Average-Case Complexity Theory of Approximation Problems

Birgit Schelm

Technische Universität Berlin
Fakultät IV – Elektrotechnik und Informatik
10587 Berlin, Germany
bts@cs.tu-berlin.de

Abstract. Both average-case complexity and the study of the approximability properties of NP-optimization problems are well established and active fields of research. By applying the notion of average-case complexity to approximation problems we provide a formal framework that allows the classification of NP-optimization problems according to their average-case approximability. Thus, known results about the average-case behaviour of approximation algorithms – with respect to their running time as well as their performance ratio – can be unified, though subsuming results about problems being approximable within a certain factor *with high probability* under the new classes is not straightforward. The structural properties of this framework provide an interesting field of study on their own. We not only define appropriate average-case approximation classes, but also a reduction that preserves approximability within average polynomial time. We show that the class of NP-optimization problems with P-computable input distributions has complete problems with respect to this reduction. The inclusion structure of the average-case approximation classes is investigated. E.g. using a new result by Buhrman, Fortnow and Pavan [8] we are able to show that DistNPO $\subseteq$ Avg$_t$-PTAS if DistNP $\subseteq$ AvgP.

1 Introduction

There are several ways of dealing with the fact that many computational problems are not solvable in polynomial time, unless P = NP. One way is to consider algorithms whose running time is polynomial on most inputs, though it is still super-polynomial on some instances that rarely occur. Another way, applicable to optimization problems, is to search for polynomial-time algorithms that do not necessarily yield an optimal solution, but come close to the optimum, for example up to a constant factor.

This paper combines these two approaches. To do so, one has to give formal definitions of average-case approximation classes. Reflecting the two parameters of interest, namely the running time and the quality of the solution found, there are three possibilities of relaxing worst-case requirements to average-case requirements. First, we allow the running time to be polynomial on average rather than bounded by a polynomial everywhere. The notion of average polynomial time for decision problems is well established, see [24] for a survey, and recent results like [8] attest to the relevance of this field of research. In this setting we examine average-case variants of worst-case approximation classes like PTAS, APX, poly-APX and exp-APX as well as a reduction that preserves approximability in average polynomial time. Using this reduction we define a notion of completeness for average-case approximation classes, and show that DistNPO, the class of optimization problems with P-computable input distributions, has complete problems. We examine the inclusion structure of the average polynomial-time approximation classes assuming that NP is easy on average, that is DistNP $\subseteq$ AvgP, and under the assumption that DistNP $\not\subseteq$ AvgP as well.

Second, we relax the requirement on the quality of solutions by allowing solutions to meet the desired quality only on average. Finally, we combine these two relaxations. In these settings we focus on average-case variants of APX, the worst-case class of optimization problems approximable within a constant factor.

While defining average polynomial time versions of worst-case approximation classes is straightforward, capturing the notion of approximability within a constant factor *on average* is not that obvious. We give a formal definition for approximability within a constant factor on average. This definition allows a more structural view on results on the average value of the performance ratio of polynomial-time approximation strategies as given in [15, 21, 19], see also the corresponding chapter in [2].

This paper is structured as follows. In Section 2 we review definitions and results from approximation and average-case complexity theory, and define average polynomial-time approximation classes. Section 3 introduces an average-case approximability preserving reduction and shows that DistNPO has complete problems. In Section 4 the inclusion structure of the average polynomial-time approximation classes is examined. A formal definition of approximability within a constant factor on average is given in Section 5, a comparison of the resulting classes is given in Section 6.

2 Preliminaries

We follow standard definitions of complexity theory, see for example [22], [3], or [13]. We consider only nonempty words over $\Sigma = \{0, 1\}$. By $|x|$ we denote the length of a word x, by Σ^n the set of all words of length n, and by $\Sigma^{\geq n}$ the set of all words of length at least n. By $\leq$ we denote the standard ordering on Σ^+.

A *probability function* on Σ^+ is a function $\mu : \Sigma^+ \to [0, 1]$ such that $\sum_{x\in\Sigma^+} \mu(x) = 1$. Its corresponding *probability distribution* $\mu^*(x)$ is given by $\mu^*(x) = \Sigma_{y\leq x}\mu(y)$. For $D \subseteq \Sigma^+$, the conditional probability function $\mu_{|D}$ is defined as $\mu_{|D}(x) = \mu(x)/\mu(D)$ for $x \in D$ and $\mu_{|D}(x) = 0$ otherwise. Let $\mu_n = \mu_{|\Sigma^n}$ and $\mu_{\geq n} = \mu_{|\Sigma^{\geq n}}$. The conditional expectations of random variables are denoted likewise by E_n and $\mathrm{E}_{\geq n}$ respectively. We will use subscripts to describe the probability of an event with respect to a conditional probability function as well. A sequence of events B_n for $n \in \mathbb{N}$ *holds with high probability* if $\lim_{n\to\infty} \Pr[B_n] = 1$.

A function $f : \Sigma^+ \to [0, 1]$ is P-*computable* if there exists a deterministic Turing machine M such that $|M(x, k) - f(x)| \leq 2^{-k}$ for all $x \in \Sigma^+, k \geq 1$ whose time is bounded by a polynomial in $|x|$ and k. Note that if a probability distribution μ^* is P-computable, the corresponding probability function μ is P-computable as well. Since the converse is not true unless P = NP, see [16], we say that a probability function μ is P-computable if both μ and μ^* are, thus following the standard notation used in the literature.

The *standard probability functions* on positive integers, words and graphs are defined as follows. For positive integers we use $\tilde{\mu}(n) = c/n^2$ with $c = 6/\pi^2$ as a standard probability function. This function does not put too much weight on small numbers, but its sum over all positive integers is 1. A probability function μ_0 on Σ^+ is obtained by first selecting a length according to $\tilde{\mu}$, then a word of this length with uniform distribution from Σ^n, that is, $\mu_0(x) = \tilde{\mu}(|x|) \cdot 2^{-|x|}$. When dealing with random graphs, we use – unless stated otherwise

– the $\mathcal{G}_{n,p}$-model [7], where p may depend on n. According to this model, a graph with n vertices is selected by randomly inserting an edge between two vertices with probability p, independently of the existence or absence of other edges in the graph. So the probability function μ_{G_n} on graphs with n vertices is $\mu_{G_n}(G) = p^m(1-p)^{\binom{n}{2}-m}$, where m is the number of edges in G. We obtain a probability function μ_G on the set of all graphs by first selecting n according to $\tilde{\mu}$, and then a graph according to μ_{G_n}.

