Texts in Computational Complexity: Average-Case Complexity

Oded Goldreich
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot, ISRAEL.

January 28, 2006

Teaching note: We view average-case complexity as referring to the performance on average (or typical) instances, and not as the average performance on random instances. This choice is justified in Section 1.1. Thus, the current theory may be termed typical-case complexity.

Introduction

Our approach so far is termed worst-case complexity, because it refers to the performance of potential algorithms on each legitimate instance (and hence to the performance on the worst possible instance). That is, computational problems were defined as referring to a set of instances and performance guarantees were required to hold for each instance in this set. In contrast, average-case complexity allows ignoring a negligible measure of the possible instances, where *the identity of the ignored instances is determined by the analysis of potential solvers and not by the problem's statement.*

A few comments are in place. Firstly, as just hinted, the standard statement of the worst-case complexity of a computational problem (especially one having a promise) may also ignores some instances (i.e., those considered inadmissible or violating the promise), but these instances are determined by the problem's statement. In contrast, the inputs ignored in average-case complexity are not inadmissible in any inherent sense (and are certainly not identified as such by the problem's statement). It is just that they are viewed as exceptional when claiming that a specific algorithm solve the problem; furthermore, these exceptional instances are determined by the analysis of that algorithm. Needless to say, these exceptional instances ought to be rare (i.e., occur with negligible probability).

The last sentence raises a couple of issues. Firstly, a distribution on the set of admissible instances has to be specified. In fact, we shall consider a new type of computational problems, each consisting of a standard computational problem coupled with a probability distribution on instances. Consequently, the question of which distributions should be considered arises. These and numerous other definitional issues will be addressed in Section 1.1.

Before proceeding, let us spell out the rather straightforward motivation to the study of the average-case complexity of computational problems. It is that, in real-life applications, one may be perfectly happy with an algorithm that solves the problem fast on almost all instances that arise in the application. That is, one may be willing to tolerate error provided that it occurs with negligible probability, where the probability is taken over the distribution of instances encountered in the application. We stress that a key aspect in this approach is a good modeling of the type of distributions of instances that are encountered in natural algorithmic applications.

At this point a natural question arises: *can natural computational problems be solve efficiently when considering typical instances?* The bottom-line of this section is that, for a well-motivated choice of definitions, our conjecture is that the "distributional version" of NP is not contained in the average-case (or typical-case) version of P. This means that some NP problems are not merely hard in the worst-case, but rather "typically hard" (i.e., hard on typical instances drawn from some simple distribution). Specifically, hard instances may occur in natural algorithmic applications (and not only in cryptographic (and other "adversarial") applications that are design on purpose to produce hard instances). This conjecture motivates the development of an average-case analogue of NP-completeness, which will be presented in this section.

Organization. A major part of our exposition is devoted to the definitional issues that arise when developing a general theory of average-case complexity. These issues are discussed in Section 1.1. In Section 1.2 we prove the existence of a distributional problem that is "NP-complete" in the average-case complexity sense. In Section 1.3 we extend the treatment to randomized algorithms. Additional ramifications are presented in Section 2.

1 The basic theory

In this section we provide a basic treatment of the theory of average-case complexity, while postponing important ramifications to Section 2. The basic treatment contains the preferred definitional choices for the main notions as well as the identification of a complete problem for a natural class of average-case computational problems.

1.1 Definitional issues

The theory of average-case complexity is more subtle than may appear in first thought. In addition to the generic difficulty involved in defining relaxations, difficulties arise from the "interface" between standard probabilistic analysis and the conventions of complexity theory. This is most striking in the definition of the class of feasible average-case computations. Referring to the theory of worst-case complexity as a guideline, we shall address the following aspects of the analogous theory of average-case complexity.

1. *Setting the general framework.* We shall consider distributional problems, which are standard computational problems coupled with distributions on the relevant instances.

2. *Identifying the class of feasible* (distributional) *problems.* Seeking an average-case analogue of classes such as $\mathcal{P}$, we shall reject the first definition (of "average polynomial time") that comes to mind, briefly discuss several related alternatives, and adopt one of them for the main treatment.

