1

Smoothed Analysis of Algorithms and Heuristics: Progress and Open Questions

Daniel A. Spielman
Applied Mathematics and Computer Science, Yale University
New Heaven, Connecticut, USA
e-mail: spielman@cs.yale.edu

Shang-Hua Teng
Computer Science, Boston University
and Akamai Technologies Inc
Boston, Massachusetts, USA
e-mail: steng@cs.bu.edu

Abstract

In this paper, we survey some recent progress in the smoothed analysis of algorithms and heuristics in mathematical programming, combinatorial optimization, computational geometry, and scientific computing. Our focus will be more on problems and results rather than on proofs. We discuss several perturbation models used in smoothed analysis for both continuous and discrete inputs. Perhaps more importantly, we present a collection of emerging open questions as food for thought in this field.

1.1 Prelinminaries

The quality of an algorithm is often measured by its time complexity (Aho, Hopcroft & Ullman (1983) and Cormen, Leiserson, Rivest & Stein (2001)). There are other performance parameters that might be important as well, such as the amount of space used in computation, the number of bits needed to achieve a given precision (Wilkinson (1961)), the number of cache misses in a system with a memory hierarchy (Aggarwal et al. (1987), Frigo et al. (1999), and Sen et al. (2002)), the error probability of a decision algorithm (Spielman & Teng (2003a)), the number of random bits needed in a randomized algorithm (Motwani & Raghavan (1995)), the number of calls to a particular "oracle" program, and the number of iterations of an iterative algorithm (Wright (1997), Ye (1997), Nesterov & Nemirovskii (1994), and Golub & Van Loan (1989)). The quality of an approximation algorithm could be its approximation ratio (Vazirani (2001)) and the quality of an online algorithm could be

its competitive ratio (Sleator & Tarjan (1985) and Borodin & El-Yaniv (1998)).

Once we fix a quality parameter Q, there might still be more than one way to measure an algorithm A. If our universe of inputs happens to have only one instance $\mathbf{x}$ then the most natural measure is the *instance-based complexity*, given by $Q(A, \mathbf{x})$. In such a case, if we have a few algorithms $A_1, \ldots, A_k$ in our repertoire, we can easily decide which one is better. If our universe of inputs has two instances $\mathbf{x}$ and $\mathbf{y}$, then the instance-based measure of an algorithm A defines a two dimensional vector $(Q(A, \mathbf{x}), Q(A, \mathbf{y}))$. For two algorithms A_1 and A_2, if $Q(A_1, \mathbf{x}) < Q(A_2, \mathbf{x})$ but $Q(A_1, \mathbf{y}) > Q(A_2, \mathbf{y})$, then strictly speaking, they are not comparable.

The universe D of inputs is much more complex, both in theory and in practice. The instance-based measure defines a high-dimensional vector when D is finite. Otherwise, it can be viewed as a function from D to $\mathbb{R}$. How should one measure the quality of an algorithm? How should one compare two algorithms?

Traditionally, one partitions an input domain D into a collection of subdomains $\{D_1, \ldots, D_n, \ldots\}$ according to the *input size*. The set D_n represents all instances in D whose input size is n. Given an algorithm A, for each D_n, one comes up with a scalar $t_{Q,A}(n)$ that "summarizes" the performance of A over D_n, as given by the restriction $Q_n(A)$ of $Q(A, \cdot)$ to D_n. Then $t_{Q,A}(n)$ is a function of n. With the help of big-O or big-Θ notations, one often characterizes the behavior of A by evaluating $t_{Q,A}(n)$ asymptomatically.

The definition of input sizes could be a source of discussion, for example,

- in optimization, scientific computing, and computational geometry, the input size could be the number of real scalars in the input;
- in number-theoretical algorithms, it could be the total number of bits in the input;
- in comparison-based sorting, it could the number of elements, while in some other sorting algorithms, it could be the total number of letters in the input;
- in the knapsack problem, it could be the total magnitude (or the size of the unary representation) of the input.

Whatever the definition of the input size is, we need to find a way to measure and to summarize the performance of an algorithm over an input subdomain D_n.

The most commonly used measure is the *worst-case measure*. It is given by

$$\mathrm{W}\left[Q_n(A)\right] = \max_{\mathbf{x} \in D_n} Q(A, \mathbf{x}).$$

When the worst-case measure of an algorithm A is small†, we have an absolute guarantee on the performance of algorithm A no matter which input it is given. Algorithms with good worst-case performance have been developed for a great number of problems including some seemingly difficult ones such as primality testing (Solovay & Strassen (1977), Miller (1975), Adleman & Huang (1987), and Agrawal, Kayal & Saxena (2004)) and convex programming (Nesterov & Nemirovskii (1994)). These algorithms have time complexity upper-bounded by a (low-degree) polynomial function in n.

However, with an even greater number of problems, ranging from network design to industrial optimizations, we have been less lucky. Scientists and engineers often use heuristic algorithms for these problems. Most of these algorithms, after years of improvements, work well in practice. But, their worst-case complexities might be still be very poor. For example, they could be exponential in their input sizes. For theorists who are also concerned about the practical performance of algorithms, it has long been observed that the worst-case instances of such an algorithm might not be "typical" and might never occur in practice. Thus, worst-case analysis can pessimistically suggest that the performance of the algorithm is poor. Trying to rigorously understand and model the practical performance of heuristic algorithms has been a major challenge in Theoretical Computer Science‡ (cf. the report of Condon et al. (1999)).

Average-case analysis was introduced to overcome this difficulty. In it, one first determines a distribution of inputs and then measures the expected performance of an algorithm assuming inputs are drawn from this distribution. If we suppose that $\mathcal{S}$ is a distribution over D_n, the

† For example, the number of comparisons needed by the merge-sort algorithm to sort any sequence of n elements is bounded above by $n \log n$.

‡ The theory-practice gap is not limited to heuristics with exponential complexities. Many polynomial time algorithms, such as the interior-point method for linear programming (Karmarkar (1984)) and the conjugate gradient method for linear systems (Hestenes & Stiefel (1952)), are often much faster than their worst-case bounds. In addition, various heuristics are used to speed up the practical performance of codes that are based on worst-case polynomial time algorithms. These heuristics might in fact worsen the worst-case performance, or make the worst-case complexity hard to analyze.

average-case measure according to $\mathcal{S}$ is

$$\mathrm{AVG}_{\mathcal{S}}\left[Q_n(A)\right] = \mathrm{E}_{\mathbf{x}\in_{\mathcal{S}} D_n}\left[Q(A,\mathbf{x})\right],$$

where we use $\mathbf{x} \in_{\mathcal{S}} D_n$ to denote that $\mathbf{x}$ is randomly chosen from D_n according to distribution $\mathcal{S}$.

Ideally, one should use a mathematically analyzable distribution that is also the same as or close to the "practical distribution." But finding such a distribution and analyzing it could be a difficult or even impossible task. As most average-case analyses are conducted on simpler distributions than what might occur in practice, the inputs encountered in applications may bear little resemblance to the random inputs that dominate the analysis. For example, a randomly chosen graph with average degree around six is rarely similar to a finite-element graph in two dimensions, even though the latter also has average degree around six. Random objects such as random graphs or random matrices might have some special properties with all but exponentially low probability, and these special properties might dominate the average-case analysis.

Smoothed analysis (Spielman & Teng (2004)) is a recently developed framework for analyzing algorithms and heuristics. It is partially motivated by the observation that input parameters in practice are often subject to a small degree of random noise: In industrial optimization and market predictions, the input parameters could be obtained by physical measurements, and measurements usually have some random uncertainties of low magnitudes. In computer aided design, the input parameters could be the output of another computer program, e.g., a geometric modeling program, that might have numerical imprecision due to rounding and approximation errors. Even in applications where inputs are discrete, there might be randomness in the formation of inputs. For example, the network structure of the Internet may very well be governed by some "blueprints" of the government and industrial giants, but it is still "perturbed" by the involvements of smaller Internet service providers. Thus it may be neither completely random nor arbitrary.

In smoothed analysis, we assume that an input to an algorithm is subject to a slight random perturbation. The *smoothed measure* of an algorithm on an input instance is its expected performance over the perturbations of that instance. We define the *smoothed complexity* of an algorithm as the maximum smoothed measure over its inputs.

In this paper, we survey the progress made in smoothed analysis in recent years. We discuss several perturbation models considered for both

continuous and discrete problems. We then present some open questions in this field.

1.2 Basic Perturbation Models and Polynomial Smoothed Complexity

To conduct smoothed analysis, we need a perturbation model that can capture the randomness and imprecision in the formation of inputs. To be concrete in the discussion below, we first consider the case when our sub-universe is $D_n = \mathbb{R}^n$, as often considered in optimization, scientific computing, and computational geometry. For these continuous inputs, for example, the family of Gaussian distributions (cf. Feller (1968, 1970)) provide a perturbation model for noise.

Recall that a univariate Gaussian distribution with mean 0 and standard deviation σ has density

$$\frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}.$$

A *Gaussian random vector* of variance σ^2 centered at the origin in $\mathbb{R}^n$ is a vector where each entry is a Gaussian random variable of standard deviation σ and mean 0. It has density

$$\frac{1}{\left(\sqrt{2\pi}\sigma\right)^d} e^{-\|\mathbf{x}\|^2/2\sigma^2}.$$

Definition 1.1 (Gaussian Perturbations) *Let $\bar{\mathbf{x}} \in \mathbb{R}^n$. A σ-Gaussian perturbation of $\bar{\mathbf{x}}$ is a random vector $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{g}$, where $\mathbf{g}$ is a Gaussian random vector of variance σ^2.*

Definition 1.2 (Smoothed Complexity with Gaussian Perturbations) *Suppose $Q_n : D_n = \mathbb{R}^n \to \mathbb{R}^+$ is a quality function. Then the smoothed complexity of Q_n under σ-Gaussian perturbations is given as*

$$\max_{\bar{\mathbf{x}} \in \mathbb{R}^n} \mathrm{E}_{\mathbf{g}} \left[Q_n \left(\bar{\mathbf{x}} + \|\bar{\mathbf{x}}\|_2 \, \mathbf{g} \right) \right],$$

where $\mathbf{g}$ is a Gaussian random vector of variance σ^2.

Each instance $\bar{\mathbf{x}}$ of a computational problem has a neighborhood which, intuitively, contains the set of instances that are close to and similar to $\bar{\mathbf{x}}$. A perturbation model defines a distribution over the neighborhood of $\bar{\mathbf{x}}$. The closer $\mathbf{x}$ is to $\bar{\mathbf{x}}$, the higher $\mathbf{x}$ and $\bar{\mathbf{x}}$ might be correlated due to the randomness in the formation of input instances. In Gaussian

perturbations, the closeness and similarity among inputs are measured by their Euclidean distance. As the density function decreases exponentially in distance, the variance parameter σ defines the magnitude of perturbations and also captures the radius of the most likely neighborhood of an instance. The smoothed complexity is measured in terms of the input length n as well as σ, the magnitude of the perturbations. As σ increases continuously starting from 0, the smoothed complexity interpolates between the worst-case and average-case complexities (Spielman & Teng (2004)).

Of course, not all computational problems deal with continuous inputs. A commonly used communication model with a noisy channel assumes inputs are subject to Boolean perturbations of probability σ:

Definition 1.3 (Boolean Perturbations) *Let* $\bar{\mathbf{x}} = (\bar{x}_1, \ldots, \bar{x}_n) \in \{0,1\}^n$ *or* $\{-1,1\}^n$ *A* σ*-Boolean perturbation of* $\bar{\mathbf{x}}$ *is a random string* $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ *or* $\{-1,1\}^n$*, where* $x_i = \bar{x}_i$ *with probability* $1 - \sigma$.

In Boolean perturbations, the closeness and similarity of instances are measured by their Hamming distances. Again, the parameter σ defines the magnitude of perturbations as well as the radius of the most likely neighborhood of an instance.

In scheduling, packing, and sorting, the inputs are often integers of certain magnitudes. Banderier, Beier, and Mehlhorn (2003) propose to use the partial bit randomization model:

Definition 1.4 (Partial Bit Randomization) *Let* $\bar{z}$ *be an integer and* k *be a positive integer indicating the magnitude of the perturbation. A* k*-partial bit randomization of* $\bar{z}$ *is an integer* z *obtained from* $\bar{z}$ *by replacing its* k *least significant bits by a random number in* $[0 : 2^{k-1}]$ *according to some distribution over* $[0 : 2^{k-1}]$.

In comparison-based sorting and online problems, each input consists of a sequence of elements. Banderier, Beier, and Mehlhorn (2003) introduce the following *partial permutation model*:

Definition 1.5 (Partial Permutation Perturbations) *Let* $\bar{\mathbf{s}}$ *be a sequence of* n *elements. Let* $0 \leq \sigma \leq 1$ *be the magnitude of perturbations. A* σ*-partial permutation of* $\bar{\mathbf{s}}$ *is a random sequence* $\mathbf{s}$ *obtained from* $\bar{\mathbf{s}}$ *by first building a subset* S *by independently selecting each index number from* $\{1, 2, ..., n\}$ *with probability* σ*, and then randomly permut-*

ing elements of $\bar{\mathbf{s}}$ *in position* S *while retaining the positions of all other elements.*

The perturbation model that most naturally captures the imprecision in the formation of inputs can vary from application to application. For instance, it might be more suitable to use uniform random perturbations within a properly-centered ball to analyze some computational geometry algorithms.

Definition 1.6 (Uniform Ball Perturbations) *Let* $\bar{\mathbf{x}} \in \mathbb{R}^n$. *A uniform ball perturbation of radius* σ *of* $\bar{\mathbf{x}}$ *is a random vector* $\mathbf{x}$ *chosen uniformly from the ball of radius* σ *centered at* $\bar{\mathbf{x}}$.

For any of the basic perturbation models we have discussed, there might be some refinements and variants worthy of considerations.

For example, Eppstein† proposed the following refinement of the partial permutation model: Let $\bar{\mathbf{s}}$ be a sequence of n elements that have a total ordering. Let $\|\mathbf{s}\|$ denote the number of elements of the input that must be moved to make the input sorted or reverse-sorted. To obtain a perturbed element, one randomly chooses a set S of $(\sigma \cdot \|\mathbf{s}\|)$ elements, and randomly permutes them. In this model, one does not perturb the already-sorted input or the reverse-sorted input at all, and the perturbations of other inputs depend on their distance to these inputs. This definition is inspired by the definition of smoothed analysis for problems that take inputs from $\mathbb{R}^n$: we do not perturb the zero vector, and perturb other vectors in proportion to their norm. For sorting, one may view the already-sorted input as a zero, and distance-to-sorted as a norm.

In analyzing scientific computing algorithms that take advantage of the sparsity in the problem instances, one may find relative Gaussian perturbations or zero-preserving Gaussian perturbations better models of imprecision:

Definition 1.7 (Relative Gaussian Perturbations) *Let* $\bar{\mathbf{x}}$ *be a vector* $(\bar{x}_1, \ldots, \bar{x}_n) \in \mathbb{R}^n$. *A relative* σ*-Gaussian perturbation of* $\bar{\mathbf{x}}$ *is a random vector* $\mathbf{x} = (x_1, \ldots, x_n)$ *where* $x_i = \bar{x}_i(1 + g_i)$, *where* g_i *is a Gaussian random variable with standard deviation* σ.

Definition 1.8 (Zero-Preserving Gaussian Perturbations) *For any* $\bar{\mathbf{x}} = (\bar{x}_1, \ldots, \bar{x}_n) \in \mathbb{R}^n$, *a zero-preserving* σ*-Gaussian perturbation of* $\bar{\mathbf{x}}$ *is a vector* $\mathbf{x} = (x_1, \ldots, x_n)$ *where* $x_i = \bar{x}_i + (1 - \mathbf{IsZero}\,(\bar{x}_i))\, g_i$,

† Personal Communication

where g_i is a Gaussian random variable with standard deviation σ and $\mathbf{IsZero}(x) = 1$ *if* $x = 0$, *and* $\mathbf{IsZero}(x) = 0$, *otherwise.*

When time complexity is the main concern, the central questions in smoothed analysis naturally are:

Does an algorithm have polynomial smoothed complexity? Is a decision or search/optimization problem in smoothed polynomial time?

In addition to the notion of input size, one needs a model of perturbations and a notion of magnitudes of perturbations to define polynomial smoothed complexity. Given a model and notion of magnitudes of perturbations, there might still be several possible definitions of polynomial smoothed complexity.

Spielman and Teng (2004) define polynomial smoothed complexity as:

Definition 1.9 (Polynomial Smoothed Complexity) *Given a problem P with input domain $D = \cup_n D_n$ where D_n represents all instances whose input size is n. Let $\mathcal{R} = \cup_{n,\sigma} R_{n,\sigma}$ be a family of perturbations where $R_{n,\sigma}$ defines for each $\bar{\mathbf{x}} \in D_n$ a perturbation distribution of $\bar{\mathbf{x}}$ with magnitude σ. Let A be an algorithm for solving P and $T_A(\mathbf{x})$ be the time complexity for solving an instance $\mathbf{x} \in D$. Then algorithm A has polynomial smoothed complexity if there exist constants n_0, σ_0, c, k_1 and k_2 such that for all $n \geq n_0$ and $0 \leq \sigma \leq \sigma_0$,*

$$\max_{\bar{\mathbf{x}} \in D_n} \left(\mathrm{E}_{\mathbf{x} \leftarrow R_{n,\sigma}(\bar{\mathbf{x}})} \left[T_A(\mathbf{x}) \right] \right) \leq c \cdot \sigma^{-k_2} \cdot n^{k_1}, \tag{1.1}$$

where $\mathbf{x} \leftarrow R_{n,\sigma}(\bar{\mathbf{x}})$ means $\mathbf{x}$ is chosen according to distribution $R_{n,\sigma}(\bar{\mathbf{x}})$.

The problem P is in smoothed polynomial time with perturbation model $\mathcal{R}$ if it has an algorithm with polynomial smoothed complexity.

For example, Spielman and Teng show that the simplex method with the shadow-vertex pivoting rule (Gass & Saaty (1955)) has polynomial smoothed complexity under Gaussian perturbations. We can relax or strengthen the dependency on σ in the definition of the polynomial smoothed complexity.

Definition 1.10 (Polynomial Smoothed Complexity: II) *Let P, A, D, D_n, $\mathcal{R}$, $R_{n,\sigma}$ be the same as in Definition 1.9. Then algorithm A has polynomial smoothed complexity if there exist constants n_0, σ_0, c, k, and a function $g : \mathbb{R}^+ \to \mathbb{R}^+$ such that for all $n \geq n_0$ and $0 \leq \sigma \leq \sigma_0$,*

$$\max_{\bar{\mathbf{x}} \in D_n} \left(\mathrm{E}_{\mathbf{x} \leftarrow R_{n,\sigma}(\bar{\mathbf{x}})} \left[T_A(\mathbf{x}) \right] \right) \leq c \cdot g(\sigma) \cdot n^k.$$

In particularly, when $g(\sigma)$ is a poly-logarithmic function of $1/\sigma$, we say the algorithm has polynomial smoothed complexity with poly-logarithmic dependency on $1/\sigma$.

By Markov's inequality (cf. Alon & Spencer (1992) and Feller (1968)), if an algorithm A has smoothed complexity $T(n,\sigma)$, then

$$\min_{\bar{\mathbf{x}}\in D_n} \Pr_{\mathbf{x}\leftarrow R_{n,\sigma}(\bar{\mathbf{x}})} \left[T_A(\mathbf{x}) \leq \delta^{-1} T(n,\sigma)\right] \geq 1-\delta. \tag{1.2}$$

In other words, if A has polynomial smoothed complexity, then for any $\bar{\mathbf{x}}$, with high probability, say with $(1-\delta)$, A can solve a random perturbation of $\bar{\mathbf{x}}$ in time polynomial in n, $1/\sigma$, and $1/\delta$.

However, the probabilistic upper bound given in (1.2) does not usually imply that the smoothed complexity of A is $O(T(n,\sigma))$. In fact Eqn (1.2) may not even imply that

$$\max_{\bar{\mathbf{x}}\in D_n} \left(\mathrm{E}_{\mathbf{x}\leftarrow R_{n,\sigma}(\bar{\mathbf{x}})} \left[T_A(\mathbf{x})\right]\right) \text{ is finite.}$$

Eqn (1.2) suggests a relaxed extension of polynomial smoothed complexity.

Definition 1.11 (Probably Polynomial Smoothed Complexity) *Let P, A, D, D_n, $\mathcal{R}$, $R_{n,\sigma}$ be the same as in Definition 1.9. Then algorithm A has probably polynomial smoothed complexity if there exist constants n_0, σ_0, c, k_1, k_2, k_3, such that for all $n \geq n_0$ and $0 \leq \sigma \leq \sigma_0$,*

$$\max_{\bar{\mathbf{x}}\in D_n} \left(\Pr_{\mathbf{x}\leftarrow R_{n,\sigma}(\bar{\mathbf{x}})} \left[T_A(\mathbf{x}) > c\cdot\sigma^{-k_1}\cdot\delta^{-k_2}\cdot n^{k_3}\right]\right) \leq \delta. \tag{1.3}$$

Equivalently, there exist constants n_0, σ_0, c, and α, such that for all $n \geq n_0$ and $0 \leq \sigma \leq \sigma_0$,

$$\max_{\bar{\mathbf{x}}\in D_n} \left(\mathrm{E}_{\mathbf{x}\leftarrow R_{n,\sigma}(\bar{\mathbf{x}})} \left[(T_A(\mathbf{x}))^{\alpha}\right]\right) \leq c\cdot\sigma^{-1}\cdot n \tag{1.4}$$

The relaxation of polynomial smoothed complexity given in Eqn (1.3) is introduced by Blum and Dunagan (2002) in their analysis of the perceptron algorithm. They show that the perceptron algorithm has probably polynomial smoothed complexity, in spite of the fact that its smoothed complexity according to Definition 1.9 is unbounded. Beier and Vöcking (2004), in their study of the binary optimization problem, introduce the alternative form given in Eqn (1.4).

1.3 Progress in Smoothed Analysis

We cluster the materials in this section into four subsections.

- Linear Programming.
- Combinatorial Optimization.
- Scientific Computing.
- Discrete and Geometric Structures.

Although these topics appear to be diverse, the approaches developed for conducting smoothed analysis in these areas are quite similar. In fact, most approaches consist of two basic steps:

- **Geometric/Combinatorial Conditions of Nice Instances:** Establish a set of analyzable geometric or combinatorial conditions under which the algorithm performs well on an instance.
- **Probabilistic Analysis:** Prove that for every input, these geometric/combinatorial conditions hold with high probability over its perturbations.

The challenge in the first step is to establish *manageable* conditions. The instance-based complexity itself provides the most accurate characterization of nice and bad input instances of an algorithm, but this characterization is hardly useful in analysis. What we often look for are conditions that are accurate enough for predicting the performance and simple enough for probabilistic analysis. For example, the number of iterations of the Conjugate Gradient Method (CG) (Hestenes & Stiefel (1952)) for solving a symmetric positive definite linear system $\mathbf{Ax} = \mathbf{b}$ can be bounded above by $O\left(\sqrt{\kappa(\mathbf{A})}\right)$ (Golub & Van Loan (1989)), where $\kappa(\mathbf{A})$ is the condition number of $\mathbf{A}$ – the ratio of the largest eigenvalue of $\mathbf{A}$ to the smallest eigenvalue of $\mathbf{A}$. Thus, if $\mathbf{A}$ is from a distribution where $\kappa(\mathbf{A})$ is small with high probability, then we can use $\kappa(\mathbf{A})$ as our condition of nice inputs, even though there might exist $\mathbf{A}$ with very large $\kappa(\mathbf{A})$ and $\mathbf{b}$ for which the CG converges rapidly. But if $\mathbf{A}$ is from a distribution where the condition numbers are mostly very large, and the CG has been observed and believed to perform well, then we need to find some other conditions for its good performance.