Next, we give a brief description of the basic definitions and notations of optimization and approximation problems. A survey can be found for example in [2].

An NP*-optimization problem* F over an alphabet Σ is defined as a tuple (I, S, m, type) such that

1. $I \subseteq \Sigma^*$ is the set of input instances with $I \in \mathrm{P}$.
2. $S \subseteq I \times \Sigma^*$ is a relation of inputs and their feasible solutions. $S(x)$ denotes the set of feasible solutions for any $x \in I$. We require $S \in \mathrm{P}$ and $|y| \leq q(|x|)$ for all $y \in S(x)$ and some polynomial q.
3. $m : S \to \mathbb{N}^+$ is the objective function with $m \in \mathrm{FP}$.
4. *type* states whether F is a maximization or minimization problem.

NPO denotes the class of all NP-optimization problems. The optimal value $\mathrm{opt}(x)$ of the objective function for $x \in I$ is $\mathrm{opt}(x) = \max\{m(x,y) \mid y \in S(x)\}$ for maximization and $\mathrm{opt}(x) = \min\{m(x,y) \mid y \in S(x)\}$ for minimization problems.

If not stated otherwise, an *approximation algorithm* is a deterministic algorithm running in polynomial time. For an approximation algorithm A, we denote by $A(x)$ the solution computed by A on input x. Clearly, $A(x) \in S(x)$ has to hold for $x \in I$ if $S(x) \neq \emptyset$.

For $x \in I$ with $S(x) \neq \emptyset$, the *performance ratio* R of an approximation algorithm A is defined as $R\big(x, A(x)\big) = \max\{\mathrm{opt}(x)/m\big(x, A(x)\big),\ m\big(x, A(x)\big)/\,\mathrm{opt}(x)\}$. Note that $R\big(x, A(x)\big) \geq 1$ for all $x \in I$. If $R\big(x, A(x)\big) = 1$, the solution found by A is optimal.

Let $r : \mathbb{N} \to [1, \infty)$ be a function. We say that an approximation algorithm computes an *r-approximate solution* for $F \in \mathrm{NPO}$ if $R\big(x, A(x)\big) \leq r$ for every $x \in I$. Then A is said to be an *r-approximate algorithm* and F is *r-approximable.*

A problem $F \in \mathrm{NPO}$ is in *APX* if F is c-approximable for a constant $c \geq 1$. It is in *poly-APX* if there exists a polynomial p such that F is $p(n)$-approximable, and it is in *exp-APX* if there exists a polynomial p such that F is $2^{p(n)}$-approximable. F is in *PTAS* if there exists an approximation algorithm that returns a c-approximate solution for x in time polynomial in $|x|$ on input (x, c) for any $x \in I$ and constant $c > 1$.

The notion of approximation preserving reductions we will use was first presented in [11], see also [2]. Let $F_1, F_2 \in \mathrm{NPO}$. F_1 *AP-reduces* to F_2 ($F_1 \leq_{\mathrm{AP}} F_2$) if there exist functions f, g that are computable in time polynomial in $|x|$ for r fixed and a constant $\alpha \geq 1$ such that for every $x \in I_{F_1}$ and for every $r \geq 1$

1. $f(x, r) \in I_{F_2}$.
2. If $S_{F_1}(x) \neq \emptyset$, then $S_{F_2}\big(f(x, r)\big) \neq \emptyset$.
3. $g(x, y, r) \in S_{F_1}(x)$ for every $y \in S_{F_2}\big(f(x, r)\big)$.
4. For every $y \in S_{F_2}\big(f(x, r)\big)$

$$R_{F_2}\big(f(x, r), y\big) \leq r \;\Rightarrow\; R_{F_1}\big(x, g(x, y, r)\big) \leq 1 + \alpha(r - 1).$$

Note that the additional parameter r is only required to preserve membership in PTAS. If the quality of the required solution is not needed for the reduction, the additional parameter will be omitted. This also implies that the time needed to compute f and g does not depend on it.

The AP-reduction is transitive, and PTAS, APX, poly-APX and exp-APX are closed under AP-reduction.

Since we will deal with average-case properties of optimization problems, we cannot separate the problems from the probability functions on their inputs. The pairs consisting of an optimization problem or a language and a probability function are called *distributional optimization problems* and *distributional problems* respectively.

A distributional problem (L, μ) is efficiently solvable on average if there exists an algorithm whose running time is polynomial on μ-average. A function $f : \Sigma^+ \to \mathbb{N}$ is *polynomial on μ-average*, where μ is a probability function on Σ^+, if there exist constants $k, c > 0$ such that

$$\sum_{x \in \Sigma^+} \frac{f^{1/k}(x)}{|x|} \mu(x) \leq c.$$

This definition was first stated by Levin in [20]. For a detailed discussion of the various ways to define functions polynomial on average see for example [16], [5], and [24].

A distributional problem (L, μ) belongs to *AvgP* if L is decidable by a deterministic algorithm whose running time is polynomial on μ-average. Besides AvgP, the best studied average-case complexity class is *DistNP*. It consists of all distributional problems (L, μ) where $L \in \mathrm{NP}$ and μ is P-computable. Ben-David et al. showed in [5] that if DistNP $\subseteq$ AvgP then E $=$ NE.

We first extend these notions to classify distributional optimization problems according to their approximability in average polynomial time.

Definition 1. *Let (F, μ) be a distributional optimization problem, $F \in \mathrm{NPO}$, A an approximation algorithm for F with performance ratio R whose running time is polynomial on μ-average.*

$(F, \mu) \in$ AvgPO if $R\big(x, A(x)\big) = 1$ for every $x \in I$.
$(F, \mu) \in$ Avg$_t$-PTAS if $R\big(x, A(x, c)\big) \leq c$ for every $x \in I, c > 1$.
$(F, \mu) \in$ Avg$_t$-APX if $R\big(x, A(x)\big) \leq c$ for every $x \in I$ and some constant $c \geq 1$.
$(F, \mu) \in$ Avg$_t$-poly-APX if there exists a polynomial p such that $R\big(x, A(x)\big) \leq p(|x|)$ for every $x \in I$.
$(F, \mu) \in$ Avg$_t$-exp-APX if there exists a polynomial p such that $R\big(x, A(x)\big) \leq 2^{p(|x|)}$ for every $x \in I$.
$(F, \mu) \in$ DistNPO if $F \in \mathrm{NPO}$ and μ is P-computable.

