Smoothed Complexity Theory*

Markus Bläser[1] and Bodo Manthey[2]

[1] Saarland University, mblaeser@cs.uni-saarland.de
[2] University of Twente, b.manthey@utwente.nl

Abstract. 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 results.

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. 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: The algorithm will 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.

* Supported by DFG research grant BL 511/7-1. A full version of this paper is available at http://arxiv.org/abs/1202.1936.

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 give a comprehensive survey of average-case complexity [7].

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 have proposed a new way to analyze algorithms called *smoothed analysis* [27]. 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 have proved that the simplex method has a running time that is polynomial in the size of the instance and the perturbation parameter [27]. 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 [1, 2, 5, 12, 13, 16, 23, 26, 29]. We refer to two recent surveys for a broader picture of smoothed analysis [22, 28]. However, with only few exceptions [3, 25], smoothed analysis has only been applied yet to single algorithms or single problems. 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 $\mathsf{Dist\text{-}NP}_{\mathrm{para}}$, 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 [27], the strength of the perturbation has been controlled in terms of the standard de-

viation 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 [2]: 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. Our perturbation models are families of distributions $\mathcal{D} = (D_{n,x,\phi})$. The length of x is n (so 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. For every n, x, and ϕ, the support of the distribution $D_{n,x,\phi}$ should be contained in the set $\{0,1\}^{\leq \mathrm{poly}(n)}$. Let $S_{n,x} = \big\{y \mid D_{n,x,\phi}(y) > 0 \text{ for some } \phi\big\}$, and let $N_{n,x} = |S_{n,x}|$.

For all n, x, ϕ, and y, we demand $D_{n,x,\phi}(y) \leq \phi$. This controls the strength of the perturbation and restricts the adversary. We allow $\phi \in [1/N_{n,x}, 1]$. Furthermore, the values of ϕ are discretized, so that they can be described by at most $\mathrm{poly}(n)$ bits. 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 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. We call such families $(D_{n,x,\phi})_{n,x,\phi}$ of probability distributions *parameterized families of distributions*.

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 [3]. 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 [20]. We refer to Bogdanov and Trevisan [7] 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 $\phi = O(\mathrm{poly}(n)/N_{n,x})$. 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 [27] or Beier and Vöcking's analysis of integer programs [3]; they achieve polynomial time running time only if they perturb all but at most $O(\log n)$ digits: If we perturb a number with, say, a Gaussian of standard deviation $\sigma = 1/\mathrm{poly}(n)$, then we expect that the $O(\log n)$ most significant bits remain untouched, but the less significant bits are random.

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 the average case: an algorithm has smoothed polynomial running-time if and only if its running-time has polynomially decreasing tail bounds.

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

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

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

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 [24, 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 $\varepsilon > 0$ and for every n, x, and ϕ, we have $\mathbb{E}_{y\sim D_{n,x,\phi}}\big(t_A(y;n,\phi)^\varepsilon\big) = O\big(n \cdot N_{n,x} \cdot \phi\big)$.* 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 $\varepsilon > 0$ and for all n, we have $\mathbb{E}_{y\sim D_n}\big(t_A(y)^\varepsilon\big) = O(n)$.* Avg-PBM *contains all $(L,\mathcal{D})$ for which there exists a deterministic algorithm with robust smoothed polynomial running time.*

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 [24], 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.

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 causes 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 $\operatorname{supp} D_{n,x,\phi}$. 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, every x, every ϕ and every $y \in \operatorname{supp} D_{n,x,\phi}$ 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, and ϕ and every $y' \in \operatorname{supp} D'_{m(n),f_1(\langle x,y\rangle;n,\phi),\phi}$, we have*

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

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_{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 $|x|$.)*

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

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 $(L, \mathcal{D}) \in$ Smoothed-P implies $(L_{\mathrm{ds}}, \mathcal{D}) \in$ Smoothed-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 Smoothed-P_{ds} whose counterparts are in Smoothed-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 Smoothed-$\mathsf{P}^{\mathrm{obl}}_{\mathrm{ds}}$ be the resulting subset of problems in Smoothed-P_{ds} that have such an oblivious algorithm with smoothed polynomial running time.

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

Note that almost all algorithms, for which a smoothed analysis has been carried out, do not know the x from which y was drawn; in particular, there is an oblivious algorithm for them. Thus, a begging questions is if there is a problem $(L, \mathcal{D}) \notin \mathsf{Smoothed\text{-}P}$ but $(L_{\mathrm{ds}}, \mathcal{D}) \in \mathsf{Smoothed\text{-}P}_{\mathrm{ds}}$.

