Smoothed Complexity Theory

MARKUS BLÄSER, Saarland University
BODO MANTHEY, University of Twente

Smoothed analysis is a new way of analyzing algorithms introduced by Spielman and Teng (*J. ACM*, 2004). Classical methods like worst-case or average-case analysis have accompanying complexity classes, like P and Avg-P, respectively. While worst-case or average-case analysis give us a means to talk about the running time of a particular algorithm, complexity classes allows us to talk about the inherent difficulty of problems.

Smoothed analysis is a hybrid of worst-case and average-case analysis and compensates some of their drawbacks. Despite its success for the analysis of single algorithms and problems, there is no embedding of smoothed analysis into computational complexity theory, which is necessary to classify problems according to their intrinsic difficulty.

We propose a framework for smoothed complexity theory, define the relevant classes, and prove some first hardness results (of bounded halting and tiling) and tractability results (binary optimization problems, graph coloring, satisfiability) within this framework.

1. INTRODUCTION

The goal of computational complexity theory is to classify computational problems according to their intrinsic difficulty. While the analysis of algorithms is concerned with analyzing, say, the running time of a particular algorithm, complexity theory rather analyses the amount of resources that all algorithms need at least to solve a given problem.

Classical complexity classes, like P, reflect worst-case analysis of algorithms. A problem is in P if there is an algorithm whose running time on all inputs of length n is bounded by a polynomial in n. Worst-case analysis has been a success story: The bounds obtained are valid for every input of a given size, and, thus, we do not have to think about typical instances of our problem. If an algorithm has a good worst-case upper bound, then this is a very strong statement, and, up to constants and lower order terms in the running-time, the algorithm should also perform well in practice.

However, some algorithms work well in practice despite having a provably high worst-case running time. The reason for this is that the worst-case running time can be dominated by a few pathological instances that rarely or never occur in practice. An alternative to worst-

---

case analysis is average-case analysis. Many of the algorithms with poor worst-case but good practical performance have a good average running time. This means that the expected running time with instances drawn according to some fixed probability distribution is low.

In complexity-theoretic terms, P is the class of all problems that can be solved with polynomial worst-case running time. In the same way, the class Avg-P is the class of all problems that have polynomial average-case running time. Average-case complexity theory studies the structural properties of average-case running time. Bogdanov and Trevisan [2006] give a comprehensive survey of average-case complexity.

While worst-case complexity has the drawback of being often pessimistic, the drawback of average-case analysis is that random instances have often very special properties with high probability. These properties of random instances distinguish them from typical instances. Since a random and a typical instance is not the same, a good average-case running time does not necessarily explain a good performance in practice. In order to get a more realistic performance measure, (and, in particular, to explain the speed of the simplex method), Spielman and Teng [2004] have proposed a new way to analyze algorithms called *smoothed analysis*. In smoothed analysis, an adversary chooses an instance, and then this instance is subjected to a slight random perturbation. We can think of this perturbation as modeling measurement errors or random noise or the randomness introduced by taking, say, a random poll. The perturbation is controlled by some parameter ϕ, called the *perturbation parameter*. Spielman and Teng [2004] have proved that the simplex method has a running time that is polynomial in the size of the instance and the perturbation parameter. (More precisely, for any given instance, the expected running time on the perturbed instance is bounded by a polynomial.) Since then, the framework of smoothed analysis has been applied successfully to a variety of algorithms that have a good behavior in practice (and are therefore widely used) but whose worst-case running time indicates poor performance [Arthur et al. 2011; Banderier et al. 2003; Beier and Vöcking 2004; Bläser et al. 2013; Blum and Dunagan 2002; Brunsch et al. 2013a; 2013b; Brunsch and Röglin 2015; Damerow et al. 2012; Englert et al. 2014; Fouz et al. 2012; Manthey and Reischuk 2007; Manthey and Röglin 2013; Manthey and Veenstra 2013; Moitra and O'Donnell 2011; Spielman and Teng 2003; Vershynin 2009]. We refer to two surveys for a broader picture of smoothed analysis [Spielman and Teng 2009; Manthey and Röglin 2011].

However, with only few exceptions [Beier and Vöcking 2006; Röglin and Vöcking 2007], smoothed analysis has only been applied yet to single algorithms or single problems. Furthermore, with the exception of the complexity of finding Nash equilibria [Chen et al. 2009] and integer programming [Beier and Vöcking 2006], there are very few hardness results. Up to our knowledge, there is currently no attempt to formulate a smoothed complexity theory and, thus, to embed smoothed analysis into computational complexity.

This paper is an attempt to define a smoothed complexity theory, including notions of intractability, reducibility, and completeness. We define the class Smoothed-P (Section 2), which corresponds to problems that can be solved smoothed efficiently, we provide a notion of reducibility (Section 3), and define the class Dist-NP$_{para}$, which is a smoothed analogue of NP, and prove that it contains complete problems (Section 4). We continue with some basic observations (Section 5). We also add examples of problems in Smoothed-P (Sections 6 and 7) and discuss the relationship of smoothed complexity to semi-random models (Section 8). Finally, we conclude with a discussion of extension, shortcomings, and difficulties of our definitions (Section 9).

2. SMOOTHED POLYNOMIAL TIME AND SMOOTHED-P

2.1. Basic Definitions

In the first application of smoothed analysis to the simplex method [Spielman and Teng 2004], the strength of the perturbation has been controlled in terms of the standard deviation

σ of the Gaussian perturbation. While this makes sense for numerical problems, this model cannot be used for general (discrete problems). A more general form of perturbation models has been introduced by Beier and Vöcking [2004]: Instead of specifying an instance that is afterwards perturbed (which can also be viewed as the adversary specifying the mean of the probability distribution according to which the instances are drawn), the adversary specifies the whole probability distribution. Now the role of the standard deviation σ is taken over by the parameter ϕ, which is an upper bound for the maximum density of the probability distributions. For Gaussian perturbation, we have $\sigma = \Theta(1/\phi)$. Because we do not want to restrict our theory to numerical problems, we have decided to use the latter model.

Let us now define our model formally.

For a family $\mathcal{D} = (D_{n,x,\phi})$, let

$$S_{n,x} = \big\{ y \mid D_{n,x,\phi}(y) > 0 \text{ for some } \phi \big\},$$

and let $N_{n,x} = |S_{n,x}|$. In principle, $S_{n,x}$ can be the set of all strings of length bounded by a polynomial in n. However, for many problems, not all strings are meaningful encodings of instances. In this case, $N_{n,x}$ is the number of possible meaningful perturbations of x.

A perturbation model is a family $\mathcal{D} = (D_{n,x,\phi})$ of distributions that satisfies the following properties for all $n \in \mathbb{N}$, $x \in \{0,1\}^*$, and $\phi \in [0,1]$:

(1) The length of x is n (we could omit the index n, but we keep it for clarity). Note that length does not necessarily mean bit length, but depends on the problem. For instance, it can be the number of vertices of the graph encoded by x.
(2) The support of $D_{n,x,\phi}$ is a subset of $\{0,1\}^{\leq \mathrm{poly}(n)}$.
(3) For all n, x, ϕ, and y, we require $D_{n,x,\phi}(y) \leq \phi$. This controls the strength of the perturbation and restricts the adversary.
(4) We allow $\phi \in [1/N_{n,x}, 1]$. Furthermore, the range of ϕ are discretized, so that they can be described by at most $\mathrm{poly}(n)$ bits.

We call such families $(D_{n,x,\phi})_{n,x,\phi}$ of probability distributions *parameterized families of distributions*.

The case $\phi = 1$ corresponds to the worst-case complexity; we can put all the mass on one string. The case $\phi = 1/N_{n,x}$ models the standard average case; here we usually have to put probability on an exponentially large set of strings. In general, the larger ϕ, the more powerful the adversary.

Now we can specify what it means that an algorithm has smoothed polynomial running-time. The following definition can also be viewed as a discretized version of Beier and Vöcking's definition [Beier and Vöcking 2006]. Note that we do not speak about expected running-time, but about expected running-time to some power ε. This is because the notion of expected running-time is not robust with respect to, e.g., quadratic slowdown. The corresponding definition for average-case complexity is due to Levin [1986]. We refer to Bogdanov and Trevisan [2006] for a thorough discussion of this issue.

Definition 2.1.   An algorithm A has *smoothed polynomial running time* with respect to the family $\mathcal{D}$ if there exists an $\varepsilon > 0$ such that, for all n, x, and ϕ, we have

$$\mathbb{E}_{y \sim D_{n,x,\phi}}\big(t_A(y; n, \phi)^\varepsilon\big) = O\big(n \cdot N_{n,x} \cdot \phi\big).$$

This definition implies that (average-)polynomial time is only required if we have $\phi = O(\mathrm{poly}(n)/N_{n,x})$. For the case $\phi = 1$, an exponential running-time is allowed if $N_{n,x}$ is exponential, which it usually is. This seems to be quite generous at first glance, but it is in accordance with, e.g., Spielman and Teng's analysis of the simplex method [Spielman and Teng 2004] or Beier and Vöcking's analysis of integer programs [Beier and Vöcking 2006]. They achieve polynomial running time only if they perturb all but at most $O(\log n)$ digits per coefficient: If we perturb a number with, say, a Gaussian of standard deviation

$\sigma = 1/\operatorname{poly}(n)$, then we expect that the $O(\log n)$ most significant bits remain untouched, but the less significant bits are random. Of course, there remain roughly $O(\log n)$ bits per coefficient untouched, while our perturbation models leaves in this case only roughly $O(\log n)$ bits overall untouched. However, the former can only be achieved if we include, e.g., the notion of coefficients in the input. For the purpose of generality, we decided against including notions such as coefficients in the input. We also refer to Section 6.2 for a discussion of this matter.

In average-case complexity, one considers not decision problems alone, but decision problems together with a probability distribution. The smoothed analogue of this is that we consider tuples $(L, \mathcal{D})$, where $L \subseteq \{0,1\}^*$ is a decision problem and $\mathcal{D}$ is a parameterized family of distributions. We call such problems *parameterized distributional problems*. The notion of smoothed polynomial running-time (Definition 2.1) allows us to define what it means for a parameterized distributional problem to have polynomial smoothed complexity.