The subscripts denote that the average is taken over the time rather than the ratio. We skipped the subscript in the definition of the class AvgPO since the parameter on which the average is taken is clear.

The inclusion structure of these classes is examined in Section 4. Average constant ratio approximation classes are defined and examined in Sections 5 and 6. In the next section we show that DistNPO, like DistNP (see [20]), contains complete problems.

3 Reductions and DistNPO-Completeness

In order to define DistNPO-completeness, we present an average-case approximability preserving reduction, a distributional version of the AP-reduction introduced in the previous section. Though we will speak of an AP-reduction here as well, it will become clear from the context whether we refer to the distributional or non-distributional version.

Definition 2. *Let $(F_1, \mu_1), (F_2, \mu_2)$ be distributional optimization problems, $F_1, F_2 \in \mathrm{NPO}$. Then (F_1, μ_1) AP-reduces to (F_2, μ_2) (we write $(F_1, \mu_1) \leq_{\mathrm{AP}} (F_2, \mu_2)$ for short) if*

Reducibility *F_1 AP-reduces to F_2 via functions f, g and a constant $\alpha \geq 1$, and*
Dominance *μ_2 dominates μ_1 with respect to f, that means that there exists a polynomial p such that for every $r \geq 1$ and every $y \in \mathrm{range}(f)$*

$$\mu_2(y) \geq \sum_{x \in I_{F_1} \cdot f(x,r)=y} \frac{\mu_1(x)}{p(|x|)} \ .$$

Note that the dominance-requirement is the same as for the many-one reduction for distributional decision problems (see for example [16] for further details).

Lemma 1. *The AP-reduction for distributional optimization problems is transitive.*

Proof. Let $(F_1, \mu_1) \leq_{\mathrm{AP}} (F_2, \mu_2)$ via (f_1, g_1, α_1) and $(F_2, \mu_2) \leq_{\mathrm{AP}} (F_3, \mu_3)$ via (f_2, g_2, α_2). Since the AP-reduction for non-distributional optimization problems is transitive, it follows that $F_1 \leq_{\mathrm{AP}} F_3$. Let p_1 and p_2 be the polynomials verifying the dominance requirements. Then for all $r \geq 1$ and $z \in \mathrm{range}(f_1 \circ f_2)$

$$\mu_3(z) \geq \sum_{x . f_2(f_1(x,r))=z} \frac{\mu_1(x)}{(p_1(|x|) p_2(|f_1(x,r)|))} \ .$$

□

Lemma 2. $\mathrm{Avg_t}$-APX *is closed under AP-reduction.*

Proof. Let $(F_1, \mu_1) \leq_{\mathrm{AP}} (F_2, \mu_2)$ via (f, g, α) and $(F_2, \mu_2) \in \mathrm{Avg_t}$-APX which implies that there exists an approximation algorithm A that approximates F_2 within a constant factor c in time t that is polynomial on μ_2-average. Let f and g be computable in time bounded by polynomials p_f and p_g. Using A to approximate F_1 yields an algorithm A' that approximates F_1 within the constant factor $1+\alpha(c-1)$. The running time of A' on input x is then bounded by $p_f(|x|) + t(f(x,c)) + p_g(|x|)$ which was shown to be polynomial on μ_1-average in [16] if f is computable in polynomial time and μ_2 dominates μ_1 with respect to f. □

By the same argument it can be shown that the classes $\mathrm{Avg_t}$-PTAS, $\mathrm{Avg_t}$-poly-APX and $\mathrm{Avg_t}$-exp-APX are closed under AP-reduction as well.

We define hardness and completeness for classes of distributional optimization problems in the obvious way.

Definition 3. *Let $\mathcal{C}$ be a class of distributional optimization problems (F, μ) where $F \in$* NPO. *(F, μ) is* hard *for $\mathcal{C}$ with respect to AP-reduction if $(F', \mu') \leq_{\mathrm{AP}} (F, \mu)$ for all $(F', \mu') \in \mathcal{C}$. Furthermore, (F, μ) is* complete *for $\mathcal{C}$ if it is hard for $\mathcal{C}$ and $(F, \mu) \in \mathcal{C}$.*

The main contribution of this section is to show that DistNPO has complete problems. This is accomplished by showing that a distributional version of the Universal Maximization Problem (called *MaxU* for short, see [18], [12]) is complete for DistNPO.

Definition 4 (MaxU, $\mu_{\mathbf{MaxU}}$).

Input: *A tuple $(M, x, 1^k)$, where M is a nondeterministic Turing machine with a designated output tape that makes at most two nondeterministic choices in each computation step, and x is an input for M.*

Solution: *Any sequence $y \in \{0,1\}^*$ of nondeterministic choices of M that corresponds to a halting computation of at most k steps.*

Objective function: *The value $M(x, y)$ that M computes on input x following the computation path y.*

Probability distribution:

$$\mu_{\mathrm{MaxU}}(M, x, 1^k) = \left(\frac{1}{|M|^2} \cdot 2^{-|M|}\right) \left(\frac{1}{|x|^2} \cdot 2^{-|x|}\right) \frac{1}{k^2}.$$

Note that MaxU itself is NPO-complete as was shown in [12].

Theorem 1. $(\mathrm{MaxU}, \mu_{\mathrm{MaxU}})$ *is complete for* DistMaxNPO*, where* DistMaxNPO *is the class of all distributional maximization problems contained in* DistNPO.

Proof. Obviously, $(\mathrm{MaxU}, \mu_{\mathrm{MaxU}}) \in$ DistMaxNPO since MaxU $\in$ MaxNPO and the probability function μ_{MaxU} is P-computable.

Since we have to deal with the problem of reducing *any* P-computable probability distribution to μ_{MaxU} in a way that the dominance-requirement is fulfilled, we first need some technical details before we can show that any DistMaxNPO-problem reduces to $(\mathrm{MaxU}, \mu_{\mathrm{MaxU}})$. The same problem arises when proving DistNP-completeness, and we adapt the technique used in this context, see for example [16] and [5]. The key to solving this problem is an efficiently computable encoding of words that reflects – with respect to the given probability function – the probability with which the encoded word was chosen. The existence and properties of such an encoding were stated in the Coding Lemma in [5].