3. *Identifying the class of interesting* (distributional) *problems.* Seeking an average-case analogue of the class $\mathcal{NP}$, we shall avoid both the extreme of allowing arbitrary distributions (which collapses average-case complexity to worst-case complexity) and the opposite extreme of confining the treatment to the uniform distribution (which is misguided by the naive assumption that this distribution is the only one relevant to applications).

4. *Developing an adequate notion of reduction among* (distributional) *problems.* As in the theory of worst-case complexity, this notion should preserve feasible solveability (in the current distributional context).

We now turn to the actual treatment of each of the aforementioned aspects.

Distributional problems. Focusing on decision problems, we define distributional problems as pairs consisting of a decision problem and a probability ensemble.[1] For simplicity, here a probability ensemble $\{X_n\}_{n\in\mathbb{N}}$ is a sequence of random variables such that X_n ranges over $\{0,1\}^n$. Thus, $(S, \{X_n\}_{n\in\mathbb{N}})$ is the distributional problem consisting of the problem of deciding membership in the set S with respect to the probability ensemble $\{X_n\}_{n\in\mathbb{N}}$. (The treatment of search problem is similar; see Section 2.1.) We denote the uniform probability ensemble by $U = \{U_n\}_{n\in\mathbb{N}}$; that is, U_n is uniform over $\{0,1\}^n$.

Identifying the class of feasible problems. The first idea that comes to mind is defining the problem $(S, \{X_n\}_{n\in\mathbb{N}})$ as feasible (on the average) if there exists an algorithm A that solves S such that the *average running time* of A on X_n is bounded by a polynomial in n (i.e., there exists a polynomial p such that $\mathsf{E}[t_A(X_n)] \leq p(n)$, where $t_A(x)$ denotes the running-time of A on input x). The problem with this definition is that it very sensitive to the model of computation and is not closed under algorithmic composition. Both deficiencies are a consequence of the fact that t_A may be polynomial on the average with respect to $\{X_n\}_{n\in\mathbb{N}}$ but t_A^2 may fail to be so (e.g., consider $t_A(x'x'') = 2^{|x'|}$ if $x' = x''$ and $t_A(x'x'') = |x'x''|^2$ otherwise, coupled with the uniform distribution over $\{0,1\}^n$). We conclude that the average running-time of algorithms is not a robust notion. We also doubt the naive appeal of this notion, and view the typical running time of algorithms (as defined next) as a more natural notion. Thus, we shall consider an algorithm as feasible if its running-time is typically polynomial.[2]

We say that A is typically polynomial-time on $X = \{X_n\}_{n\in\mathbb{N}}$ if there exists a polynomial p such that the probability that A runs more that $p(n)$ steps on X_n is *negligible* (i.e., for every polynomial q and all sufficiently large n it holds that $\Pr[t_A(X_n) > p(n)] < 1/q(n)$). The question is what is required in the "untypical" cases, and two possible definitions follow.

1. The simpler option is saying that $(S, \{X_n\}_{n\in\mathbb{N}})$ is (typically) feasible if there exists an algorithm A that solves S such that A is typically polynomial-time on $X = \{X_n\}_{n\in\mathbb{N}}$. This effectively requires A to correctly solve S on each instance, which is more than was required in the motivational discussion. (Indeed, if the underlying reasoning is ignoring rare cases, then we should ignore them altogether rather than partially (i.e., only ignore their affect on the running-time).)

[1]We mention that even this choice is not evident. Specifically, Levin [8] (see discussion in [3]) advocates the use of a single probability distribution defined over the set of all strings. His argument is that this makes the theory less representation-dependent. At the time we were convinced of his argument (see [3]), but currently we feel that the representation-dependent effects discussed in [3] are legitimate. Furthermore, the alternative formulation of [3] comes across as unnatural and tends to be confusing.

[2]An alternative choice, taken by Levin [8] (see discussion in [3]), is considering as feasible (w.r.t $X = \{X_n\}_{n\in\mathbb{N}}$) any algorithm that runs in time that is polynomial in a function that is linear on the average (w.r.t X); that is, requiring that there exists a polynomial p and a function $\ell : \{0,1\}^* \to \mathbb{N}$ such that $t(x) \leq p(\ell(x))$ and $\mathsf{E}[\ell(X_n)] = O(n)$. This definition is robust (i.e., it does not suffer from the aforementioned deficiencies) and is arguably as justified as the naive definition (i.e., $\mathsf{E}[t_A(X_n)] \leq \mathrm{poly}(n)$).