Definition 2.2. Smoothed-P is the class of all $(L, \mathcal{D})$ such that there is a deterministic algorithm A with smoothed polynomial running time that decides L.

We start with an alternative characterization of smoothed polynomial time as it is known for average case running time. It basically says that an algorithm has smoothed polynomial running-time if and only if its running-time has polynomially decreasing tail bounds. Though smoothed polynomial time is a generalization of average case polynomial time, the characterization and the proof of equivalence are similar.

THEOREM 2.3. *An algorithm A has smoothed polynomial running time if and only if there is an $\varepsilon > 0$ and a polynomial p such that for all n, x, ϕ, and t,*

$$\Pr_{y \sim D_{n,x,\phi}}[t_A(y; n, \phi) \geq t] \leq \frac{p(n)}{t^{\varepsilon}} \cdot N_{n,x} \cdot \phi.$$

PROOF. Let A be an algorithm whose running time t_A fulfills

$$\mathbb{E}_{y \sim D_{n,x,\phi}}\left(t_A(y; n, \phi)^{\varepsilon}\right) = O\left(n \cdot N_{n,x}\phi\right).$$

The probability that the running time exceeds a certain value t can be bounded by Markov's inequality:

$$\mathbb{P}(t_A(y; n, \phi) \geq t) = \mathbb{P}\left(t_A(y; n, \phi)^{\varepsilon} \geq t^{\varepsilon}\right)$$

$$\leq \frac{\mathbb{E}_{y \sim D_{n,x,\phi}}\left(t_A(y; n, \phi)^{\varepsilon}\right)}{t^{\varepsilon}} = O\left(n \cdot N_{n,x}\phi \cdot t^{-\varepsilon}\right).$$

For the other direction, assume that

$$\Pr_{y \sim D_{n,x,\phi}}[t_A(y; n, \phi) \geq t] \leq \frac{n^c}{t^{\varepsilon}} \cdot N_{n,x}\phi$$

for some constants c and ε. Let $\varepsilon' = \varepsilon/(c+2)$. Then we have

$$\mathbb{E}_{y \sim D_{n,x,\phi}}\left(t_A(y; n, \phi)^{\varepsilon'}\right) = \sum_t \mathbb{P}(t_A(y; n, \phi)^{\varepsilon'} \geq t)$$

$$\leq n + \sum_{t \geq n} \mathbb{P}(t_A(y; n, \phi) \geq t^{1/\varepsilon'})$$

$$\leq n + \sum_{t \geq n} t^{-2} \cdot N_{n,x}\phi = n + O(N_{n,x}\phi).$$

□

2.2. Heuristic Schemes

A different way to think about efficiency in the smoothed setting is via so-called heuristic schemes. This notion comes from average-case complexity [Bogdanov and Trevisan 2006], but can be adapted to our smoothed setting. The notion of a heuristic scheme comes from the observation that, in practice, we might only be able to run our algorithm for a polynomial number of steps. If the algorithms does not succeed within this time bound, then it "fails", i.e., it does not solve the given instance. The failure probability decreases polynomially with the running time that we allow. The following definition captures this.

Definition 2.4. Let $(L, \mathcal{D})$ be a smoothed distributional problem. An algorithm A is an errorless heuristic scheme for $(L, \mathcal{D})$ if there is a polynomial q such that

(1) For every n, x, ϕ, $\delta > 0$, and $y \in \operatorname{supp} D_{n,x,\phi}$, we have $A(y; n, \phi, \delta)$ outputs either $L(y)$ or $\perp$.
(2) For n, x, ϕ, $\delta > 0$, and $y \in \operatorname{supp} D_{n,x,\phi}$, we have $t_A(y; n, \delta) \leq q(n, N_{n,x}\phi, 1/\delta)$.
(3) For n, x, ϕ, $\delta > 0$, and $y \in \operatorname{supp} D_{n,x,\phi}$, we have $\Pr_{y \sim D_{n,x,\phi}}[A(y; n, \phi, \delta) = \perp] \leq \delta$.

With the definition of a heuristic scheme, we can prove that heuristic schemes precisely characterize Smoothed-P.

THEOREM 2.5. $(L, \mathcal{D}) \in$ Smoothed-P *if and only if* $(L, \mathcal{D})$ *has an errorless heuristic scheme.*

PROOF.   Let A be an algorithm for $(L, \mathcal{D})$. By Theorem 2.3, the probability that

$$\mathbb{P}(t_A(y; n, \phi) \geq t) = O\left(n \cdot N_{n,x}\phi \cdot t^{-\varepsilon}\right).$$

We get an errorless heuristic scheme B from A as follows: Simulate A for $(n \cdot N_{n,x}\phi/\delta)^{1/\varepsilon}$ steps. If A stops within these number of steps, then output whatever A outputs. Otherwise, output $\perp$. By the choice of the parameters, the probability that B outputs $\perp$ is bounded by δ.

For the other direction, let A be an errorless heuristic scheme for $(L, \mathcal{D})$. We get an algorithm with smoothed polynomial running time by first running A with $\delta = 1/2$, then with $\delta = 1/4$, and in the ith iteration with $\delta = 1/2^i$. Whenever A does not answer $\perp$, B gives the same answer and stops. B will eventually stop, when $\delta < D_{n,x,\phi}(y)$. For i iterations, B needs

$$\sum_{j=1}^{i} q(n, N_{n,x}\phi, 2^j) \leq \operatorname{poly}(n, N_{n,x}\phi) \cdot 2^{ci}$$

for some constant c. B stops after i iterations for all but a 2^{-i} fraction of the input. Thus B has smoothed polynomial running time. (Choose $\varepsilon < 1/c$.)   □

2.3. Alternative Definition: Bounded Moments

At first glance, one might be tempted to use "expected running time" for the definition of Avg-P and Smoothed-P. However, as mentioned above, simply using the expected running time does not yield a robust measure. This is the reason why the expected value of the running time raised to some (small) constant power is used. Röglin and Teng [2009, Theorem 6.2] have shown that for integer programming (more precisely, for binary integer programs with a linear objective function), the expected value indeed provides a robust measure. They have proved that a binary optimization problem can be solved in expected polynomial time if and only if it can be solved in worst-case pseudo-polynomial time. The reason for this is that all finite moments of the Pareto curve are polynomially bounded. Thus, a polynomial slowdown does not cause the expected running time to jump from polynomial to exponential.

As far as we are aware, this phenomenon, i.e., the case that all finite moments have to be bounded by a polynomial, has not been studied yet in average-case complexity. Thus, for completeness, we define the corresponding average-case and smoothed complexity classes as an alternative to Avg-P and Smoothed-P.

Definition 2.6.

(1) An algorithm has *robust smoothed polynomial running time* with respect to $\mathcal{D}$ if, for all fixed $c > 0$ and for every n, x, and ϕ, we have
$$\mathbb{E}_{y \sim D_{n,x,\phi}}\big(t_A(y; n, \phi)^c\big) = O\big(n \cdot N_{n,x} \cdot \phi\big).$$
Smoothed-PBM is the class of all $(L, \mathcal{D})$ for which there exists a deterministic algorithm with robust smoothed polynomial running time. (The "PBM" stands for "polynomially bounded moments".)

(2) An algorithm A has *robust average polynomial running time* with respect to $\mathcal{D}$ if, for all fixed $c > 0$ and for all n, we have $\mathbb{E}_{y \sim D_n}\big(t_A(y)^c\big) = O(n)$. Avg-PBM contains all $(L, \mathcal{D})$ for which there exists a deterministic algorithm with robust smoothed polynomial running time.

Instead of having multi-linear right-hand sides, we could choose any polynomial dependence on the parameters since the bounds for the moments have to hold for all c.

From the definition, we immediately get Smoothed-PBM $\subseteq$ Smoothed-P and Avg-PBM $\subseteq$ Avg-P. Moreover, if $L \in$ P, then L together with any family of distributions is also in Smoothed-P and Avg-P and also in Smoothed-PBM and Avg-PBM. From Röglin and Teng's result [Röglin and Teng 2009], one might suspect Avg-P = Avg-PBM and Smoothed-P = Smoothed-PBM, but this does not hold.

THEOREM 2.7.   Avg-PBM $\subsetneq$ Avg-P *and* Smoothed-PBM $\subsetneq$ Smoothed-P.

PROOF. We only prove the theorem for average-case complexity. The proof for the smoothed complexity case is almost identical.

By the time hierarchy theorem [Arora and Barak 2009], there is a language $L' \in$ DTime(2^n) such that $L' \notin$ DTime$(2^{o(n)})$. Consider the following language $L = \{x0^n \mid |x| = n, x \in L'\}$. Let $\mathcal{D}' = (D'_n)$ be a hard probability distribution for L', i.e., $(L', \mathcal{D}')$ is as hard to solve as L' in the worst case [Li and Vitányi 1992; Li and Vitányi 1993].

Let $\mathcal{D} = (D_n)$ be given as follows:
$$D_n(xy) = \begin{cases} 2^{-n} \cdot D'_n(x) & \text{if } |x| = n \text{ and } y = 0^n \text{ and} \\ 2^{-2n} & \text{otherwise.} \end{cases}$$

Since $L' \in$ DTime(2^n), we have $(L, \mathcal{D}) \in$ Avg-P: L can be decided in expected time $2^{-n} \cdot 2^n + O(n) = O(n)$. Now we prove that $(L, \mathcal{D}) \notin$ Avg-PBM. If $(L, \mathcal{D}) \in$ Avg-PBM were true, then $2^{-n} \cdot \mathbb{E}_{x \sim D'_n}(t(x)^c)$ would be bounded by a polynomial for all fixed c. Here, t is the time needed to solve the L' instance x.