To establish a lower bound on the worst-case complexity of an algorithm we rely a lot on our intuition of the properties for bad instances. In contrast, to prove a smoothed upper bound, we need to work with our imagination to find properties of nice instances. However, these two studies are not completely unrelated, and if all of our worst-case

instances are unstable in a perturbation model, then there might be reasons to believe that the smoothed measure is good.

1.3.1 Linear Programming

In a linear program, one is asked to optimize a linear objective function subject to a set of linear constraints. Mathematically, according to one standard form of linear programming, one is solving

$$\max \quad \mathbf{c}^T\mathbf{x} \qquad \text{subject to} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b},$$

where $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{b}$ is an m-dimensional vector, and $\mathbf{c}$ is an n-dimensional vector.

If the constraints are feasible, then they define a convex polyhedron $\{\mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$. This polyhedron could be unbounded in the direction of $\mathbf{c}$ in which case the optimal value of the linear program is infinite. Otherwise, the optimal value is finite and the solution point $\mathbf{x}$ that achieves this optimal value must be a vertex of the polyhedron $\{\mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$. Note that a vertex is determined by a subset of equations from $\mathbf{A}\mathbf{x} = \mathbf{b}$.

Linear programming is perhaps the most fundamental optimization problem (Dantzig (1991)). Several methods for solving linear programs have been developed since its introduction (Dantzig (1951), Khachiyan (1979), and Karmarkar (1984)). The most commonly used approaches for solving a linear program are the simplex method (Dantzig (1951)) and the interior-point method (Karmarkar (1984)).

We start our discussion with results in the smoothed analysis of the simplex method. We then continue with three other methods for solving linear programs: the perceptron method, its variant with scaling, and the interior-point method.

Smoothed Analysis of the Simplex Method

The simplex method provides a family of linear programming algorithms. Most of them are two-phase algorithms: In Phase I, they determine whether a given linear program is infeasible and, if the program is feasible, they also compute an initial vertex $\mathbf{v}_0$ of the feasible region and enter Phase II, where they iterate: in the i^{th} iteration, they find a neighboring vertex $\mathbf{v}_i$ of $\mathbf{v}_{i-1}$ with better objective value, or terminate with an extreme ray from $\mathbf{v}_{i-1}$ on which the objective function is unbounded above, or terminate with an optimal solution $\mathbf{v}_{i-1}$. Some two-phase

simplex methods can determine whether a feasible linear program is unbounded in the objective direction in Phase I.

Spielman and Teng (2004) consider the smoothed complexity of the simplex method under Gaussian perturbations: For any $\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\mathbf{c}}$, the perturbations of the linear program defined by $(\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\mathbf{c}})$ is

$$\max \quad \mathbf{c}^T\mathbf{x} \qquad \text{subject to} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b},$$

where $\mathbf{A}$, $\mathbf{b}$, and $\mathbf{c}$, respectively, are obtained from $\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\mathbf{c}}$ by a Gaussian perturbations of variance

$$\left(\left\|\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\mathbf{c}}\right\|_F \sigma\right)^2,$$

where $\|(\mathbf{A}, \mathbf{b}, \mathbf{c})\|_F$ is the square root of the sum of squares of the entries in $\mathbf{A}, \mathbf{b}$, and $\mathbf{c}$.

In this smoothed setting, with probability 1, every vertex of the feasible region is determined by exactly n equations. Two vertices $\mathbf{v}$ and $\mathbf{u}$ of the feasible region are neighbors if their associated sets of equations differ by only one equation. So with probability 1, apart from the *extreme* vertices from which there is a feasible ray, each vertex of a perturbed linear program has n neighbors.

Spielman and Teng prove the following theorem.

Theorem 1.1 (Smoothed Complexity of the Simplex Method) *There exists a two-phase simplex algorithm with polynomial smoothed complexity under Gaussian perturbations.*

Let $\mathbf{A}$ be a σ-Gaussian perturbation of an $m \times n$ matrix $\bar{\mathbf{A}}$ with $\left\|\bar{\mathbf{A}}\right\|_F \leq 1$ and $\mathbf{1}$ be the m-vector all of whose entries are equal to 1. Then the polyhedron $\{\mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{1}\}$ is always feasible with $\mathbf{0}$ as a feasible point. For any two n-vectors $\mathbf{c}$ and $\mathbf{t}$ the projection of the polyhedron $\{\mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{1}\}$ on the two-dimensional plane spanned by $\mathbf{c}$ and $\mathbf{t}$ is called the *shadow* of the polyhedron onto the plane spanned by $\mathbf{c}$ and $\mathbf{t}$. We denote this shadow by $\mathbf{Shadow}_{\mathbf{t},\mathbf{c}}(\mathbf{A})$, and its size, the number of its vertices, by $|\mathbf{Shadow}_{\mathbf{t},\mathbf{c}}(\mathbf{A})|$. Theorem 1.1 is built upon the smoothed analysis of $|\mathbf{Shadow}_{\mathbf{t},\mathbf{c}}(\mathbf{A})|$.

Theorem 1.2 (Smoothed Shadow Size) *For any $m \times n$ matrix $\bar{\mathbf{A}}$ with $\left\|\bar{\mathbf{A}}\right\|_F \leq 1$, let $\mathbf{A}$ be a σ-Gaussian perturbation of $\bar{\mathbf{A}}$. For any two n-dimensional vectors $\mathbf{c}$ and $\mathbf{t}$*

$$\mathrm{E}_{\mathbf{A}}\left[|\mathbf{Shadow}_{\mathbf{t},\mathbf{c}}(\mathbf{A})|\right] = O\left(\frac{mn^3}{\min(\sigma, 1/\sqrt{n\log m})^6}\right).$$

This probabilistic geometric theorem provides a smoothed upper bound on the Phase II complexity of the simplex method algorithm with the shadow-vertex pivot rule (Gass & Saaty (1955)). Theorem 1.1 was established by a reduction of Phase I computation to Phase II computation in $n+1$ dimensions.

Recently, Deshpande and Spielman (2005) improve Theorem 1.2 with a greatly simplified proof.

Theorem 1.3 (Deshpande-Spielman) *For any $m \times n$ matrix $\bar{\mathbf{A}}$ with $\left\|\bar{\mathbf{A}}\right\|_F \leq 1$, let $\mathbf{A}$ be a σ-Gaussian perturbation of $\bar{\mathbf{A}}$. For any two n-dimensional vectors $\mathbf{c}$ and $\mathbf{t}$*

$$\mathrm{E}_{\mathbf{A}}\left[|\mathbf{Shadow}_{\mathbf{t},\mathbf{c}}(\mathbf{A})|\right] = O\left(\frac{m^3 n^{1.5}}{\min(\sigma, 1/\sqrt{n\log m})^3}\right).$$

Perceptron Algorithms

The perceptron algorithm (Agmon (1954) and Rosenblatt (1962)) is an iterative algorithm for finding a feasible solution to linear programs in the form

$$\mathbf{A}\mathbf{x} \geq \mathbf{0}, \quad \mathbf{x} \neq \mathbf{0}. \tag{1.5}$$

It is commonly used in Machine Learning (Minsky & Papert (1988)) for finding a linear separator of two sets of points $R = \{\mathbf{p}_1, \ldots, \mathbf{p}_{m_1}\}$ and $B = \{\mathbf{q}_{m_1+1}, \ldots, \mathbf{q}_{m_2}\}$ in $\mathbb{R}^n$. If R and B are separable by a hyperplane through $\mathbf{0}$ with normal vector $\mathbf{x}$, then

$$\begin{aligned} &\mathbf{p}_i^T\mathbf{x} \geq 0 \text{ for each } i \in \{1 : m_1\}, \quad \text{and} \\ &\mathbf{q}_i^T\mathbf{x} \leq 0 \text{ for each } i \in \{m_1+1 : m_1+m_2\}. \end{aligned}$$

Letting $\mathbf{a}_i = \mathbf{p}_i$ for $i \in \{1 : m_1\}$ and $\mathbf{a}_i = -\mathbf{q}_i$ for $\{m_1+1 : m_1+m_2\}$, the problem becomes the linear program given in (1.5).

The perceptron algorithm starts with an initial unit vector $\mathbf{x}_0$. During the k^{th} iteration, if there exists a row $\mathbf{a}_i^T$ of $\mathbf{A}$ with $\mathbf{a}_i^T\mathbf{x}_{k-1} < 0$ then it sets $\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{a}_i/\left\|\mathbf{a}_i\right\|_2$. The complexity question is: how many iterations does the perceptron algorithm take when given a feasible linear program of form (1.5)? The following theorem of Block (1962) and Novikoff (1962) gives an upper bound in term of a geometric quantity.

Theorem 1.4 (Block-Novikoff) *For a linear program of form (1.5), let*

$$\rho(\mathbf{A}) = \max_{\mathbf{x}:\|\mathbf{x}\|_2=1, \mathbf{A}\mathbf{x}\geq 0} \min_i \left(\frac{\mathbf{a}_i^T}{\|\mathbf{a}_i\|_2} \mathbf{x} \right).$$

The perceptron algorithm terminates in $O(1/\rho^2(\mathbf{A}))$ iterations.

The parameter $\rho(\mathbf{A})$ is known as the *wiggle room* of the perceptron problem. By establishing a probabilistic lower bound on the wiggle room, Blum and Dunagan (2002) obtain the following result.

Theorem 1.5 (Blum-Dunagan) *For any $\bar{\mathbf{A}} \in \mathbb{R}^{m \times n}$ with $\left\|\bar{\mathbf{A}}\right\|_F \leq 1$, let $\mathbf{A}$ be a σ-Gaussian perturbation of $\bar{\mathbf{A}}$ for $\sigma \leq \sqrt{1/2n}$. Then, for any δ, with probability at least $1 - \delta$, in*

$$O\left(\frac{n^3 m^2 \log^2(m/\delta)}{\sigma^2 \delta^2} \right)$$

iterations, the perceptron algorithm finds a feasible solution or correctly concludes that the linear program defined by $\mathbf{A}$ is infeasible.

Blum and Dunagan's result does not imply that the smoothed complexity of the perceptron algorithm is polynomial in m, n, and $1/\sigma$. It only states that with high probability, the perceptron algorithm with a polynomial number of iterations would correctly solve a perturbed linear program. See Definition 1.11. But this discrepancy in the definitions of polynomial smoothed complexities in the results of the perceptron method (Theorem 1.5) and the simplex method (Theorem 1.1) might provide some insights on why the simplex method usually performs better in practice than the perceptron algorithm.

Recently, Dunagan and Vempala (2004) improve the performance of a randomized version of the perceptron algorithm by applying periodic rescalings. They show that, with high probability, their algorithm finds a feasible solution to feasible linear programs of form (1.5) in time $O(mn^4 \log n \log(1/\rho(\mathbf{A})))$. It is not hard to combine the analysis of Blum-Dunagan with the the result of Dunagan-Vempala to prove the following result.

Theorem 1.6 (Smoothed complexity perceptron algorithms with rescaling) *For any $\bar{\mathbf{A}} \in \mathbb{R}^{m \times n}$ with $\left\|\bar{\mathbf{A}}\right\|_F \leq 1$, let $\mathbf{A}$ be a σ-Gaussian perturbation of $\bar{\mathbf{A}}$, for $\sigma \leq \sqrt{1/2n}$. Then, for any δ, with probability at*

least $1-\delta$*, in random*

$$O\left(mn^4 \log\left(\frac{nm}{\sigma\delta}\right)\right)$$

time, the Dunagan-Vempala perceptron algorithm finds a feasible solution or correctly concludes that the linear program defined by $\mathbf{A}$ *is infeasible.*

Condition Number of Linear Programs and the Smoothed Complexity of Interior-Point Methods

The parameter $\rho(\mathbf{A})$ aforementioned is a special case of the condition number of a linear program introduced by Renegar (1994), (1995a). Consider the following four common canonical forms of linear programs and their dual forms:

$$\begin{aligned} &\max\ \mathbf{c}^T\mathbf{x}\ \text{ s.t. }\ \mathbf{A}\mathbf{x} \le \mathbf{b} \\ &\min\ \mathbf{b}^T\mathbf{y}\ \text{ s.t. }\ \mathbf{A}^T\mathbf{y} = \mathbf{c},\ \ \mathbf{y} \ge \mathbf{0}, \end{aligned} \tag{1}$$

$$\begin{aligned} &\max\ \mathbf{c}^T\mathbf{x}\ \text{ s.t. }\ \mathbf{A}\mathbf{x} \le \mathbf{b},\ \mathbf{x} \ge \mathbf{0} \\ &\min\ \mathbf{b}^T\mathbf{y}\ \text{ s.t. }\ \mathbf{A}^T\mathbf{y} \ge \mathbf{c},\ \mathbf{y} \ge \mathbf{0}, \end{aligned} \tag{2}$$

$$\begin{aligned} &\max\ \mathbf{c}^T\mathbf{x}\ \text{ s.t. }\ \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{x} \ge \mathbf{0} \\ &\min\ \mathbf{b}^T\mathbf{y}\ \text{ s.t. }\ \mathbf{A}^T\mathbf{y} \ge \mathbf{c}, \end{aligned} \tag{3}$$

$$\begin{aligned} &\text{find } \mathbf{x} \ne \mathbf{0}\ \text{ s.t. }\ \mathbf{A}\mathbf{x} \le \mathbf{0} \\ &\text{find } \mathbf{y} \ne \mathbf{0}\ \text{ s.t. }\ \mathbf{A}^T\mathbf{y} = \mathbf{0},\ \mathbf{y} \ge \mathbf{0}. \end{aligned} \tag{4}$$

A key concept in defining the condition number is the set of ill-posed linear programs. A linear program is *ill-posed* if the program can be made both feasible and infeasible by arbitrarily small changes to its data.

In his pioneering work (Renegar (1994), (1995a), (1995b)), Renegar defines the condition number of a linear program as the scale-invariant reciprocal of the distance of that program to "ill-posedness". Any linear program may be expressed in each of the first three canonical forms. However, transformations among formulations do not in general preserve their condition numbers (Renegar (1995a)). Therefore, the condition number is defined for each normal form.

Let F be the property that a linear program is feasible. For each $(\mathbf{A}, \mathbf{b}, \mathbf{c})$ and $i \in \{1, 2, 3\}$, let $\mathbf{PLP}_i(\mathbf{A}, \mathbf{b}, \mathbf{c})$ and $\mathbf{DLP}_i(\mathbf{A}, \mathbf{b}, \mathbf{c})$ be the primal and dual linear programs, respectively, in normal form (i) defined by data $(\mathbf{A}, \mathbf{b}, \mathbf{c})$. Let

$$\begin{aligned}\rho_i(\mathbf{A}, \mathbf{b}, \mathbf{c}) = \sup \{\delta : \|\Delta\mathbf{A}, \Delta\mathbf{b}, \Delta\mathbf{c}\|_F \leq \delta \text{ implies} \\ F(\mathbf{PLP}_i(\mathbf{A}, \mathbf{b}, \mathbf{c})) = F(\mathbf{PLP}_i(\mathbf{A} + \Delta\mathbf{A}, \mathbf{b} + \Delta\mathbf{b}, \mathbf{c} + \Delta\mathbf{c})) \ \& \\ F(\mathbf{DLP}_i(\mathbf{A}, \mathbf{b}, \mathbf{c})) = F(\mathbf{DLP}_i(\mathbf{A} + \Delta\mathbf{A}, \mathbf{b} + \Delta\mathbf{b}, \mathbf{c} + \Delta\mathbf{c}))\}.\end{aligned}$$

The condition number of the linear program defined by data $(\mathbf{A}, \mathbf{b}, \mathbf{c})$ in normal form (i) is

$$C_i(\mathbf{A}, \mathbf{b}, \mathbf{c}) = \frac{\|\mathbf{A}, \mathbf{b}, \mathbf{c}\|_F}{\rho_i(\mathbf{A}, \mathbf{b}, \mathbf{c})}.$$

We can similarly define the distance $\rho_4(\mathbf{A})$ to ill-posedness and the condition number $C_4(\mathbf{A})$ of a linear program in normal form (4).

Dunagan, Spielman and Teng (2002) prove the following theorem:

Theorem 1.7 (Smoothed condition number) *For any $(\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\mathbf{c}})$ with $\left\|\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\mathbf{c}}\right\|_F \leq 1$ and $\sigma \leq 1$, let $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{c}$ be σ-Gaussian perturbations of $\bar{\mathbf{A}}$, $\bar{\mathbf{b}}$ and $\bar{\mathbf{c}}$, respectively. Then,*

$$\mathrm{E}_{\mathbf{A},\mathbf{b},\mathbf{c}}[\log C_i(\mathbf{A}, \mathbf{b}, \mathbf{c})] = O(\log(mn/\sigma)).$$

For any linear program of form (i), $i \in \{1, 2, 3\}$, specified by $(\mathbf{A}, \mathbf{b}, \mathbf{c})$ and parameter $\epsilon \leq 1$, there is an interior-point algorithm that determines whether the program is infeasible or unbounded or finds a feasible solution $\mathbf{x}$ with duality gap $\epsilon \|\mathbf{A}, \mathbf{b}, \mathbf{c}\|_F$ in $O(N^3 \log(N \cdot C_i(\mathbf{A}, \mathbf{b}, \mathbf{c})/\epsilon))$ operations (Renegar (1994), (1995a), (1995b), Vera (1996), Freund & Vera (1999)). where $N = \max(m, n)$. Let $T_i((\mathbf{A}, \mathbf{b}, \mathbf{c}), \epsilon)$ be the time complexity of these interior-point algorithms. For a linear program of form (4) given by $\mathbf{A}$, there is an algorithm that finds a feasible solution $\mathbf{x}$ or determines that the program is infeasible in $O(N^3 \log(N \cdot C_4(\mathbf{A})))$ operations (Cucker & Peña (2001) and Freund & Vera (1999)). Let $T_i((\mathbf{A}, \mathbf{b}, \mathbf{c}), \epsilon)$ be the time complexity of these interior-point algorithms. We have,

Theorem 1.8 (Smoothed Complexity of IPM: Approximation) *For any $\sigma \leq 1$, for any $\bar{\mathbf{A}} \in \mathbb{R}^{m \times n}$, $\bar{\mathbf{b}} \in \mathbb{R}^m$ and $\bar{\mathbf{c}} \in \mathbb{R}^n$ such that $\left\|\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\mathbf{c}}\right\|_F \leq 1$, let $(\mathbf{A}, \mathbf{b}, \mathbf{c})$ be a σ-Gaussian perturbation of $(\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\mathbf{c}})$.*

Then,

$$\mathrm{E}_{(\mathbf{A},\mathbf{b},\mathbf{c})}\left[T_i((\mathbf{A},\mathbf{b},\mathbf{c}),\epsilon)\right] = O\left(\max(m,n)^3 \log\left(\frac{mn}{\sigma\epsilon}\right)\right).$$

When an exact solution of a linear program is desired, one can find an optimal solution in two steps: First, apply the interior-point method to find a feasible point that is close enough to optimal solution. Then run a termination algorithm that "jumps" from the close-to-optimal solution to the optimal solution. For a feasible program defined by $(\mathbf{A},\mathbf{b},\mathbf{c})$, there is a precision quantity $\delta(\mathbf{A},\mathbf{b},\mathbf{c})$ such that for all $\epsilon \leq \delta(\mathbf{A},\mathbf{b},\mathbf{c})$, one could jump from any ϵ-accurate solution to an exact solution. Spielman and Teng (2003b) show that under σ-Gaussian perturbations, the smoothed value of $\max(1, \log(1/\delta(\mathbf{A},\mathbf{b},\mathbf{c})))$ is $O(\log nm/\sigma)$. Putting everything together, we obtain the following theorem.

Theorem 1.9 (Smoothed Complexity of IPM: Exact Solution) *For any $\bar{\mathbf{A}} \in \mathbb{R}^{m\times n}$, $\bar{\mathbf{b}} \in \mathbb{R}^m$ and $\bar{\mathbf{c}} \in \mathbb{R}^n$ such that $\left\|\bar{\mathbf{A}},\bar{\mathbf{b}},\bar{\mathbf{c}}\right\|_F \leq 1$ and $\sigma \leq 1$, let $(\mathbf{A},\mathbf{b},\mathbf{c})$ be a σ-Gaussian perturbation of $(\bar{\mathbf{A}},\bar{\mathbf{b}},\bar{\mathbf{c}})$. One can apply the interior-point algorithm with periodic applications of a termination procedure to exactly solve a linear program in normal form 1, 2, or 3 in an expected*

$$O\left(\max(m,n)^3 \log\left(\frac{mn}{\sigma}\right)\right)$$

arithmetic operations.

Smoothed Complexity of Linear Programming in Low Dimensions

A linear program is often referred to as a low-dimensional linear program if $m \gg n$. Clarkson (1995) introduces a remarkable reduction algorithm and proves the following lemma.

Lemma 1.1 (Clarkson's Reduction) *For any linear program with m constraints in n variables, with random sampling, one can reduce the problem of solving this program to the problems of solving $O(n^2 \log m)$ programs with $9n^2$ constraints in n variables.*

One can solve a low-dimensional linear program in two steps:

1. Apply Clarkson's reduction to the input program.
2. Use an interior-point algorithm to solve these smaller linear programs.

We can use Theorem 1.9 to prove the following theorem.

Theorem 1.10 (Smoothed complexity of low-dimensional linear programming) *There is a linear programming algorithm with smoothed complexity*

$$O\left(n^2m + n^8 \log m \log(mn/\sigma)\right).$$

So far, this is the best smoothed bound for low-dimensional linear programming. In contrast, the best worst-case upper bound for low-dimensional linear programming is obtained by the combination of Clarkson's reduction with the randomized simplex algorithm of Kalai (1992) and Matoušek, Sharir, and Welzl (1992). The complexity of this combination is

$$n^2m + n^{O\left(\sqrt{n/\log n} + \log\log m\right)}.$$

1.3.2 Combinatorial Optimization

In combinatorial optimization, the solutions are discrete. However, not all input parameters of combinatorial optimization problems are necessarily discrete. For example, in optimization problems defined on weighted graphs such as the Traveling Salesman Problem, the Minimum Steiner Tree Problem, and the Multi-Commodity Flow Problem, the weights could be continuous while the graph structure remains discrete (Papadimitriou & Steiglitz (1982)). In integer linear programming (Schrijver (1986)), all input parameters can be continuous, though the solutions must be integer vectors. For these problems, we can still consider the effects of Gaussian perturbations. In this subsection, we discuss results on the smoothed analysis of the binary optimization problem, integer programming, and some problems in online optimization.