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. On the other hand, 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 seem to help.

4 Parameterized Distributional NP

In this section, we define the smoothed analogue of the worst-case class NP and the average-case class DistNP [18, 20]. 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 [21]. We transfer the notion of computable ensembles to smoothed complexity, which allows us to define the smoothed analogue of NP and DistNP.

Definition 4.1. *A parameterized family of distributions is in* $\mathsf{PComp}_{\mathrm{para}}$ *if the cumulative probability* $F_{D_{n,x,\phi}} = \sum_{z \le x} D_{n,x,\phi}$ *can be computed in polynomial time (given* n*,* x *and* ϕ *in binary).*

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

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 $\mathsf{Dist\text{-}NP}_{\mathrm{para}}$. Bounded halting is the canonical NP-complete language, and it has been the first problem that has been shown to be $\mathsf{Avg\text{-}P}$-complete [20]. Formally, let

$$\mathtt{BH} = \{\langle g, x, 1^t\rangle \mid \text{NTM with Gödel number } g \text{ accepts } x \text{ within } t \text{ steps}\}.$$

For a specific parameterized family $U^{\mathtt{BH}}$ of distributions, we can prove the following theorem.

Theorem 4.3. $(\mathtt{BH}, U^{\mathtt{BH}})$ *is* $\mathsf{Dist\text{-}NP}_{\mathrm{para}}$*-complete for some* $U^{\mathtt{BH}} \in \mathsf{PComp}_{\mathrm{para}}$.

The original DistNP-complete problem by Levin [20] was `Tiling`: 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 $n \le t$ such that 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. Again, we need a special family $U^{\mathtt{Tiling}}$ of distributions.

Theorem 4.4. $(\texttt{Tiling}, U^{\texttt{Tiling}})$ *is* $\mathsf{Dist\text{-}NP}_{\mathrm{para}}$*-complete for some* $U^{\texttt{Tiling}} \in \mathsf{PComp}_{\mathrm{para}}$ *under polynomial-time smoothed reductions.*

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.

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* $|x_n| \leq \mathrm{poly}(n)$*.*

It is known that $\mathsf{DistNP} \subseteq \mathsf{Avg\text{-}P}$ implies $\mathsf{NE} = \mathsf{E}$ [4]. This can be transferred to smoothed complexity.

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

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 [3]: 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.

A binary optimization problem is an optimization problem of the form "maximize $c^T x$ subject to $w_i^T x \leq t_i$ for $i \in [k]$ and $x \in S \subseteq \{0,1\}^n$. The set S should be viewed as containing the "structure" of the problem. 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 $c^T x \geq b$ as well as $w_i^T x \leq t_i$ for all $i \in \{1, \ldots, k\}$. In the following, we treat the budget constraint $c^T x \geq b$ as an additional linear constraint for simplicity. We call this type of problems *binary decision problems.*

Let us now describe the perturbation model. 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. The coefficients of the left-hand sides of the constraints are n-bit binary numbers. 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: $w_1, \ldots, w_n$ 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. 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).

Theorem 6.1. *If a binary decision problem can be solved in pseudo-polynomial time, then it is in* Smoothed-P.

Beier and Vöcking [3] 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* $G_{n,p}$ [28]: 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.)

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 $k \geq 3$ [17, GT 4].

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

Remark 7.2. Bohman et al. [8] and Krivelevich et al. [19] 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

Feige [14] and Coja-Oghlan et al. [11] 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.)

Now, let kUNSAT$_\beta$ be kUNSAT 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 [14] 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 kUNSAT$_\beta$ 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. *kUNSAT$_\beta \in$ Smoothed-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 [15], graph coloring [6, 9, 15], or finding sparse induced subgraphs [10]. 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.

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 important and useful to use the usual notion of input length such that smoothed complexity fits with average-case and worst-case complexity.