2. The alternative, which fits the motivational discussion, is saying that (S, X) is (typically) feasible if there exists an algorithm A such that typically A solves S on X in polynomial-time; that is, there exists a polynomial p such that the probability that on input X_n algorithm A either errs or runs more that $p(n)$ steps is negligible. This formulation totally ignores the untypical instances. Indeed, in this case we may assume, without loss of generality, that A always runs in polynomial-time (see Exercise 12), but we shall not do so here (in order to facilitate viewing the first option as a special case of the current option).

We note that both alternatives actually define typical feasibility and not average-case feasibility. To illustrate the difference between the two options, consider the distributional problem of deciding whether a uniformly selected (n-vertex) graph contains a Hamiltonian path. Intuitively, this problem is "typically trivial" because the algorithm may always say `yes` and be wrong with exponentially vanishing probability.[3] Indeed, this trivial algorithm is admissible by the second approach, but not by the first approach. In light of the foregoing, we adopt the second approach.

Definition 1 (the class tpc$\mathcal{P}$): *We say that A* typically solves $(S, \{X_n\}_{n\in\mathbb{N}})$ in polynomial-time *if there exists a polynomial p such that the probability that on input X_n algorithm A either errs or runs more that $p(n)$ steps is negligible.*[4] *We denote by* tpc$\mathcal{P}$ *the class of distributional problems that are typically solvable in polynomial-time.*

Clearly, for every $S \in \mathcal{P}$ and every probability ensemble X, it holds that $(S, X) \in$ tpc$\mathcal{P}$. However, tpc$\mathcal{P}$ contains also distributional problems (S, X) with $S \notin \mathcal{P}$ (see Exercises 13 and 14). The big question motivating the study of average-case complexity is whether natural distributional versions of $\mathcal{NP}$ are in tpc$\mathcal{P}$. Thus, we turn to identify such versions.

Identifying the class of interesting problems. Seeking to identify reasonable distributional versions of $\mathcal{NP}$, we note that two extreme choices should be avoided. On one hand, we must limit the class of admissible distributions so to prevent the collapse of average-case complexity to worst-case complexity (by a selection of a pathological distribution that resides on the "worst case" instances). On the other hand, we should allow for various types of natural distributions rather than confining attention merely to the uniform distribution. Recall that our aim is to address all possible input distributions that may occur in applications, and there is no justification to confining attention to the uniform distribution. Still, arguably, the distributions occuring in applications are "relatively simple" and so we seek to identify a class of simple distributions. One such notion (of simple distributions) underlies the following definition, while a more liberal notion will be presented in Section 2.2.

Definition 2 (the class dist$\mathcal{NP}$): *We say that a probability ensemble $X = \{X_n\}_{n\in\mathbb{N}}$ is* simple *if there exists a polynomial time algorithm that, on any input $x \in \{0,1\}^*$, outputs $\Pr[X_{|x|} \leq x]$, where the inequality refers to the standard lexicographic order of strings. We denote by* dist$\mathcal{NP}$ *the class of distributional problems consisting of decision problems in $\mathcal{NP}$ coupled with simple probability ensembles.*

[3]In contrast, testing whether a given graph contains a Hamiltonian path seems "typically hard" for other distributions (see Exercise 23). Needless to say, in the latter distributions both yes-instances and no-instances appear with noticeable probability.

[4]Recall that a function $\mu : \mathbb{N} \to \mathbb{N}$ is negligible if for every positive polynomial q and all sufficiently large n it holds that $\mu(n) < 1/q(n)$. We say that A errs on x if $A(x)$ differs from the indicator value of the predicate $x \in S$.

