length n. The minimal distance $d(C)$ is the minimal Hamming distance between two elements $x, y \in C$. The same definition extends when the set $\{0, 1\}$ is replaced by a larger alphabet Σ. When the minimal distance is d the code C is capable of correcting $[d/2]$ arbitrary errors. The rate of a code C of vectors of length n is defined as $R(C) = \log |C|/n$. Codes have important theoretical aspects. The classical geometric problem of densest sphere packing is closely related to the problem of finding error-correcting codes. So is the classical question of constructing "block design," which arose in recreational mathematics and later resurfaced in the design of statistical tests.

Error-correcting codes have important applications in theoretical computer science, which have enriched both areas. Codes are crucial in the area of "Hardness of approximations and PCP," and quantum analogs of error correcting codes are expected to be crucial in the engineering and building of quantum computers.

Expanders. Expanders are special types of graphs. Roughly speaking, a graph with n vertices is an expander if every set A of vertices $|A| \leq n/2$ has at least ϵn neighbors outside A. While the initial motivation came from coding theory, expanders quickly found many applications and connections in mathematics and theoretical computer science. See [13] for a comprehensible survey. Pinsker gave a simple probabilistic proof of expander graphs with bounded degree. The first explicit construction was given by Margulis [31]. There are important connections between the expansion properties, spectral properties of graphs-Laplacian [4, 3], and random walks [16]. Number theory enables the construction of a remarkable optimal class of expanders called "Ramanujan graphs" [28].

3.2. From expanders to codes.

An important class of codes are those of minimal distance αn errors $\alpha < 1/2$. Finding the largest rate of these codes is a famous open problem. Probabilistic constructions due to Gilbert and Varshamov give the highest known rates for binary codes with minimal-distance αn. The first explicit construction for codes with positive rate that correct a positive fraction of errors (for large n) was achieved by Justesen [17] and was one of the major discoveries of coding theory in the seventies.[2] A major open problem was to find such codes which admit a linear time algorithm for decoding and for encoding.

In the early sixties Gallager [14] described a method to move from graphs to codes. Michael Sipser and Spielman [49] were able to show that codes that are based on expander graphs (thus called expander codes) have positive rate, correct a positive fraction of errors, and have a simple decoding algorithm. A remarkable subsequent result by Spielman is

Theorem 3.1 (Spielman 1995, [40]). *There is a construction of positive rate linear codes which correct a positive fraction of errors and which admit a linear time algorithm for decoding and for encoding.*

Spielman's full result is quite difficult and requires a substantial extension of Gallager's original construction which is similar to the construction of "superconcentrators" from expanders. Let me explain how Sipser and Spielman were able to construct high-rate codes via expanders.

Start with an expander bipartite graph G with two unbalanced sides A and B. Suppose that vertices in A have degree c and vertices in B have degree d and that $d > c$. (Thus, $|A|d = |B|c$; put $|A| = n$.) The code C will consist of all 0,1 vectors indexed by vertices in A such that for every vertex b of B the coordinates indexed by the neighbors of b sum up to zero. This gives us a linear code (namely, C is a linear subspace of $\{0,1\}^{|A|}$) of dimension $|A| - |B|$. The minimum distance

[2]Later, constructions based on algebraic geometry were found which give, for large alphabets, even higher rates than the Gilbert-Varshamov bound.

between two vectors of C is simply the minimal number of ones in a vector in C. Now enters the expansion property. Assume that for every set B' of vertices in B such that $|B| \leq \alpha|B|$ the number of its neighbors in A is at least $\delta|B'|$, where $\delta = \epsilon c/d$. This implies that every vector in the code C has at least ϵn ones.

3.3. Tornado codes. Dan Spielman has made other important contributions to the theory of error-correcting codes and to connections between codes and computational complexity. Some of his subsequent work on error-correcting codes took a more practical turn. Spielman, together with Michael G. Luby, Michael Mitzenmacher and M. Amin Shokrollahi constructed [29] codes that approach the capacity of erasure channels. Their constructions, which are now called *tornado codes*, have various practical implications. For example, they can be useful for compensating for packet loss in Internet traffic.