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

1. David Arthur, Bodo Manthey, and Heiko Röglin. Smoothed analysis of the k-means method. *J. ACM*, 58(5), 2011.
2. René Beier and Berthold Vöcking. Random knapsack in expected polynomial time. *J. Comput. System Sci.*, 69(3):306–329, 2004.
3. René Beier and Berthold Vöcking. Typical properties of winners and losers in discrete optimization. *SIAM J. Comput.*, 35(4):855–881, 2006.
4. Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. On the theory of average case complexity. *J. Comput. System Sci.*, 44(2):193–219, 1992.
5. Markus Bläser, Bodo Manthey, and B. V. Raghavendra Rao. Smoothed analysis of partitioning algorithms for Euclidean functionals. *Algorithmica*, to appear.
6. Avrim L. Blum and Joel Spencer. Coloring random and semi-random k-colorable graphs. *J. Algorithms*, 19(2):204–234, 1995.
7. Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Foundations and Trends in Theoret. Comput. Sci.*, 2(1):1–106, 2006.
8. Tom Bohman, Alan M. Frieze, Michael Krivelevich, and Ryan Martin. Adding random edges to dense graphs. *Random Struct. Algorithms*, 24(2):105–117, 2004.
9. Amin Coja-Oghlan. Colouring semirandom graphs. *Combin. Probab. Comput.*, 16(4):515–552, 2007.

10. Amin Coja-Oghlan. Solving NP-hard semirandom graph problems in polynomial expected time. *J. Algorithms*, 62(1):19–46, 2007.
11. Amin Coja-Oghlan, Uriel Feige, Alan M. Frieze, Michael Krivelevich, and Dan Vilenchik. On smoothed k-CNF formulas and the Walksat algorithm. *Proc. 20th Ann. Symp. on Discrete Algorithms (SODA)*, pp. 451–460. SIAM, 2009.
12. Valentina Damerow, Bodo Manthey, Friedhelm Meyer auf der Heide, Harald Räcke, Christian Scheideler, Christian Sohler, and Till Tantau. Smoothed analysis of left-to-right maxima with applications. *ACM Trans. Algorithms*, to appear.
13. Matthias Englert, Heiko Röglin, and Berthold Vöcking. Worst case and probabilistic analysis of the 2-Opt algorithm for the TSP. *Proc. 18th Ann. Symp. on Discrete Algorithms (SODA)*, pp. 1295–1304. SIAM, 2007.
14. Uriel Feige. Refuting smoothed 3CNF formulas. *Proc. 48th Ann. Symp. on Foundations of Computer Science (FOCS)*, pp. 407–417. IEEE, 2007.
15. Uriel Feige and Joe Kilian. Heuristics for semirandom graph problems. *J. Comput. System Sci.*, 63(4):639–671, 2001.
16. Mahmoud Fouz, Manfred Kufleitner, Bodo Manthey, and Nima Zeini Jahromi. On smoothed analysis of quicksort and Hoare's find. *Algorithmica*, 62(3–4):879–905, 2012.
17. Michael R. Garey and David S. Johnson. *Computers and Intractability.* W. H. Freeman and Company, 1979.
18. Yuri Gurevich. Average case completeness. *J. Comput. System Sci.*, 42(3):346–398, 1991.
19. Michael Krivelevich, Benny Sudakov, and Prasad Tetali. On smoothed analysis in dense graphs and formulas. *Random Struct. Algorithms*, 29(2):180–193, 2006.
20. Leonid A. Levin. Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, 1986.
21. Ming Li and Paul M. B. Vitányi. Average case complexity under the universal distribution equals worst-case complexity. *Inform. Process. Lett.*, 42(3):145–149, 1992.
22. Bodo Manthey and Heiko Röglin. Smoothed analysis: Analysis of algorithms beyond worst case. *it – Information Technology*, 53(6), 2011.
23. Ankur Moitra and Ryan O'Donnell. Pareto optimal solutions for smoothed analysts. In *Proc. 43rd Ann. Symp. on Theory of Computing (STOC)*, pp. 225–234. ACM, 2011.
24. Heiko Röglin and Shang-Hua Teng. Smoothed analysis of multiobjective optimization. *Proc. 50th Ann. Symp. on Foundations of Computer Science (FOCS)*, pp. 681–690. IEEE, 2009.
25. Heiko Röglin and Berthold Vöcking. Smoothed analysis of integer programming. *Math. Prog.*, 110(1):21–56, 2007.
26. Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of termination of linear programming algorithms. *Math. Prog.*, 97(1–2):375–404, 2003.
27. Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004.
28. Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis: An attempt to explain the behavior of algorithms in practice. *Commun. ACM*, 52(10):76–84, 2009.
29. Roman Vershynin. Beyond Hirsch conjecture: Walks on random polytopes and smoothed complexity of the simplex method. *SIAM J. Comput.*, 39(2):646–678, 2009.