Note that the uniform probability ensemble is simple, but so are many other "simple" probability ensembles. Actually, it makes sense to relax the definition such that the algorithm is only required to output an approximation of $\mathsf{Pr}[X_{|x|} \leq x]$, say, to within a factor of $1 \pm 2^{-|x|}$. We note that although Definition 2 does not refer explicitly to any structural property of the probability ensemble, it imposes a computational restriction on the admissible ensembles (which, in turn, limits their "complexity"). In Section 2.2 we shall consider the more intuitive and robust class of all polynomial-time sampleable ensembles (and show that it contains all simple ensembles). We believe that the results presented in Section 1.2 and Section 2.2 retrospectively justify the choice underlying Definition 2. We articulate this point next.

We note that a wider class of distributions weakens the conjecture that some distributional version of NP are not feasible. On the other hand, the conclusion that some distributional problem is not feasible becomes stronger when restricting the admissible class of distributions. The results presented in Section 1.2 and Section 2.2 assert that a conjecture that refers to polynomial-time sampleable ensembles implies a conclusion that refers to a (very) simple probability ensemble. The current setting in which both the conjecture and the conclusion refer to simple probability ensembles is thus but an intermediate step.

Indeed, the big question in the current context is whether dist$\mathcal{NP}$ is contained in tpc$\mathcal{P}$. A positive answer (especially if extended to sampleable ensembles) would deem the P-vs-NP Question of little practical significant. However, our daily experience as well as much research effort indicate that some NP problems are not merely hard in the worst-case, but rather "typically hard". This supports the *conjecture that* dist$\mathcal{NP}$ *is not contained in* tpc$\mathcal{P}$.

Needless to say, the latter conjecture implies $\mathcal{P} \neq \mathcal{NP}$, and thus we should not expect to see a proof of it. What we may hope to see is "dist$\mathcal{NP}$-complete" problems; that is, problems in dist$\mathcal{NP}$ that are not in tpc$\mathcal{P}$ unless the entire class dist$\mathcal{NP}$ is contained in tpc$\mathcal{P}$. An adequate notion of a reduction is used towards formulating this notion.

Reductions among (distributional) problems. Intuitively, such reductions must preserve average-case feasibility. Thus, in addition to the standard conditions (i.e., that the reduction be efficiently computable and yield a correct result), we require that the reduction "respects" the probability distribution of the corresponding distributional problems. Specifically, the reduction should not map very likely instances of the first ("starting") problem to rare instances of the second ("target") problem. Otherwise, having a typically polynomial-time algorithm for the second distributional problem does not necessarily yield such an algorithm for the first distributional problem. Following is the adequate analogue of a Cook reduction (i.e., general polynomial-time reduction), where the analogue of a Karp-reduction (many-to-one reduction) can be easily derived as a special case.

Teaching note: One may prefer presenting in class only the special case of many-to-one reductions, which suffices for Theorem 4. See Footnote 6.

Definition 3 (reductions among distributional problems): *We say that the oracle machine M* reduces *the distributional problem (S, X) to the distributional problem (T, Y) if the following three conditions hold.*

1. Efficiency: *The machine M runs in polynomial-time.*[5]

[5]In fact, one may relax the requirement and only require that M is typically polynomial-time with respect to X. The validity condition may also be relaxed similarly.

2. Validity: *For every $x \in \{0,1\}^*$, it holds that $M^T(x) = 1$ if an only if $x \in S$, where $M^T(x)$ denotes the output of the oracle machine M on input x and access to an oracle for T.*

3. Domination:[6] *There exists a polynomial p such that, for every $y \in \{0,1\}^*$, it holds that*

$$\max_{n\in\mathbb{N}}\{\mathsf{Pr}[Q(X_n) \ni y]\} \leq p(|y|) \cdot \mathsf{Pr}[Y_{|y|} = y], \tag{1}$$

where $Q(x)$ denotes the set of queries made by M on input x and oracle access to T. Furthermore, if $y \in Q(x)$ then $|x| \leq p(|y|)$.

The l.h.s. of Eq. (1) refers to the probability that, on input distributed as X_n, the reduction makes the query y. This probability is required not to exceed the probability that y occurs in the distribution $Y_{|y|}$ by more than a polynomial factor in $|y|$. In this case we say that the l.h.s. of Eq. (1) is dominated by $\mathsf{Pr}[Y_{|y|} = y]$.