Our choice of $\mathcal{D}'$, Jensen's inequality, and the fact that $L' \notin$ DTime$(2^{o(n)})$ imply that $\mathbb{E}_{x \sim D'_n}(t(x)^c) \geq \mathbb{E}_{x \sim D'_n}(t(x))^c = 2^{c \cdot \Omega(n)}$. Thus, for some sufficiently large c, $2^{-n} \cdot \mathbb{E}_{x \sim D'_n}(t(x)^c)$ exceeds any polynomial.   □

A natural question to ask is: Does there exist a language $L \in$ NP together with some ensemble $\mathcal{D}$ such that $(L, \mathcal{D})$ separates Avg-P from Avg-PBM and Smoothed-P from Smoothed-PBM? Does there exist some L together with a computable ensemble $\mathcal{D}$ that separates these classes?

We conjecture that, assuming the exponential time hypothesis (ETH) [Impagliazzo et al. 2001], such an $L \in$ NP exists to separate Avg-P from Avg-PBM and Smoothed-P from Smoothed-PBM. Given the ETH, 3SAT requires time $2^{\Omega(n)}$ in the worst case, thus also on

average if we use the universal distribution. This holds even if we restrict 3SAT to $O(n)$ clauses. However, n is here the number of variables, not the bit length of the input, which is roughly $\Theta(n \log n)$. Thus, a direct application of the ETH seems to be impossible here, as, with respect to bit length n, the ETH states only that satisfiability cannot be solved in time $2^{o(n/\log n)}$, which seems to be too weak.

3. DISJOINT SUPPORTS AND REDUCIBILITY

The same given input y can appear with very high and with very low probability at the same time. What sounds like a contradiction has an easy explanation: $D_{n,x,\phi}(y)$ can be large whereas $D_{n,x',\phi}(y)$ for some $x' \neq x$ is small. But if we only see y, we do not know whether x or x' was perturbed. This may cause some problems when one wants to develop a notion of reduction and completeness.

For a parameterized distributional problem $(L, \mathcal{D})$, let

$$L_{\mathrm{ds}} = \{\langle x, y\rangle \mid y \in L \text{ and } |y| \leq \mathrm{poly}(|x|)\}.$$

The length of $|y|$ is bounded by the same polynomial that bounds the length of the strings in any $\mathrm{supp}\, D_{n,x,\phi}$. We will interpret a pair $\langle x, y\rangle$ as "y was drawn according to $D_{n,x,\phi}$". With the notion of L_{ds}, we can now define a reducibility between parameterized distributional problems. We stress that, although the definition below involves L_{ds} and L'_{ds}, the reduction is defined for pairs L and L' and neither of the two is required to be a disjoint-support language. This means that, for $(L, \mathcal{D})$, the supports of $D_{n,x,\phi}$ for different x may intersect. And the same is allowed for $(L', \mathcal{D}')$.

Definition 3.1. Let $(L, \mathcal{D})$ and $(L', \mathcal{D}')$ be two parameterized distributional problems. $(L, \mathcal{D})$ reduces to $(L', \mathcal{D}')$ (denoted by "$(L, \mathcal{D}) \leq_{\mathrm{smoothed}} (L', \mathcal{D}')$") if there is a polynomial time computable function f such that for every n, x, ϕ and $y \in \mathrm{supp}\, D_{n,x,\phi}$ the following holds:

(1) $\langle x, y\rangle \in L_{\mathrm{ds}}$ if and only if $f(\langle x, y\rangle; n, \phi) \in L'_{\mathrm{ds}}$.
(2) There exist polynomials p and m such that, for every n, x, ϕ, and every $y' \in \mathrm{supp}\, D'_{m(n),f_1(\langle x,y\rangle;n,\phi),\phi}$, we have

$$\sum_{y: f_2(\langle x,y\rangle;n,\phi)=y'} D_{n,x,\phi}(y) \leq p(n) D'_{m(n),f_1(\langle x,y\rangle;n,\phi),\phi}(y'),$$

where $f(\langle x, y\rangle; n, \phi) = \langle f_1(\langle x, y\rangle; n, \phi), f_2(\langle x, y\rangle; n, \phi)\rangle$.

Remark 3.2. We could also allow that ϕ on the right-hand side is polynomially transformed. However, we currently do not see how to benefit from this.

It is easy to see that $\leq_{\mathrm{smoothed}}$ is transitive. Ideally, Smoothed-P should be closed under this type of reductions. However, we can only show this for the related class of problems with disjoint support.

Definition 3.3. Smoothed-P$_{\mathrm{ds}}$ is the set of all distributional problems with disjoint supports such that there is an algorithm A for L_{ds} with smoothed polynomial running time. (Here, the running time on $\langle x, y\rangle$ is defined in the same way as in Definition 2.1. Since $|y| \leq \mathrm{poly}(|x|)$ for a pair $\langle x, y\rangle \in L_{\mathrm{ds}}$, we can as well measure the running time in $|x|$.)

Now, Smoothed-P$_{\mathrm{ds}}$ is indeed closed under the above type of reductions.

THEOREM 3.4. *If* $(L, \mathcal{D}) \leq_{\mathrm{smoothed}} (L', \mathcal{D}')$ *and* $(L'_{\mathrm{ds}}, \mathcal{D}') \in$ Smoothed-P$_{\mathrm{ds}}$, *then* $(L_{\mathrm{ds}}, \mathcal{D}) \in$ Smoothed-P$_{\mathrm{ds}}$.

PROOF. Let A' be a an errorless heuristic scheme for $(L'_{\mathrm{ds}}, \mathcal{D}')$. Let f be the reduction from $(L, \mathcal{D})$ to $(L', \mathcal{D}')$ and let p and m be the corresponding polynomials.

We claim that $A(\langle x, y \rangle; n, \phi, \delta) = A'(f(\langle x, y \rangle; n, \phi); m(n), \phi, \delta/p(n))$ is an errorless heuristic scheme for $(L_{\mathrm{ds}}, \mathcal{D})$. To prove this, let

$$B = \{y' \in \operatorname{supp} D'_{m(n), f_1(\langle x, y \rangle; n, \phi), \phi} \mid A'(\langle f(\langle x, y \rangle; n, \phi), y' \rangle; m(n), \phi, \delta/p(n)) = \bot\}$$

be the set of string on which A' fails.

Because A' is a heuristic scheme, we have $D'_{m(n), f_1(\langle x, y \rangle; n, \phi), \phi}(B) \leq \delta/p(n)$. Therefore,

$$\Pr_{y \sim D_{n,x,\phi}} (A(\langle x, y \rangle; n, \phi, \delta) = \bot) = \Pr_{y \sim D_{n,x,\phi}} (A'(f(\langle x, y \rangle; n, \phi); m(n), \phi, \delta/p(n) = \bot)$$

$$= \sum_{y: f_2(\langle x, y \rangle; n, \phi) \in B} D_{n,x,\phi}(y)$$

$$\leq \sum_{y' \in B} p(n) D'_{m(n); f_1(\langle x, y \rangle; n, \phi); \phi}(y')$$

$$= p(n) D'_{m(n); f_1(\langle x, y \rangle; n, \phi); \phi}(B) \leq \delta.$$

Thus, $(L_{\mathrm{ds}}, \mathcal{D}) \in \mathsf{Smoothed\text{-}P_{ds}}$. $\square$

With the definition of disjoint support problems, a begging question is how the complexity of L and L_{ds} are related. It is obvious that $(L, \mathcal{D}) \in \mathsf{Smoothed\text{-}P}$ implies $(L_{\mathrm{ds}}, \mathcal{D}) \in \mathsf{Smoothed\text{-}P_{ds}}$. However, the converse is not so obvious. The difference between L and L_{ds} is that for L_{ds}, we get the x from which the input y was drawn. While this extra information does not seem to be helpful at a first glance, we can potentially use it to extract randomness from it. So this question is closely related to the problem of derandomization.

But there is an important subclass of problems in $\mathsf{Smoothed\text{-}P_{ds}}$ whose counterparts are in $\mathsf{Smoothed\text{-}P}$, namely those which have an oblivious algorithm with smoothed polynomial running time. We call an algorithm (or heuristic scheme) for some problem with disjoint supports *oblivious* if the running time on $\langle x, y \rangle$ does not depend on x (up to constant factors). Let $\mathsf{Smoothed\text{-}P_{ds}^{obl}}$ be the resulting subset of problems in $\mathsf{Smoothed\text{-}P_{ds}}$ that have such an oblivious algorithm with smoothed polynomial running time.

PROPOSITION 3.5. *For any parameterized problem* $(L, \mathcal{D})$, $(L, \mathcal{D}) \in \mathsf{Smoothed\text{-}P}$ *if and only if* $(L_{\mathrm{ds}}, \mathcal{D}) \in \mathsf{Smoothed\text{-}P_{ds}^{obl}}$.

PROOF. Let A be an oblivious algorithm with smoothed polynomial running time for L_{ds}. Since A is oblivious, we get an algorithm for L with the same running time (up to constant factors) by running A on $\langle x_0, y \rangle$ on input y, where x_0 is an arbitrary string of length n. $\square$

Note that almost all algorithms, for which a smoothed analysis has been carried out, the algorithm does not know the x from which y was drawn. This means that these algorithms are oblivious. Still, this does not rule out the possibility that there is a problem $(L, \mathcal{D}) \notin \mathsf{Smoothed\text{-}P}$ with $(L_{\mathrm{ds}}, \mathcal{D}) \in \mathsf{Smoothed\text{-}P_{ds}}$.

What could be a potential attempt to find such a problem? Note that in L_{ds}, each y is paired with *every* x, so there is no possibility to encode information by omitting some pairs. This prohibits attempts for constructing such a problem like considering pairs $\langle x, f(x) \rangle$ where f is some one-way function. However, a pair $\langle x, y \rangle$ contains randomness that one could extract. Therefore, such an attempt seems to be connected to derandomisation issues, a notoriously hard problem in complexity theory. To avoid the derandomisation problem, we could look at probabilistic or nonuniform smoothed classes: For the classes $\mathsf{Smoothed\text{-}BPP}$ or $\mathsf{Smoothed\text{-}P/poly}$, which can be defined in the obvious way, it seems plausible that knowing x does not help; for these two classes, it should be possible to use the internal random bits (or the advice) to find an x' that is good enough.