Binary Optimization Problems

Beier and Vöcking (2004) consider the following Binary Optimization Problem:

$$\max \quad \mathbf{c}^T\mathbf{x} \qquad \text{subject to} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b} \text{ and } \mathbf{x} \in \{0,1\}^n$$

Several classical discrete optimization problems can be expressed as binary optimization problems. One example is the 0/1-Knapsack Problem: Given a set of n items $\{(w_1, v_1), ..., (w_n, v_n)\}$ where item i has weight $w_i \geq 0$ and value $v_i \geq 0$, and a knapsack of capacity c, find a

subset $S \subseteq \{1, ..., n\}$ with $\sum_{i \in S} w_i \leq c$ that maximizes $\sum_{i \in S} v_i$. By setting $x_i = 1$ if $i \in S$ and $x_i = 0$ if $i \notin S$, one can express the knapsack problem as a binary optimization problem:

$$\max \sum_i v_i x_i, \text{ subject to } \sum_i w_i x_i \leq c \text{ and } x_i \in \{0,1\} \ \forall \ i \in \{1, ..., n\}.$$

Another example is the Constrained Shortest Path Problem (Ziegelmann (2001)): Given a graph $G = (V, E)$ where each edge $e \in E$ has distance $d_e > 0$ and latency $l_e \geq 0$, a source vertex s, a destination vertex t, and a latency tolerance parameter L, find a path P from s to t with $\sum_{e \in P} l_e \leq L$ that minimizes $\sum_{e \in P} d_e$.

Let $\mathcal{P}$ be the set of all simple paths from s to t. In any feasible solution, there must be a path in $\mathcal{P}$ with all of its edges chosen. Let $\mathcal{C}$ be all subsets of the edges whose removal disconnects s from t. By the duality relation between cuts and paths, in any feasible solution and for any cut $C \in \mathcal{C}$, at least one of its edges is chosen. For each $e \in E$, let x_e be a binary variable with $x_e = 1$ if e is chosen. One can then reformulate the constrained shortest path problem as:

$$\begin{aligned} \max \quad -\textstyle\sum_{e \in E} d_e x_e \quad \text{subject to} & \sum_{e \in E} l_e x_e \leq L \\ & \sum_{e \in C} x_e \geq 1 \quad \text{for all } C \in \mathcal{C} \quad \text{and} \\ & x_e \in \{0,1\} \text{ for all } e \in E. \end{aligned}$$

In their smoothed analysis of the binary optimization problem, Beier and Vöcking distinguish two types of expressions: *deterministic expressions* and *stochastic expressions*. Unlike the stochastic constraints, the deterministic constraints are not subject to perturbations. For instance, in the smoothed analysis of the Constrained Shortest Path Problem, one could assume the distances and latencies are subject to some perturbations while the set of combinatorial structure of the graph is not subject to any perturbation, making the constraints $\sum_{e \in C} x_e \geq 1$ for all $C \in \mathcal{C}$ deterministic.

One way to capture the deterministic constraints in the binary optimization problem is the following reformulation:

$$\min \quad \mathbf{c}^T \mathbf{x} \qquad \text{subject to} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b} \text{ and } \mathbf{x} \in S \cap \{0,1\}^n, \qquad (1.6)$$

where S is the intersection of the feasible sets of the deterministic linear constraints. Then, in smoothed analysis, one assumes entries of $(\mathbf{A}, \mathbf{b})$

are always subject to perturbations and only needs to consider the possibility of whether the objective function is also subject to perturbations.

Beier and Vöcking (2004) also introduce a quite general way to extend Gaussian perturbations: Let f be a piece-wise continuous univariate density function of a probability distribution (i.e., $f(x) \geq 0$ and $\int_{\mathbb{R}} f(x)dx = 1$), with finite mean $\int_{\mathbb{R}} |x| f(x)dx$ and $\sup_x f(x) = 1$. For $\sigma \leq 1$, let f_σ be a scaling of f such that $f_\sigma(x) = f(x/\sigma)/\sigma$. It is easy to check that the mean of f_σ satisfies

$$\int_{\mathbb{R}} |x| \cdot f_\sigma(x)dx = \sigma \left(\int_{\mathbb{R}} |x| \cdot f(x)dx \right),$$

and

$$\int_{\mathbb{R}} f_\sigma(x)dx = \int_{\mathbb{R}} f(x/\sigma)/\sigma dx = \int_{\mathbb{R}} f(y)dy = 1.$$

For any $\bar{x}$, an f-perturbation with magnitude $\sigma < 1$ is a random variable $x = \bar{x} + r$, where r is randomly chosen according to a density f_σ. To perturb a vector, one independently perturbs each of its entries. For example, the σ-Gaussian perturbation is an f-perturbation with magnitude σ when $f(x) = e^{-x^2/2}/\sqrt{2\pi}$. By setting f to be $1/2$ in $[-1, 1]$ and 0 outside $[-1, 1]$, one obtains a family of uniform perturbations within a box of side-length 2σ centered at a vector. By setting f to be 1 in $[0, 1]$ and 0 outside $[0, 1]$, one obtains a family of uniform perturbations with a box of side-length σ in the positive quadrant of a vector.

Before stating the main result of Beier and Vöcking, let us first review a few concepts from complexity theory (Papadimitriou (1994) and Sipser (1996)). Let RP denote the class of decision problems solvable by a randomized polynomial time algorithm such that for every "yes"-instance, the algorithm accepts with probability at least $1/2$, and for every "no"-instance, the algorithm always rejects. Let coRP be the complement of RP. Let ZPP be the intersection of RP and coRP. In other words, ZPP is the class of decision problems solvable by a randomized algorithm that always returns the correct answer, and whose expected running time (on every input) is polynomial.

For a binary optimization problem Π, let Π_u be the "unary" representation of Π – in Π_u, all parameters in the stochastic expressions are assumed to be integers in unary representation.

Theorem 1.11 (Beier and Vöcking's Characterization) *For every density function f with finite mean and $\sup_x f(x) \leq 1$, in the perturbation model defined by f-perturbations, a binary optimization problem Π*

is in smoothed polynomial time probabilistically (see (1.4)) if and only if $\Pi_u \in$ ZPP.† *Moreover, a binary optimization problem* Π *is in smoothed polynomial time as in (1.1), if* Π_u *can be solved in linear time.*

For instance, because the unary version of the 0/1-knapsack problem can be solved in linear time using dynamic programming, as a corollary of Theorem 1.11, the 0/1-knapsack problem is in smoothed polynomial time. Like the 0/1-knapsack problem, the Constrained Shortest Path Problem is NP-complete while its unary version is in P. Thus, the Constrained Shortest Path Problem is in smoothed polynomial time probabilistically. One can similarly prove the Constrained Minimum Spanning Tree Problem (Ravi & Goemans (1996)), the Constrained Minimum Weighted Matching Problem, and several other instances of packing problems are in smoothed polynomial time in the sense of Definition 1.11.

In contrast, even though 0/1-integer programming with a fixed number of constraints is in smoothed polynomial time, general 0/1-integer program is not (unless NP = RP).

In the proof of Theorem 1.11, Beier and Vöcking examine the distribution of three quantities of a binary optimization problem Π:

- **Winner Gap:** If Π is feasible and has more than one feasible solution, then the *winner gap* is the difference between the objective value of the optimal solution and the objective value of the second best feasible solution.
- **Loser Gap:** Let $I^+(\Pi) \subseteq \{0,1\}^n$ be the set of infeasible binary vectors with better objective values than the optimal solution. Suppose the feasible region is given by $\mathbf{a}_i^T\mathbf{x} \leq b_i$, for $i \in [1,m]$. The *loser gap* is then equal to
$$\min_{\mathbf{x}\in I^+(\Pi)} \max_i (\mathbf{a}_i^T\mathbf{x} - b_i),$$
that is, the minimum amount of violation of binary vectors in $I^+(\Pi)$.
- **Feasibility Gap:** Suppose $\mathbf{x}^*$ is the optimal solution. Then the *feasibility gap* is equal to
$$\min_{i:\ \mathbf{a}_i^T\mathbf{x} \leq b_i \text{ is stochastic}} (b_i - \mathbf{a}_i^T\mathbf{x}^*),$$

† Usually by saying Π has a *pseudo-polynomial time* algorithm, one means $\Pi_u \in$ P. So $\Pi_u \in$ ZPP means that Π are solvable by a randomized *pseudo-polynomial time* algorithm. We say a problem Π is strongly NP-hard, if Π_u is NP-hard. For example, 0/1-integer programming with fixed number of constraints is in pseudo-polynomial time, while general 0/1-integer programming is strongly NP-hard.

that is, the minimum slack of the optimal solution with respect to stochastic constraints.

Note that these three quantities are closely related with the concept of the condition numbers studied in the smoothed analysis of the perceptron algorithm and the interior-point algorithms. Beier and Vöcking prove that the reciprocal of each of these quantities is polynomial with high probability. Consequently, if the binary optimization problem has k stochastic equations and n variables, then with high probability the winner is uniquely determined when revealing $O(\log(nk/\sigma))$ bits of each stochastic coefficient. So, if Π_u is in ZPP, then the ZPP algorithm would solve almost all perturbed instances.

Integer Programming

Röglin and Vöcking (2005) extend the result of Beier and Vöcking to integer linear programming. They consider programs of the form

$$\max \quad \mathbf{c}^T\mathbf{x} \qquad \text{subject to} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b} \text{ and } \mathbf{x} \in \mathcal{D}^n, \tag{1.7}$$

where $\mathbf{A}$ is an $m \times n$ real matrix, $\mathbf{b} \in \mathbb{R}^m$, and $\mathcal{D} \subset \mathbb{Z}$.

Theorem 1.12 (Röglin and Vöcking) *For any constant c, let Π be a class of integer linear programs of form (1.7) with $|D| = O(n^c)$. Then, Π is in smoothed polynomial time in the probabilistic sense if and only if $\Pi_u \in$ ZPP.*

Smoothed Competitive Ratio of Online Scheduling

In online computing (Sleator & Tarjan (1985)) an input is given as a sequence of events $\mathbf{x} = x_1 \circ x_2 \circ \ldots \circ x_i \circ \ldots$. For all i, an online algorithm A must generate a response r_i based on $\{x_1, \ldots, x_i\}$ only. Let $\mathbf{r}_A(\mathbf{x}) = r_1 \circ r_2 \circ \ldots \circ r_i \circ \ldots$ be its response sequence. There is a positive-valued cost function $\mathbf{COST}(\mathbf{r})$ for each response $\mathbf{r}$.

Let $\mathbf{OPT}(\mathbf{x})$ be the cost of the best response, possibly generated by an optimal offline algorithm that has access of all events in $\mathbf{x}$. Sleator and Tarjan (1985) define the worst-case competitive ratio of an online algorithm to be

$$\sup_{\mathbf{x}} \left\{ \frac{\mathbf{COST}(\mathbf{r}_A(\mathbf{x}))}{\mathbf{OPT}(\mathbf{x})} \right\}.$$

Becchetti, Leonardi, Marchetti-Spaccamela, Schäfer, and Vredeveld

(2003) apply smoothed analysis to evaluate the performance of online algorithms. For a perturbation model $\mathcal{D}$, they define the smoothed competitive-ratio as

$$\sup_{\bar{\mathbf{x}}} \left\{ \mathrm{E}_{\mathbf{x} \in_D \mathcal{D}(\bar{\mathbf{x}})} \left[\frac{\mathbf{COST}(\mathbf{r}_A(\mathbf{x}))}{\mathbf{OPT}(\mathbf{x})} \right] \right\}.$$

They then apply this measure to the following online scheduling problem. The input is a collection of jobs $\{j_1, \ldots, j_n\}$. Each job j_i has a release time R_i and processing time T_i. An online scheduler only knows the existence of j_i at its release time R_i. Its processing time T_i is not known until the job is completed. In the system, one is allowed to interrupt a running job and resume it later on the system. The system could have only one machine or m parallel machines. The scheduler decides when and which uncompleted job should be executed at an available machine, as well as when and which running job to interrupt. Each machine can only process at most one job at any time. Suppose the completion time of job j_i is C_i. The *flow time* of j_i is then $F_i = C_i - R_i$. The objective of the scheduler is to minimize the total flow time

$$\sum_i F_i = \sum_i (C_i - R_i).$$

Since the Multi-Level Feedback algorithm (MLF) is one of the commonly used processor-scheduling algorithms in operating systems such as Unix and Windows NT, Becchetti et al. (2003) choose to analyze its smoothed competitive ratio.

MLF maintains a set of queues $Q_0, \ldots, Q_{\ldots}$ and uses Q_i to book-keep jobs that have recently processed for 2^{i-1} time units. At each stage, the scheduler processes the job at the front of the lowest non-empty queue. This algorithm is *non-clairvoyant* as it makes decisions without full knowledge of the running time of each job.

Although MLF performs well in practice, its worst-case competitive ratio is rather poor. In fact no deterministic non-clairvoyant preemptive scheduling algorithm has good competitive ratio due to a lower bound of $\Omega(2^K)$ on the competitive ratio of any such scheduling algorithm when processing times of jobs are chosen from $[1, 2^K]$, as shown by Motwani, Phillips, and Torng (1993).

In their smoothed analysis, Becchetti et al. use the partial bit perturbation model introduced by Banderier, Beier, and Mehlhorn (2003) with magnitude parameter $k \leq K$. See Definition 1.4. Becchetti et al.'s results hold for any *well-conditioned distribution* over $[0, 2^{k-1}]$ whose

density function f satisfies that f is symmetric around its mean $\mu(f) \in [0, 2^{k-1}]$ and f is non-decreasing in $[1, \mu(f)]$. Let σ denote the standard deviation of f.

Theorem 1.13 (Smoothed Performance of MLF) *For any K, k, and σ, under a partial k-bit perturbation model with a well-conditioned distribution of standard deviation σ, the smoothed competitive ratio of MLF is $O\left(2^{2k}\sigma^{-2}2^{K-k} + 2^{2k}\sigma^{-3}\right)$.*

For example, Theorem 1.13 implies that the smoothed competitive ratio of MLF is $O(2^{K-k})$ for the uniform partial k-bit randomization as the standard deviation of this distribution is $\Theta(2^k)$.

Metrical Task Systems

The *metrical task systems* introduced by Borodin, Linial, and Saks (1992) provide a general framework for modeling many online problems including various scheduling and paging problems. A metrical task system is defined by a weighted, connected and undirected graph $G = (V, E, \mathbf{d})$, where, for each $e \in E$, $\mathbf{d}(e) > 0$ specifies the length of edge e. For simplicity, one can assume $V = \{1, ..., n\}$. Naturally, via shortest paths, $\mathbf{d}$ also defines the distance $\mathbf{d}(u, v)$ between any two vertices u and v in the graph. A task can then be represented by an n-dimensional vector $\tau = (\tau(1), ..., \tau(n))$, where $\tau(i)$ specifies the service cost of performing τ at vertex i. In an online metrical task system, a server is initially positioned at a starting vertex v_0, and needs to service a sequence $(\tau_1, \ldots, \tau_i, ...)$ of tasks. Upon receiving τ_i, an online algorithm must decide which vertex v_i to service τ_i. So the cost to service τ_i depends on v_{i-1} and is equal $\mathbf{d}(v_i, v_{i-1}) + \tau_i(v_i)$. The objective of the optimization problem is to minimize the total service cost $\sum_i \left(\mathbf{d}(v_i, v_{i-1}) + \tau_i(v_i)\right)$.

The deterministic online algorithm with the best possible worst-case competitive ratio is the Work Function Algorithm (WFA) developed by Borodin, Linial, and Saks. The idea of this algorithm is very simple. Let $w_i(v)$ be the minimum offline cost to process $(\tau_1, \ldots, \tau_i)$ with starting position v_0 and ending position v. The vector $\mathbf{w}_i = (..., w_i(v), ...)$ is called the work function. One can compute $\mathbf{w}_i$ incrementally by dynamic programming. The optimal off-line cost to process $(\tau_1, \ldots, \tau_i)$ is then $\min_v w_i(v)$. WFA simply chooses s_i to be the vertex that realizes $\min_v(w_i(v) + \mathbf{d}(s_{i-1}, v))$. Borodin et al. proved that the worst-case

competitive ratio of WFA is $2n-1$, and also proved that $2n-1$ is the best possible competitive ratio for any deterministic online algorithm.

Schäfer and Sivadasan (2004) consider the smoothed competitive ratio assuming that the service cost of each task is subject to a random perturbation with mean 0 and standard deviation σ. If the perturbation makes a cost negative, then the cost would be reassigned to 0.

Theorem 1.14 (Schäfer and Sivadasan) *There exist constants c_1 and c_2 such that for all $(\ldots, \bar{\tau}_i, \ldots)$ with $\bar{\tau}_i \leq 1$, the smoothed competitive ratio of WFA is bounded above by the smaller of the following two quantities:*

$$c_1 \cdot \left(\frac{\mathbf{DIAMETER}\,(G)}{\lambda_{\min}} \left(\frac{\lambda_{\min}}{\sigma} + \log \Delta \right) \right), \quad \textit{and}$$

$$c_2 \cdot \left(\sqrt{n \cdot \frac{\lambda_{\max}}{\lambda_{\min}} \left(\frac{\lambda_{\min}}{\sigma} + \log \Delta \right)} \right),$$

where $\lambda_{\min} = \min_{e \in E} \mathbf{d}(e)$, $\lambda_{\max} = \max_{e \in E} \mathbf{d}(e)$, $\mathbf{DIAMETER}\,(G)$ is the diameter, $\max_{u,v} \mathbf{d}(u,v)$, of G, and Δ is the maximum vertex degree of G.

Furthermore, if the service cost vector of each task contains at most k non-zero entries, then the smoothed competitive ratio of WFA is

$$O\left(k \cdot \frac{\lambda_{\max}}{\lambda_{\min}} \left(\frac{\lambda_{\min}}{\sigma} + \log \Delta \right) \right).$$

1.3.3 Scientific Computing

Scientific computing is another area where input parameters are often continuous. In addition, due to round-off errors in scientific computing, inputs to numerical algorithms are subject to random perturbations (Wilkinson (1963)). We now discuss results of smoothed analysis in solving linear systems and in parallel mesh generation.

Growth Factor and Bit-Complexity of Gaussian Elimination

Solving linear systems is the most fundamental problem in computational science and numerical linear algebra (Strang (1980), Golub & Van Loan (1989), Demmel (1997)). The most common method used to find a solution to a system $\mathbf{Ax} = \mathbf{b}$ is the classical Gaussian elimination. The method first uses elimination to reduce an n variables and n

equations system to a smaller $n-1$ by $n-1$ system and then recursively solves the smaller system. In the elimination, it chooses one of the equations and one of the variables, and uses the chosen equation to eliminate the variable from other equations. The choice of the equation and the variable is determined by a pivoting rule.

The simplest pivoting rule is to use the i^{th} equation to eliminate the i^{th} variable. This process, often referred as *Gaussian elimination without pivoting*, factors the coefficient matrix $\mathbf{A}$ into

$$\mathbf{A} = \mathbf{LU},$$

where $\mathbf{L}$ is a lower triangular matrix and $\mathbf{U}$ is an upper triangular matrix.

The pivoting rule most used in practice is *partial pivoting*. In the i^{th} step, it chooses the equation in which the i^{th} variable has the largest coefficient in absolute value, and uses that equation to eliminate the i^{th} variable. Gaussian elimination with partial pivoting defines a row-permutation matrix $\mathbf{P}$ and factors $\mathbf{PA}$ into

$$\mathbf{PA} = \mathbf{LU}.$$

Because of the partial pivoting, all entries in $\mathbf{L}$ have absolute value at most 1.

Another quite natural pivoting rule is the complete pivoting rule. In the i^{th} step, it chooses the equation containing the the largest coefficient (in absolute value) from the entire system uses it to eliminate the variable that has that coefficient. Gaussian elimination with complete pivoting produces a row permutation matrix $\mathbf{P}$ (for the choices of equations) and a column permutation matrix $\mathbf{Q}$ (for the choices of variables) and factors $\mathbf{PAQ}$ into

$$\mathbf{PAQ} = \mathbf{LU}.$$

In his seminal work, Wilkinson (1961) considers the number of bits needed to obtain a solution of a given accuracy. He proves that it suffices to carry out Gaussian elimination with

$$b + \log_2(5n\kappa(\mathbf{A}) \left\|\mathbf{L}\right\|_\infty \left\|\mathbf{U}\right\|_\infty / \left\|\mathbf{A}\right\|_\infty + 3)$$

bits of accuracy to obtain a solution that is accurate to b bits. In the formula, $\kappa(\mathbf{A}) = \left\|\mathbf{A}\right\|_2 \left\|\mathbf{A}^{-1}\right\|_2$ is the condition number of $\mathbf{A}$ where $\left\|\mathbf{A}\right\|_2 = \max_{\mathbf{x}} \left\|\mathbf{Ax}\right\|_2 / \left\|\mathbf{x}\right\|_2$, and $\left\|\mathbf{A}\right\|_\infty$ is the maximum absolute row sum. The reciprocal of $\left\|\mathbf{A}^{-1}\right\|_2$ is also known as the smallest singular value of $\mathbf{A}$.

The quantity $\left\|\mathbf{L}\right\|_\infty \left\|\mathbf{U}\right\|_\infty / \left\|\mathbf{A}\right\|_\infty$ is called the growth factor of the

elimination. It depends on the pivoting rule. We will use $\rho_{GEWP}(\mathbf{A})$, $\rho_{GEPP}(\mathbf{A})$, and $\rho_{GECP}(\mathbf{A})$ to respectively denote the growth factors of Gaussian elimination without pivoting, with partial pivoting, and with complete pivoting.

For some nonsingular matrix $\mathbf{A}$, $\rho_{GEWP}(\mathbf{A})$ could be unbounded as the pivoting coefficient could be zero or arbitrarily close to zero.

Wilkinson constructs a family of matrices, $\mathbf{W}_n = \mathbf{L}_n + \mathbf{C}_n$, where $\mathbf{L}_n$ is an $n \times n$ lower triangular matrix with diagonal entries equal to 1 and off-diagonal entries equal to -1, and $\mathbf{C}_n$ is a matrix with $\mathbf{C}_n[n,n] = 0$, $\mathbf{C}_n[i,j] = 0$ for $j < n$, and $\mathbf{C}_n[i,n] = 1$ for $i < n$. For $\mathbf{W}_n$, $\rho_{GEPP}(\mathbf{W}_n) = \Omega(2^n)$. On the positive side, Wilkinson also proves that for any non-singular matrix $\mathbf{A}$, $\rho_{GECP}(\mathbf{A}) = n^{O(\log n)}$.

Wilkinson's counterexample shows that in the worst-case one must use at least $\Omega(n)$ bits to accurately solve every linear system using the partial pivoting rule. However, in practice one usually obtains accurate answers using much less precision†. In fact, it is rare to find an implementation of Gaussian elimination that uses more than double precision, and high-precision solvers are rarely used or needed (Trefethen & Schreiber (1990) and Trefethen & Bau (1997)). For example, LAPACK uses 64 bits (Anderson et al. (1999)).

Sankar, Spielman, and Teng (2005) conduct smoothed analysis of growth factors for Gaussian eliminations without pivoting and with partial pivoting. They prove the following results.