Coding Lemma. *Let μ be a* P*-computable probability function. Then there exists a coding function C_μ satisfying the following three conditions.*

Compression: $|C_\mu(x)| \leq 1 + \min\{|x|, -\log_2 \mu(x)\}$ *for all $x \in \Sigma^+$.*
Efficient Encoding: *The function C_μ is computable in polynomial time.*
Unique Decoding: *The function C_μ is one-to-one (i.e. $C_\mu(x) = C_\mu(x')$ implies $x = x'$).*

The function $C_{\mu(x)} = 0x$ if $\mu(x) \leq 2^{-|x|}$, and $C_\mu(x) = 1z$ otherwise, where z is the shortest binary string such that $\mu^*(x^-) < 0.z1 \leq \mu^*(x)$, has the desired properties. For the proof see [5], or [4] where a slightly different version of the Coding Lemma is presented.

Next, we can associate with any MaxNPO-problem F a nondeterministic Turing machine M_F that, on input x, nondeterministically guesses y and tests (in polynomial time) if $y \in S_F(x)$. If not, it loops infinitely, else computes $m_F(x, y)$ and writes that value on its output tape.

Let $(F, \mu) \in$ DistMaxNPO and x be an input for F with respect to the probability function μ. Using the encoding function C_μ and the nondeterministic Turing machine M_F, we construct a nondeterministic Turing machine $M_{F,\mu}$ with input $C_\mu(x)$. Given an input w, it computes x such that $C_\mu(x) = w$. If no such x exists, it loops infinitely, else simulates M_F on input x and writes the value computed by M_F on the output tape. Note that every y that encodes a halting computation of $M_{F,\mu}$ on input $C_\mu(x)$ also encodes a halting computation of M_F on input x, so $y \in S_F(x)$ and the output of $M_{F,\mu}$ is $m_F(x,y)$.

Let q_F, q_C be the polynomial time bounds for M_F and the computation of C_μ respectively. Let $p(|x|) = q_F(|x|) + q_C(|x|)$. So for $(F,\mu) \in$ DistMaxNPO, the functions $f(x) = (M_{F,\mu}, C_\mu(x), 1^{p(|x|)})$ for $x \in I_F$, $g(x,y) = y$ for all $y \in S_{\mathrm{MaxU}}(f(x))$ and the constant $\alpha = 1$ provide the desired reduction to $(\mathrm{MaxU}, \mu_{\mathrm{MaxU}})$.

That F AP-reduces to MaxU via f, g and α is easy to verify, so it remains to show that the dominance-requirement is fulfilled. According to the definition of μ_{MaxU} it holds that

$$\mu_{\mathrm{MaxU}}(M_{F,\mu}, C_\mu(x), 1^{p(|x|)}) = \left(\frac{1}{|M_{F,\mu}|^2} 2^{-|M_{F,\mu}|}\right) \left(\frac{1}{|C_\mu(x)|^2} 2^{-|C_\mu(x)|}\right) \frac{1}{p^2(|x|)},$$

where $|M_{F,\mu}|^2 2^{-|M_{F,\mu}|}$ is in fact a constant. Since $|C_\mu(x)| \leq 1 + \min\{|x|, -\log_2 \mu(x)\}$ according to the Coding Lemma, we have both $|C_\mu(x)| \leq \log_2 2\mu^{-1}(x)$ and $|C_\mu(x)| \leq 1 + |x|$. Thus

$$\mu_{\mathrm{MaxU}}(M_{F,\mu}, C_\mu(x), 1^{p(|x|)}) \geq \frac{c}{p^2(|x|)} \frac{1}{(1+|x|)^2} \frac{\mu(x)}{2} \quad \geq \quad \frac{\mu(x)}{p'(|x|)}$$

for some polynomial p' and constant c. Since C_μ is one-to-one, the function f is one-to-one as well and the dominance-requirement holds. □

A similar completeness result can be stated for $(\mathrm{MinU}, \mu_{\mathrm{MinU}}) \in$ DistMinNPO, the minimization version of MaxU, where $\mu_{\mathrm{MinU}} = \mu_{\mathrm{MaxU}}$. Using an auxiliary maximization or minimization problem respectively, we can show that $(\mathrm{MaxU}, \mu_{\mathrm{MaxU}})$ and $(\mathrm{MinU}, \mu_{\mathrm{MinU}})$ are complete for DistNPO.

Theorem 2. $(\mathrm{MaxU}, \mu_{\mathrm{MaxU}})$ *and* $(\mathrm{MinU}, \mu_{\mathrm{MinU}})$ *are complete for* DistNPO.

Proof. We reduce $(\mathrm{MaxU}, \mu_{\mathrm{MaxU}})$ to $(\mathrm{MinU}', \mu_{\mathrm{MinU}'})$. The minimization problem MinU′ is defined like MinU except for the objective function which is $\lfloor 2^{2k}/M(x,y) \rfloor$ on input $(M, x, 1^k)$ and computation path y. Let $X = (M, x, 1^k)$. The functions required for this reduction are the identity functions $f(X) = X$ and $g(X,y) = y$. Since the output of M is bounded by 2^k on computations of length k, there can be no feasible solutions $y, y' \in S_{\mathrm{MinU}'}(f(X)) = S_{\mathrm{MaxU}}(X)$ such that $m_{\mathrm{MinU}'}(f(X), y) = m_{\mathrm{MinU}'}(f(X), y')$ but $m_{\mathrm{MaxU}}(X,y) \neq m_{\mathrm{MaxU}}(X,y')$. Hence $(\mathrm{MaxU}, \mu_{\mathrm{MaxU}})$ is AP-reducible to $(\mathrm{MinU}', \mu_{\mathrm{MinU}'})$ which is in turn AP-reducible to $(\mathrm{MinU}, \mu_{\mathrm{MinU}})$, so $(\mathrm{MinU}, \mu_{\mathrm{MinU}})$ is complete for DistNPO.

Similarly, the DistNPO-completeness of $(\mathrm{MaxU}, \mu_{\mathrm{MaxU}})$ can be shown. □

4 Average Polynomial Time Approximation Classes

After we have shown that DistNPO has complete problems, we take a closer look at the inclusion structure of the average polynomial time approximation classes introduced in Definition 1.

The inclusions $\mathrm{AvgPO} \subseteq \mathrm{Avg_t\text{-}PTAS} \subseteq \mathrm{Avg_t\text{-}APX} \subseteq \mathrm{Avg_t\text{-}poly\text{-}APX} \subseteq \mathrm{Avg_t\text{-}exp\text{-}APX}$ follow immediately.

We show that if $\mathrm{DistNP} \nsubseteq \mathrm{AvgP}$, there are distributional optimization problems that are not even approximable within an exponential factor in average polynomial time, and that the inclusion $\mathrm{Avg_t\text{-}APX} \subseteq \mathrm{Avg_t\text{-}poly\text{-}APX}$ is strict. If $\mathrm{DistNP} \subseteq \mathrm{AvgP}$ on the other hand, we show that $\mathrm{DistNPO} \subseteq \mathrm{Avg_t\text{-}PTAS}$.