4. PARAMETERIZED DISTRIBUTIONAL NP

4.1. Dist-NP$_{\text{para}}$

In this section, we define the smoothed analogue of the worst-case class NP and the average-case class DistNP [Levin 1986; Gurevich 1991]. First, we have to restrict ourself to "natural" distributions. This rules out, for instance, probability distributions based on Kolmogorov complexity that (the *universal distribution*), under which worst-case complexity equals average-case complexity for all problems [Li and Vitányi 1992]. We transfer the notion of computable ensembles to smoothed complexity.

Definition 4.1. A parameterized family of distributions is in PComp$_{\text{para}}$ if the cumulative probability

$$F_{D_{n,x,\phi}} = \sum_{z \leq x} D_{n,x,\phi}$$

can be computed in polynomial time (given n, x and ϕ in binary). Here $z \leq x$ means that $|z| < |x|$ or $|z| = |x|$ and z comes lexigraphically before x or $z = x$.

With this notion, we can define the smoothed analogue of NP and DistNP.

Definition 4.2. Dist-NP$_{\text{para}} = \{(L, \mathcal{D}) \mid L \in$ NP and $\mathcal{D} \in$ PComp$_{\text{para}}\}$.

4.2. Dist-NP$_{\text{para}}$-Complete Problems

4.2.1. Bounded Halting. Having defined Dist-NP$_{\text{para}}$ in the previous section, we now prove that *bounded halting* – given a Turing machine, an input, and a running-time bound, does the Turing machine halt on this input within the given time bound – is complete for Dist-NP$_{\text{para}}$. Bounded halting is the canonical NP-complete language, and it has been the first problem that has been shown to be Avg-P-complete [Levin 1986]. Formally, let

$$\text{BH} = \{\langle g, x, 1^t \rangle \mid \text{ the nondeterministic Turing machine with Gödel number } g$$
$$\text{accepts } x \text{ within } t \text{ steps}\}.$$

In order to show that BH is Dist-NP$_{\text{para}}$-complete, we need a "compression function" for probability distributions [Bogdanov and Trevisan 2006]. This is the purpose of the following lemma.

LEMMA 4.3. *Let $\mathcal{D} = (D_{n,x,\phi}) \in$ PComp$_{\text{para}}$ be an ensemble. There exists a deterministic algorithm C such that the following holds:*

(1) $C(y; n, x, \phi)$ runs in time polynomial in n and ϕ for all $y \in \text{supp } D_{n,x,\phi}$.
(2) For every $y, y' \in \text{supp } D_{n,x,\phi}$, $C(y; n, x, \phi) = C(y'; n, x, \phi)$ implies $y = y'$.
(3) If $D_{n,x,\phi}(y) < 2^{-|y|}$, then $|C(y; n, x, \phi)| = 1 + |y|$. Else, $|C(y; n, x, \phi)| = \log \frac{1}{D_{n,x,\phi}(y)} + c \cdot \log n + 1$ for some constant c.

PROOF. Consider any string $y \in \text{supp}(D_{n,x,\phi})$. If $D_{n,x,\phi}(y) \leq 2^{-|y|}$, then we let $C(y; n, x, \phi) = 0y$. If $D_{n,x,\phi}(y) > 2^{-|x|}$, then let y' be the string that precedes y in lexicographic order, and let $p = F_{D_{n,x,\phi}}(y')$. Then we set $C(y; n, x, \phi) = 1a$, where a is the longest common prefix of the binary representation of p and $F_{D_{n,x,\phi}}(y) = p + D_{n,x,\phi}(y)$. Since $\mathcal{D} \in$ PComp$_{\text{para}}$, the string z can be computed in polynomial time. Thus, C can be computed in polynomial time. (This also shows that $|C(x, n)|$ is bounded by a polynomial in $|x|$.)

We have $D_{n,x,\phi}(y) \leq 2^{-|a|}$, since adding $D_{n,x,\phi}(y)$ leaves the first $|a|$ bits of p unchanged.

Let z be another string, z' its predecessor and b the longest common prefix of $q = F_{D_{n,x,\phi}}(z')$ and $q + D_{n,x,\phi}(z')$. The intervals $[p, p + D_{n,x,\phi}(y))$ and $[q, q + D_{n,x,\phi}(y))$ are

disjoint by construction. Therefore, a and b have to be different, because otherwise these intervals would intersect.

Let c be such that $|y| \le n^c$ for all $y \in \mathrm{supp}(D_{n,x,\phi})$. We set

$$C(y; n, x, \phi) = 1 \, \mathrm{bin}(|a|) a 0^{\log \frac{1}{D_{n,x,\phi}(y)} - |a|}.$$

(Note that $\log \frac{1}{D_{n,x,\phi}(y)} \ge |a|$.) Here $\mathrm{bin}(|a|)$ is a fixed length binary encoding of a. We can bound this length by $c \log n$. The total length of $C(y; n, x, \phi)$ is

$$|C(y; n, x, \phi)| = 1 + c \log n + \log \frac{1}{D_{n,x,\phi}(y)}.$$

It remains to be proved that C is injective. Let $C(y; n, x, \phi) = C(z; n, x, \phi)$. If $C(y; n, x, \phi)$ starts with a 0, then obviously $y = z$. If $C(y; n, x, \phi)$ starts with a 1, then the prefixes a and b are the same. Therefore $y = z$ by the consideration above. $\square$

The instances of BH are triples $\langle g, x, 1^t \rangle$ of length $2 \log |g| + 2 \log |x| + |x| + |g| + t + \Theta(1)$ using a standard pairing function for strings, where the $2 \log |g| + 2 \log |x|$ stems from encoding the lengths of g and of x. Note that the instances of BH can be made prefix-free. Let

$$U^{\mathrm{BH}}_{N, \langle g, x, 1^t \rangle, \phi}(\langle g', x', 1^{t'} \rangle) = \begin{cases} c_\phi \cdot 2^{-|x'|} & \text{if } g = g', \ N = |\langle g', x', 1^{t'} \rangle|, \text{ and } |x'| \ge \log \frac{1}{\phi}, \\ 0 & \text{otherwise.} \end{cases}$$

Above, c_ϕ is an appropriate scaling factor. More precisely, c_ϕ is the reciprocal of the number of possible lengths for a string x', i.e., it is of order $\frac{1}{N - \log \phi}$. In particular, $U^{\mathrm{BH}}_{N, \langle g, x, 1^t \rangle, \phi}(y) \le \phi$ for all y.

What is the intuition behind the definition of U^{BH}: First we want that tuples with Gödel number g should be perturbed to tuples with Gödel number $g' = g$ again. Second, the length of the perturbed tuples should be the same as the length of the original tuple. And third, the string x' should not be too short, since the length of the string produced by the compression function is not too short either.

THEOREM 4.4. $(\mathrm{BH}, U^{\mathrm{BH}})$ is $\mathrm{Dist\text{-}NP}_{\mathrm{para}}$-complete under polynomial-time smoothed reductions.

PROOF. Let $(L, \mathcal{D}) \in \mathrm{Dist\text{-}NP}_{\mathrm{para}}$ be arbitrary. Let $p(n)$ be an upper bound for the length of the strings in any $\mathrm{supp}(D_{n,x,\phi})$. Let M be a nondeterministic machine that accepts an input a if and only if there is a string $y \in L$ with $C(y; n, x, \phi) = a$. Let q be an upper bound on the running time of M. Let g be the Gödel number of M. Our reduction maps a string y to

$$f(\langle x, y \rangle; n, x, \phi) = \left\langle \langle g, C(x; n, x, \phi), 1^t \rangle, \langle g, C(y; n, x, \phi), 1^{t'} \rangle \right\rangle$$

where t and t' chosen such that, first, both are larger than $q(p(|x|))$ and, second, that all tuples have the same length $N(n)$.

The Turing machine M gets a "compressed" input a and behaves on it like on the "uncompressed" input y. Note that M can guess the uncompressed input y, since y is unique. By construction, $\langle x, y \rangle \in L_{\mathrm{ds}}$ if and only if $f(\langle x, y \rangle; n, x, \phi) \in \mathrm{BH}_{\mathrm{ds}}$.

Domination remains to be verified. Since C is injective, at most one y is mapped to $\langle g, a, 1^t \rangle$ given n, x, and ϕ. We have

$$U^{\mathrm{BH}}_{N, \langle g, C(x; n, x, \phi), 1^t \rangle, \phi}(\langle g, C(y; n, x, \phi), 1^{t'} \rangle) = c_\phi \cdot 2^{-|C(y; n, x, \phi)|}.$$

If $|C(y; n, x, \phi)| \le \log \frac{1}{D_{n,x,\phi}(y)} + c \log n + 1$, then

$$U^{\mathrm{BH}}_{N, \langle g, C(x; n, x, \phi), 1^t \rangle, \phi}(\langle g, C(y; n, x, \phi), 1^{t'} \rangle) \ge c_\phi \cdot \frac{D_{n,x,\phi}(x)}{2n^c}$$

and domination is fulfilled. If $|C(y; n, x, \phi)| = 1 + |y|$, then

$$U^{\mathrm{BH}}_{N, \langle g, C(x; n, x, \phi), 1^t \rangle, \phi}(\langle g, C(y; n, x, \phi), 1^{t'} \rangle) = c_\phi \cdot 2^{-|y|-1} \geq \frac{1}{2} c_\phi \cdot D_{n, x, \phi}(y).$$

This completes the hardness proof. The completeness follows since $(\mathtt{BH}, U^{\mathrm{BH}})$ is indeed contained in $\mathsf{Dist\text{-}NP}_{\mathrm{para}}$. $\square$

4.2.2. Tiling. The original DistNP-complete problem by Levin [1986] was $\mathtt{Tiling}$ (see also Wang [1997]): An instance of the problem consists of a finite set T of square tiles, a positive integer t, and a sequence $s = (s_1, \ldots, s_n)$ for some $n \leq t$ such that $s_i \in T$ for all $i \in \{1, \ldots, n\}$ and s_i matches s_{i+1} (the right side of s_i equals the left side of s_{i+1}). The question is whether s can be extended to tile an $n \times n$ square using tiles from T. Here, all four sides of a square tile are labeled, and tiles must not be rotated. Two tiles match horizontally if the label on the right-hand side of the left tile matches the label on the left-hand side of the right tile. Two tiles match vertically if the bottom label of the upper tile matches the top label of the lower tile.