Theorem 1.15 (Gaussian Elimination without Pivoting) *For $n > e^4$, let $\bar{\mathbf{A}}$ be an n-by-n matrix for which $\left\|\bar{\mathbf{A}}\right\|_2 \leq 1$, and let $\mathbf{A}$ be a σ-Gaussian perturbation of $\bar{\mathbf{A}}$, for $\sigma \leq 1/2$. Then,*

$$\mathrm{E}\left[\log \rho_{GEWP}(\mathbf{A})\right] \leq 3\log_2 n + 2.5\log_2\left(\frac{1}{\sigma}\right) + \frac{1}{2}\log_2\log_2 n + 1.81.$$

Theorem 1.16 (Gaussian Elimination with Partial Pivoting) *For any n-by-n matrix $\bar{\mathbf{A}}$ such that $\left\|\bar{\mathbf{A}}\right\|_2 \leq 1$, let $\mathbf{A}$ be a σ-Gaussian perturbation of $\bar{\mathbf{A}}$. Then, for $x > 1$*

$$\Pr\left[\rho_{GEPP}(\mathbf{A}) > x^{21}\left(\frac{n(1+\sigma\sqrt{n})}{\sigma}\right)^{12\log n}\right] \leq x^{-\log n}.$$

† Wright (1993) gives a collection of natural problems for which Gaussian elimination with partial pivoting is unstable.

Condition Number of Matrices

A key step in the smoothed analysis of the growth factor is to obtain a smoothed bound on the condition number of a square matrix. The condition number $\kappa(\mathbf{A}) = \|\mathbf{A}\|_2 \left\|\mathbf{A}^{-1}\right\|_2$ measures how much the solution to a system $\mathbf{Ax} = \mathbf{b}$ changes as one makes slight changes to $\mathbf{A}$ and $\mathbf{b}$. A consequence is that if one solves the linear system using fewer than $\log(\kappa(\mathbf{A}))$ bits of precision, one is likely to obtain a result far from a solution (Golub & Van Loan (1989), Trefethen & Bau (1997), Demmel (1997)).

Sankar, Spielman, and Teng (2005) establish the following smoothed bound on the condition number:

Theorem 1.17 (Smoothed Analysis of Condition number) *Let $\bar{\mathbf{A}}$ be an $n \times n$ matrix satisfying $\left\|\bar{\mathbf{A}}\right\|_2 \leq \sqrt{n}$, and let $\mathbf{A}$ be a σ-Gaussian perturbation of $\bar{\mathbf{A}}$, with $\sigma \leq 1$. Then,*

$$\Pr\left[\kappa(\mathbf{A}) \geq x\right] \leq \frac{14.1n\left(1 + \sqrt{2\ln(x)/9n}\right)}{x\sigma}.$$

As bounds on the norm of a random matrix are standard, to prove Theorem 1.17, one only needs to focus on the norm of the inverse. Notice that $1/\left\|\mathbf{A}^{-1}\right\|_2 = \min_{\mathbf{x}} \|\mathbf{Ax}\|_2 / \|\mathbf{x}\|_2$ is the smallest singular value of $\mathbf{A}$. Sankar, Spielman, and Teng prove the following theorem:

Theorem 1.18 (Smallest singular value) *Let $\bar{\mathbf{A}}$ be an arbitrary square matrix in $\mathbb{R}^{n\times n}$, and let $\mathbf{A}$ be a σ-Gaussian perturbation of $\bar{\mathbf{A}}$. Then*

$$\Pr_{\mathbf{A}}\left[\left\|\mathbf{A}^{-1}\right\|_2 \geq x\right] \leq 2.35\frac{\sqrt{n}}{x\sigma}$$

Wschebor (2004) improves the smoothed bound on the condition number.

Theorem 1.19 (Wschebor) *Let $\bar{\mathbf{A}}$ be an $n \times n$ matrix and let $\mathbf{A}$ be a σ-Gaussian perturbation of $\bar{\mathbf{A}}$ for $\sigma \leq 1$. Then,*

$$\Pr\left[\kappa(\mathbf{A}) \geq x\right] \leq \frac{n}{x}\left(\frac{1}{4\sqrt{2\pi n}} + 7\left(5 + \frac{4\left\|\bar{\mathbf{A}}\right\|_2^2(1+\log n)}{\sigma^2 n}\right)^{1/2}\right).$$

When $\left\|\bar{\mathbf{A}}\right\|_2 \leq \sqrt{n}$, his result implies

$$\Pr\left[\kappa(\mathbf{A}) \geq x\right] \leq O\left(\frac{n \log n}{x\sigma}\right).$$

Zero-Preserving Perturbations and Symmetric Linear Systems

Many matrices that occur in practice are both symmetric and sparse. Moreover, numerical algorithms take advantage of any assumed sparseness. Thus, it is natural to study the smoothed complexity of algorithms under perturbations that respect symmetry and non-zero structure. See Definition 1.8 for zero-preserving perturbations. One can express a symmetric matrix $\mathbf{A}$ as $\mathbf{T} + \mathbf{D} + \mathbf{T}^T$, where $\mathbf{T}$ is a lower-triangular matrix with zeros on the diagonal and $\mathbf{D}$ is a diagonal matrix. By making a zero-preserving perturbation to $\bar{\mathbf{T}}$, we preserve the symmetry and the zero-structure of the matrix.

Sankar, Spielman, Teng (2005) extend their results on condition number and growth factor to symmetric matrices with zero-preserving and symmetry-preserving perturbations.

Theorem 1.20 (Condition number and growth factor of symmetric matrices) *Let $\bar{\mathbf{A}} = \bar{\mathbf{T}} + \bar{\mathbf{D}} + \bar{\mathbf{T}}^T$ be an arbitrary n-by-n symmetric matrix satisfying $\left\|\bar{\mathbf{A}}\right\|_2 \leq \sqrt{n}$. Let $\sigma^2 \leq 1$, let $\mathbf{T}$ be a zero-preserving σ-Gaussian perturbation of $\bar{\mathbf{T}}$, let $\mathbf{G}_D$ be a diagonal matrix of Gaussian random variables of standard deviation σ and mean 0 that are independent of $\mathbf{T}$ and let $\mathbf{D} = \bar{\mathbf{D}} + \mathbf{G}_D$. Then, for $\mathbf{A} = \mathbf{T} + \mathbf{D} + \mathbf{T}^T$ and $x \geq 2$,*

$$\Pr\left[\kappa(\mathbf{A}) \geq x\right] \leq O\left(\frac{n^2\sqrt{\log x}}{x\sigma}\right).$$

In addition,

$$\mathrm{E}\left[\log \rho_{GEWP}(\mathbf{A})\right] = O\left(\log\left(\frac{n}{\sigma}\right)\right).$$

Well-Shaped Mesh Generation: Parallel Delaunay Refinement

Mesh generation is a key problem in scientific computing (Edelsbrunner (2001) and Teng & Wong (2000)). In mesh generation, one is given a geometric domain, specified by its boundary. The goal is to produce a triangulation of the domain, wherein all triangles are well-shaped. One

standard definition of well-shapedness is that the ratio of the circumradius to the shortest side of each triangle is bounded by a prespecified constant, such as 2 (Miller et al. (1995)). One would also like to minimize the number of triangles in the mesh.

A special family of input domains is the periodic point set: If P is a finite set of points in the half-open unit box $[0,1)^d$ and $\mathbb{Z}^d$ is the d-dimensional integer grid, then $S = P + \mathbb{Z}^d$ is a periodic point set (Edelsbrunner (2001)). The periodic set S contains all points $p + v$, where $p \in P$ and v is an integer vector. Periodic point sets are often used to study some basic issues in well-shaped mesh generation and to simplify the presentation of several mesh generation algorithms (Cheng et al. (2000) and Edelsbrunner et al. (2000)).

The Delaunay triangulation of a periodic point set is also periodic. In general, this triangulation might not be well-shaped. A practical and popular technique to generate a well-shaped mesh is to iteratively apply Delaunay refinement†: Choose a triangle that is not well-shaped, add its circumcenter to the domain, and recompute the Delaunay triangulation. This process will be repeated until there are no more poorly-shaped elements.

The standard Delaunay refinement algorithms are inherently sequential. Spielman, Teng, and Üngör (2002) define a parallel rule‡ for adding more points for refinement. They prove that if the minimum pair-wise distance of the input point set is s, then the use of the parallel rule takes $O(\log^2(1/s))$ parallel time. In the smoothed setting where input points are subject to small perturbations, their result implies:

Theorem 1.21 (Parallel Delaunay Refinements) *For any point set $\bar{P} \subseteq \mathbb{R}^2$, let P be a σ-Gaussian perturbation of $\bar{P}$. Then there is a parallel Delaunay refinement algorithm that, in $O(\log^2(1/\sigma))$ time, produces a well-shaped mesh for $P+\mathbb{Z}^2$. Moreover, the number of elements in this mesh is within a constant factor of the optimal solution. The number of processors needed is equal to the size of the resulting mesh.*

† Chew (1989) and Ruppert (1993) pioneer this approach in two dimensions and Shewchuk (1998) extends it to three dimensions. Li and Teng (2001), Cheng and Dey (2002) resolve a major difficulty of Shewchuk's extension to ensure that all tetrahedra are well-shaped.

‡ By relaxing the refinement rule that the circumcenters of poorly shaped triangles must be added, then in Spielman, Teng, and Üngör (2004), they show another approach with $(\log(1/s))$ parallel time.

1.3.4 Discrete and Geometric Structures

The notion of perturbations and neighborhoods is far less straightforward for discrete structures than for continuous structures. Perhaps, the simplest model for perturbing a graph $\bar{G} = (V, \bar{E})$ is to insert every non-edge $(i, j) \notin \bar{E}$ into the graph with some probability σ and to delete every edge $(i, j) \in E$ from the graph with some probability σ. We denote this distribution on graphs by $\mathcal{P}(\bar{G}, \sigma)$ and call the resulting graph $G = (V, E)$ a σ-perturbation of $\bar{G}$.

Unfortunately, such perturbations can considerably change the properties of a graph. For example, the above perturbation model can not be used if we would like to study the performance of a bisection algorithm for planar graphs, because almost all perturbed graphs are non-planar.

We now discuss some modifications of this model, which have been proposed in an attempt to make the analysis more meaningful.

- **Property-Preserving Perturbations** (Spielman & Teng (2003a)): Given a property P and a basic model of perturbation, a P-preserving perturbation of an object $\bar{X}$ is a perturbation of $\bar{X}$ according to the basic perturbation model but subject to the condition $P(X) = P(\bar{X})$. In the case when $\bar{G}$ is a graph and the basic model is the σ-perturbation of $\bar{G}$, the probability of a graph G with $P(G) = P(\bar{G})$ is

$$\frac{\Pr_{G \leftarrow \mathcal{P}(\bar{G},\sigma)} \left[G \text{ and } (P(G) = P(\bar{G})) \right]}{\Pr_{G \leftarrow (\bar{G},\sigma)} \left[P(G) = P(\bar{G}) \right]}$$

  In the property preserving perturbation model for graphs, P can be any graph property such as planarity or can be a combination of several properties such as being planar and having a bisection of size at most B.
- **The Semi-Random Model** (Blum & Spencer (1995)): Blum and Spencer's semi-random graph model combines Santha and Vazirani's (1986) semi-random source with the random graph model that has a "planted solution" (Boppana (1987)). Since this is best described by an example, let us consider the k-Coloring Problem.

  An adversary plants a solution by partitioning the set V of n vertices into k subsets $V_1, \ldots, V_k$. Let

$$F = \{(u, v) | u \text{ and } v \text{ are in different subsets}\}$$

  be the set of potential inter-subset edges. A graph is then constructed by the following semi-random process that perturbs the decisions of the adversary: Initially let $H = F$. Then while H is not empty,

(i) the adversary chooses an edge e from H, and decides whether it would like to include the edge in the graph.
(ii) The semi-random process then reverses the decision with probability σ.
(iii) The edge e is then removed from H.

We will refer such a graph as a *semi-random k-colorable graph*. In this model, one can also require that each V_i has size n/k or $\Theta(n/k)$. Such a graph is called a *balanced semi-random k-colorable graph*.

All semi-random k-colorable graphs have the same planted coloring: $\Pi(v) = i$ for all $v \in V_i$, because both the adversary and the semi-random process preserve this solution by only considering edges from F.

As with the smoothed model, one can work with the semi-random model, by varying σ from 0 to 1/2, to interpolate between worst-case and average-case complexity for k-coloring. In fact, the semi-random model is related to the following perturbation model that *partially preserves* a particular solution:

Let $\bar{G} = (V, \bar{E})$ be a k-colorable graph. Let $\Pi : V \to \{1, ..., k\}$ be a k-coloring of $\bar{G}$ and let $V_i = \{v \mid \Pi(v) = i\}$. The model then returns a graph $G = (V, E)$ that is a σ-perturbation of $\bar{G}$ subject to Π also being a valid k-coloring of G. In other words, the perturbation is subject to

$$E \subseteq F = \{(u, v) | u \text{ and } v \text{ are in different subsets}\}.$$

This perturbation model is equivalent to the semi-random model in which the adversary is oblivious. An oblivious adversary simply chooses a set $\bar{E} \subseteq F$, and sends the decisions that only include edges in $\bar{E}$ (and hence exclude edges in $F - \bar{E}$) through the semi-random process. Thus, if a k-coloring algorithm can successfully color semi-random k-colorable graphs (with high probability), it must also be able to color graphs generated by the perturbation model that partially preserves a particular solution.

- **Solution-Preserving Perturbations**: The semi-random model only generates graphs that have the planted solution and hence only assigns non-zero probabilities to graphs that have the planted solution. In this model, the decision problem, such as whether a graph is k-colorable, is trivial. The search problem, in which one is asked to find the planted solution, is the subject of the study.

  An alternative model is to apply perturbations that preserve a planted solution. Continuing to use the graph coloring problem as

our example, let $\Pi : V \to \{1, ...k\}$ be a k-coloring assignment. For a graph $\bar{G} = (V, \bar{E})$, if it is k-colored by Π, then G is a σ-perturbation of $\bar{G}$ subject to G having Π as a solution; otherwise G is a σ-perturbation of $\bar{G}$ subject to G not having Π as a solution.

- **Monotone Adversary Semi-Random Model** (Blum & Spencer (1995), Feige & Kilian (1998) and Feige & Krauthgamer (2002)): Blum and Spencer define another generation process of semi-random graphs of k-colorable graphs: First, partition the set of n vertices into k subsets $V_1, \ldots, V_k$ each having n/k vertices. Second, choose a set E_1 of edges by selecting each edge in

$$F = \{(u, v) | u \text{ and } v \text{ are in different subsets}\}$$

independently with probability σ. Then, the adversary chooses another set E_2 from F and returns $G = (V, E_1 \cup E_2)$.

The monotone adversary semi-random model can be applied to a graph problem P with a particular "planted" solution S. It is a two step model. In the first step, it generates a "random" graph $\tilde{G} = (V, \tilde{E})$ with S as a solution according to some distribution. In the second step, the adversary can only modify $\tilde{G}$ in a limited way – the modification has to respect the planted solution S.

For example, the following is the semi-random model for graph bisection analyzed by Feige and Kilian (1998): Let $0 \leq q < p \leq 1$, let V_1 be a subset of V of size $n/2$ and let $V_2 = V - V_1$. The random process builds a graph $\tilde{G} = (V, \tilde{E})$ by selecting each edge in $V_1 \times V_2$ with independent probability q and each edge in $(V_1 \times V_1) \cup (V_2 \times V_2)$ with probability p. The adversary is then given the chance to add a subset of edges from $(V_1 \times V_1) \cup (V_2 \times V_2)$ to the graph and remove a subset of edges of $V_1 \times V_2$ from the graph.

Results in the Semi-Random Model

We now summarize some results in the monotone adversary semi-random model.

Theorem 1.22 (Semi-Random Coloring: Blum-Spencer) *For any $\epsilon > 0$ and $p \geq n^{\frac{-2k}{(k+1)k-2}+\epsilon}$, there is a polynomial-time algorithm to k-color a balanced semi-random k-colorable graph with probability $1 - o(1)$. When $k = 3$, the condition on p is $p \geq n^{-1/3+\epsilon}$.*

Feige and Kilian (1998) improve the above result and show:

Theorem 1.23 (Semi-random Coloring: Feige and Kilian) *For any constant k, $\epsilon > 0$ and $p \geq ((1+\epsilon)k \ln n)/n$, there is a polynomial-time algorithm to k-color a balanced semi-random k-colorable graph with high probability. But if $p < (1-\epsilon)\ln/n$, every random polynomial time algorithm will fail with high probability to k-color a balanced semi-random k-colorable graph, unless $NP \subseteq BPP$.*

Feige and Kilian (1998) also extend their semi-random analysis to the maximum independent set and Graph Bisection Problem. For the independent set problem, they develop a new two-phase algorithm. The algorithm first uses semidefinite programming as a tool to compute a constant number of nearly independent sets. It then uses a matching based optimization technique to purify the output of Phase I to extract the embedded maximum independent set.

Theorem 1.24 (Semi-Random Maximum Independent Set) *For any $\alpha > 0$, $\epsilon > 0$, and $p = (1+\epsilon)\ln n/\alpha n$, there is a randomized polynomial time algorithm that finds the embedded independent set of size αn in a semirandom graph with high probability.*

For graph bisection, Feige and Kilian consider the monotone adversary semirandom model and analyze the performance of a semidefinite-programming-based bisection algorithm.

Theorem 1.25 (Semi-Random Bisection) *There exists a constant c such that for any $1 \geq p > q$ satisfying $p - q \geq c\sqrt{p \log n/n}$, with high probability, Feige-Kilian's algorithm finds the embedded bisection in a semi-random graph in polynomial time.*

Feige and Krauthgamer (2002) examine the Bandwidth Problem:. One is given a undirected graph $G = (V, E)$ of n vertices and is asked to find a linear ordering π from V onto $\{1, \ldots, n\}$ to minimize the *bandwidth*: $\max\{|\pi(u) - \pi(v)| : (u, v) \in E\}$.

In the semi-random model, using parameters B and p, a graph is generated by choosing a linear ordering π of V and selecting each pair (u, v) satisfying $|\pi(u) - \pi(v)| \leq B$ with probability p. Then the monotone adversary may add a set of pairs (w, z) satisfying $|\pi(w) - \pi(z)| \leq B$.

Theorem 1.26 (Bandwidth Problem) *There exists a constant c such that for any ϵ, B, and p satisfying $\ln^2 n \leq Bn/\ln n$ and $p \geq c \ln n/B$, with high probability, Feige and Krauthgamer's algorithm, in polynomial*

time, returns a linear ordering of vertices with bandwidth $(1+\epsilon)B$ *for semirandom graphs.*

Results in the Property-Preserving Model

Spielman and Teng (2003a) prove that under property-preserving perturbations, several property-testing algorithms become sublinear-time decision algorithms with low smoothed error probability.

In a decision problem of a property P over an instance domain D_n, one is given an instance $x \in D_n$ and is asked to decide whether $P(x)$ is true or false. In graph theory, some graph properties such as Bipartite, the property of being bipartite, have a polynomial-time decision procedure. Other properties such as ρ-Clique, the property of having a clique of size ρn and ρ-Bisection, the property of having a bisection of at most ρn^2 edges, have no polynomial-time decision procedure unless NP = RP.

Rubinfeld and Sudan (1996) introduce *property testing*, a relaxation of the standard decision problem. The objective of property testing is to efficiently distinguish those instances that have a property from those that are "far" from having the property. An algorithm A is said to be a property tester for the property P with parameter $\epsilon \geq 0$ if

(i) for all x with property P, then $\Pr[A(x,\epsilon)=1] \geq 2/3$; and
(ii) for all x of distance at least ϵ from every instance that has property P, $\Pr[A(x,\epsilon)=1] \leq 1/3$,

under some appropriate measure of distance on inputs. It follows from this definition that a property testing algorithm A satisfies

$$\Pr[A(X,\epsilon) \neq P(X)] < 1/3$$

for all instances that have property P and for all instances that are at least ϵ distance from those with property P. For graphs, one possible measure of the distance between two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ on n vertices is the fraction of edges on which G_1 and G_2 differ: $|(E_1 \cup E_2) \setminus (E_1 \cap E_2)| / \binom{n}{2}$.

A typical property-testing algorithm will use a randomized process to choose a small number of facets of x to examine, and then make its decision. For example, a property tester for a graph property may query whether or not certain edges exist in the graph. The quality of a property-testing algorithm is measured by its query complexity (the number of queries to the input) and its time complexity.

Under this relaxation, many properties can be tested by sub-linear algorithms that examine random portions of their input. For example, Goldreich, Goldwasser, and Ron (1998) prove the following theorem.

Theorem 1.27 (Goldreich-Goldwasser-Ron) *The properties ρ-Clique and ρ-Bisection have property-testing algorithms with query complexity polynomial in $1/\epsilon$ and time complexity $2^{\tilde{O}(1/\epsilon^3)}$, and the property Bipartite has a property testing algorithm with query and time complexities polynomial in $1/\epsilon$.*

Let P be a graph property. Let $\mathcal{P}_P(X, \sigma)$ be the distribution of P-preserving σ-perturbations of X. Spielman and Teng (2003a) use the following lemma to relate the smoothed error probability of using a testing algorithm for P as a decision procedure with the probability that a P-preserving perturbed instance is far from one having property P.

Lemma 1.2 (Testing for Decision: Smoothed Error Probability) *Let P be a property and $\mathcal{P}(\bar{X}, \sigma)$ be a family of distributions satisfying for all $\bar{X}$ without property P,*

$$\Pr_{X \leftarrow \mathcal{P}(\bar{X},\sigma)} \left[X \text{ is } \epsilon\text{-close to } P | P(X) = P(\bar{X}) \right] \leq \lambda(\epsilon, \sigma, n).$$

Then for every P-testing algorithm A and every input $\bar{X}$,

$$\Pr_{X \leftarrow \mathcal{P}(\bar{X},\sigma)} \left[A(X) \neq P(X) | P(X) = P(\bar{X}) \right] < 1/3 + \lambda(\epsilon, \sigma, n).$$

Spielman and Teng (2003a) show that for any graph $\bar{G}$ and for $P \in$ {Bipartite, ρ-Clique, ρ-Bisection}, then $\bar{G}$ not satisfying P implies that (for any ϵ for the first property, and $\epsilon < \sigma(1/4 - 2\rho)$ for the last two properties),

$$\Pr_{G \leftarrow \mathcal{P}_P(\bar{G},\sigma)}[G \text{ is } \epsilon \text{ close to a graph with property } P] < 2^{-\Omega(n^2)}.$$

Clearly, if $\bar{G}$ satisfies P then G also satisfies P. Therefore,

Theorem 1.28 (Testing for Decision: Smoothed Perspective) *There exists an algorithm A that takes as input a graph G, examines $poly(1/\sigma)$ edges of G and runs in time $\tilde{O}(1/\epsilon^3)$ when P is Bipartite, and in $2^{\tilde{O}(1/\epsilon^2)}$ time when P is ρ-Clique or ρ-Bisection such that for every $\bar{G}$, if G is a P-property preserving σ-perturbation of $\bar{G}$, then*

$$\Pr\left[A(G) \neq P(G)\right] < 1/3 + o(1).$$

Applying the techniques developed by Feige and Kilian (1998) Spielman and Teng (2003a) demonstrate a testing algorithm with faster smoothed complexity for ρ-Clique.