Theorem 3. *If* $\mathrm{DistNPO} \subseteq \mathrm{Avg_t\text{-}exp\text{-}APX}$*, then* $\mathrm{DistNP} \subseteq \mathrm{AvgP}$.

Proof. Let $(L, \mu) \in \mathrm{DistNP}$. Then there exists a nondeterministic Turing machine M_L and a polynomial p such that $x \in L$ iff there exists an accepting computation of M that requires at most $p(|x|)$ steps. M_L can be easily modified to a nondeterministic Turing machine M'_L that writes its accepting path on an output tape. Using the same technique as in the proof of Theorem 1, we construct a nondeterministic Turing machine $M'_{L,\mu}$ that expects inputs $C_\mu(x)$ rather than x, and reduce the question whether $x \in L$ to the question whether $S_{\mathrm{MaxU}}(M'_{L,\mu}, C_\mu(x), 1^{p'(|x|)}) \neq \emptyset$, where $p'(|x|) = p(|x|) + q(|x|)$ and $q(|x|)$ is the polynomial time bound needed to compute C_μ.

If the assumption holds, there exists an $\mathrm{Avg_t\text{-}exp\text{-}APX}$-algorithm for $(\mathrm{MaxU}, \mu_{\mathrm{MaxU}})$ that returns a valid solution for an input $(M, x, 1^k)$ whenever it exists. This algorithm can then be used to decide whether $S_{\mathrm{MaxU}}(M'_{L,\mu}, C_\mu(x), 1^{p'(|x|)}) \neq \emptyset$ and hence whether $x \in L$. □

To show that the inclusion $\mathrm{Avg_t\text{-}APX} \subseteq \mathrm{Avg_t\text{-}poly\text{-}APX}$ is strict if $\mathrm{DistNP} \nsubseteq \mathrm{AvgP}$, we use the maximum independent set problem (MaxIndSet for short). In an undirected graph $G = (V, E)$ a vertex set $V' \subseteq V$ is *independent* if $(u, v) \notin E$ for every $u, v \in V'$. To solve MaxIndSet we need to find an independent set of maximal size in the given graph. MaxIndSet is trivially approximable within a factor of n in the worst case, where n is the number of vertices in the graph. It follows that $(\mathrm{MaxIndSet}, \mu) \in \mathrm{Avg_t\text{-}poly\text{-}APX}$ for every distribution μ.

Theorem 4. *If* $\mathrm{DistNP} \nsubseteq \mathrm{AvgP}$*, then* $\mathrm{Avg_t\text{-}APX} \subsetneq \mathrm{Avg_t\text{-}poly\text{-}APX}$.

Proof. We show that if $(\mathrm{MaxIndSet}, \nu) \in \mathrm{Avg_t\text{-}APX}$, where ν will be given below, then $\mathrm{DistNP} \subseteq \mathrm{AvgP}$.

For the proof we use DistNP-complete problems and the technique for showing that MaxIndSet is not approximable within any constant factor unless $\mathrm{P} = \mathrm{NP}$ (see [14]), which is based on the PCP-characterization of NP [1]. Let (L, μ) be complete for DistNP, and suppose that $(\mathrm{MaxIndSet}, \nu)$ is approximable by an algorithm A within a constant ratio $d \in \mathbb{N}^+$ in time polynomial on ν-average. This algorithm can then be used to decide membership in L in time polynomial on μ-average thus implying $\mathrm{DistNP} \subseteq \mathrm{AvgP}$.

Since $L \in \mathrm{NP} = \mathrm{PCP}(\log n, 1)$, there exists a PCP-verifier with error probability less than δ for any positive constant $\delta < 1$, that requires $c_r \log n$ random bits and c_q queries for constants c_r, c_q that depend on δ. A PCP-verifier is a probabilistic Turing machine that has access to a proof via an oracle that, given a proof position as input, returns the corresponding bit of the proof. For every $x \in L$ there exists a proof such that the verifier accepts with probability 1. If $x \notin L$, for every proof the verifier accepts with probability $\leq \delta$.

Using this verifier for $\delta = 1/d$, we can construct a graph G_x^δ for every x in polynomial time such that if $x \in L$, the size $\alpha(G_x^\delta)$ of the largest independent set in G_x^δ is $2^{c_r \log n}$ and

$\alpha(G_x^\delta) < \delta 2^{c_r \log n}$ if $x \notin L$. Every accepting run of the verifier is encoded as a vertex. If two vertices are not consistent to the same proof, they are connected via an edge. For details of the original construction see [14].

We set $\nu(G) = \delta^2\mu(x)$ if $G = G_x^\delta$ and 0 otherwise. We can now decide whether $x \in L$ by first constructing G_x^δ and then using A to decide whether $\alpha(G_x^\delta) \geq \delta 2^{c_r \log n}$. If the running time t_A of A is polynomial on ν-average, the running time $t_L(x) = |x|^l + t_A(G_x^\delta)$ of this procedure is polynomial on μ-average, where $|x|^l$ is the time required to compute G_x^δ. □

If DistNP $\subseteq$ AvgP, combining a result by Schuler and Watanabe [23] with a recent result by Buhrman, Fortnow and Pavan [8] shows that then every DistNPO-problem is in Avg_t-PTAS.

Theorem 5. *If* DistNP $\subseteq$ AvgP *then* DistNPO $\subseteq$ Avg_t-PTAS.

Proof. (Sketch) In [23] it was shown that DistNP $\subseteq$ AvgP implies that every $(F, \mu) \in$ DistNPO has a randomized μ-average polynomial-time approximation scheme, a randomized algorithm that for every x and $c > 1$ computes a solution $A(x, c)$ such that $\Pr\big[R\big(x, A(x)\big) \leq c\big] \geq 2/3$ and for every c the running time $t_c(x)$ of A is polynomial on μ-average.