We use the following probability distribution for $\mathtt{Tiling}$:

$$U^{\mathtt{Tiling}}_{N, \langle T, s, 1^t \rangle, \phi}(\langle T', s', 1^{t'} \rangle) = \begin{cases} c_\phi \cdot a^{-|s'|} & \text{if } T = T', \ N = |\langle T', s', 1^{t'} \rangle|, \ |T'| \geq \log \frac{1}{\phi}, \\ 0 & \text{otherwise.} \end{cases}$$

Here, a is the number of possible choices in T for each initial tile s_i.

THEOREM 4.5. $(\mathtt{Tiling}, U^{\mathtt{Tiling}})$ *is* $\mathsf{Dist\text{-}NP}_{\mathrm{para}}$-*complete for some* $U^{\mathtt{Tiling}} \in \mathsf{PComp}_{\mathrm{para}}$ *under polynomial-time smoothed reductions.*

PROOF. By construction, we have $(\mathtt{Tiling}, U^{\mathtt{Tiling}}) \in \mathsf{Dist\text{-}NP}_{\mathrm{para}}$. For simplicity, we assume that the set T of tiles always contains two tiles encoding the input bits "0" and "1" and that these are the only possible tiles for the initial tiling $(s_1, \ldots, s_n)$. (The problem does not become easier without this restriction, but the hardness proof becomes more technical.)

For the hardness, $(\mathtt{BH}, U^{\mathrm{BH}})$ reduces to $(\mathtt{Tiling}, U^{\mathtt{Tiling}})$ because the Turing machine computations can be encoded as tiling problems in a straightforward way [Wang 1997] (the Gödel number g maps to some set T of tiles, and the input x maps to the initial tiling s). Finally, Item 2 of the reduction (Definition 3.1) is fulfilled because of the similarity between the two probability distributions. $\square$

5. BASIC RELATIONS TO WORST-CASE COMPLEXITY

In this section, we collect some simple facts about $\mathsf{Smoothed\text{-}P}$ and $\mathsf{Dist\text{-}NP}_{\mathrm{para}}$ and their relationship to their worst-case and average-case counterparts. First, $\mathsf{Smoothed\text{-}P}$ is sandwiched between P and $\mathsf{Avg\text{-}P}$, which follows immediately from the definitions.

THEOREM 5.1. *If* $L \in \mathsf{P}$, *then* $(L, \mathcal{D}) \in \mathsf{Smoothed\text{-}P}$ *for any* $\mathcal{D}$. *If* $(L, \mathcal{D}) \in \mathsf{Smoothed\text{-}P}$ *with* $\mathcal{D} = (D_{n, x, \phi})_{n, x, \phi}$, *then* $(L, (D_{n, x_n, \phi})_n) \in \mathsf{Avg\text{-}P}$ *for* $\phi = O(\mathrm{poly}(n)/N_{n, x})$ *and every sequence* $(x_n)_n$ *of strings with* $|x_n| \leq \mathrm{poly}(n)$.

Second, for unary languages, i.e., languages $L \subseteq \{1\}^*$, the classes $\mathsf{Avg\text{-}P}$ and P coincide. The reason is that for unary languages, we have just one single instance 1^n for each length n, and this instance has a probability of 1. Also $\mathsf{Smoothed\text{-}P}$ coincides with $\mathsf{Avg\text{-}P}$ and P for unary languages. Because the set of instances is just a singleton, the parameter ϕ is fixed to 1 in this case. The observation that $\mathsf{Avg\text{-}P}$, P, and $\mathsf{Smoothed\text{-}P}$ coincide for unary languages allows us to transfer the result that $\mathsf{DistNP} \subseteq \mathsf{Avg\text{-}P}$ implies $\mathsf{NE} = \mathsf{E}$ [Ben-David et al. 1992] to smoothed complexity. (The latter classes are defined as $\mathsf{NE} = \mathsf{NTime}(2^{O(n)})$ and $\mathsf{E} = \mathsf{DTime}(2^{O(n)})$.) The transfer of their result to smoothed complexity is straightforward and therefore omitted.

THEOREM 5.2.   *If* $\mathsf{Dist\text{-}NP}_{\mathrm{para}} \subseteq \mathsf{Smoothed\text{-}P}$, *then* $\mathsf{NE} = \mathsf{E}$.

This gives some evidence that $\mathsf{Dist\text{-}NP}_{\mathrm{para}}$ is not a subset of $\mathsf{Smoothed\text{-}P}$. (We cannot have equality because $\mathsf{Dist\text{-}NP}_{\mathrm{para}}$ is restricted to computable ensembles and problems in NP, while $\mathsf{Smoothed\text{-}P}$ does not have these restrictions.)

6. TRACTABILITY 1: INTEGER PROGRAMMING

Now we deal with tractable – in the sense of smoothed complexity – optimization problems: We show that if a binary integer linear program can be solved in pseudo-polynomial time, then the corresponding decision problem belongs to $\mathsf{Smoothed\text{-}P}$. This result is similar to Beier and Vöckings characterization [Beier and Vöcking 2006]: Binary optimization problems have smoothed polynomial complexity (with respect to continuous distributions) if and only if they can be solved in randomized pseudo-polynomial time. We follow their notation and refer to their lemmas wherever appropriate.

6.1. Setup and Probabilistic Model.

A binary optimization problem is an optimization problem of the form

$$
\begin{aligned}
\text{maximize } & c^T x \\
\text{subject to } & w_i^T x \le t_i \text{ for } i \in [k] \text{ and} \\
& x \in S \subseteq \{0,1\}^n.
\end{aligned}
$$

Here, $c^T x = \sum_{j=1}^{n} c_j x_j$ is the linear objective function and $w_i^T x = \sum_{j=1}^{n} w_{i,j} x_j \le t_i$ are linear constraints. Furthermore, we have the constraint that the binary vector x must be contained in the set S. This set S should be viewed as containing the "structure" of the problem. Examples are that S contains all binary vectors representing spanning trees of a graph of n vertices or that S represents all paths connecting two given vertices or that S contains all vectors corresponding to perfect matchings of a given graph. Maybe the simplest case is $k = 1$ and $S = \{0,1\}^n$; then the binary program above represents the knapsack problem.

We assume that S is adversarial (i.e., non-random). Since we deal with decision problems in this paper rather than with optimization problems, we use the standard approach and introduce a threshold for the objective function. This means that the optimization problem becomes the question whether there is an $x \in S$ that fulfills $c^T x \ge b$ as well as $w_i^T x \le t_i$ for all $i \in \{1, \dots, k\}$. In the following, we treat the budget constraint $c^T x \ge b$ as an additional linear constraint for simplicity. We call this type of problems *binary decision problems*.

For ease of presentation, we assume that we have just one linear constraint (whose coefficients will be perturbed) and everything else is encoded in the set S. This means that the binary decision problem that we want to solve is the following: Does there exist an $x \in S$ with $w^T x \le t$?

The values $w_1, \dots, w_n$ are n-bit binary numbers. Thus, $w_i \in \{0, 1, \dots, 2^n - 1\}$. While we can of course vary their length, we choose to do it this way as it conveys all ideas while avoiding another parameter.

Let us now describe the perturbation model. We do not make any assumption about the probability distribution of any single coefficient. Instead, our result holds for any family of probability distribution that fulfills the following properties:

— $w_1, \dots, w_n$ are drawn according to independent distributions. The set S and the threshold t are part of the input and not subject to randomness. Thus, $N_{n,(S,w,t)} = 2^{n^2}$ for any instance (S, w, t) of size n. We assume that S can be encoded by a polynomially long string. (This is fulfilled for most natural optimization problems, like TSP, matching, shortest path, or knapsack.)

— The fact that $w_1, \ldots, w_n$ are drawn independently means that the probability for one
  coefficient to assume a specific value is bounded from above by $\phi^{1/n}$.

Since $N_{n,(S,w)} = 2^{n^2}$, the perturbation parameter ϕ can vary between 2^{-n^2} (for the average
case) and 1 (for the worst case).

  The idea is as follows: If we have a pseudo-polynomial algorithm, then we can solve
instances with $O(\log n)$ bits per coefficient efficiently. Our goal is thus to show that $O(\log n)$
bits suffice with high probability. (This is for the average case, i.e., $\phi = 2^{-n^2}$. For larger ϕ,
more but not too many bits are needed.) The proofs in the following are similar to proofs by
Beier and Vöcking [2006]. However, at various places it gets slightly more technical because
we have discrete rather than continuous probability distributions.

  The following simple lemma bounds the probability that a certain coefficient assumes a
value in a given small interval.

  LEMMA 6.1.   *Let $\delta, z \in \mathbb{N}$. Let a be an n-bit coefficient drawn according to some discrete
probability distribution bounded from above by $\phi^{1/n}$. Then $\Pr(a \in [z, z + \delta)) \leq \phi^{1/n}\delta$.*

  PROOF.   There are exactly δ outcomes of a that lead to $a \in [z, z + \delta)$. Thus, $\Pr(a \in [z, z + \delta)) \leq \phi^{1/n}\varepsilon$.   □

  Our goal is to show that $O(\log(n\phi^{1/n}2^n))$ bits for each coefficient suffice to determine
whether a solution exists. (For the average case, we have $\phi = 2^{-n^2}$, thus $O(\log n)$ bits per
coefficient.) To do this, it is not sufficient for an $x \in S$ to just satisfy $w^T x \leq t$: Because of the
rounding, we might find that x is feasible with respect to the rounded coefficients whereas
x is infeasible with respect to the true coefficients. Thus, what we need is that $w^T x$ is
sufficiently smaller than t. Then the rounding does not affect the feasibility. Unfortunately,
we cannot rule out the existence of solutions $x \in S$ that are very close to the threshold (after
all, there can be an exponential number of solutions, and it is likely that some of them are
close to the threshold). But it is possible to prove the following: Assume that there is some
ranking among the solutions $x \in S$. Let the winner be the solution $x^\star \in S$ that fulfills
$w^T x^\star \leq t$ and is ranked highest among all such solutions. Then it is likely that $t - w^T x^\star$ is
not too small. Now, any solution that is ranked higher than $x^\star$ must be infeasible because it
violates the linear constraint $w^T x \leq t$. Let $\hat{x}$ be the solution that minimizes $w^T x - t$ among
all solutions ranked higher than $x^\star$. Then it is also unlikely that $w^T \hat{x} - t$ extremely small,
i.e., that $\hat{x}$ violates the linear constraint by only a small margin.