4. Fast linear-system solvers and spectral graph theory

Spielman recently focused his attention to one of the most fundamental problems in computing: the problem of solving a system of linear equations. Solving large-scale linear systems is central to scientific and engineering simulation, mathematical programming, and machine learning.

Inspired by Pravin Vaidya's work on iterative solver that uses combinatorial techniques to build preconditioners, Spielman and Teng were able to settle an open problem raised by Vaidya in 1990:

Theorem 4.1 (Spielman and Teng [46]). *There is a nearly linear time algorithm for solving diagonally dominant linear systems.*

Their solver incorporates basic numerical techniques with combinatorial techniques including random walks, separator theory, and low-stretch spanning trees. This endeavor have grown into a body of interdisciplinary work, and in the process

- Numerically motivated combinatorial concepts such as spectral sparsifiers were introduced.
- Efficient graph-theoretic algorithms for constructing sparsifiers were found and applied to matrix approximation.
- Nearly linear-time clustering and partitioning algorithms for massive graphs were developed with the guidance of spectral analysis.

A recent practical fruit is the development of an asymptotically and practically efficient nearly-linear time algorithm by Koutis, Miller and Peng [25], which incorporates Spielman and collaborators' constructions of low-stretch spanning trees [12] and of sparsifiers [45].

In the rest of this section we will describe three themes in this endeavor, but let me first make a remark about the relation of Spielman's work with numerical analysis. Numerical analysis may well deserve the title of "queen of applied mathematics" and *thinking numerically* often gives a whole new dimension to mathematical understanding. Algorithms discovered in numerical analysis and scientific computing are among the most important and most useful algorithms known to mankind. Many of the notions and results discussed in Section 1 and in this section are related to numerical analysis. Spielman and Teng [43] proposed smoothed analysis as a framework for the theoretical study of certain numerical algorithms and Spielman's recent endeavor have built new bridges between numerical analysis and discrete mathematics.

The following themes are heavily entangled in Spielman's work, but I will describe them separately. For more, see Spielman's article in these proceedings [41].

Sparsifiers. What is a sparsifier? Very roughly, given a graph G (typically dense, with a quadratic number of edges in terms of the vertices), a sparsifier H is a sparse graph with a linear or nearly linear number of edges that captures (in a sense that needs to be specified) structural properties of G. So expanders can be thought of as sparsifiers of the complete graph.

Spielman and Teng [45] gave a spectral definition of sparsifiers and this definition turned out to be very fruitful. Recent works by Spielman with several coauthors [6, 42] give answers to the following questions: How are sparsifiers constructed? What are they good for? If sparsifiers are analogs of expanders what are the analogs of Ramanujan graphs? One of the major new results is nearly linear-time algorithms to construct sparsifiers.

Graph partitioning via spectral graph theory. Splitting graphs into a structured collection of subgraphs is an important area in pure and applied graph theory. A tree-like structure is often a goal. And the Tarjan-Lipton separator theorem [27] for planar graphs is an early graph-partitioning result that often serves as a role model.

By bounding the eigenvalue of the Laplacian of bounded-degree planar graphs, Spielman and Teng [45] gave an algebraic proof of the Lipton-Tarjan's theorem and its extensions, providing an explanation of why the spectral partitioning works on practical graphs including finite-element meshes and planar-like graphs.

Solvers for linear systems of equations based on graph Laplacians. In a series of papers Spielman and his coauthors considered systems of linear equations based on Laplacian matrices of graphs. This may look rather special but it is a fascinating class of linear systems and various others systems of linear equations can be reduced to this case. See [41].

Conclusion.

The beautiful interface between theory and practice, be it in mathematical programming, error-correcting codes, the search for Ramanujan-quality sparsifiers, the analysis of algorithms, computational complexity theory, or numerical analysis, is characteristic of Dan Spielman's work.

References

[1] I. Adler and R. M. Karp and R. Shamir, A simplex variant solving an m x d linear program in $o(min(m^2, d^2))$ expected number of pivot steps, *J. Complexity*, 3 (1987) 372–387.

[2] I. Adler and N. Megiddo, A simplex algorithm whose average number of steps is bounded between two quadratic functions of the smaller dimension, *Journal of the ACM*, 4 (1985), 871-895.