Buhrman, Fortnow and Pavan proved in [8] that if DistNP $\subseteq$ AvgP, efficient pseudorandom generators exist, so the randomized μ-average PTAS can be efficiently derandomized. □

5 Constant on Average Performance Ratio

In order to relax worst-case approximability within a constant factor to average-case approximability within a constant factor, we introduce functions that are *constant on average.* Generalizations of functions that are polynomial on average to some non-polynomial functions that are not constant have already been studied by Cai, Selman, and Ben-David et al. in [9] and [5]. Their basic idea is that, given functions $f : \Sigma^+ \to \mathbb{N}$ and $T : \mathbb{N} \to \mathbb{N}$, one can say that f is T *on average*, if $T^{-1}\big(f(x)\big)$ is linear on average. However, this generalization does not directly capture the notion of functions that are constant on average since constant functions are not invertible. But we can express constant functions in terms of linear functions nevertheless. Two plausible ways of calling a function f *constant on μ-average* are the following. Either

1. the function $|x|^{\frac{1}{k}f(x)}$ is linear on μ-average for some $k > 0$, or
2. the function $|x|f(x)$ is linear on μ-average.

Every function that is constant on average according to the first condition is also constant on average according to the second condition. But the first condition turns out to be too strict. If the lengthwise expectation of a function is bounded by a constant, the function itself is not necessarily constant on average according to that condition. For the second condition on the other hand, this implication holds. We state this, in a more general form, in Lemma 3.

So we adapt the definition of functions polynomial on μ-average from [9] in order to define functions that are constant on μ-average using the second condition and considering $f^{1/k}$ for some $k > 1$ rather than f which ensures several closure properties that are stated below.

Definition 5. *Let μ be a probability function on Σ^+. A function $f : \Sigma^+ \to \mathbb{R}$ is* constant on μ-average *if there exist constants $k, c > 0$ such that for every $n > 0$*

$$\sum_{x \in \Sigma^{\geq n}} f^{1/k}(x) \mu_{\geq n}(x) \leq c.$$

Note that it is not sufficient that the expectation over Σ^+ is bounded by c instead of the expectation over $\Sigma^{\geq n}$ for every $n > 0$. Else, every polynomial would be constant on average with respect to the standard probability function μ_0. But if we consider functions that depend solely on the length of a word rather than on the word itself, with Definition 5 no such function is constant on average with respect to μ_0 if it is unbounded, but every function bounded by a constant is constant on μ-average for any distribution μ. Furthermore, if two functions $f, g : \Sigma^+ \to \mathbb{R}$ are constant on μ-average for some probability function μ, so are the functions $\max\{f, g\}, f + g, f \cdot g$ and f^r for every constant r.

The relationship between functions f whose lengthwise expectation $\mathrm{E}_n[f^{1/k}]$ is bounded by a constant and functions that are constant on average is stated in the following lemma which is often helpful for showing that a function is constant on μ-average.

Lemma 3. *Let $f : \Sigma^+ \to \mathbb{R}$ be a function and μ a probability function on Σ^+. If there exist constants $k, c > 0$ such that for every $n > 0$*

$$\mathrm{E}_n[f^{1/k}] = \sum_{x \in \Sigma^n} f^{1/k}(x) \mu_n(x) \ \leq \ c,$$

then f is constant on μ-average.

Proof. It is easy to see that for every $n > 0$

$$\sum_{x \in \Sigma^{\geq n}} f^{1/k}(x) \mu(x) = \sum_{m \geq n} \mu(\Sigma^m) \sum_{x \in \Sigma^m} f^{1/k}(x) \mu_m(x) \ \leq \ c\mu(\Sigma^{\geq n})$$

and that this inequality is equivalent to the inequality in Definition 5. $\square$

Combining the above lemma with a general upper bound on the expectation of the performance ratio R of an approximation algorithm yields a sufficient condition for R being constant on μ-average.

Lemma 4. *Let (F, μ) be a distributional maximization problem, $F \in$ NPO, and A an approximation algorithm for F. Let $g : \mathbb{N}^+ \to \mathbb{N}^+$ be an upper bound on m, that is $m(x, y) \leq g(|x|)$ for all $x \in I$, $y \in S(x)$. If there exist constants $k, c \geq 1$ and thresholds $\theta_{\mathrm{opt}}(n), \theta_{\mathrm{A}}(n)$ on $\mathrm{opt}(x)$ and $m(x, A(x))$ respectively such that*

1. $\left(\theta_{\mathrm{opt}}(n)/\theta_{\mathrm{A}}(n)\right)^{1/k} \leq c$,
2. $\lim_{n\to\infty} \left(g^{1/k}(n) - \theta_{\mathrm{opt}}^{1/k}(n)\right) \mathrm{Pr}_n\left[\mathrm{opt}(x) > \theta_{\mathrm{opt}}(n)\right] = 0$,
3. $\lim_{n\to\infty} \theta_{\mathrm{opt}}^{1/k}(n)\, \mathrm{Pr}_n\left[m\big(x, A(x)\big) < \theta_{\mathrm{A}}(n)\right] = 0$,

then the performance ratio R of A is constant on μ-average.

Proof. For every $k \geq 1$ and $n > 0$ the expectation $\mathrm{E}_n[R^{1/k}]$ is bounded above by

$$\mathrm{E}_n[R^{1/k}] = \sum_{x \in \Sigma^n} R^{1/k}\big(x, A(x)\big)\mu_n(x) \;\leq\; \big(g^{1/k}(n) - \theta_{\mathrm{opt}}^{1/k}(n)\big)\Pr_n\big[\mathrm{opt}(x) > \theta_{\mathrm{opt}}(n)\big]$$
$$+\,\theta_{\mathrm{opt}}^{1/k}(n)\Pr_n\big[m\big(x, A(x)\big) < \theta_{\mathrm{A}}(n)\big] + \left(\frac{\theta_{\mathrm{opt}}(n)}{\theta_{\mathrm{A}}(n)}\right)^{1/k}\Pr_n\big[m\big(x, A(x)\big) \geq \theta_{\mathrm{A}}(n)\big].$$

This bound is obtained by splitting the sum and using the threshold values and g as bounds. If the conditions hold, $\lim_{n\to\infty}\mathrm{E}_n[R^{1/k}] = c$ and $\mathrm{E}_n[R^{1/k}]$ can be bounded by a constant. Applying Lemma 3 concludes the proof. □

A similar sufficient condition can be given for minimization problems as well.

We illustrate the usefulness of this lemma by analyzing the greedy algorithm for approximating MaxIndSet as an example. Given $G = (V, E)$ as input, this algorithm picks a vertex from V, deletes all its neighbours, picks the next vertex from V, deletes all its neighbours and so on, until V is empty.

Let α_n denote the size of the maximum independent set in a random graph $G \in \mathcal{G}_{n,p}$, σ_n the size of the independent set found by the greedy algorithm, $q(n) = 1 - p(n)$, and $b = -\ln q(n)$. In [21] McDiarmid showed that the performance ratio of the greedy algorithm is bounded with high probability by $2(1-\delta)^{-1}$ for every δ such that $0 < \delta < 1$ by proving the following.