  Remark 6.2. In Beier and Vöcking's analysis [Beier and Vöcking 2006], the ranking was
given by the objective function. We do not have an objective function here because we deal
with decision problems. Thus, we have to introduce a ranking artificially. In the following,
we use the lexicographic ordering (if not mentioned otherwise), which satisfies the following
monotonicity property that simplifies the proofs: if $x \in S$ is ranked higher than $y \in S$, then
there is an i with $x_i = 1$ and $y_i = 0$.

  Now let $x^\star$ be the winner (if it exists), i.e., the highest ranked (with respect to lexi-
cographic ordering) solution among all feasible solutions. Then we define the *winner gap*
as

$$\Gamma(t) = \begin{cases} t - w^T x^\star & \text{if there exists a feasible solution and} \\ \bot & \text{otherwise.} \end{cases}$$

The goal is to show that it is unlikely that Γ is small. In order to analyze Γ, it is useful
to define also the loser gap Λ. The loser $\hat{x} \in S$ is a solution that is ranked higher than $x^\star$
but cut off by the constraint $w^T x \leq t$. It is the solution with minimal $w^T x - t$ among all
such solutions. (If there is a tie, which can happen because we have discrete probability

distributions, then we take the highest-ranked solution as the loser.) We define

$$\Lambda(t) = \begin{cases} w^T \hat{x} - t & \text{if there exists a loser } \hat{x} \text{ and} \\ \bot & \text{otherwise.} \end{cases}$$

The probability that Λ or Γ is smaller than some value δ is bounded by $\delta\phi^{1/n}n$, which we will prove in the following.

The following lemma states that it suffices to analyze Λ in order to get bounds for both Λ and Γ. In fact, for the setting with just one linear constraint with non-negative coefficients, we do not even need the winner gap. But the winner gap is needed for more general cases, which we discuss in Section 6.2 but do not treat in detail for conciseness.

LEMMA 6.3 (DISCRETE VERSION OF [BEIER AND VÖCKING 2006, LEMMA 7]).   *For all* t *and* δ, *we have* $\Pr(\Gamma(t) < \delta) = \Pr(\Lambda(t - \delta) \le \delta)$.

PROOF.   A solution $x \in S$ is called *Pareto-optimal* if there is no other solution $x' \in S$ such that $w^T x' \le w^T x$ and x' is ranked higher than x. Let us make two observations. First, we observe that both winners and losers are Pareto-optimal. Second, for every Pareto-optimal solution x, there exists a threshold t such that x is the loser for this particular threshold. To see this, simply set $t = w^T x - 1$.

Let $P \subseteq S$ be the set of Pareto-optimal solutions. Then

$$\Gamma(t) = \min\{t - w^T x \mid x \in P, w^T x \le t\} \quad \text{and}$$
$$\Lambda(t) = \min\{w^T x - t \mid x \in P, w^T x > t\} = \min\{w^T x - t \mid x \in P, w^T x \ge t + 1\}.$$

Now $\Gamma(t) < \delta$ if and only if there is an $x \in P$ with $t - w^T x \in \{0, \dots, \delta - 1\}$. This is equivalent to $w^T x - t \in \{-\delta + 1, \dots, 0\}$ and to $w^T x - (t - \delta) \in \{1, \dots, \delta\}$. In turn, this is equivalent to $\Lambda(t - \delta) \le \delta$.   □

Now we analyze $\Lambda(t)$. The following lemma makes this rigorous. It is a discrete counterpart to Beier and Vöcking's separating lemma [Beier and Vöcking 2006, Lemma 5]. We have to assume that the all-zero vector is not contained in S. The reason for this is that its feasibility does not depend on any randomness.

LEMMA 6.4 (SEPARATING LEMMA).   *Suppose that* $(0, \dots, 0) \notin S$. *For every* $\delta, t \in \mathbb{N}$, *we have* $\Pr(\Gamma(t) < \delta) \le \delta\phi^{1/n}n$ *and* $\Pr(\Lambda(t) \le \delta) \le \delta\phi^{1/n}n$.
If we use a non-monotone ranking, then the bounds for the probabilities become $\delta\phi^{1/n}n^2$.

PROOF.   Because of Lemma 6.3, it suffices to analyze the loser gap Λ. We only give a proof sketch for monotone rankings as that emphasis the differences to the continuous counterpart [Beier and Vöcking 2006, Lemma 5].

Let $S_i = \{x \in S \mid x_i = 1\}$, and let $\overline{S}_i = S \setminus S_i = \{x \in S \mid x_i = 0\}$. Let $x^{\star i} \in \overline{S}_i$ be the winner from $\overline{S}_i$: $x^{\star i}$ is ranked highest in $\overline{S}_i$ and satisfies the linear constraint $w^T x^{\star i} \le t$. Let $\hat{x}^i \in S_i$ be the loser with respect to $x^{\star i}$, i.e., a solution that is ranked higher than $x^{\star i}$ and minimizes $w^T \hat{x}^i - t$ (if such a solution exists). Let

$$\Lambda_i = \begin{cases} w^T \hat{x}^i - t & \text{if } \hat{x}^i \text{ exists and} \\ \bot & \text{otherwise.} \end{cases}$$

Note that $\hat{x}^i$ can be feasible and, thus, Λ_i can be negative.

To analyze Λ_i, we assume that all w_j with $j \ne i$ are fixed by an adversary. The winner $x^{\star i}$ does not depend w_i because all solutions $x \in \overline{S}_i$ have $x_i = 0$. Once $x^{\star i}$ is fixed, also $\hat{x}^i$ is fixed. Because w_j for $j \ne i$ is fixed and $\hat{x}_i^i = 1$, we can rewrite $w^T \hat{x}^i - t = z + w_i$. Now $\Lambda_i \in \{1, \dots, \delta\}$ if w_i assumes a value in some interval of length δ, which happens with a probability of at most $\delta\phi^{1/n}$.

Furthermore, if $\Lambda \neq \perp$, then there exists an i with $\Lambda_i = \Lambda$ [Beier and Vöcking 2006, Claim B]. Thus, a union bound over all n possibilities for i yields $\Pr(\Lambda(t) \leq \delta) \leq \delta \phi^{1/n} n$. □

For the probabilistic constraint $w^T x \leq t$, it is not sufficient for an x to satisfy it. Instead, we want that only a few bits of each coefficient of w suffice to find an x that satisfies that constraint. Here, "few" means roughly $O(\log(n\phi^{1/n} 2^n))$. (Note that this is roughly $O(\log n)$ if we are close to the average case, where $\phi \approx 2^{-n^2}$.) Different from Beier and Vöcking's continuous case (where the real-valued coefficients where revealed by an oracle), we have the true coefficients at hand. Thus, we do not need their certificates that a solution is indeed feasible, but we can simply test with the true coefficients. Clearly, this testing can be done in polynomial time.

For an n-bit natural number a and $b \in \mathbb{N}$, let $\lfloor a \rfloor_b =$ be the number obtained from a by only taking the b most significant bits. This means that $\lfloor a \rfloor_b = 2^{n-b} \cdot \lfloor a/2^{n-b} \rfloor$.

In order to show that pseudo-polynomiality implies smoothed polynomial complexity, we use a pseudo-polynomial algorithm as a black box in the following way: We run the pseudo-polynomial algorithm with the highest $O(\log n)$ bits. (To do this, we scale the rounded coefficients of w down. Furthermore, we also have to scale t down appropriately.) If we find a solution, then we check it against the true coefficients of w. If it remains feasible, we output "yes". If it becomes unfeasible, then we take one more bit for each coefficient and continue. The following lemma gives a tail bound for how long this can go on.

LEMMA 6.5. *Assume that we use b bits for each coefficient of w. Let $x^\star$ be the winner (with respect to the true w without rounding). The probability that solving the problem with b bits for each coefficient yields a solution different from $x^\star$ is bounded from above by $2^{n-b}\phi^{1/n}n^2$.*

PROOF. We only get a solution different from $x^\star$ if there is a solution $\hat{x}$ ranked higher than $x^\star$ that is feasible with respect to the rounded coefficients. By rounding, we change each coefficient by at most 2^{n-b}. Thus, $w^T \hat{x} - \lfloor w \rfloor_b^T \hat{x} \leq 2^{n-b}n$.

We can conclude that we find $\hat{x}$ instead of $x^\star$ only if the loser gap Λ is at most $2^{n-b}n$, which happens with a probability of at most $2^{n-b}\phi^{1/n}n^2$ (or $2^{n-b}\phi^{1/n}n^3$ if the ranking is not monotone). □

With this preparation, we can prove the main result of this section.

THEOREM 6.6. *If a binary decision problem can be solved in pseudo-polynomial time, then it is in* Smoothed-P.

PROOF. We have to show that the running time of the algorithm sketched above, which uses the pseudo-polynomial algorithm as a black box, fulfills Theorem 2.3.

If b bits for each coefficient are used, the running time of the pseudo-polynomial algorithm is bounded from above by $O((n2^b)^c)$ for some constant c. (Even the total running time summed over all iterations up to b bits being revealed is bounded by $O((n2^b)^c)$, because it is dominated by the last iteration.)