[3] N. Alon. Eigenvalues and expanders, *Combinatorica*, 6 (1986) 83-96, 1986.

[4] N. Alon and V. D. Milman, λ_1, isoperimetric inequalities for graphs, and superconcentrators. *J. Combin. Theory Ser. B*, 38 (1985), 73-88.

[5] D. Arthur, B. Manthey, and H. Röglin k-means has polynomial smoothed complexity, in *Proc. of the 50th FOCS (Atlanta, USA)*, pp. 405-414, 2009.

[6] J. Baston, D. Spielman, and N. Srivastava, Twice-Ramanujan sparsifiers, STOC 2009.

[7] R. Beier and B. Vöcking, Typical properties of winners and losers in discrete optimization, in *STOC '04: the 36th Annual ACM Symposium on Theory of Computing,* pp. 343-352, 2004.)

[8] K. H. Borgwardt,*The Simplex Method, a Probabilistic Analysis*, Algorithms and Combinatorics 1, Springer-Verlag, Berlin, 1987.

[9] K. L. Clarkson, A Las Vegas algorithm for linear programming when the dimension is small, *J. ACM* **42**(2) (1995) 488–499.

[10] G. B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, N.J., 1963.

[11] J. Dunagan, D. A. Spielman, and S.-H. Teng, Smoothed analysis of condition numbers and complexity implications for linear programming, *Mathematical Programming,* Series A, 2009. To appear.

[12] M. Elkin, Y. Emek, D. Spielman, and S.-H. Teng, Lower-stretch spanning trees, *SIAM Journal on Computing,* Vol 32 (2008), 608–628.

[13] S. Hoory, N. Linial and A. Wigderson, Expander graphs and their applications, Bull. Amer. Math. Soc., 43 (2006), 439–561.

[14] R. G. Gallager, *Low Density Parity Check Codes*, MIT Press, Cambridge, MA, 1963.

[15] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of Algorithms and Combinatorics. Springer-Verlag, Berlin, second edition, 1993.

[16] M. Jerrum and A. Sinclair. Approximate counting, uniform generation and rapidly mixing Markov chains. *Inform. and Comput.*, 82 (1989), 93-133.

[17] J. Justesen, A class of constructive asymptotically good algebraic codes, *IEEE Trans. Information Theory* 18 (1972), 652–656.

[18] L. G. Khachiyan, A polynomial algorithm in linear programming, *Soviet Math. Doklady*, 20 (1979), 191-194.

[19] G. Kalai, A subexponential randomized simplex algorithm, *Proceedings of the 24-th Ann. ACM symp. on the Theory of Computing*, pp. 475-482, ACM Press, Victoria, 1992.

[20] N. Karmarkar, A new polynomial time algorithm for linear programming, *Combinatorica*, 4 (1984), 373-397.

[21] R. Kannan, S. Vempala, and A. Vetta, On clustering: Good, bad, and spectral, *J. ACM*, 51 (2004), 497–515.

[22] G. Kalai and D. J. Kleitman, A quasi-polynomial bound for diameter of graphs of polyhedra, *Bull. Amer Math. Soc.* 26 (1992), 315-316.

[23] J. Kelner and D.A. Spielman, A randomized polynomial-time simplex algorithm for linear programming, *Proceedings of the 38th annual ACM symposium on Theory of computing.* 2006

[24] V. Klee and G.J. Minty, How good is the simplex algorithm, in: *Inequalities III,* O. Shisha (ed.) Academic Press, New-York ,1972, pp. 159-175.

[25] I. Koutis, G. Miller,and R. Peng, Approaching optimality for solving SDD systems, *FOCS* 2010.

[26] M. Krivelevich, B. Sudakov, and P. Tetali, On smoothed analysis in dense graphs and formulas, *Random Struct. Algorithms* 29 (2005), 180–193.

[27] R. J. Lipton, and R. E. Tarjan, A separator theorem for planar graphs, *SIAM J. Appl. Math.* 36 (1979), 177–189.

[28] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs, *Combinatorica*, 8 (1988), 261–277.