Theorem 1.29 (Fast Clique Tester) *Let ρ and $\sigma < 1/2$ be constants. For any graph $\bar{G}$, let G be a ρ-Clique preserving σ-perturbation of $\bar{G}$. Then, there exists an algorithm A that examines the induced subgraph of G on a randomly chosen set of $\frac{8}{\rho\sigma}\log\left(\frac{4}{\rho\sigma}\right)$ vertices and runs in time polynomial in $\frac{1}{\rho\sigma}$ to achieve*

$$\Pr\left[A(G) \neq \rho\text{-}Clique(G)\right] < 1/4 + o(1).$$

Comparison of Perturbation Models

Suppose we have two models M_1 and M_2 for measuring a performance quality Q of an algorithm A. Then, M_1 is said to be *more adversarial*† than M_2 if $M_1(Q, A, \sigma) \geq M_2(Q, A, \sigma)$. For example, the worst-case measure is more adversarial than the smoothed measure and than an average measure. If M_1 is more adversarial than M_2, then any upper bound for model M_1 is also an upper bound for M_2. Conversely, any lower bound for M_2 is an lower bound for M_1.

As noted in Blum & Spencer (1995) and Feige & Kilian (1998), the monotone adversary semi-random model is at least as adversarial as the semi-random model. In turn, the semi-random model is more adversarial than the semi-random model with an oblivious adversary.

However, results for the semi-random model may not always extend to the property-preserving perturbation model, even though these two are seemingly related. This is partially due to the fact that the semi-random model only produces "positive" instances which share a common, planted solution. For example, for the k-coloring problem, the semi-random model only generates graphs that are k-colorable and all graphs from this distribution share a common, planted k coloring. Thus, this model does not assign probabilities to graphs without this planted solution. As some of these "unassigned" graphs are still k-colorable, results in the semi-random model may not provide any performance information on them. If we want to extend a result from the semi-random

† One might extend our standard complexity notions such as O, o, Ω, and Θ for models of measures. For example, we could say $M_1 = \Theta(M_2)$ if there exist constants n_0, σ_0, and $C_1 \leq C_2$ such that for all $n \geq n_0$ and $\sigma \leq \sigma_0$, $C_1 M_2(Q, A, \sigma) \leq M_1(Q, A, \sigma) \leq C_2 M_2(Q, A, \sigma)$ when input size is n.

model to the smoothed model with property-preserving perturbations, we need to understand the contribution of these instances. For some graph problems, such as the bisection problem, it remains open whether results in the semi-random model still hold in the property-preserving perturbation model. In fact, it is not even clear whether the results would extend to the more closely related solution-preserving perturbation model.

Finding a proper perturbation model for modeling practical discrete algorithms can be a challenging task. For certain graph problems such as coloring and bisection, the semi-random model defines a family of distributions using a random process that can be efficiently implemented. In contrast, the conditional density of an instance in the property-preserving distribution might be hard to compute in polynomial time. As argued in Spielman & Teng (2003a), the "practical" distributions may not have efficient descriptions. So the requirement that a perturbation model be efficiently implementable might not be reasonable or necessary. Of course, if one would like to conduct some experimental studies of these models, it would be helpful if one had an efficient procedure for the perturbation.

For certain graph properties such as planarity, it is relatively hard to define a semi-random model to study them. For more complex studies, such as those on the performance of a planar bisection algorithm, one might need to be more creative to define a suitable semi-random model. Being able to model these studies might be the strength of the property-preserving perturbation framework.

Number of Left-to-Right Maxima and Height of Binary Search Trees

For a sequence $\mathbf{a} = (a_1, \ldots, a_n)$, an element a_i is a *left-to-right maximum* of $\mathbf{a}$ if $a_i > a_j$, for all $j < i$. When $\mathbf{a} = (1, 2, ..., n)$, the number of left-to-right maxima is n. Manthey and Reischuk (2005) prove the following bounds that improve an early result of Banderier, Beier, and Mehlhorn (2003).

Theorem 1.30 (Manthey-Reischuk) *For any sequence $\bar{\mathbf{a}}$, let $\mathbf{a}$ be an σ-partial permutation of $\bar{\mathbf{a}}$. Let $\phi(\mathbf{a})$ be the number of left-to-right maxima of $\mathbf{a}$. Then for all $0 < \sigma < 1$,*

$$0.4(1-\sigma)\sqrt{n/\sigma} \leq \mathrm{E}\left[\phi(\mathbf{a})\right] \leq 3.6(1-\sigma)\sqrt{n/\sigma}.$$

The most commonly used data structure for storing a set whose elements have a total ordering is the binary search tree. Perhaps the simplest way to form a binary search tree for a sequence $\mathbf{a} = (a_1, \ldots, a_n)$ is to insert elements of $\mathbf{a}$ one by one into an initially empty binary search tree. For $i = 2$ to n, we insert a_i into the binary search tree T_{i-1} of $\{a_1, ..., a_{i-1}\}$ to produce tree T_i. An important parameter of this data structure is the height of the tree. Let $T(\mathbf{a})$ denote the binary search tree so constructed. If $\mathbf{a}$ is sorted, either in increasing or decreasing order, then the height of $T(\mathbf{a})$ is n. Manthey and Reischuk (2005) prove the following result.

Theorem 1.31 (Manthey-Reischuk) *For any sequence $\bar{\mathbf{a}}$, let $\mathbf{a}$ be an σ-partial permutation of $\bar{\mathbf{a}}$. Let $h(\mathbf{a})$ be the height of the binary search tree $T(\mathbf{a})$. Then for all $0 < \sigma < 1$,*

$$0.8(1-\sigma)\sqrt{n/\sigma} \leq \mathrm{E}\left[h(\mathbf{a})\right] \leq 6.7(1-\sigma)\sqrt{n/\sigma}.$$

Number of Extreme Points in d Dimensions

Recall that the vertices of the convex hull of a set P of points in $\mathbb{R}^d$ are points in P that cannot be expressed as a convex combination of other points in P. These points are also called the *extreme points* of P. Let $v(P)$ be the number of extreme points of P. A well known result (Bentley et al. (1978)) in geometry states: If P is a set of n points chosen uniformly at random from the unit d-cube, then the expected value of $v(P)$ is $O(\log^{d-1} n)$.

Damerow and Sohler (2004) consider the following version of the smoothed value of $v(P)$ and prove the following theorem.

Theorem 1.32 (Damerow-Sohler) *For any set $\bar{P} = \{\bar{\mathbf{p}}_1, ..., \bar{\mathbf{p}}_n\}$ of n points from the unit d-cube, let $\mathbf{r}_1, ..., \mathbf{r}_n$ be n vectors chosen uniformly at random from the cube $[-\epsilon, \epsilon]^d$. Let $P = \{\bar{\mathbf{p}}_1 + \mathbf{r}_1, ..., \bar{\mathbf{p}}_n + \mathbf{r}_n\}$ be an perturbation of $\bar{P}$. Then*

$$\mathrm{E}\left[v(P)\right] = O\left(\left(\frac{n \log n}{\epsilon}\right)^{1-1/(d+1)}\right).$$

Motion Complexity

Kinetic data structures have become subjects of active research in computational geometry since their introduction by Basch, Guibas, and

Hershberger (1997). The aim of these dynamic data structures is to efficiently maintain combinatorial and geometric information of continuously moving objects. The motion of objects is typically specified by some algebraic function. The complexity of the kinetic data structures depends on the initial positions of objects as well as the functions that govern their motion.

As a first step to study the smoothed complexity of kinetic data structures, Damerow, Meyer auf der Heide, Räcke, Scheideler, and Sohler (2003) consider the following problem: Given a set $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ of n points in $\mathbb{R}^d$ and a set $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ of initial velocity vectors, the position of the i^{th} object at time t is $\mathbf{p}_i(t) = \mathbf{p}_i + t\mathbf{v}_i$. The *motion complexity* for maintaining the orthogonal bounding box is defined to be the number of combinatorial changes of the $2d$ sides of the bounding box.

In the worst case, this motion complexity is $\Omega(dn)$. When $\mathbf{v}_i$ and $\mathbf{p}_i$ are chosen uniformly at random from the unit cube $[-1,1]^d$, the average complexity is $O(d \log n)$. Damerow et al. prove the following result.

Theorem 1.33 (Smoothed Motion Complexity) *For any positive integer d, let $\bar{P} = \{\bar{\mathbf{p}}_1, \ldots, \bar{\mathbf{p}}_n\} \subset [-1,1]^d$ and $\bar{V} = \{\bar{\mathbf{v}}_1, \ldots, \bar{\mathbf{v}}_n\} \subset [-1,1]^d$. Let $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ and $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ be σ-Gaussian perturbations of $\bar{P}$ and $\bar{V}$, respectively. Then the expected motion complexity for maintaining the orthogonal bounding box of (P, V) is $O(d(1 + 1/\sigma)\log^{1.5} n)$ and $\Omega(d\sqrt{\log n})$.*

Properties of Perturbed Graphs and Formula

Each perturbation model in the smoothed analysis of graph algorithms can be used to define a distribution of random graphs. For example, let $H = (V, E)$ be an undirected graph over vertices $V = \{1, ..., n\}$. For any $\sigma < 1$, the σ-perturbations of H can be viewed as a distribution of random graphs. Let us refer it as $G_{H,\sigma}$. Similarly, for any graph property P, the P-preserving σ-perturbations of a graph H is also a distribution of random graphs, denoted by $G_{P,H,\sigma}$.

For any m and $H = (V, E)$, Bohman, Frieze and Martin (2003) define a distribution $G_{H,m}$ of random graphs $(V, E \cup R)$ where R is a set of m edges chosen uniformly at random from the complement of E, i.e., chosen from $\bar{E} = \{(i,j) \notin E\}$.

Krivelevich, Sudakov, and Tetali (2005) prove a sharp threshold for the appearance of a fixed subgraph. For a fixed graph Γ with n_Γ ver-

tices and e_Γ edges, let $m(\Gamma) = \max\{e_{\Gamma'}/n_{\Gamma'} | \Gamma' \subseteq \Gamma, n_{\Gamma'} > 0\}$. For any positive integer r, let

$$m_r(\Gamma) = \min_{r\text{-way partition } (V_1,...,V_r) \text{ of } V(\Gamma)} \max_{|V_i|>0} m(\Gamma(V_i))$$

Theorem 1.34 (Emerging of a Subgraph) *Let r and α be constants such that $r \geq 2$ and $\alpha \in \left(\frac{r-2}{r-1}, \frac{r-1}{r}\right]$. Let Γ be a graph of no more than n vertices. Then, for every graph H over $\{1,...,n\}$ of average degree αn, where $m = \omega(n^{2-1/m_r(\Gamma)})$, $G_{H,m}$ almost surely contains a copy of Γ. Moreover, there exists a graph H' of average degree αn such that if $m = o(n^{2-1/m_r(\Gamma)})$, then $G_{H',m}$ almost surely does not contain a copy of Γ. Here $f = \omega(g)$ is equivalent to $g = o(f)$.*

Krievlevich, Sudakov, and Tetalli also consider the problem of adding m randomly chosen disjunctions of k literals to a satisfiable instance, F, of a k-SAT formula on n variables. They obtain the following result.

Theorem 1.35 (Transition From Happy to Unhappy) *For some $c > 0$ and $\epsilon < 1/k$, let F be a satisfiable k-SAT formula on n variables with at least $cn^{k-\epsilon}$ clauses. Then almost surely the conjunction of F with a randomly chosen k-SAT formula of $m = \omega(n^{k\epsilon})$ clauses is not satisfiable. Moreover, there exists a satisfiable k-SAT formula F' of $\Omega(n^{k-\epsilon})$ clauses such that the conjunction of F with a randomly chosen k-SAT formula of $m = o(n^{k\epsilon})$ clauses is satisfiable with high probability.*

Several other discrete properties have been studied in the smoothed setting: for example, Flaxman and Frieze (2004) consider the diameter of perturbed digraphs and Sudakov and Vondrak (2005) examine the 2-colorability of dense hypergraphs.

Very recently, Arthur and Vassilvitskii (2005) analyze the smoothed iteration complexity of the k-means method (Lloyd (1982)) for clustering: Given a set P of n points in $\mathbb{R}^d$, the k-means method first chooses an arbitrary set of k centers and uses the Voronoi diagram of these centers to partition P into k clusters. Then the center-of-mass of each of these clusters becomes a new center, and the k-means method re-clusters the points and repeats this process until it stabilizes. They show that in the worst-case, the k-means method requires $2^{\Omega(\sqrt{n})}$ iterations to converge. In contrast, they prove that the k-means method has probably polynomial smoothed complexity.

1.4 Open Problems

In this section, we discuss some emerging open questions in smoothed analysis. For some problems and algorithms, we will present concrete conjectures, whereas for others, we will discuss possible directions for research.

P1: The Simplex Method and its Pivoting Rules

During a Phase II iteration of a simplex algorithm, a vertex $\mathbf{v}_{i-1}$ may have several neighboring vertices with better objective values. The algorithm needs to decide which of them should be chosen as the next vertex $\mathbf{v}_i$. Simplex algorithms differ by their pivoting rules which guide this decision when there are multiple choices. Several pivoting rules have been proposed in the literature (Dantzig (1991) and Chvatal (1983)). They include

- *Greedy*: Choose the vertex that maximizes the improvement of the objective value, $\mathbf{c}^T(\mathbf{v}_i - \mathbf{v}_{i-1})$.
- *Steepest-Edge*: Choose the vertex $\mathbf{v}_i$ whose edge $(\mathbf{v}_{i-1}, \mathbf{v}_i)$ forms the smallest angle with $\mathbf{c}$, that is, $\mathbf{v}_i$ maximizes
$$\frac{\mathbf{c}^T(\mathbf{v}_i - \mathbf{v}_{i-1})}{||\mathbf{v}_i - \mathbf{v}_{i-1}||_2}.$$
- *Random*: Choose randomly from the neighboring vertices with better objective values according to some distribution.

There are other rules such as the shadow-vertex rule (Gass & Saaty (1955)), Bland's rule (1977) and the self-dual simplex rule (Dantzig (1991) and Lemke (1965)).

The steepest-edge rule (Forrest & Goldfarb (1992)) is among the most commonly used in practice. But most existing pivoting rules have been shown to have exponential worst-case complexity (Klee & Minty (1972), Goldfarb & Sit (1979), Goldfarb (1983), Jeroslow (1973), Murty (1980), Amenta & Ziegler (1999)).

A natural open question is whether our results such as Theorem 1.1 on the smoothed complexity of the shadow-vertex rule can be extended to other pivoting rules.

Conjecture 1 (Smoothed Simplex Conjecture) *The smoothed complexity of the simplex algorithms with greedy, steepest-descent, and random pivoting rules are polynomial under Gaussian perturbations.*

The key step for proving Conjecture 1 is the Phase II complexity, since Phase I computation can often be reduced to a Phase II computation in which we already know that the constraints are feasible and have an initial vertex of the feasible region. Like the shadow-vertex simplex algorithm, these simplex algorithms iteratively produce a path of vertices of the feasible region that monotonically improves the objective value. We refer to such a path as a $\mathbf{c}$-*monotonic path* of the feasible polyhedron, recalling that $\mathbf{c}$ is the objective direction.

There is a geometric reason, as pointed out in Spielman & Teng (2004), that the shadow-vertex simplex algorithm is the first simplex algorithm considered in both the average-case (Borgwardt (1980)) and smoothed analyses: The length of the $\mathbf{c}$-monotonic path constructed by the shadow-vertex algorithm can be bounded above by the size of the shadow. Moreover, the shadow depends only on the initial vertex and the objective direction. So in the probabilistic analysis, we do not need to explicitly consider the iterative steps taken by the simplex algorithm.

The key to analyze the smoothed complexity of simplex algorithms with, say, the steepest-edge pivoting rule is to search for a simpler geometric parameter that upper bounds the length of its $\mathbf{c}$-monotonic path. We could start with linear programs of form

$$\max \quad \mathbf{c}^T\mathbf{x} \qquad \text{subject to } \mathbf{A}\mathbf{x} \leq \mathbf{1}.$$

One approach is to relate the length of the steepest-edge $\mathbf{c}$-monotonic path with the size of the shadow analyzed in Theorem 1.1.

Conjecture 2 (Shadow and Steepest-Edge Path: I) *For any $m \times n$ matrix $\bar{\mathbf{A}}$ and an n-vector $\bar{\mathbf{c}}$ such that $\left\|\bar{\mathbf{A}}, \bar{\mathbf{c}}\right\|_F \leq 1$, let $\mathbf{A}$ and $\mathbf{c}$ be σ-Gaussian perturbations of $\bar{\mathbf{A}}$ and $\bar{\mathbf{c}}$, respectively. For an m-vector $\mathbf{z}$, let $\mathbf{v}$ be the vertex of the polyhedron $\{\mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{1}\}$ that maximizes $\mathbf{z}^T\mathbf{v}$ and let P be the steepest-edge $\mathbf{c}$-monotonic path of the polyhedron $\{\mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{1}\}$ starting from $\mathbf{v}$. Then, there exists a constant $\alpha \geq 1$ and an $x_0(m, n, 1/\sigma)$ polynomial in m,n, and $1/\sigma$, such that for all $x > x_0(m, n, 1/\sigma)$*

$$\Pr\left[|P| > x\right] \leq \alpha \cdot \Pr\left[|\mathbf{Shadow}_{\mathbf{z},\mathbf{c}}\left(\mathbf{A}\right)| > x\right].$$

Conjecture 3 (Shadow and Steepest-Edge Path: II) *Let $\mathbf{A}$, $\mathbf{c}$, $\mathbf{z}$, P be the same as defined in Conjecture 2. Then, there exists an*

$h(m,n,1/\sigma)$ polynomial in m,n, and $1/\sigma$ and constants k_1 and k_2

$$\Pr\left[|P| > h(m,n,1/\sigma)\cdot|\mathbf{Shadow}_{\mathbf{z},\mathbf{c}}(\mathbf{A})|\right] = \frac{1}{m^{k_1}n^{k_2}}.$$

One can try to prove similar conjectures for the greedy **c**-monotonic path.

One might, of course, consider more direct approaches to solve the challenge posted by iterative methods. The main difficulty in conducting probabilistic analyses for iterative methods is not unique to the simplex method. In the study of the growth factor of Gaussian elimination with partial pivoting and interior-point methods for linear and convex programming, we have been facing a similar challenge: How do we resolve the dependency of the structures before and after an iterative step? In each iterative step, the algorithm explores some fraction of data which is initially either random or subject to some random perturbations. However, this exploration of the algorithm in determining an iterative step spoils the "randomness" in the same fraction of the data. We can be lucky with some iterative algorithms, such as the shadow-vertex simplex algorithm and Gaussian elimination without pivoting. For each of these, we have found a "flat" upper bound parameter to support an iteration-free analysis of the iterative algorithm. Developing a systematic framework for probabilistic analysis of general iterative methods is a challenging and rewarding research direction.

One simplex algorithm that comes with a flat geometric characterization is Lemke's self-dual parametric simplex algorithm (Dantzig (1991) and Lemke (1965)). Polynomial expected bound was established for a lexicographic variant of the self-dual model for solving a random linear program whose matrices are drawn from a spherically-symmetric distribution (Adler, Karp, & Shamir (1987), Adler & Megiddo (1985), and Todd (1986)).

Conjecture 4 (Lemke's Self-Dual Parametric Simplex Algorithm) *Lemke's self-dual parametric simplex algorithm has polynomial smoothed complexity under Gaussian perturbations.*

Another line of research in the smoothed analysis of the simplex algorithm is to extend the result from Gaussian perturbations to other perturbations. Recall the family of distributions introduced by Beier and Vöcking (2004): Let f be a piece-wise continuous univariate density function with finite mean $\int_{\mathbb{R}}|x|\,f(x)dx$ and $\sup_x f(x) = 1$. For $\sigma \leq 1$, let f_σ be a scaling of f such that $f_\sigma(x) = f(x/\sigma)/\sigma$. For any $\bar{x}$, an

f-perturbation with magnitude $\sigma < 1$ is a random variable $x = \bar{x} + r$, where r is randomly chosen according to density f_σ.

Conjecture 5 (Perturbations and Simplex Methods) *For any piece-wise continuous univariate density function f with finite mean and $\sup_x f(x) = 1$, there exists a σ_0 such that for any $0 \leq \sigma \leq \sigma_0$ the simplex algorithm with the shadow-vertex pivoting rule has smoothed time complexity polynomial in m, n, and $1/\sigma$ under f-perturbations with magnitude σ.*

P2: Smoothed Hirsch Conjecture

The Hirsch conjecture states that for any convex polytope of m facets in n dimensions, the diameter of the graph induced by the 1-skeleton of the polytope is bounded above by $(m-n)$. In other words, any two vertices on the polytope are connected by a path of length at most $(m-n)$. The best bound on the diameter of the polyhedron known today is given by Kalai and Kleitman (1992). Their bound is $m^{\log_2 n+1}$. In the smoothed setting, we would like to prove the following conjecture.

Conjecture 6 (Smoothed Hirsch Conjecture) *For any $m \times n$ matrix $\bar{\mathbf{A}}$ and any m-vector $\bar{\mathbf{b}} \in \mathbb{R}^m$ such that $\left\|\bar{\mathbf{A}}\right\|_F \leq 1$ and $\left\|\bar{\mathbf{b}}\right\|_2 \leq 1$, let $\mathbf{A}$ and $\mathbf{b}$ be σ-Gaussian perturbations of $\bar{\mathbf{A}}$ and $\bar{\mathbf{b}}$, respectively. If $\mathbf{Ax} \leq \mathbf{b}$ is feasible, then the expected diameter of the polyhedron defined by $\mathbf{Ax} \leq \mathbf{b}$ is polynomial in m, n, and $1/\sigma$.*

A special case of the Smoothed Hirsch Conjecture closely related with Theorem 1.2 is:

Conjecture 7 (Smoothed Hirsch Conjecture: Special Case) *For any $m \times n$ matrix $\bar{\mathbf{A}}$ such that $\left\|\bar{\mathbf{A}}\right\|_F \leq 1$, let $\mathbf{A}$ be a σ-Gaussian perturbation of $\bar{\mathbf{A}}$. Then the expected diameter of the polyhedron defined by $\mathbf{Ax} \leq \mathbf{1}$ is polynomial in m, n, and $1/\sigma$.*

Note that Conjecture 7 does not directly follow from Theorem 1.2, although the latter states that on the polyhedron $\{\mathbf{x} \mid \mathbf{Ax} \leq \mathbf{1}\}$, for any two n-vectors $\mathbf{c}$ and $\mathbf{z}$, the expected length of the shortest path connecting the vertex optimized by $\mathbf{c}$ and the vertex optimized by $\mathbf{z}$ is polynomial in m, n and $1/\sigma$. There could be an exponential number of pairs of vertices on the polyhedron. Without using additional geometric

properties, the probability bound in Theorem 1.2 may not be strong enough.

P3: Number of Iterations of IPM and Parallel Linear Programming Algorithms

Several interior-point methods take $O(\sqrt{n} \cdot L)$ number of iterations to solve a linear program. Some experiments suggest that the number of iterations could be as low as $O\left(\log^2(mn)\right)$ when long-step interior-point methods are used. Although progress has been made in recent years concerning the convergence of these methods, it remains open whether, in the worst-case, these long-step methods are better than the more prudent short-step methods (Wright (1997) and Ye (1997)). If practical observations are any indication (Todd (1991)), then the following conjecture could be true.

Conjecture 8 (IPM: Optimistic Convergence) *Under Gaussian perturbations, there is an interior-point algorithm with smoothed iteration complexity* $O\left(\log^2(mn/\sigma)\right)$ *for solving linear programming with input dimensions* $m \times n$.