The probability that more than time $t = O((n2^b)^c)$ is needed is bounded from above by $2^{n-b}\phi^{1/n}n^2$ according to Lemma 6.5. We can rewrite this as

$$2^{n-b}\phi^{1/n}n^2 = n^2 2^{-b}\left(2^{n^2}\phi\right)^{1/n} = \frac{n^3}{O(t^{1/c})} \cdot \left(2^{n^2}\phi\right)^{1/n} \leq \frac{n^3}{O(t^{1/c})} \cdot 2^{n^2}\phi.$$

The last inequality holds because $\phi \geq 2^{-n^2}$. The theorem is proved because this tail bound for the running time is strong enough according to Theorem 2.3. □

6.2. Examples and Discussion

Examples of problems in Smoothed-P are the decision problems associated with the following NP-hard optimization problems:

— knapsack, where the goal is to find a subset of a given collection of items that maximizes the profit while obeying a budget for its weight;
— constrained shortest path, where the goal is to find a path of minimum length that obeys a certain a budget;
— constrained minimum-weight spanning tree.

These problems can be solved in pseudo-polynomial time using dynamic programming, even if we insist on a lexicographically maximal solution (as we have to for Lemma 6.4).

Let us now discuss some extensions of the model. We have restricted ourselves to deterministic pseudo-polynomial algorithms, which yield smoothed polynomial complexity. These deterministic algorithms can be replaced without any complication by randomized errorless algorithms that have expected pseudo-polynomial running time.

So far, we have not explicitly dealt with constraints of the form "$w^T x \geq t$". But they can be treated in the same way as "$w^T x \leq t$", except that winner and loser gap change their roles. Furthermore, we did not include the case that coefficients can be positive or negative. This yields additional technical difficulties (we have to round more carefully and take both winner and loser gap into account), but we decided to restrict ourselves to the simpler form with non-negative coefficients for the sake of clarity. Moreover, we have not considered the case of multiple linear constraints [Beier and Vöcking 2006, Section 2.3] for the same reason. Finally, Röglin and Vöcking [2007] have extended the smoothed analysis framework to integer programming. We believe that the same can be done for our discrete setting.

The main open problem concerning Smoothed-P and integer optimization is the following: Beier and Vöcking [2006] have proved that (randomized) pseudo-polynomiality and smoothed polynomiality are equivalent. The reason why we do not get a similar result is as follows: Our "joint density" for all coefficients is bounded by ϕ, and the density of a single coefficient is bounded by $\phi^{1/n}$. In contrast, in the continuous version, the joint density is bounded by ϕ^n while a single coefficient has a density bounded by ϕ.

However, our goal is to devise a general theory for arbitrary decision problems. This theory should include integer optimization, but it should not be restricted to integer optimization. The problem is that generalizing the concept of one distribution bounded by ϕ for each coefficient to arbitrary problems involves knowledge about the instances and the structure of the specific problems. This knowledge, however, is not available if we want to speak about classes of decision problems as in classical complexity theory.

7. TRACTABILITY 2: GRAPHS AND FORMULAS

7.1. Graph Coloring and Smoothed Extension of $G_{n,p}$

The perturbation model that we choose is the *smoothed extension of $G_{n,p}$* [Spielman and Teng 2009]: Given an adversarial graph $G = (V, E)$ and an $\varepsilon \in (0, 1/2]$, we obtain a new graph $G' = (V, E')$ on the same set of vertices by "flipping" each (non-)edge of G independently with a probability of ε. This means the following: If $e = \{u, v\} \in E$, then e is contained in E' with a probability of $1 - \varepsilon$. If $e = \{u, v\} \notin E$, then $\Pr(e \in E') = \varepsilon$.

Transferred to our framework, this means the following: We represent a graph G on n vertices as a binary string of length $\binom{n}{2}$, and we have $N_{n,G} = 2^{\binom{n}{2}}$. The flip probability ε depends on ϕ: We choose $\varepsilon \leq 1/2$ such that $(1 - \varepsilon)^{\binom{n}{2}} = \phi$. (For $\phi = 2^{-\binom{n}{2}} = 1/N_{n,G}$, we have a fully random graph with edge probabilities of $1/2$. For $\phi = 1$, we have $\varepsilon = 0$, thus the worst case.)

We will not present an exhaustive list of graph problems in Smoothed-P, but we will focus on graph coloring as a very simple example. k-Coloring is the decision problem whether the vertices of a graph can be colored with k colors such that no pair of adjacent vertices get the same color. k-Coloring is NP-complete for any $k \geq 3$ [Garey and Johnson 1979, GT 4].

THEOREM 7.1. *For any* $k \in \mathbb{N}$, *k-Coloring* $\in$ Smoothed-P.

PROOF. To show that k-Coloring $\in$ Smoothed-P, we analyze the following simple algorithm: First, we check whether the input graph contains a clique of size $k + 1$. This can be done easily in polynomial time. If yes, we output no. If no, we perform exhaustive search. The analysis is similar to Wilf's analysis [Wilf 1985] of the coloring problem.

A graph is k-colorable only if it does not contain a clique of size $k + 1$. The probability that a specific set of $k + 1$ vertices form a $k + 1$ clique is at least $\varepsilon^{\binom{k+1}{2}}$. Thus, the probability that a graph G on n vertices does not contain a $k + 1$ clique is at most $\left(1 - \varepsilon^{\binom{k+1}{2}}\right)^{\frac{n}{k+1}}$.

We distinguish two cases: First, $\varepsilon \geq 0.1$. In this case, $\left(1 - \varepsilon^{\binom{k+1}{2}}\right)^{\frac{n}{k+1}}$ can be bounded from above by c^n for some positive constant $c < 1$ that depends on k. Brute-force testing whether a graph can be k-colored can be done in time $\text{poly}(n) \cdot k^n$. The probability that we need brute force is at most c^n. Thus, the expected running-time, raised to the power $\log_k(1/c)$, is bounded from above by a polynomial.

Second, $\varepsilon < 0.1$. Then we have $\phi = (1 - \varepsilon)^{\binom{n}{2}} \geq 0.9^{\binom{n}{2}}$. The allowed running-time (raised to some constant power) is $N_{n,G}\phi n = 2^{\binom{n}{2}}\phi n \geq 1.8^{\binom{n}{2}}$. Thus, we can afford exhaustive search in every run. □

Remark 7.2. Bohman et al. [2004] and Krivelevich et al. [2006] consider a slightly different model for perturbing graphs: Given an adversarial graph, we add random edges to the graph to obtain our actual instance. No edges are removed.

They analyze the probability that the random graph thus obtained is guaranteed to contain a given subgraph H. By choosing H to be a clique of size $k + 1$ and using a proof similar to Theorem 7.1's, we obtain that k-Coloring $\in$ Smoothed-P also with respect to this perturbation model.

7.2. Unsatisfiability and Smoothed-RP

Besides the smoothed extension of $G_{n,p}$ discussed above, there exist various other models for obtaining graphs and also Boolean formulas that are neither fully random nor adversarial.

Feige [2007] and Coja-Oghlan et al. [2009] have considered the following model: We are given a (relatively dense) adversarial Boolean k-CNF formula. Then we obtain our instance by negating each literal with a small probability. It is proved that such smoothed formulas are likely to be unsatisfiable, and that their unsatisfiability can be proved efficiently. However, their algorithms are randomized, thus we do not get a result that kUNSAT (this means that unsatisfiability problem for k-CNF formulas) for dense instances belongs to Smoothed-P. However, it shows that kUNSAT for dense instance belongs to Smoothed-RP, where Smoothed-RP is the smoothed analogue of RP: A pair $(L, \mathcal{D})$ is in Smoothed-RP if there is a randomized polynomial algorithm A with the following properties:

(1) For all $x \notin L$, A outputs "no". (This property is independent of the perturbation.)
(2) For all $x \in L$, A outputs "yes" with a probability of at least $1/2$. (This property is also independent of the perturbation.)
(3) A has smoothed polynomial running time with respect to $\mathcal{D}$. (This property is independent of the internal randomness of A.)

Note that we have two sources of randomness in Smoothed-RP: The instance is perturbed, and the algorithm A is allowed to use randomness. Item 1 and 2 depend only on A's own randomness. Item 3 depends only on the perturbation $\mathcal{D}$.

Now, let $k\mathsf{UNSAT}_\beta$ be $k\mathsf{UNSAT}$ restricted to instances with at least βn clauses, where n denotes the number of variables. Let ε be the probability that a particular literal is negated. Feige [2007] has presented a polynomial-time algorithm with the following property: If $\beta = \Omega(\sqrt{n \log\log n}/\varepsilon^2)$ and the perturbed instance of $k\mathsf{UNSAT}_\beta$ is unsatisfiable, which it is with high probability, then his algorithm proves that the formula is unsatisfiable with a probability of at least $1 - 2^{\Omega(-n)}$. The following result is a straightforward consequence.

THEOREM 7.3. $k\mathsf{UNSAT}_\beta \in \mathsf{Smoothed\text{-}RP}$ *for* $\beta = \Omega(\sqrt{n \log\log n})$.

8. SMOOTHED ANALYSIS VS. SEMI-RANDOM MODELS

Semi-random models for graphs and formulas exist even longer than smoothed analysis and can be considered as precursors to smoothed analysis. The basic concept is as follows: Some instance is created randomly that possesses a particular property. This property can, for instance, be that the graph is k-colorable. After that, the adversary is allowed to modify the instance without destroying the property. For instance, the adversary can be allowed to add arbitrary edges between the different color classes. Problems that have been considered in this model or variants thereof are independent set [Feige and Kilian 2001], graph coloring [Blum and Spencer 1995; Feige and Kilian 2001; Coja-Oghlan 2007a], or finding sparse induced subgraphs [Coja-Oghlan 2007b]. However, we remark that these results do not easily fit into a theory of smoothed analysis. The reason is that in these semi-random models, we first have the random instance, which is then altered by the adversary. This is in contrast to smoothed analysis in general and our smoothed complexity theory in particular, where we the adversarial decisions come before the randomness is applied.

9. DISCUSSION

Our framework has many of the characteristics that one would expect. We have reductions and complete problems and they work in the way one expects them to work. To define reductions, we have to use the concept of disjoint supports. It seems to be essential that we know the original instance x that the actual instance y was drawn from to obtain proper domination. Although this is somewhat unconventional, we believe that this is the right way to define reductions in the smoothed setting. The reason is that otherwise, we do not know the probabilities of the instances, which we need in order to apply the compression function. The compression function, in turn, seems to be crucial to prove hardness results. Still, an open question is whether a notion of reducibility can be defined that circumvents these problems. Moreover, many of the positive results from smoothed analysis can be cast in our framework, like it is done in Sections 6 and 7.