[29] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, Efficient erasure correcting codes, *IEEE Transactions on Information Theory*, 47 (2001), 569-584.

[30] B. Ma, Why greed works for shortest common superstring problem, *Theor. Computer Science* 410 (2009), 5374–3975.

[31] G. A. Margulis, Explicit constructions of graphs without short cycles and low density codes, *Combinatorica*, 2 (1982), 71–78.

[32] N. Megiddo, Toward a genuinely polynomial algorithm for linear programming, *SIAM J. Comp.* 12 (1983), 347-353.

[33] N. Megiddo, Linear programming in linear time when the dimension is fixed, *J. Assoc. Comp. Mach.* 31 (1984), 114-127.

[34] J. Matoušek, M. Sharir and E. Welzl, A subexponential bound for linear programming, *Proc. 8-th Annual Symp. on Computational Geometry*, 1992, pp. 1-8.

[35] J. Renegar, Condition numbers, the barrier method and the conjugate gradient method, *SIAM Journal on Optimization* 6, 879 - 912 (1996).

[36] F. Santos, A counterexample to the Hirsch conjecture, preprint 2010, arXiv:1006.2814.

[37] A. Schrijver, *Theory of Linear and Integer Programming,* Wiley-Interscience 1986.

[38] S. Smale, On the average number of steps in the simplex method of linear programming, *Mathematical Programming*, 27 (1983), 241–262.

[39] S. Smale, Mathematical problems for the next century, *Mathematics: frontiers and perspectives*, pp. 271-294, American Mathematics Society, Providence, 2000.

[40] D. A. Spielman, Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Inform. Theory*, 42 (1996), 1723-1731.

[41] D. A. Spielman, Algorithms, graph theory, and linear equations, *Proceedings of the International Congress of Mathematicians,* 2010

[42] D. A. Spielman and N, Srivastava, Graph sparcification by effective resistances, *SIAM J. on Computing*, to appear.

[43] D. Spielman and S.-H. Teng, Smoothed analysis of algorithms, *Proceedings of the International Congress of Mathematicians,* vol I, pp. 597-606. Beijing 2002.

[44] D. Spielman and S.-H. Teng, Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time, *Journal of the ACM*, 51 (2004), 385 - 463.

[45] D. Spielman and S.-H. Teng, Spectral partitioning works: planar graphs and finite element meshes *Linear Algebra and its Applications* 421 (2007), 284-305.

[46] D. Spielman and S.-H. Teng, Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems, preprint 2006.

[47] D. Spielman and S.-H. Teng, Spectral sparsification of graphs, preprint, http://arxiv.org/abs/0808.4134.

[48] D. Spielman and S.-H. Teng, Smoothed analysis: an attempt to explain the behavior of algorithms in practice, *Comm. of the ACM*, 52 (2009), no. 10 pp. 76-84.

[49] M. Sipser and D. A. Spielman, Expander codes, *IEEE Trans. Inform. Theory*, 42 (1996), 1710-1722.

[50] E. Tardos, A strongly polynomial algorithm to solve combinatorial linear programs, *Oper. Res.*, 34 (1986), 250-256.

[51] T. Tao, and V. H. Vu, The condition number of a randomly perturbed matrix. In *STOC '07: the 39th Annual ACM Symposium on Theory of Computing* (2007), 248–255.

[52] M. J. Todd, Polynomial expected behavior of a pivoting algorithm for linear complementarity and linear programming problems, *Math. Prog.* 35 (1986), 173–192.

[53] M. J. Todd, The many facets of linear programming, *Math. Prog.* 91 (2002), 417–436.

[54] J. H. van Lint, *Introduction to Coding Theory*, third edition, Graduate Texts in Mathematics, 86, Springer-Verlag, Berlin, 1999.

[55] R. Vershynin, Beyond Hirsch conjecture: Walks on random polytopes and smoothed complexity of the simplex method, In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science* (2006), 133–142.

[56] G. M. Ziegler, *Lectures on Polytopes*, Graduate Texts in Mathematics **152**, Springer-Verlag, New York 1995.

Hebrew University of Jerusalem and Yale University
E-mail: kalai@math.huji.ac.il