Fact 1. *If* $\lim_{n\to\infty} np(n) = \infty$ *then*

$$\lim_{n\to\infty}\Pr_n[\alpha_n \leq 2b^{-1}\ln np(n)] = 1 \textit{ and}$$
$$\lim_{n\to\infty}\Pr_n[\sigma_n \geq (1-\delta)b^{-1}\ln np(n)] = 1 \textit{ for every } \delta \textit{ with } 0 < \delta < 1.$$

But this result does not already imply that the performance ratio of the greedy algorithm for MaxIndSet is constant on average as well. To prove this, we need to show that the probabilities of the optimal value being too large and the value of the solution found being too small converge to 0 fast enough so that Lemma 4 holds. Reviewing the proofs in [21] yields the following result.

Theorem 6. *Let μ_G be a probability function on random graphs according to the $\mathcal{G}_{n,p}$-model such that $p(n) \geq n^{-(1-\varepsilon)}$ for some $\varepsilon > 0$ and $\lim_{n\to\infty} p(n) = 0$. Then the performance ratio of the greedy algorithm for* MaxIndSet *is constant on μ_G-average.*

Proof. In [21] it was shown for $\theta_{\mathrm{opt}}(n) = 2b^{-1}\log np(n)$, and $\theta_{\mathrm{A}}(n) = (1-\delta)b^{-1}\log np(n)$ for some $\delta \in (0,1)$ that

$$\Pr_n[\alpha_n > \theta_{\mathrm{opt}}] \leq \binom{n}{\theta_{\mathrm{opt}}} q(n)^{\binom{\theta_{\mathrm{opt}}}{2}}, \text{ and}$$
$$\Pr_n[\sigma_n < \theta_{\mathrm{A}}] \leq \theta_{\mathrm{A}}\left(e^{-(np(n))^{\delta}/(p(n)\theta_{\mathrm{A}})}\right).$$

With $g(n) = n$, further examination of this proof yields

$$\lim_{n\to\infty} g(n)\Pr_n[\alpha_n > \theta_{\mathrm{opt}}] = \lim_{n\to\infty}(np(n))^{1/\varepsilon+1+2b^{-1}(2-\ln\ln np(n))} = 0,$$

and for every $\delta \in (0,1)$ with $c = (2+2\varepsilon)\varepsilon^{-1}$

$$\lim_{n\to\infty} \theta_{\mathrm{opt}} \Pr_n[\sigma_n < \theta_{\mathrm{A}}] = \lim_{n\to\infty} \left(\exp\left(c \ln np(n) - \frac{(np(n))^{\delta}}{\ln np(n)} \right) \right) = 0$$

which proves that condition 2 and 3 of Lemma 4 hold. Condition 1 of Lemma 4 holds for some $\delta \in (0,1)$ since $\theta_{\mathrm{opt}}(n)/\theta_{\mathrm{A}}(n) = 2/(1-\delta)$ is bounded by a constant. Applying Lemma 4 for $k = 1$ concludes the proof. □

Using functions that are constant on average, we define the following average case approximation classes.

Definition 6. *Let (F,μ) be a distributional optimization problem, $F \in$ NPO, A an approximation algorithm for F with running time T and performance ratio R.*

$(F,\mu) \in Avg_r$-APX if R is constant on μ-average and T is bounded by a polynomial.
$(F,\mu) \in Avg_{r,t}$-APX if R is constant on μ-average and T is polynomial on μ-average.

6 Average Constant Ratio Approximation Classes

In this section we examine the inclusion structure of average case approximation classes obtained by relaxing the requirements on the performance ratio from worst case to average case.

The inclusions $\mathrm{Avg_r}$-APX $\subseteq$ $\mathrm{Avg_{r,t}}$-APX and $\mathrm{Avg_t}$-APX $\subseteq$ $\mathrm{Avg_{r,t}}$-APX are obvious. A closer look at the nature of $\mathrm{Avg_r}$-APX-algorithms yields the following theorem.

Theorem 7. *If* DistNPO $\subseteq$ $\mathrm{Avg_r}$-APX, *then* P = NP.

Proof. In [2] it is stated that NPO $\subseteq$ exp-APX implies P = NP. So it remains to show that DistNPO $\subseteq$ $\mathrm{Avg_r}$-APX implies NPO $\subseteq$ exp-APX. Since every $\mathrm{Avg_r}$-APX-algorithm returns a valid solution whenever it exists, the performance ratio of such an algorithm is bounded by $2^{p(|x|)}$ for some polynomial p in the worst case. So if $(F,\mu) \in$ DistNPO is in $\mathrm{Avg_r}$-APX, it follows that $F \in$ exp-APX. □

For NP-optimization problems that belong to poly-APX, membership in $\mathrm{Avg_t}$-APX or $\mathrm{Avg_{r,t}}$-APX already induces membership in $\mathrm{Avg_r}$-APX if their probability functions are well-behaved. Note that poly-APX contains – among others – those NP-optimization problems whose objective function is polynomially bounded, and for which at least a trivial solution can be computed in polynomial time for every $x \in I$.

Theorem 8. *Let (F,μ) be a distributional optimization problem with $F \in$* poly-APX *and $\mu(\Sigma^{\geq n}) \geq n^{-s}$ for some $s > 0$. If $(F,\mu) \in$* $\mathrm{Avg_{r,t}}$-APX *then $(F,\mu) \in$* $\mathrm{Avg_r}$-APX.

Proof. Since $F \in$ poly-APX, there exists an approximation algorithm A with polynomial running time t_A such that $R\big(x, A(x)\big) \leq |x|^l$ for some $l \geq 1$. Since $(F,\mu) \in$ $\mathrm{Avg_{r,t}}$-APX, there exists an approximation algorithm B for F with running time t_B and performance ratio $R_B\big(x, B(x)\big)$ such that t_B is polynomial on μ-average and R_B is constant on μ-average via constants $r, d > 0$, that is for every $n > 0$

$$\sum_{x \in \Sigma^{\geq n}} R_B^{1/r}\big(x, B(x)\big)\mu_{\geq n}(x) \leq d.$$

Cai and Selman proved in [9] that if $\mu(\Sigma^{\geq n}) \geq n^{-s}$ there exist $k, c > 0$ such that for every $n > 0$

$$\sum_{x \in \Sigma^{\geq n}} \frac{t_B^{1/k}(x)}{|x|} \mu_{\geq n}(x) \leq c.$$

Now, simulate B on input x for $|x|^{2k}$ steps. If B succeeds, return the solution computed by B. Otherwise run A on input x and return the solution computed by A. This procedure requires $\mathcal{O}(|x|^{2k} + t_A(x))$ steps.