The best worst-case iteration lower bound is $\Omega(n^{1/3})$ due to Todd (1994) and Todd and Ye (1996). However, the programs for which these lower bounds hold are very ill-conditioned. Dunagan, Spielman, and Teng (2002) observe that small perturbations could improve the condition numbers of the resulting linear program. Thus, the lower bound of Todd-Ye does not hold in the smoothed model.

There has not yet been a lower bound of $\Omega(n^{\epsilon})$ for well-conditioned linear programs. We would like to know whether such a lower bound holds for well-conditioned programs, or whether there are interior-point algorithms that take fewer iterations when their input is well-conditioned. Perhaps one could start with the following weaker version of Conjecture 8:

Conjecture 9 (IPM: First step?) *There is an interior-point algorithm with smoothed iteration complexity* $O\left(n^{\epsilon}\log^2(mn/\sigma)\right)$ *for solving linear programs, for some* $\epsilon < 1/2$.

A closely related theoretical question is the smoothed parallel complexity of linear programming. In the worst-case, linear programming is complete for P under log-space reductions (Dobkin, Lipton & Reiss

(1979)). In other words, it is unlikely that one can solve linear programs in poly-logarithmic time using a polynomial number of processors. Solving linear programming approximately is also complete for P (Serna (1991) and Megiddo (1992)).

If Conjecture 8 is true, then linear programming has a *smoothed NC* algorithm – one can solve a linear program in poly-logarithmic (in m, n, and $1/\sigma$)) time using a polynomial number of processors. One might relax the poly-logarithmic dependency on $1/\sigma$ and conjecture that:

Conjecture 10 (Parallel Linear Programming) *There exists a linear programming algorithm with smoothed parallel complexity, under σ-Gaussian perturbations, $O\left(\sigma^{-k_1}\log^{k_2}(mn)\right)$ for some constants k_1 and k_2.*

Luby and Nisan (1993) consider the parallel complexity of a special family of linear programs

$$\max \quad \mathbf{c}^T\mathbf{x} \qquad \text{subject to } \mathbf{A}\mathbf{x} \leq \mathbf{b},$$

where all input entries in $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{c}$ are positive. They give a parallel algorithm that, for any $\epsilon > 0$, finds a $(1+\epsilon)$-approximate solution in time polynomial in $1/\epsilon$ and $\log(mn)$ using $O(mn)$ processors. Although positive linear programming is also complete for P under log-space reductions (Trevisan & Xhafa (1998)) the following conjecture may still be true.

Conjecture 11 (Positive Linear Programming) *There exists a positive linear programming algorithm with smoothed parallel complexity $O\left(\sigma^{-k_1}\log^{k_2}(mn)\right)$ for some constant k_1 and k_2.*

The conjecture above may follow from the results of Spielman & Teng (2003b), Dunagan, Spielman, & Teng (2002) and Luby & Nisan (1993). It might be more challenging, however, to prove the following stronger conjecture:

Conjecture 12 (Positive Linear Programming: Stronger Version) *There is a positive linear programming algorithm with smoothed parallel complexity $O\left(\log^c(mn/\sigma)\right)$ for some constant c.*

We would like to conclude this subsection with what might be a simple question. Vaidya (1987) proves that any linear program $(\mathbf{A}, \mathbf{b}, \mathbf{c})$ of m constraints and n variables can be solved in $O((n+m)n^2+(n+m)^{1.5}n)L)$

time. Is it true that the L in Vaidya's bound can be replaced by $\log C(\mathbf{A}, \mathbf{b}, \mathbf{c})$? Recall that $C(\mathbf{A}, \mathbf{b}, \mathbf{c})$ is the condition number of the linear program defined by $(\mathbf{A}, \mathbf{b}, \mathbf{c})$. If true, we can apply the smoothed analysis of Dunagan, Spielman and Teng (2002) to show Vaidya's linear programming algorithm runs in $O((n+m)n^2 + (n+m)^{1.5}n)\log(mn/\sigma))$ smoothed time. Combining with Clarkson's reduction (1995), this would reduce the smoothed complexity for linear programming in low dimensions to

$$O\left(n^2 m + n^6 \log m \log(mn/\sigma)\right).$$

P4: Convex and Conic Programming

Can we extend Theorem 1.7 and Theorem 1.8 from linear programming to conic convex programming (Cucker & Peña (2001) and Freund & Vera (2000)) and to general convex programming (Nesterov & Nemirovskii (1994))? For conic convex programming, one can first consider the following form

$$\min \quad \mathbf{c}^T\mathbf{x} \qquad \text{subject to } \mathbf{Ax} - \mathbf{b} \in \mathbf{C}_1 \text{ and } \mathbf{x} \in \mathbf{C}_2$$

where $\mathbf{C}_1 \subset \mathbb{R}^m$ and $\mathbf{C}_2 \subset \mathbb{R}^n$ are closed convex cones.

The dual of this program is

$$\max \quad \mathbf{b}^T\mathbf{y} \qquad \text{subject to } \mathbf{c} - \mathbf{A}^T\mathbf{y} \in \mathbf{C}_2^* \text{ and } \mathbf{y} \in \mathbf{C}_1^*,$$

where for $i \in \{1, 2\}$, $\mathbf{C}_i^*$ is the dual cone of $\mathbf{C}_i$:

$$\mathbf{C}_i^* = \left\{\mathbf{u} : \mathbf{u}^T\mathbf{v} \geq 0 \text{ for any } \mathbf{v} \in \mathbf{C}_i.\right\}$$

Note that in this form the primal and dual programs have the same format. As the concept of the distance to ill-posedness can be easily extended from linear programs to conic programs, Renegar's condition number for linear programs naturally extends to conic programs. Similarly, it has been shown (Freund & Vera (200) and Nesterov & Nemirovskii (1994)) that the complexity of the interior-point method for conic programming depends logarithmically on the condition number of the input program.

For a convex body $\mathbf{K}$ in $\mathbb{R}^d$, let $\partial\mathbf{K}$ be its boundary. For any $\epsilon \geq 0$, let

$$\partial\left(\mathbf{K}, \epsilon\right) = \{\mathbf{x} : \exists \mathbf{x}' \in \partial\mathbf{K}, \|\mathbf{x} - \mathbf{x}'\|_2 \leq \epsilon\}$$

The proof of Theorem 1.7 uses the following key probabilistic bound of Ball (1993) in convex geometry.

Theorem 1.36 (Ball (1993)) *Let μ be the density function of a n-dimensional Gaussian random vector with center $\mathbf{0}$ and variance σ^2. Then for any convex body $\mathbf{K}$ in $\mathbb{R}^n$,*

$$\int_{\partial \mathbf{K}} \mu \leq 4n^{1/4}.$$

The smoothed analysis of Dunagan, Spielman and Teng (2002) applied the following corollary of Theorem 1.36 to estimate the probability that a perturbed linear program is poorly conditioned.

Corollary 1.1 (Darting the boundary of a convex set) *For a vector $\bar{\mathbf{x}} \in \mathbb{R}^n$, let $\mathbf{x}$ be a σ-Gaussian perturbation of $\bar{\mathbf{x}}$. Then for any convex body $\mathbf{K}$ in $\mathbb{R}^n$,*

$$\Pr_{\mathbf{x}}\left[\mathbf{x} \in \partial\left(\mathbf{K}, \epsilon\right) \setminus \mathbf{K}\right] \leq \frac{4n^{1/4}\epsilon}{\sigma} \text{ (outside boundary),}$$

$$\Pr_{\mathbf{x}}\left[\mathbf{x} \in \partial\left(\mathbf{K}, \epsilon\right) \cap \mathbf{K}\right] \leq \frac{4n^{1/4}\epsilon}{\sigma} \text{ (inside boundary).}$$

Proving the following conjecture would allow us to extend Theorem 1.7 and Theorem 1.8 to conic convex programming.

Conjecture 13 (Linear Transformation of Convex Cones) *For any convex cone $\mathbf{K}$ in $\mathbb{R}^n$, let $\mathbf{A}$ be a σ-Gaussian perturbation of an $m \times n$ matrix $\bar{\mathbf{A}}$ with $\left\|\bar{\mathbf{A}}\right\|_F \leq 1$. Then, there exist constants σ_0, m_0, n_0, c, k_1, k_2, and k_3 such that for any convex cone $\mathbf{C}$ with angle Θ, $n \geq n_0$, $m \geq m_0$, and $0 \leq \sigma \leq \sigma_0$*

$$\Pr\left[(\mathbf{A} \cdot \mathbf{K}) \cap \mathbf{C} = \emptyset \ \mathit{\&}\ (\mathbf{A} \cdot \mathbf{K}) \cap \partial\left(\mathbf{C}, \epsilon\right) \neq \emptyset\right] \leq c \cdot m^{k_1} n^{k_2} \left(\frac{\epsilon}{\sigma}\right)^{k_3}, \text{ and}$$

$$\Pr\left[(\mathbf{A} \cdot \mathbf{K}) \cap \mathbf{C} \neq \emptyset \ \mathit{\&}\ ((\mathbf{A} \cdot \mathbf{K}) \cap \mathbf{C}) \subseteq \partial\left(\mathbf{C}, \epsilon \cdot \Theta\right)\right] \leq c \cdot m^{k_1} n^{k_2} \left(\frac{\epsilon}{\sigma}\right)^{k_3}.$$

Note that when $\mathbf{K}$ is a single vector, $(\mathbf{A} \cdot \mathbf{K})$ is a Gaussian perturbation of the vector $(\bar{\mathbf{A}} \cdot \mathbf{K})$. Thus, in this case, Conjecture 13 is a special case of Corollary 1.1 and hence is true.

Conjecture 14 (Smoothed condition number of conic programming) *For any $(\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\mathbf{c}})$ and $\sigma \leq 1$, let $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{c}$ be σ-Gaussian perturbations of $\bar{\mathbf{A}}$, $\bar{\mathbf{b}}$ and $\bar{\mathbf{c}}$. Then, for any closed convex cones $\mathbf{C}_1$ and $\mathbf{C}_2$, the expectation of the the logarithm of the condition of the conic program defined by $\mathbf{A}, \mathbf{b}, \mathbf{c}$ together with $\mathbf{C}_1$ and $\mathbf{C}_2$ is $O(\log(mn/\sigma))$.*

An important family of conic convex programming problems is semi-definite programming (SDP). The standard primal form of a semi-definite program is (Todd (2001))

$$\max_{\mathbb{X}} \quad \mathbf{C} \bullet \mathbb{X} \qquad \text{subject to } \mathbf{A}_i \bullet \mathbb{X} = b_i,\ i = 1, \ldots, m \text{ and } \mathbb{X} \succeq \mathbf{0}$$

where $\mathbf{C}$, $\mathbf{A}_i$, and $\mathbb{X}$ are symmetric matrices and $\mathbb{X}$ is required to be positive semi-definite. Because the set of positive semi-definite matrices forms a convex cone, a semi-definite program is a conic convex program. One can define the condition number of a semi-definition program for $(\mathbf{C}, \{\mathbf{A}_i\}, \mathbf{b})$. We conjecture that the expectation of the the logarithm of this condition number is $O(\log(mn/\sigma))$ under Gaussian perturbations.

Even though every convex optimization problem can be transformed into an equivalent instance of conic programming, the transformation, like those among normal forms of linear programming, may not preserve the condition number of the programs. Freund and Ordóñez (2005) explicitly consider the condition number of convex programming in the following non-conic form:

$$\min \quad \mathbf{c}^T\mathbf{x} \qquad \text{subject to } \mathbf{Ax} - \mathbf{b} \in \mathbf{C} \text{ and } \mathbf{x} \in \mathbf{K}$$

where $\mathbf{C} \subset \mathbb{R}^m$ is a convex cone while $\mathbf{K}$ could be any closed convex set (including a cone).

It would be interesting to understand the smoothed behavior of the condition number of convex programs in this form.

P5: Two-Person Games and Multi-Person Games

A two-person game or bimatrix game (Nash (1951) and Lemke (1965)) is specified by two $m \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$, where the m rows represent the pure strategies for the first player, and the n columns represent the pure strategies for the second player. In other words, if the first player chooses strategy i and the second player chooses strategy j, then the payoffs to the first and the second players are a_{ij} and b_{ij}, respectively.

Nash's theorem (1951) on non-cooperative games when specialized to two-person games states that there exists a profile of possibly mixed strategies so that neither player can gain by changing his/her (mixed) strategy, while the other player stays put. Such a profile of strategies is called a *Nash equilibrium*.

Mathematically, a mixed strategy for the first player can be expressed by a column probability vector $\mathbf{x} \in \mathbb{R}^m$, that is, a vector with non-negative entries that sum to 1, while a mixed strategy for the second

player is a probability vector $\mathbf{y} \in \mathbb{R}^n$. A Nash equilibrium is then a pair of probability vectors $(\mathbf{x}, \mathbf{y})$ such that for all probabilities vectors $\mathbf{x}' \in \mathbb{R}^m$ and $\mathbf{y}' \in \mathbb{R}^n$,

$$\mathbf{x}^T\mathbf{A}\mathbf{y} \geq (\mathbf{x}')^T\mathbf{A}\mathbf{y} \quad \text{and} \quad \mathbf{x}^T\mathbf{B}\mathbf{y} \geq \mathbf{x}^T\mathbf{B}\mathbf{y}'.$$

The complexity of finding a Nash equilibrium of a two-person game remains open and is considered to be a major open question in theoretical computer science. Recently, Savani and von Stengel (2004) show that the classical Lemke-Howson algorithm (1964) needs an exponential number of steps in the worst case.

In smoothed analysis with Gaussian perturbations, we assume the payoff matrices $\mathbf{A}$ and $\mathbf{B}$ are subject to small Gaussian perturbations. The most optimistic conjecture is:

Conjecture 15 (Smoothed 2-Nash Conjecture) *The problem of finding a Nash equilibrium of a two-person game, 2-Nash, is in smoothed polynomial time under Gaussian perturbations.*

As the first step of this research, one could examine the worst-case instances of Savani-von Stengel in the smoothed model. We would like to understand whether such worst-case instances are stable when subject to perturbations. If one can build a stable instance with poor complexity for Lemke-Howson's algorithm, then its smoothed complexity would be poor.

Open Question 1 (Smoothed Complexity of Lemke-Howson) *Does Lemke-Howson's algorithm for 2-Nash have polynomial smoothed complexity under Gaussian perturbations?*

One could consider other algorithms in order to prove the Smoothed 2-Nash Conjecture. An encouraging recent development is the work of Barany, Vempala, and Vetta (2005) who show that when entries of $\mathbf{A}$ and $\mathbf{B}$ are Gaussian variables with mean 0, then 2-Nash has polynomial average-case complexity. So far, their technique does not quite apply to the smoothed model.

As a more general research direction, one can ask similar questions about other games, such as the combinatorially defined graphical games (Kearns, Littman & Singh (2001)), general multi-player finite games (Nash (1951)), or stochastic games (Jurdzinski (2005)).

The 4-person game, 4-Nash, was recently shown to be PPAD-complete

by Daskalakis, Goldberg, and Papadimitriou (2005). This result implies that 4-Nash is as hard as the computation of Brouwer fixed points.

Open Question 2 (Game Theory and Algorithms) *What is the smoothed complexity of the computation of Nash equilibria? What is the impact of perturbations to mechanism design?*

In particular,

Open Question 3 (Smoothed Complexity of 4-Nash) *Is 4-Nash, or 3-Nash, in smoothed polynomial time under Gaussian perturbations?*

P6: Gaussian Elimination and Condition Numbers

Several very basic questions on the stability and growth factors of Gaussian elimination remain open. In the worst-case, there are matrices for which Gaussian elimination with partial pivoting has a larger growth factor than Gaussian elimination without pivoting. Similarly, there are matrices for which Gaussian elimination with complete pivoting has a larger growth factor than Gaussian elimination with partial pivoting. Experimentally, partial pivoting has been shown to be much more stable than no pivoting but less stable than complete pivoting.

The most important open problem in the smoothed analysis of Gaussian elimination is to improve the bound of Theorem 1.16. Experimental work seems to suggest that it is exponentially unlikely that Gaussian elimination with partial pivoting has a superpolynomial growth factor.

Conjecture 16 (Exponential Stability of GEPP) *For any $n \times n$ matrix $\bar{\mathbf{A}}$ such that $\|\bar{\mathbf{A}}\|_2 \leq 1$, let $\mathbf{A}$ be a σ-Gaussian perturbation of $\bar{\mathbf{A}}$. Then, there exists constants c_1 and c_2 such that*

$$\Pr_{\mathbf{A}}\left[\rho_{GEPP}(\mathbf{A}) > x\left(\frac{n}{\sigma}\right)^{c_1}\right] \leq 2^{-c_2 x}.$$

There are several other matrix factorization methods that also enjoy practical success. One example is the Bruhat's decomposition that factors $\mathbf{A}$ as $\mathbf{A} = \mathbf{V\Pi U}$ where $\mathbf{\Pi}$ is a permutation matrix, $\mathbf{V}$ and $\mathbf{U}$ are upper triangular matrices, and $\mathbf{\Pi}^T\mathbf{V\Pi}$ is a lower triangular matrix (van den Driessche,Odeh & Olesky (1998)). Another example is the superLU algorithm developed by Li and Demmel (Li (2005)). It first permutes a matrix $\mathbf{A}$ in order to move large elements to the diagonal. A maximum

weighted matching algorithm is used for this step to produce a permutation matrix $\mathbf{P}$. The algorithm then symmetrically permutes $\mathbf{PA}$ into $\mathbf{Q}(\mathbf{PA})\mathbf{Q}^T$ to improve the sparsity for elimination. Then $\mathbf{Q}(\mathbf{PA})\mathbf{Q}^T$ is factored into $\mathbf{LU}$ using Gaussian elimination with no pivoting but with one modification: if during the elimination the current pivoting diagonal entry is smaller than $\epsilon \|\mathbf{A}\|_F$, for some ϵ, then it is replaced by $\sqrt{\epsilon}\|\mathbf{A}\|_2$ before the elimination step proceeds. To solve a linear system $\mathbf{Ax} = \mathbf{b}$, one can use this factorization to obtain an approximate solution by solving the two triangular systems, one defined by $\mathbf{U}$ and one defined by $\mathbf{L}$. Finally the algorithm may apply a few iterations to improve its solution.

Open Question 4 (Stability of Linear Solvers) *What is the smoothed performance of these practically-used factorization algorithms and linear solvers under Gaussian perturbations or under zero-preserving Gaussian perturbations?*

An alternative approach to improve the stability of LU factorization is to use randomization. For example, in the i^{th} step of elimination, instead of choosing the equation with the largest i^{th} coefficient (in absolute value) as in partial pivoting, one can select the next equation from a random distribution that depends on the magnitudes the i^{th} coefficients of the equation. Intuitively, the larger the magnitude of its i^{th} coefficient, the higher is the chance that the equation is chosen. For example, suppose the i^{th} coefficients are $a_{i,i}^{(i-1)}, ..., a_{n,i}^{(i-1)}$. For each $p > 0$, the *p-normal partial pivoting* chooses the equation with coefficient $a_{k,i}^{(i-1)}$ for $k \geq i$ with probability

$$\frac{\left(a_{k,i}^{(i-1)}\right)^p}{\left(a_{i,i}^{(i-1)}\right)^p + ... + \left(a_{n,i}^{(i-1)}\right)^p}.$$

Open Question 5 (Gaussian elimination with p-normal partial pivoting) *What is the expected growth factor of Gaussian elimination with p-normal partial pivoting?*

Is there a p such that the expected growth factor of Gaussian elimination with p-normal partial pivoting is polynomial in n?

Under Gaussian perturbations, what is the smoothed growth factor of Gaussian elimination with p-normal partial pivoting?

Does Gaussian elimination with p-normal partial pivoting have exponential stability as defined in Conjecture 16?

There are several questions still open dealing with the condition numbers and the smallest singular value of a square matrix.

Conjecture 17 (Condition Number) *Let $\bar{\mathbf{A}}$ be an $n \times n$ matrix satisfying $\|\bar{\mathbf{A}}\|_2 \leq \sqrt{n}$, and let $\mathbf{A}$ be a σ-Gaussian perturbation of $\bar{\mathbf{A}}$ for $\sigma \leq 1$. Then,*

$$\Pr\left[\kappa(\mathbf{A}) \geq x\right] \leq O\left(\frac{n}{x\sigma}\right).$$

Conjecture 18 (Smallest Singular Value) *Let $\bar{\mathbf{A}}$ be an arbitrary square matrix in $\mathbb{R}^{n\times n}$, and let $\mathbf{A}$ be a σ-Gaussian perturbation of $\bar{\mathbf{A}}$. Then*

$$\Pr_{\mathbf{A}}\left[\left\|\mathbf{A}^{-1}\right\|_2 \geq x\right] \leq \frac{\sqrt{n}}{x\sigma}.$$

In the average case where $\mathbf{G}$ is a Gaussian matrix with each of its entries an independent univariate Gaussian variable with mean 0 and standard deviation σ, Edelman (1988) proves

$$\Pr_{\mathbf{G}}\left[\left\|\mathbf{G}^{-1}\right\|_2 \geq x\right] \leq \frac{\sqrt{n}}{x\sigma}.$$

One possible way to prove Conjecture 18 would be to show that the Gaussian matrix considered by Edelman, is in fact, the worst-case distribution, as stated in the next conjecture.

Conjecture 19 (Gaussian Matrices and Gaussian Perturbations) *Let $\bar{\mathbf{A}}$ be an arbitrary square matrix in $\mathbb{R}^{n\times n}$, and let $\mathbf{A}$ be a σ-Gaussian perturbation of $\bar{\mathbf{A}}$. Let $\mathbf{G}$ be a Gaussian matrix of variance σ^2 as above. Then for all $x \geq 1$*

$$\Pr_{\mathbf{A}}\left[\left\|\mathbf{A}^{-1}\right\|_2 \geq x\right] \leq \Pr_{\mathbf{G}}\left[\left\|\mathbf{G}^{-1}\right\|_2 \geq x\right].$$

Finally, we have a conjecture on the smallest singular value of a Boolean perturbation of binary matrices.

Conjecture 20 (Smallest Singular Value of Binary Matrices) *Let $\bar{\mathbf{A}}$ be an arbitrary square matrix in $\{-1,+1\}^{n\times n}$, and let $\mathbf{A}$ be a σ-Boolean perturbation of $\bar{\mathbf{A}}$. Then there exists a constant $\alpha < 1$ such that*

$$\Pr_{\mathbf{A}}\left[\left\|\mathbf{A}^{-1}\right\|_2 \geq x\right] \leq \frac{\sqrt{n}}{x\sigma} + \alpha^n.$$

P7: Algebraic Eigenvalue Problems

Steve Vavasis† suggests studying the smoothed complexity of the classical QR iteration algorithm for solving algebraic eigenvalue problems. The eigenvalue problem is to find all eigenvalue-eigenvector pairs of a given $n \times n$ matrix $\mathbf{A}$, where the entries of $\mathbf{A}$ could be either complex or real. A scalar λ and an n-dimensional vector $\mathbf{x}$ form an eigenvalue-eigenvector pair of $\mathbf{A}$ if $\mathbf{Ax} = \lambda\mathbf{x}$. Note that the eigenvalue λ and the entries of its eigenvector $\mathbf{x}$ could be complex. The famous Schur decomposition theorem states:

Theorem 1.37 (Schur Decomposition) *If $\mathbf{A}$ is an $n \times n$ complex matrix, then there exists a unitary matrix $\mathbf{Q}$ such that*

$$\mathbf{Q}^H\mathbf{AQ} = \mathbf{T},$$

where $\mathbf{T}$ is an upper triangular matrix with all the eigenvalues of $\mathbf{A}$ appearing on its diagonal.