One question that frequently arises is whether the original, unperturbed x should have the highest probability in $D_{n,x,\phi}$. While this is natural for perturbing real numbers, we think that this just puts an extra technical restriction, which complicates the framework. And for the general probabilistic input model introduced by Beier and Vöcking [2006], which we have adapted, there is not even a natural notion of an "original instance" as the adversary merely specifies density functions.

Many positive results in the literature state their bounds in the number of "entities" (like number of nodes, number of coefficients) of the instance. However, in complexity theory, we measure bounds in the length (number of symbols) of the input in order to get a theory for arbitrary problems, not only for problems of a specific type. To state bounds in terms of bit length makes things less tight, for instance the reverse direction of integer programming does not work. But still, we think it is more reasonable to use the customary notion of input length such that smoothed complexity fits with average-case and worst-case complexity.

Finally, the results by Röglin and Teng [2009] show that, for binary optimization problems, expected polynomial is indeed a robust measure. We have shown that this is in general not the case. To do this, we have used a language in E. The obvious question is now whether Avg-P and Avg-PBM as well as Smoothed-P and Smoothed-PBM coincide for problems in NP.

We hope that the present work will stimulate further research in smoothed complexity theory in order to get a deeper understanding of the theory behind smoothed analysis.

REFERENCES

Sanjeev Arora and Boaz Barak. 2009. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA.

David Arthur, Bodo Manthey, and Heiko Röglin. 2011. Smoothed Analysis of the k-Means Method. *J. ACM* 58, 5, Article 19 (2011), 31 pages.

Cyril Banderier, René Beier, and Kurt Mehlhorn. 2003. Smoothed Analysis of Three Combinatorial Problems. In *Proc. of the 28th Int. Symp. on Mathematical Foundations of Computer Science (MFCS) (Lecture Notes in Computer Science)*, Branislav Rovan and Peter Vojtás (Eds.), Vol. 2747. Springer, 198–207.

René Beier and Berthold Vöcking. 2004. Random Knapsack in Expected Polynomial Time. *J. Comput. System Sci.* 69, 3 (2004), 306–329.

René Beier and Berthold Vöcking. 2006. Typical Properties of Winners and Losers in Discrete Optimization. *SIAM J. Comput.* 35, 4 (2006), 855–881.

Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. 1992. On the Theory of Average Case Complexity. *J. Comput. System Sci.* 44, 2 (1992), 193–219.

Markus Bläser, Bodo Manthey, and B. V. Raghavendra Rao. 2013. Smoothed Analysis of Partitioning Algorithms for Euclidean Functionals. *Algorithmica* 66, 2 (2013), 397–418.

Avrim L. Blum and John D. Dunagan. 2002. Smoothed Analysis of the Perceptron Algorithm for Linear Programming. In *Proc. of the 13th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA)*. SIAM, San Francisco, CA, USA, 905–914.

Avrim L. Blum and Joel Spencer. 1995. Coloring Random and Semi-Random k-Colorable Graphs. *Journal of Algorithms* 19, 2 (1995), 204–234.

Andrej Bogdanov and Luca Trevisan. 2006. Average-Case Complexity. *Foundations and Trends in Theoretical Computer Science* 2, 1 (2006), 1–106.

Tom Bohman, Alan M. Frieze, Michael Krivelevich, and Ryan Martin. 2004. Adding Random Edges to Dense Graphs. *Random Structures and Algorithms* 24, 2 (2004), 105–117.

Tobias Brunsch, Kamiel Cornelissen, Bodo Manthey, and Heiko Röglin. 2013a. Smoothed Analysis of Belief Propagation for Minimum-Cost Flow and Matching. *Journal of Graph Algorithms and Applications* 17, 6 (2013), 647–670.

Tobias Brunsch, Kamiel Cornelissen, Bodo Manthey, and Heiko Röglin. 2013b. Smoothed Analysis of the Successive Shortest Path Algorithm. In *Proc. 24th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA)*. SIAM, 1180–1189.

Tobias Brunsch and Heiko Röglin. 2015. Improved Smoothed Analysis of Multiobjective Optimization. *J. ACM* 62, 1 (2015), 4.

Xi Chen, Xiaotie Deng, and Shang-Hua Teng. 2009. Settling the Complexity of Computing Two-Player Nash Equilibria. *J. ACM* 56, 3, Article 14 (2009), 57 pages.

Amin Coja-Oghlan. 2007a. Colouring Semirandom Graphs. *Combinatorics, Probability and Computing* 16, 4 (2007), 515–552.

Amin Coja-Oghlan. 2007b. Solving NP-Hard Semirandom Graph Problems in Polynomial Expected Time. *Journal of Algorithms* 62, 1 (2007), 19–46.

Amin Coja-Oghlan, Uriel Feige, Alan M. Frieze, Michael Krivelevich, and Dan Vilenchik. 2009. On Smoothed k-CNF Formulas and the Walksat Algorithm. In *Proc. of the 20th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA)*. SIAM, New York, NY, USA, 451–460.

Valentina Damerow, Bodo Manthey, Friedhelm Meyer auf der Heide, Harald Räcke, Christian Scheideler, Christian Sohler, and Till Tantau. 2012. Smoothed Analysis of Left-To-Right Maxima with Applications. *ACM Transactions on Algorithms* 8, 3, Article 30 (2012), 28 pages.

Matthias Englert, Heiko Röglin, and Berthold Vöcking. 2014. Worst Case and Probabilistic Analysis of the 2-Opt Algorithm for the TSP. *Algorithmica* 68, 1 (2014), 190–264.

Uriel Feige. 2007. Refuting Smoothed 3CNF Formulas. In *Proc. of the 48th Ann. IEEE Symp. on Foundations of Computer Science (FOCS)*. IEEE Computer Society, Providence, RI, USA, 407–417.

Uriel Feige and Joe Kilian. 2001. Heuristics for Semirandom Graph Problems. *J. Comput. System Sci.* 63, 4 (2001), 639–671.

Mahmoud Fouz, Manfred Kufleitner, Bodo Manthey, and Nima Zeini Jahromi. 2012. On Smoothed Analysis of Quicksort and Hoare's Find. *Algorithmica* 62, 3–4 (2012), 879–905.

Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, NY, USA.

Yuri Gurevich. 1991. Average Case Completeness. *J. Comput. System Sci.* 42, 3 (1991), 346–398.

Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 2001. Which Problems Have Strongly Exponential Complexity? *J. Comput. System Sci.* 63, 4 (2001), 512–530.

Michael Krivelevich, Benny Sudakov, and Prasad Tetali. 2006. On Smoothed Analysis in Dense Graphs and Formulas. *Random Structures and Algorithms* 29, 2 (2006), 180–193.

Leonid A. Levin. 1986. Average Case Complete Problems. *SIAM J. Comput.* 15, 1 (1986), 285–286.

Ming Li and Paul M. B. Vitányi. 1992. Average Case Complexity under the Universal Distribution Equals Worst-Case Complexity. *Inform. Process. Lett.* 42, 3 (1992), 145–149.

Ming Li and Paul M. B. Vitányi. 1993. *An Introduction to Kolmogorov Complexity and its Applications*. Springer, New York, NY, USA.

Bodo Manthey and Rüdiger Reischuk. 2007. Smoothed Analysis of Binary Search Trees. *Theoretical Computer Science* 378, 3 (2007), 292–315.

Bodo Manthey and Heiko Röglin. 2011. Smoothed Analysis: Analysis of Algorithms Beyond Worst Case. *it – Information Technology* 53, 6 (2011), 280–286.

Bodo Manthey and Heiko Röglin. 2013. Worst-Case and Smoothed Analysis of k-Means Clustering with Bregman Divergences. *Journal of Computational Geometry* 4, 1 (2013), 94–132.

Bodo Manthey and Rianne Veenstra. 2013. Smoothed Analysis of the 2-Opt Heuristic for the TSP: Polynomial Bounds for Gaussian Noise. In *Proc. of the 24th Int. Symp. on Algorithms and Computation (ISAAC)*, Leizhen Cai, Siu-Wing Cheng, and Tak-Wah Lam (Eds.). Lecture Notes in Computer Science, Vol. 8283. Springer, 579–589.

Ankur Moitra and Ryan O'Donnell. 2011. Pareto Optimal Solutions for Smoothed Analysts. In *Proc. of the 43rd Ann. ACM Symp. on Theory of Computing (STOC)*. ACM, San Jose, CA, USA, 225–234.

Heiko Röglin and Shang-Hua Teng. 2009. Smoothed Analysis of Multiobjective Optimization. In *Proc. of the 50th Ann. IEEE Symp. on Foundations of Computer Science (FOCS)*. IEEE Computer Society, Atlanta, GA, USA, 681–690.

Heiko Röglin and Berthold Vöcking. 2007. Smoothed Analysis of Integer Programming. *Mathematical Programming* 110, 1 (2007), 21–56.

Daniel A. Spielman and Shang-Hua Teng. 2003. Smoothed Analysis of Termination of Linear Programming Algorithms. *Mathematical Programming* 97, 1–2 (2003), 375–404.

Daniel A. Spielman and Shang-Hua Teng. 2004. Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time. *J. ACM* 51, 3 (2004), 385–463.

Daniel A. Spielman and Shang-Hua Teng. 2009. Smoothed Analysis: An Attempt to Explain the Behavior of Algorithms in Practice. *Commun. ACM* 52, 10 (2009), 76–84.

Roman Vershynin. 2009. Beyond Hirsch Conjecture: Walks on Random Polytopes and Smoothed Complexity of the Simplex Method. *SIAM J. Comput.* 39, 2 (2009), 646–678.

Jie Wang. 1997. Average-Case Intractable NP Problems. In *Advances in Languages, Algorithms, and Complexity*, Ding-Zhu Du and Ker-I Ko (Eds.). Kluwer, Dordrecht, Netherlands, 313–378.

Herbert S. Wilf. 1985. Some Examples of Combinatorial Averaging. *The American Mathematical Monthly* 92, 4 (1985), 250–261.