If the simulation of B on input x succeeds within $|x|^{2k}$ steps, the performance ratio of B' is constant on μ-average via $r, d > 0$, otherwise it is bounded by $|x|^l$. Define for every $n > 0$ sets $S_{\geq n} := \{x \in \Sigma^{\geq n} \mid |x|^{l/l'} < t_B^{1/k}(x)|x|^{-1}\}$ and $T_{\geq n} := \Sigma^{\geq n} \setminus S_{\geq n}$. For every $n > 0$ and $l' = \max\{r, l\}$

$$\begin{aligned}\sum_{x \in \Sigma^{\geq n}} R^{1/l'}\bigl(x, B'(x)\bigr)\mu_{\geq n}(x) &\leq \sum_{x \in T_{\geq n}} R_B^{1/r}\bigl(x, B(x)\bigr)\mu_{\geq n}(x) + \sum_{x \in S_{\geq n}} \frac{t_B^{1/k}(x)}{|x|}\mu_{\geq n}(x)\\ &\leq d + c,\end{aligned}$$

and thus the performance ratio of B' is constant on μ-average. □

By the same technique we can construct an Avg$_\mathrm{r}$-APX-algorithm from an Avg$_\mathrm{t}$-APX-algorithm if the other conditions hold. But there are $(F, \mu) \in$ DistNPO with $F \in$ poly-APX but $(F, \mu) \notin$ Avg$_\mathrm{t}$-APX unless DistNP $\subseteq$ AvgP; this follows from the proof of Theorem 4, since the distribution ν used in that proof is P-computable.

The "reversal" of this construction, however, seems to be hard, since it requires measuring the quality of the solution computed so far. One has to check if this solution is close enough to the optimum or if a better solution is required. For the latter case we need another algorithm that may need more than polynomial time but computes a solution close enough to the optimum. In order to decide whether a solution is good enough or not, either the value of an optimal solution has to be computed, which is not possible in general unless P = NP, or the input instances for which the polynomial-time approximation algorithm succeeds must share some property that can be checked easily. The average polynomial-time algorithms for approximating the minimum graph colouring problem presented by Coja-Oghlan and Taraz in [10] at this year's STACS give a nice illustration of this technique.

However, in [6] and [17] it is shown that the task of determining a "good" instance for the greedy algorithm for finding a maximum independent set is hard for the class of sets solvable via parallel access to an NP-oracle. Thus it seems unlikely that a "reverse" version of Theorem 8 or a slightly weaker statement might hold in general.

References

1. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof Verification and Hardness of Approximation Problems. In *Proc. 33rd FOCS*, pages 14–23. IEEE Computer Society Press, 1992.
2. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation.* Springer Verlag, 1999.

3. J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer Verlag, second edition, 1995.
4. J. Belanger and J. Wang. On the NP-isomorphism with respect to random instances. *Journal of Computer and System Sciences*, 50:151–164, 1995.
5. S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the Theory of Average Case Complexity. *Journal of Computer and System Sciences*, 44:193–219, 1992.
6. H. L. Bodlaender, D. M. Thilikos, and K. Yamazaki. It is Hard to Know when Greedy is Good for Finding Independent Sets. *Information Processing Letters*, 61(2):101–106, 1997.
7. B. Bollobás. *Random Graphs.* Academic Press, 1985.
8. H. Buhrman, L. Fortnow, and A. Pavan. Some Results on Derandomization. In *Proc. 20th STACS*, volume 2607 of *LNCS*, pages 212–222. Springer-Verlag Berlin, 2003.
9. J.-Y. Cai and A. L. Selman. Fine Separation of Average-Time Complexity Classes. *SIAM Journal on Computing*, 28(4):1310–1325, 1999.
10. A. Coja-Oghlan and A. Taraz. Colouring Random Graphs in Expected Polynomial Time. In *Proc. 20th STACS*, volume 2607 of *LNCS*, pages 487–498. Springer-Verlag Berlin, 2003.
11. P. Crescenzi, V. Kann, R. Silvestri, and L. Trevisan. Structure in Approximation Classes. *SIAM Journal on Computing*, 28(5):1759–1782, 1999.
12. P. Crescenzi and A. Panconesi. Completeness in Approximation Classes. *Information and Computation*, 93:241–262, 1991.
13. D.-Z. Du and K. Ko. *Theory of Computational Complexity.* John Wiley and Sons, 2000.
14. U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating Clique is almost NP-complete. In *Proc. 32nd FOCS*, volume 32, pages 2–12. IEEE Computer Society Press, 1991.
15. G. Grimmet and C. McDiarmid. On Colouring Random Graphs. *Mathematical Proc. of the Cambridge Philosophical Society*, 77:313–324, 1975.
16. Y. Gurevich. Average Case Completeness. *Journal of Computer and System Sciences*, 42:346–398, 1991.
17. E. Hemaspaandra and J. Rothe. Recognizing When Greed Can Approximate Maximum Independent Sets is Complete for Parallel Access to NP. Technical Report Math/Inf/97/14, Fakultät für Mathematik und Informatik, Friedrich-Schiller Universität Jena, 1997.
18. M. W. Krentel. The Complexity of Optimization Problems. *Journal of Computer and System Sciences*, 36:490–509, 1988.
19. B. Kreuter and T. Nierhoff. Greedily Approximating the r-Independent Set and k-Center Problems on Random Instances. In *Randomization and Approximation Techniques in Computer Science*, volume 1269 of *Lecture Notes in Computer Science*. Springer Verlag, 1997.
20. L. Levin. Problems, Complete in "Average" Instance. In *Proc. 16th STOC*, page 465. ACM Press, 1984.
21. C. McDiarmid. Colouring Random Graphs. *Annals of Operations Research*, 1:183–200, 1984.
22. C. Papadimitriou. *Computational Complexity.* Addison Wesley, 1994.
23. R. Schuler and O. Watanabe. Toward Average-Case Complexity Analysis of NP Optimization Problems. In *Proc. 10th Conference on Structure in Complexity Theory*, pages 148–159. IEEE Computer Society Press, 1995.
24. J. Wang. Average-Case Computational Complexity Theory. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*, pages 295–328. Springer Verlag, 1997.