In addition, if $\mathbf{A}$ is a real matrix, then there exists an orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{n\times n}$ such that

$$\mathbf{Q}^T\mathbf{AQ} = \begin{pmatrix} \mathbf{R}_{1,1} & \mathbf{R}_{1,2} & \dots & \mathbf{R}_{1,k} \\ \mathbf{0} & \mathbf{R}_{2,2} & \dots & \mathbf{R}_{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{R}_{k,k} \end{pmatrix},$$

where $\mathbf{R}_{i,i}$ is either a scalar or a 2×2 matrix. When $\mathbf{R}_{i,i}$ is a scalar, it is an eigenvalue of $\mathbf{A}$ and when $\mathbf{R}_{i,i}$ is a 2×2 matrix, it has complex conjugate eigenvalues.

The QR iteration algorithm was first developed by Francis (1961). Its basic form is very simple. Initially, let $\mathbf{A}_0 = \mathbf{A}$. Iteratively, in the k^{th} step, the algorithm first computes an QR-decomposition of $\mathbf{A}_{i-1}$:

$$\mathbf{A}_{k-1} = \mathbf{Q}_{k-1}\mathbf{R}_{k-1},$$

where $\mathbf{Q}_{k-1}$ is a unitary matrix and $\mathbf{R}_{k-1}$ is an upper triangular matrix. It then defines

$$\mathbf{A}_k = \mathbf{R}_{k-1}\mathbf{Q}_{k-1}.$$

It is well known that in the complex case when $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$, the QR iteration algorithm converges and produces the Schur decomposition. In the real case, under some mild condition, the QR itera-

† Personal Communication.

tion algorithm converges to produce a real Schur decomposition (Wilkinson (1988) and Golub & Van Loan (1989)). Thus one can use the QR iteration algorithm to approximate all eigenvalues of $\mathbf{A}$ to an arbitrary precision.

Open Question 6 (Smoothed Complexity of QR iterations: Steve Vavasis) *What is the smoothed complexity of the QR iteration algorithm?*

The convergence of the QR iteration algorithm depends on the the minimum gaps among eigenvalues of the input matrix. For example, in the complex case when $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$, the lower off-diagonal (i,j)-entry of $\mathbf{A}_k$ $(i > j)$ is $O((\lambda_i/\lambda_j)^k)$ (Wilkinson (1988) and Golub & Van Loan (1989)).

Thus, understanding the eigenvalue gaps in the smoothed setting could hold the key to establishing the smoothed rate of convergence of the QR iteration algorithm.

Conjecture 21 (Minimum Complex Eigenvalue Gaps) *Let $\bar{\mathbf{A}}$ be an $n \times n$ complex matrix with $\left\|\bar{\mathbf{A}}\right\|_F \leq 1$. Let $\mathbf{A}$ be a σ-Gaussian perturbation of $\bar{\mathbf{A}}$. Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of $\mathbf{A}$ and assume $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$. Then, there exist positive constants c, k_1, k_2, k_3 such that, for all $x > 1$,*

$$\Pr\left[\min_{i>1}\left|\frac{\lambda_{i-1}-\lambda_i}{\lambda_{i-1}}\right| \leq \frac{1}{x}\right] \leq c \cdot n^{k_1} \cdot x^{-k_2} \cdot \sigma^{-k_3}.$$

One can similarly make a conjecture for real matrices.

Conjecture 22 (Minimum Real Eigenvalue Gaps) *Let $\bar{\mathbf{A}} \in \mathbb{R}^{n\times n}$ with $\left\|\bar{\mathbf{A}}\right\|_F \leq 1$. Let $\mathbf{A}$ be a σ-Gaussian perturbation of $\bar{\mathbf{A}}$. Then, there exist positive constants c, k_1, k_2, k_3 such that, for all $x > 1$,*

$$\Pr\left[\min_{\text{non-conjugate } \lambda_i,\, \lambda_j}\left|\frac{\lambda_i-\lambda_j}{\lambda_i}\right| \leq \frac{1}{x}\right] \leq c \cdot n^{k_1} \cdot x^{-k_2} \cdot \sigma^{-k_3}.$$

For a symmetric matrix $\mathbf{A}$, all of its eigenvalues are real and QR iterations preserve the symmetry as

$$\mathbf{A}_i = \mathbf{R}_{i-1}\mathbf{Q}_{i-1} = \mathbf{Q}_{i-1}^T\mathbf{Q}_{i-1}\mathbf{R}_{i-1}\mathbf{Q}_{i-1} = \mathbf{Q}_{i-1}^T\mathbf{A}_{i-1}\mathbf{Q}_{i-1}.$$

If the eigenvalues of $\mathbf{A}$ are $\lambda_1, \ldots, \lambda_n$, then the QR iteration algorithm converges to $\mathbf{diag}(\lambda_1, \ldots, \lambda_n)$, the diagonal matrix whose diagonal entries are $\lambda_1, \ldots, \lambda_n$.

Proving the following conjecture could be useful to establish a smoothed rate of convergence of the QR iteration algorithm under zero-preserving Gaussian perturbations.

Conjecture 23 (Minimum Symmetric Eigenvalue Gaps) *Let* $\bar{\mathbf{A}}$ *be an* $n \times n$ *real and symmetric matrix with* $\left\|\bar{\mathbf{A}}\right\|_F \leq 1$. *Let* $\mathbf{A} = \bar{\mathbf{A}} + \sigma \mathbf{diag}(g_1, \ldots, g_n)$ *be a* σ*-Gaussian perturbation of the diagonal of* $\bar{\mathbf{A}}$. *Let* $\lambda_1, \ldots, \lambda_n$ *be the eigenvalues of* $\mathbf{A}$ *such that* $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$. *Then, there exist positive constants* c, k_1, k_2, k_3 *such that, for all* $x > 1$,

$$\Pr\left[\min_i \left|\frac{\lambda_{i-1} - \lambda_i}{\lambda_i}\right| \leq \frac{1}{x}\right] \leq c \cdot n^{k_1} \cdot x^{-k_2} \cdot \sigma^{-k_3}.$$

A closely related conjecture is about the singular value gaps.

Conjecture 24 (Minimum Singular Value Gaps) *Let* $\bar{\mathbf{A}}$ *be an* $m \times n$ *real matrix with* $\left\|\bar{\mathbf{A}}\right\|_F \leq 1$ *and* $m \leq n$. *Let* $\mathbf{A}$ *be a* σ*-Gaussian perturbation of* $\bar{\mathbf{A}}$. *Let* $s_1, \ldots, s_m$ *be the singular value of* $\mathbf{A}$ *such that* $s_1 \geq s_2 \geq \ldots \geq s_m$. *Then, there exist positive constants* c, k_1, k_2, k_3 *such that, for all* $x > 1$,

$$\Pr\left[\min_i \left(\frac{s_{i-1} - s_i}{s_i}\right) \leq \frac{1}{x}\right] \leq c \cdot n^{k_1} \cdot x^{-k_2} \cdot \sigma^{-k_3}.$$

In practice, one usually does not apply the QR iteration algorithm directly to an input matrix $\mathbf{A}$. The QR computation of each iteration could take $O(n^3)$ time, which might be too expensive. In fact, most practical implementations first use an orthogonal similarity transformation to reduce the matrix $\mathbf{A}$ to an upper-Hessenberg form (Wilkinson (1988) and Golub & Van Loan (1989)) $A_0 = \mathbf{Q}_0^H \mathbf{A} \mathbf{Q}_0$. A matrix $\mathcal{H} = (h_{i,j})$ is upper Hessenberg if $h_{i,j} = 0$ for $i > j + 1$. This step is important because the QR factorization of an upper Hessenberg matrix can be computed in $O(n^2)$ time, instead of $O(n^3)$. The standard approach uses Givens rotations at each step to perform QR factorization of an upper Hessenberg matrix. A nice property of the Givens process for QR factorization is that the resulting QR iteration algorithm preserves the upper Hessenberg form.

In general, the practical QR iteration algorithms go beyond just applying an initial Hessenberg reduction. What makes them more successful in practice is the collection of shifting strategies that are used to improve the rate of convergence (Wilkinson (1988) and Golub & Van Loan (1989)). Each shifted iteration consists of the following steps.

1. Determine a scalar μ_{i-1};
2. Compute $\mathbf{A}_{i-1} - \mu_{i-1}\mathbf{I} = \mathbf{Q}_{i-1}\mathbf{R}_{i-1}$;
3. Let $\mathbf{A}_i = \mathbf{R}_{i-1}\mathbf{Q}_{i-1} + \mu_{i-1}\mathbf{I}$.

In practice, QR iteration algorithms may perform double shifts during each iteration.

Open Question 7 (Smoothed Complexity of Practical QR Iteration Algorithms)

- *What is the smoothed complexity of these practical QR iteration algorithms?*
- *Is the smoothed rate of convergence of any practical QR iteration algorithm better than that of the classical QR iteration algorithm?*
- *What is the impact of Hessenberg reduction on the smoothed rate of convergence of QR iteration Algorithms?*
- *What are the smoothed rates of convergence of the classical or practical symmetric QR iteration algorithms under symmetry-preserving Gaussian perturbations and under symmetry-preserving and zero-preserving Gaussian perturbations?*

P8: Property-Preserving Perturbations

For some discrete problems, as we have discussed in Section 1.3.4, results from the semi-random model might not always extend to the corresponding property-preserving perturbations. Perhaps the most appealing problem is the Bisection Problem.

Open Question 8 (Bisection) *Is the ρ-Bisection Problem, under ρ-Bisection preserving perturbations, in smoothed polynomial time (in the probabilistic sense), for some constant $0 < \rho < 1$?*

A closely related problem is whether a ρ-Bisection property testing algorithm exists that runs in time polynomial in $1/\epsilon$ and $1/\sigma$ in the smoothed model under ρ-Bisection-preserving σ-perturbations. Another related problem is whether the ρ-Bisection Problem is in smoothed polynomial time (in the probabilistic sense) under the solution-preserving perturbations.

The property-preserving model is not limited to discrete settings. It can be applied to the continuous setting as well. For example, one can study the smoothed complexity of a linear programming algorithm under feasibility-preserving Gaussian perturbations.

Open Question 9 (Feasibility and Linear Programming) *Is the simplex method with the shadow-vertex pivoting rule still in smoothed polynomial time under feasibility-preserving Gaussian perturbations?*

Is the smoothed value of the logarithm of the condition number of linear programs still poly-logarithmic in m, n, and $1/\sigma$ as stated in Theorem 1.7, under feasibility-preserving Gaussian perturbations?

As the purpose of smoothed analysis is to shed light on the practical performance of an algorithm, it is more desirable to use a perturbation model that better fits the input instances. See the *Final Remarks* at the end of this paper for more discussion. Thus, if all or most practical instances to an algorithm share some common structures, such as being symmetric or being planar, then to have a meaningful theory, we may have to consider perturbations that preserve these structures. For example, the fact that many scientific computing algorithms use the sparsity of the input to achieve good performance encourages us to define the zero-preserving or magnitude-preserving perturbations such as relative Gaussian perturbations. So far, however, the smoothed complexities of various problems and algorithms under these perturbations remains wide open.

Open Question 10 (Structure-Preserving Perturbations) *What is the impact of structure-preserving perturbations, such as magnitude-preserving and zero-preserving perturbations, on the smoothed complexity of an algorithm?*

P9: Smoothed Complexity and Approximation Algorithms

Open Question 11 (Smoothed Complexity and Hardness of Approximation) *Is there any connection between the smoothed complexity of an optimization problem and the hardness of its approximation? Under what conditions does "hard to approximate" imply "high smoothed complexity" and vice versa?*

As smoothed time complexity measures the performance of an algorithm A on an input x by the expected performance of A over a "neighborhood" distribution of x, intuitively, if this complexity is low, then one could first perturb an instance and solve the optimization problem over the perturbed instance. The resulting algorithm then has low randomized complexity.

How good this randomized algorithm can be as an approximation algorithm may depend on the perturbation model, the property of the objective function and the structure of the solution space.

Suppose A is an algorithm for solving a minimization problem with an objective function f over an input domain $D = \cup_n D_n$. Suppose further, there is a family of neighborhoods $N_\sigma(\bar{x}) \subseteq \cup_{n'=\Theta(n)} D_{n'}$ for every $\bar{x} \in D_n$, such that for all $x \in N_\sigma(\bar{x})$, $|f(A(x)) - f(A(\bar{x}))| \leq h(\sigma)$ where $h : \mathbb{R} \to \mathbb{R}^+$ is a monotonically increasing function.

If there is a family of perturbations $\mathcal{R} = \cup_{n,\sigma} R_{n,\sigma}$, where, for each $\bar{x} \in D_n$, $R_{n,\sigma}$ defines a perturbation distribution over $N_\sigma(\bar{x})$ such that the smoothed complexity of A under this perturbation model is $T(n,\sigma)$, then A can be used as a family of randomized approximation algorithms of expected complexity $T(n,\sigma)$ that comes within $h(\sigma)$ of the optimal value for minimizing f in instance $\bar{x}$, provided $\mathcal{R}$ can be efficiently sampled.

For example, consider a two-person game given by two $m \times n$ matrices $\bar{\mathbf{A}} = (\bar{a}_{i,j})$ and $\bar{\mathbf{B}} = (\bar{b}_{i,j})$. Suppose $\mathbf{A} = (a_{i,j})$ and $\mathbf{B} = (b_{i,j})$ are σ-*uniform-cube perturbations* of $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$, respectively, where $a_{i,j}$ (and $b_{i,j}$) is an independent random variable chosen uniformly from the interval $[\bar{a}_{i,j} - \sigma, \bar{a}_{i,j} + \sigma]$ (and $[\bar{b}_{i,j} - \sigma, \bar{b}_{i,j} + \sigma]$). Then, for every pair of mixed strategies $\mathbf{x}$ and $\mathbf{y}$, $\left|\mathbf{x}^T\mathbf{A}\mathbf{y} - \mathbf{x}^T\bar{\mathbf{A}}\mathbf{y}\right| \leq 2\sigma$. and $\left|\mathbf{x}^T\mathbf{B}\mathbf{y} - \mathbf{x}^T\bar{\mathbf{B}}\mathbf{y}\right| \leq 2\sigma$.

Now suppose $(\mathbf{x}, \mathbf{y})$ is a Nash equilibrium for $(\mathbf{A}, \mathbf{B})$. Then, for any $(\mathbf{x}', \mathbf{y}')$, we have

$$(\mathbf{x}')^T\bar{\mathbf{A}}\mathbf{y} - \mathbf{x}^T\bar{\mathbf{A}}\mathbf{y} \leq ((\mathbf{x}')^T\mathbf{A}\mathbf{y} - \mathbf{x}^T\mathbf{A}\mathbf{y}) + 4\sigma \leq 4\sigma,$$

as well as $\mathbf{x}^T\bar{\mathbf{B}}\mathbf{y}' - \mathbf{x}^T\bar{\mathbf{B}}\mathbf{y} \leq 4\sigma$. Thus, $(\mathbf{x}, \mathbf{y})$ is a (4σ)-*Nash equilibrium* for $(\bar{\mathbf{A}}, \bar{\mathbf{B}})$: a profile of mixed strategies such that no player can gain more than an amount 4σ by changing his/her strategy unilaterally. Similarly, if $(\mathbf{x}, \mathbf{y})$ is an ϵ-Nash equilibrium for $(\mathbf{A}, \mathbf{B})$, then $(\mathbf{x}, \mathbf{y})$ is an $(\epsilon + 4\sigma)$-Nash equilibrium for $(\bar{\mathbf{A}}, \bar{\mathbf{B}})$. Therefore,

Proposition 1.1 (Smoothed 2-Nash and Approximated 2-Nash) *If 2-Nash can be solved in smoothed time polynomial in m, n, and $g(1/\sigma)$ under σ-uniform-cube perturbations, then an ϵ-Nash equilibrium of two-person games can found in randomized time polynomial in m, n, and $g(1/\epsilon)$.*

However, for a constrained optimization problem, the optimal solution of a perturbed instance x may not be feasible for the original instance $\bar{x}$. Although running A on the perturbed instance x provides a good approximation of the optimal value for the original instance $\bar{x}$, one still

needs an efficient procedure to "round" the solution for x to a feasible solution for $\bar{x}$ in order to approximately solve the optimization problem. For some problems, such as linear programming, the optimal solution for x might be quite "far" away from the optimal solution for $\bar{x}$, although their objective values are be close. This discrepancy might pose algorithmic challenges to approximations.

Another interesting direction of research is to examine the worst-case instances appearing in the literature to determine whether they are stable under perturbations. If all the known worst-case instances of a problem or an algorithm are not stable under some perturbations, then one could ask whether its smoothed complexity under these perturbations is low, or if there are other bad instances that are stable.

P10: Other Algorithms and Practical Algorithms

There are many other successful practical heuristics that we cannot discuss here in great detail. For example, Berthold Vöcking suggests, as interesting and relevant research directions, considering the smoothed complexity of heuristics like branch-and-bound or cutting-plane methods on structurally simple optimization problems like packing problems with a constant number of constraints.

Other very popular methods include the multilevel algorithms (Brandt (1988) and Teng (1998)), differential evolution (Price, Storn & Lampinen (2005)), and various local search and global optimization heuristics. We would like to understand the smoothed complexity of these methods. For example, the following conjecture is at the center of our research (in this area).

Conjecture 25 (Multilevel Bisection Conjecture) *There is a multilevel bisection algorithm with smoothed polynomial time complexity that finds a $(c\cdot\rho)$-bisection (for some constant c) under ρ-bisection-preserving perturbations as well as in the semi-random model.*

Conjecture 26 (Multilevel Sparsest-Cut Conjecture) *There is a multilevel partitioning algorithm with smoothed polynomial time complexity that finds a partition with sparsity $c \cdot \rho$ under ρ-sparsest-cut-preserving perturbations as well as in the semi-random model.*

Final Remarks

Developing rigorous mathematical theory that can model the observed performance of practical algorithms and heuristics has become an increasingly important task in Theoretical Computer Science. Unlike four decades ago, when theorists were introducing asymptotic complexity but practitioners could only solve linear systems with less than 500 variables, we now have computers that are capable of solving very large-scale problems. Moreover, as heuristic algorithms become ubiquitous in applications, we have increasing opportunities to obtain data, especially on large-scale problems, from these remarkable heuristics.

One of the main objectives of smoothed analysis is to encourage the development of theories for the practical behaviors of algorithms. We are especially interested in modeling those algorithms whose practical performance is much better than their worst-case complexity measures.

A key step is to build analyzable models that are able to capture some essential aspects of the algorithms and, of equal importance, the inherent properties of practical instances. So necessarily, any such model should be more instance-oriented than our traditional worst-case and average analyses, and should consider the formation process of input instances.

However, modeling observed data and practical instances is a challenging task. Practical inputs are often complex and there may be multiple parameters that govern the process of their formation. Most of the current work in smoothed analysis focuses on the randomness in the formation of inputs and approximates the likelihood of any particular instance by its similarity or distance to a "hidden blueprint" of a "targeted" instance of the input-formation process. As the targeted instance might not be known to the algorithm, in the same spirit of worst-case analysis, we used the maximum over all possible targeted instances in the definition of the smoothed complexity.

This approach to characterize the randomness in the formation of input instances promises to be a good first step to model or to approximate the distribution of practical instances. One must understand the possible limitations of any particular perturbation model, however, and not overstate the practical implication of any particular analysis.

One way to improve the similarity-or-distance based perturbation models is to develop an analysis framework that takes into account the formation of input instances. For example, if the input instances to an algorithm A come from the output of another algorithm B, then algorithm B, together with a model of B's input instances, is the description

of A's inputs. To be concrete, consider finite-element calculations in scientific computing. The input to its linear solver A are stiffness matrices which are produced by a finite-element mesh generation algorithm B. The meshing algorithm B, which could be a randomized algorithm, receives a geometric domain Ω and a partial differential equation F as an input instances to construct a stiffness matrix. So the distribution of the stiffness matrices to algorithm A is determined by the distribution $\mathcal{D}$ of the geometric domains Ω and the set F of partial differential equations, as well as the mesh generation algorithm B. One can define the measure of the performance of A as

$$\mathrm{E}_{(\Omega,F)\leftarrow\mathcal{D}}\left[\mathrm{E}_{X\leftarrow B(\Omega,F)}\left[Q(A,X)\right]\right].$$

If, for example, $\bar{\Omega}$ is a design of an advanced rocket from a set $\mathcal{R}$ of "blueprints" and F is from a set $\mathcal{F}$ of PDEs for physical parameters such as pressure, speed, and temperature for the rocket, and Ω is generated by a perturbation model $\mathcal{P}$ of the blueprints, then one may further measure the performance of A by the smoothed value of the quantity above:

$$\max_{F\in\mathcal{F},\bar{\Omega}\in\mathcal{R}} \mathrm{E}_{\Omega\leftarrow\mathcal{P}(\bar{\Omega})}\left[\mathrm{E}_{X\leftarrow B(\Omega,F)}\left[Q(A,X)\right]\right].$$

There might be many different frameworks for modeling the formation process of the input instances. For example, one could use a Markov process, a branch tree with probabilistic nodes and binary branching nodes or some innovative diagrams or flowcharts. The better we can model our input data, the more accurately we can model the performance of an algorithm. But to be rigorous mathematically, we may have to come up with a conjecture that matches the practical observations, and find a way to prove the conjecture.

Another objective of smoothed analysis is to provide insights and motivations to design new algorithms, especially those with good smoothed complexity. For example, our analysis of the smoothed growth factor suggests a new and more stable solver for linear systems: Suppose we are given a linear system $\mathbf{Ax} = \mathbf{b}$. We first use the standard elimination-based algorithm – or software – to solve $\mathbf{Ax} = \mathbf{b}$. Suppose $\mathbf{x}^*$ is the solution computed. If $\|\mathbf{b} - \mathbf{Ax}^*\|$ is small enough, then we simply return $\mathbf{x}^*$. Otherwise, we can determine a parameter ϵ and generate a new linear system $(\mathbf{A} + \epsilon\mathbf{G})\mathbf{y} = \mathbf{b}$, where $\mathbf{G}$ is a Gaussian matrix with independent entries with mean 0 and variance 1. So instead of using the solution of $\mathbf{Ax} = \mathbf{b}$, we solve a perturbed linear system $(\mathbf{A} + \epsilon\mathbf{G})\mathbf{y} = \mathbf{b}$. It follows from the condition number analysis that if ϵ is (significantly) smaller

than $\kappa(\mathbf{A})$, then the solution to the perturbed linear system is a good approximation to the original one. One can use practical experience or binary search to set ϵ.

The new algorithm has the property that its success depends only on the machine precision and the condition number of $\mathbf{A}$, while the original algorithm may fail due to large growth factors. For example, the following is a segment of matlab code that first solves a linear system whose matrix is the 70×70 Wilkinson matrix, using the Matlab linear solver, then solves it with our new algorithm.

```
>> % Using the Matlab Solver
>> n = 70; A = 2*eye(n)-tril(ones(n)); A(:,n)=1;
>> b = randn(70,1);
>> x = A\b;
>> norm(A*x-b)
>> 2.762797463910437e+004
>> % FAILED because of large growth factor
>> %Using the new solver
>> Ap = A + randn(n)/10^9;
>> y = Ap\b;
>> norm(Ap*y-b)
>> 6.343500222435404e-015
>> norm(A*y-b)
>> 4.434147778553908e-008
```

Because the Matlab linear solver uses Gaussian elimination with partial pivoting, it fails to solve the linear system because of the large growth factor. But our perturbation-based algorithm finds a good solution.

We conclude this paper with the open question that initially led us to smoothed analysis.

Open Question 12 (Linear Programming in Strongly Random Polynomial Time?) *Can the techniques from the smoothed analysis of the simplex and interior-point methods be used to develop a randomized strongly polynomial-time algorithm for linear programming?*

1.5 Acknowledgments

We thank Berthold Vöcking for his suggestions of research direction in smoothed analysis, Steve Vavasis for his suggestion of performing

smoothed analysis on the QR iteration algorithm, Ben Hescott for mentioning the paper by Daskalakis, Goldberg, and Papadimitriou, and Jinyi Cai for bringing the Bruhat's decomposition to our attention. We thank Kyle Burke and Ben Hescott for their helpful comments and discussions on this writing. Finally, we would like to express our appreciation of all the valuable feedbacks that we received after presenting our work on smoothed analysis at various universities, institutions, and conferences.

Daniel Spielman is partially supported by the NSF grant CCR-0324914 and Shang-Hua Teng is partially supported by the NSF grants CCR-0311430 and ITR CCR-0325630.

References

L. Adleman and M.-D. Huang (1987). Recognizing primes in random polynomial time. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 462–469.

I. Adler, R. M. Karp, and R. Shamir (1987). A simplex variant solving an $m \times d$ linear program in $O(\min(m^2, d^2))$ expected number of steps. *Journal of Complexity*, 3:372–387.

I. Adler and N. Megiddo (1985). A simplex algorithm whose average number of steps is bounded between two quadratic functions of the smaller dimension. *J. ACM*, 32(4):871–895, October.

A. Aggarwal, B. Alpern, A. Chandra, and M. Snir (1987). A model for hierarchical memory. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 305–314.

S. Agmon (1954). The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6:382–392.

M. Agrawal, N. Kayal and N. Saxena (2004). Primes is in P. In *Annals of Mathematics 160(2)*, pages 781–793.

A. V. Aho, J E. Hopcroft, and J. Ullman (1983). *Data Structures and Algorithms*. Addison-Wesley Longman.

N. Alon and J. H. Spencer (1992). *The Probabilistic Method*. John Wiley and Sons.

N. Amenta and G. Ziegler (1999). Deformed products and maximal shadows of polytopes. In B. Chazelle, J.E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, number 223 in Contemporary Mathematics, pages 57–90. Amer. Math. Soc.

E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen (1999). *LAPACK Users' Guide, Third Edition*. SIAM, Philadelphia.

D. Arthur and S. Vassilvitskii (2005) On the Worst Case Complexity of the k-means Method. http://dbpubs.stanford.edu:8090/pub/2005-34.

K. Ball (1993). The reverse isoperimetric problem for gaussian measure. *Discrete and Computational Geometry*, 10(4):411–420.

C. Banderier, R. Beier, and K. Mehlhorn (2003). Smoothed analysis of three combinatorial problems. In *Proceedings of the Twenty-eighth Interna-*

tional Symposium on Mathematical Foundations of Computer Science, volume 2747, pages 198–207.

I. Barany, S. Vempala, and A. Vetta (2005). Nash equilibria in random games. In *Proceedings of Forty-sixth Annual IEEE Symposium on Foundations of Computer Science*.

J. Basch, L. J. Guibas, and J. Hershberger (1997). Data structures for mobile data. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 747–756.

L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, G. Schäfer, and T. Vredeveld (2003). Average case and smoothed competitive analysis of the multi-level feedback algorithm. In *Proceedings of the Forty-fourth Annual IEEE Symposium on Foundations of Computer Science*, page 462.

R. Beier and B. Vöcking (2004). Typical properties of winners and losers in discrete optimization. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, pages 343–352.

J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson (1978). On the average number of maxima in a set of vectors and applications. *J. ACM*, 25(4):536–543.

R. G. Bland (1977). New finite pivoting rules. *Mathematics of Operations Research*, 2:103 – 107.

H. D. Block (1962). The perceptron: A model for brain functioning. *Reviews of Modern Physics*, 34:123–135.

A. Blum and J. Dunagan (2002). Smoothed analysis of the perceptron algorithm for linear programming. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 905–914.

A. Blum and J. Spencer (1995). Coloring random and semi-random k-colorable graphs. *J. Algorithms*, 19(2):204–234.

T. Bohman, A. Frieze, and R. Martin (2003). How many random edges make a dense graph hamiltonian? *Random Struct. Algorithms*, 22(1):33–42.

R. B. Boppana (1987). Eigenvalues and graph bisection: An average-case analysis. In *Proceedings of the Forty-seventh Annual IEEE Symposium on Foundations of Computer Science*, pages 280–285.

K. H. Borgwardt (1980). *The Simplex Method: a probabilistic analysis.* Springer-Verlag.

A. Borodin and R. El-Yaniv (1998). *Online computation and competitive analysis.* Cambridge University Press, New York, NY, USA.

A. Borodin, N. Linial, and M. E. Saks (1992). An optimal on-line algorithm for metrical task system. *J. ACM*, 39(4):745–763.

A. Brandt (1988). Multilevel computations: Review and recent developments. In *Multigrid Methods: Theory, Applications, and Supercomputing*, Marcel-Dekker, S. F. McCormick, editor, 541–555.

S.-W. Cheng and T. K. Dey (2002). Quality meshing with weighted delaunay refinement. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 137–146.

S. W. Cheng, T. K. Dey, H. Edelsbrunner, M. A. Facello, and S.-H. Teng (2000). Sliver exudation. *J. ACM*, 47:883 – 904.

L.P. Chew (1989). Guaranteed-quality triangular meshes. Technical Report TR-89-983, Cornell University, Ithaca.

V. Chvatal (1983). *Linear Programming.* A Series of Books in the Mathematical Sciences. Freeman.

K. L. Clarkson (1995). Las Vegas algorithms for linear and integer program-

ming when the dimension is small. *J. ACM*, 42(2):488–499.

A. Condon, H. Edelsbrunner, E. A. Emerson, L. Fortnow, S. Haber, R. Karp, D. Leivant, R. Lipton, , N. Lynch, I. Parberry, C. Papadimitriou, M. Rabin, A. Rosenberg, J. S. Royer, J. Savage, A. L. Selman, C. Smith, E. Tardos, and J. S. Vitter (1999). Challenges for theory of computing. In *Report of an NSF-Sponsored Workshop on Research in Theoretical Computer Science*.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein (2001). *Introduction to Algorithms*. McGraw-Hill Higher Education.

F. Cucker and J. Peña (2001). A primal-dual algorithm for solving polyhedral conic systems with a finite-precision machine. *SIAM J. on Optimization*, 12(2):522–554.

V. Damerow, F. M. auf der Heide, H. Räcke, C. Scheideler, and C. Sohler (2003). Smoothed motion complexity. In *Proceedings of the Eleventh Annual European Symposium on Algorithms*, pages 161–171.

V. Damerow and C. Sohler (2004). Extreme points under random noise. In *European Symposium on Algorithms*, pages 264–274.

G. B. Dantzig (1951). Maximization of linear function of variables subject to linear inequalities. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 339–347.

G. B. Dantzig (1991). *Linear Programming and Extensions*. Springer.

C. Daskalakis, C. H. Papadimitriou, P. W. Goldberg (2005). The complexity of computing a nash equilibrium. *Electronic Colloquium on Computational Complexity*, TR05-115.

J. Demmel (1997). *Applied Numerical Linear Algebra*. SIAM.

A. Deshpande and D. A. Spielman (2005). Improved smoothed analysis of the shadow vertex simplex method. In *Proceedings of Forty-sixth Annual IEEE Symposium on Foundations of Computer Science*.

D. Dobkin, R. J. Lipton and S. Reiss (1979). Linear programming is log–space hard for P. *Information Processing Letters*, 8:96–97.

J. Dunagan, D. A. Spielman, and S.-H. Teng (2002). Smoothed analysis of renegar's condition number for linear programming. available at `http://math.mit.edu/~spielman/SmoothedAnalysis`, submitted to *Mathematical Programming*.

J. Dunagan and S. Vempala (2004). A simple polynomial-time rescaling algorithm for solving linear programs. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, pages 315–320.

A. Edelman (1988). Eigenvalues and condition numbers of random matrices. *SIAM J. Matrix Anal. Appl.*, 9(4):543–560.

H. Edelsbrunner (2001). *Geometry and topology for mesh generation*. Cambridge University Press.

H. Edelsbrunner, X.-Y. Li, G. L. Miller, A. Stathopoulos, D. Talmor, S.-H. Teng, A. Üngör, and N. Walkington (2000). Smoothing and cleaning up slivers. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, pages 273–277.

U. Feige and J. Kilian (1998). Heuristics for finding large independent sets, with applications to coloring semi-random graphs. In *Proceedings of the Thirty-ninth Annual Symposium on Foundations of Computer Science*, page 674.

U. Feige and R. Krauthgamer (2002). A polylogarithmic approximation of the minimum bisection. *SIAM J. on Computing*, 31:1090–1118.

W. Feller (1968,1970). *An Introduction to Probability Theory and Its Applications*, volume 1,2. Wiley, New York.

A. Flaxman and A. M. Frieze (2004). The diameter of randomly perturbed digraphs and some applications.. In *APPROX-RANDOM*, pages 345–356.

J. J. Forrest, D. Goldfarb, D. (1992) Steepest-edge simplex algorithms for linear programming. *Mathematical Programming* 57, 341–374.

J. G. F. Francis (1961). The qr transformation a unitary analogue to the lr transformation: Part 1. *The Computer Journal*, 4(3):256 – 271.

R. M. Freund and F. Ordóñez (2005). On an extension of condition number theory to non-conic convex optimization. *Math of OR*, 30(1).

R. Freund and J. Vera (1999). On the complexity of computing estimates of condition measures of a conic linear system. Operations Research Center Working Paper, MIT, 1999, submitted to *Mathematics of Operations Research*.

R. Freund and J. Vera (2000). Condition-based complexity of convex optimization in conic linear form via the ellipsoid algorithm. *SIAM J. on Optimization*, 10(1):155–176.

M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran (1999). Cache-oblivious algorithms. In *Proceedings of the Fortieth Annual Symposium on Foundations of Computer Science*, page 285.

S. Gass and Th. Saaty (1955). The computational algorithm for the parametric objective function. *Naval Research Logistics Quarterly*, 2:39–45.

D. Goldfarb (1983). Worst case complexity of the shadow vertex simplex algorithm. Technical report, Columbia University.

D. Goldfarb and W. T. Sit (1979). Worst case behaviour of the steepest edge simplex method. *Discrete Applied Math*, 1:277–285.

O. Goldreich, S. Goldwasser, and D. Ron (1998). Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, July.

G. H. Golub and C. F. Van Loan (1989). *Matrix Computations*. second edition.

M. Hestenes and E. Stiefel (1952). Methods of conjugate gradients for solving linear systems. *the Journal of Research of the National Bureau of Standards*.

R. G. Jeroslow (1973). The simplex algorithm with the pivot rule of maximizing improvement criterion. *Discrete Math.*, 4:367–377.

M. Jurdzinski (2005). Stochastic games: A tutorial. http://www.games.rwth-aachen.de/Events/Bordeaux/t_mju.pdf.

G. Kalai (1992). A subexponential randomized simplex algorithm (extended abstract). In *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing*, pages 475–482.

G. Kalai and D. J. Kleitman (1992). A quasi-polynomial bound for the diameter of graphs of polyhedra. *Bulletin Amer. Math. Soc.*, 26:315–316.

N. K. Karmarkar (1984). A new polynomial–time algorithm for linear programming. *Combinatorica*, 4:373–395.

M. J. Kearns, M. L. Littman, and S. P. Singh (2001). Graphical models for game theory. In *UAI '01: Proceedings of the Seventeenth Conference in Uncertainty in Artificial Intelligence*, pages 253–260.

L. G. Khachiyan (1979). A polynomial algorithm in linear programming. *Doklady Akademia Nauk SSSR*, pages 1093–1096.

V. Klee and G. J. Minty (1972). How good is the simplex algorithm ? In Shisha, O., editor, *Inequalities – III*, pages 159–175. Academic Press.

M. Krivelevich, B. Sudakov, and P. Tetali (2005). On

smoothed analysis in dense graphs and formulas. http://www.math.princeton.edu/~bsudakov/smoothed-analysis.pdf.

C. E. Lemke (1965). Bimatrix equilibrium points and mathematical programming. *Management Science*, 11:681–689.

C. E. Lemke and JR. J. T. Howson (1964). Equilibrium points of bimatrix games. *J. Soc. Indust. Appl. Math.*, 12:413–423.

X.-Y. Li and S.-H. Teng (2001). Generate sliver free three dimensional meshes. In *Proceedings of the Twelfth ACM-SIAM Symp. on Discrete Algorithms*, pages 28–37.

X. S. Li (2005). An overview of superLU: Algorithms, implementation, and user interface. *ACM Trans. Math. Softw.*, 31(3):302–325.

S. Lloyd (1982) Least squares quantization in PCM *IEEE Transactions on Information Theory*, 28 (2) pages 129 –136.

M. Luby and N. Nisan (1993). A parallel approximation algorithm for positive linear programming. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*, pages 448–457.

B. Manthey and R. Reischuk (2005). Smoothed analysis of the height of binary search tress. *Electronic Colloquium on Computational Complexity*, TR05-063.

J. Matoušek, M. Sharir, and E. Welzl (1992). A subexponential bound for linear programming. In *Proceedings of the Eighth Annual Symposium on Computational Geometry*, pages 1–8.

N. Megiddo (1986). Improved asymptotic analysis of the average number of steps performed by the self-dual simplex algorithm. *Mathematical Programming*, 35(2):140–172.

N. Megiddo (1992). A note on approximate linear programming. *Inf. Process. Lett.*, 42(1):53, 1992.

Gary L. Miller (1975). Riemann's hypothesis and tests for primality. In *Proceedings of Seventh Annual ACM Symposium on Theory of Computing*, pages 234–239.

G. L. Miller, D. Talmor, S.-H. Teng, and N. Walkington (1995). A Delaunay based numerical method for three dimensions: generation, formulation, and partition. pages 683–692.

M. L. Minsky and S. A. Papert (1988). *Perceptrons: expanded edition.* MIT Press.

R. Motwani, S. Phillips, and E. Torng (1993). Non-clairvoyant scheduling. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 422–431.

R. Motwani and P. Raghavan (1995). *Randomized algorithms.*

K. G. Murty (1980). Computational complexity of parametric linear programming. *Math. Programming*, 19:213–219.

J. Nash. Noncooperative games (1951). *Annals of Mathematics*, 54:289–295.

Y. Nesterov and A. Nemirovskii (1994). *Interior Point Polynomial Methods in Convex Programming: Theory and Applications.* Society for Industrial and Applied Mathematics, Philadelphia.

A. B. Novikoff (1962). On convergence proofs on perceptrons. In *Symposium on the Mathematical Theory of Automata, 12*, pages 615–622.

P. van den Driessche and O. H. Odeh and D. D. Olesky (1998). Bruhat decomposition and numerical stability. *SIAM J. on Matrix Analysis and Applications*, 19(1):89–98.

C. H. Papadimitriou (1994). *Computational Complexity.* Addison-Wesley.

C. H. Papadimitriou and K. Steiglitz (1982). *Combinatorial optimization: algorithms and complexity*. Prentice-Hall.

K. Price, R. Storn, and J. Lampinen (2005). *Differential Evolution - A Practical Approach to Global Optimization*. Springer.

R. Ravi and M. X. Goemans (1996). The constrained minimum spanning tree problem (extended abstract). In *Proceedings of the Fifth Scandinavian Workshop on Algorithm Theory*, pages 66–75. Springer-Verlag.

J. Renegar (1994). Some perturbation theory for linear programming. *Math. Programming*, 65(1, Ser. A):73–91.

J. Renegar (1995a). Incorporating condition measures into the complexity theory of linear programming. *SIAM J. Optim.*, 5(3):506–524.

J. Renegar (1995b). Linear programming, complexity theory and elementary functional analysis. *Math. Programming*, 70(3, Ser. A):279–351.

H. Röglin, B. Vöcking (2005). Smoothed analysis of integer programming. In *Michael Junger and Volker Kaibel, editors, Proceedings of the of the Eleventh International Conference on Integer Programming and Combinatorial Optimization, volume 3509 of Lecture Notes in Computer Science, Springer*, pages 276 – 290.

F. Rosenblatt (1962). *Principles of neurodynamics; perceptrons and the theory of brain mechanisms*. Spartan Books.

R. Rubinfeld and M. Sudan (1996). Robust characterizations of polynomials with applications to program testing. *SIAM J. on Computing*, 25(2):252–271, April.

J. Ruppert(1993). A new and simple algorithm for quality 2-dimensional mesh generation. In *Proceedings of the Fourth ACM-SIAM Symp. on Disc. Algorithms*, pages 83–92.

A. Sankar, D. A. Spielman, and S.-H Teng (2005). Smoothed analysis of the condition numbers and growth factors of matrices. *SIAM J. on Matrix Analysis and Applications*, to appear.

M. Santha and U. V Vazirani (1986). Generating quasi-random sequences from semi-random sources. *J. Comput. Syst. Sci.*, 33(1):75–87.

R. Savani and B. von Stengel (2004). Exponentially many steps for finding a nash equilibrium in a bimatrix game. In *Proceedings of the Forty-fifth Annual IEEE Symposium on Foundations of Computer Science*, pages 258–267.

F. Schäfer and N. Sivadasan (2004). Topology matters: Smoothed competitiveness of metrical task systems. In *Proceedings of the Twenty-first Annual Symposium on Theoretical Aspects of Computer Science, (Montpellier, France, March 25-27, 2004)*, volume 2996 of *LNCS*, pages 489–500. Springer-Verlag.

A. Schrijver (1986). *Theory of Linear and Integer Programming*. Wiley, 1986.

S. Sen, S. Chatterjee, and N. Dumir (2002). Towards a theory of cache-efficient algorithms. *J. ACM*, 49(6):828–858.

M. Serna (1991). Approximating linear programming is log-space complete for p. *Inf. Process. Lett.*, 37(4):233–236.

J. R. Shewchuk (1998). Tetrahedral mesh generation by Delaunay refinement. In *Proceedings of the Fourteenth Annual ACM Symposium on Computational Geometry*, pages 86–95.

M. Sipser (1996). *Introduction to the Theory of Computation*. International Thomson Publishing.

D. D. Sleator and R. E. Tarjan (1985). Amortized efficiency of list update and

paging rules. *Commun. ACM*, 28(2):202–208.

S. Smale (1982). The problem of the average speed of the simplex method. In *Proceedings of the Eleventh International Symposium on Mathematical Programming*, pages 530–539, August.

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

R. Solovay and V. Strassen (1977). A fast Monte-Carlo test for primality. 6(1):84–85, March.

D. A. Spielman and S.-H. Teng (2003a). Smoothed analysis (motivation and discrete models. In *Algorithms and Data Structures, 8th International Workshop*, pages 256–270.

D. A. Spielman and S.-H. Teng (2003b). Smoothed analysis of termination of linear programming algorithms. *Mathematical Programming, Series B*, 97:375–404.

D. A. Spielman and S.-H. Teng (2004). Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463.

D. A. Spielman, S.-H. Teng, and A. Üngör (2002). Parallel Delaunay refinement: Algorithms and analyses. In *Proceedings of the Eleventh International Meshing Roundtable, International Journal of Computational Geometry & Applications (to appear)*, pages 205–217.

D. A. Spielman, S.-H. Teng, and A. Üngör (2004). Time complexity of practical parallel steiner point insertion algorithms. In *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 267–268.

G. Strang (1980). *Linear Algebra and its Application, 2nd. Ed.* Academic Press.

B. Sudakov and J. Vondrak (2005). How many random edges make a dense hypergraph non-2-colorable?. http://www.math.princeton.edu/~bsudakov/smoothed-analysis-hyper.pdf.

S.-H. Teng and C. W. Wong (2000). Unstructured mesh generation: Theory, practice, and perspectives. *Int. J. Computational Geometry & Applications*, 10(3):227.

S.-H. Teng (1998). Coarsening, sampling, and smoothing: Elements of the multilevel method. In R. S. Schreiber M. Heath, A Ranade, editor, *Algorithms for Parallel Processing, volume 105*, pages 247 – 276. volume 105 of IMA Volumes in Mathematics and its Applications, Springer.

M. J. Todd (1986). Polynomial expected behavior of a pivoting algorithm for linear complementarity and linear programming problems. *Mathematical Programming*, 35:173–192.

M. J. Todd (1991). Probabilistic models for linear programming. *Mathematics of Operations Research*, 16(4):671–693.

M. J. Todd and Y. Ye (1996). A lower bound on the number of iterations of long-step and polynomial interior-point methods for linear programming. *Annals of Operations Research*, 62:233–252.

M. Todd (2001). Semidefinite optimization. *Acta Numerica*, 10:515–560.

M. J. Todd (1994). A lower bound on the number of iterations of primal-dual interior-point methods for linear programming. In G. A. Watson and D. F. Griffiths, editors, *Numerical Analysis 1993*, pages 237 – 259. Longman Press, Harlow.

L. N. Trefethen and D. Bau (1997). *Numerical Linear Algebra*. SIAM, Philadelphia, PA.

L. N. Trefethen and R. S. Schreiber (1990). Average-case stability of Gaussian elimination. *SIAM J. on Matrix Analysis and Applications*, 11(3):335–360.

L. Trevisan and F. Xhafa (1998). The parallel complexity of positive linear programming. *Parallel Processing Letters*, 8(4):527–533.

P. M. Vaidya (1987). An algorithm for linear programming which requires $O(((m+n)n^2+(m+n)^{1.5}n)L)$ arithmetic operations. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 29–38.

V. V. Vazirani (2001). *Approximation algorithms*. Springer-Verlag.

J. Vera (1996). Ill-posedness and the complexity of deciding existence of solutions to linear programs. *SIAM J. on Optimization*, 6(3).

J. H. Wilkinson (1961). Error analysis of direct methods of matrix inversion. *J. ACM.*, 8:261–330.

J. H. Wilkinson (1963). *Rounding Errors in Algebraic Processes*.

J. H. Wilkinson (1988). *The algebraic eigenvalue problem*. Oxford University Press.

S. J. Wright (1993). A collection of problems for which gaussian elimination with partial pivoting is unstable. *SIAM J. Sci. Comput.*, 14(1):231–238.

S. J. Wright (1997). *Primal-dual interior-point methods*. Society for Industrial and Applied Mathematics, Philadelphia.

M. Wschebor (2004). Smoothed analysis of $\kappa(\mathbf{a})$. *J. of Complexity*, 20(1):97–107, February.

Y. Ye (1997). *Interior point algorithms: theory and analysis*. John Wiley & Sons.

M. Ziegelmann (2001). *Constrained Shortest Paths and Related Problems*. PhD thesis, Universität des Saarlandes, July.