Indeed, the domination condition is the only aspect of Definition 3 that extends beyond the worst-case treatment of reductions and refers to the distributional setting. The domination condition does not insist that the distribution induced by $Q(X)$ equals Y, but rather allows some slackness that, in turn, is bounded so to guarantee preservation of typical feasibility (see Exercise 15).

We note that the reducibility arguments extensively used in the context of hardness amplification and Cryptography (see [4]) are actually reductions in the spirit of Definition 3 (except that they refer to a different type of computational tasks).

1.2 Complete problems

Recall that our conjecture is that dist$\mathcal{NP}$ is not contained in tpc$\mathcal{P}$, which in turn strengthens the conjecture $\mathcal{P} \neq \mathcal{NP}$ (making infeasibility a typical phenomenon rather than a worst-case one). Having no hope of proving that dist$\mathcal{NP}$ is not contained in tpc$\mathcal{P}$, we turn to the study of complete problems with respect to that conjecture. Specifically, we say that a distributional problem (S, X) is dist$\mathcal{NP}$-complete if $(S, X) \in$ dist$\mathcal{NP}$ and every $(S', X') \in$ dist$\mathcal{NP}$ is reducible to (S, X) (under Definition 3).

Recall that it is quite easy to prove the mere existence of NP-complete problems and many natural problems are NP-complete. In contrast, in the current context, establishing completeness results is quite hard. This should not be surprising in light of the restricted type of reductions allowed in the current context. The restriction (captured by the domination condition) requires that "typical" instances of one problem should not be mapped to "untypical" instances of the other problem. On the other hand, it is fair to say that standard Karp-reductions (used in establishing NP-completeness results) map "typical" instances of one problem to quite "bizarre" instances of the second problem. Thus, the current section may be viewed as a study of reductions that do not commit this sin.

Theorem 4 (dist$\mathcal{NP}$-completeness): dist$\mathcal{NP}$ *contains a distributional problem (S, X) such that each distributional problem in* dist$\mathcal{NP}$ *is reducible* (per Definition 3) *to (S, X). Furthermore, the reduction is deterministic and many-to-one.*

[6] For simplicity, we use the same polynomial for both bounds. Let us spell out the meaning of Eq. (1) in the special case of many-to-one reductions (i.e., $M^T(x) = 1$ if and only if $f(x) \in T$, where f is a polynomial-time computable function): in this case $\mathsf{Pr}[Q(X_n) \ni y]$ is replaced by $\mathsf{Pr}[f(X_n) = y]$. Assuming that f is one-to-one, Eq. (1) simplifies to $\mathsf{Pr}[X_{|f^{-1}(y)|} = f^{-1}(y)] \leq p(|y|) \cdot \mathsf{Pr}[Y_{|y|} = y]$ for any y in the image of f. Indeed, nothing is required for y not in the image of f.

Proof: We start by introducing such a problem, which is a natural distributional version of the decision problem $S_{\mathtt{u}}$ (used in the proof of the existence of NP-complete problems; see [5, Text 14]). Recall that $S_{\mathtt{u}}$ contains the instance $\langle M, x, 1^t\rangle$ if there exists $y \in \cup_{i \le t}\{0,1\}^i$ such that M accepts the input pair (x, y) within t steps. We couple $S_{\mathtt{u}}$ with the "quasi-uniform" probability ensemble U' that assigns to the instance $\langle M, x, 1^t\rangle$ a probability mass proportional to $2^{-(|M|+|x|)}$. Specifically, for $|\langle M, x, 1^t\rangle| = n$ it holds that $\Pr[U'_n = \langle M, x, 1^t\rangle] = 2^{-(|M|+|x|)}/\binom{n}{2}$. Note that, under a suitable encoding, the ensemble U' is indeed simple.[7]

The reader can easily verify that the generic reduction used when reducing any set in $\mathcal{NP}$ to $S_{\mathtt{u}}$ (see the proof of the existence of NP-complete problems), fails to reduce dist$\mathcal{NP}$ to $(S_{\mathtt{u}}, U')$. Specifically, in some cases (see next paragraph), these reductions do not satisfy the domination condition. Indeed, the difficulty is that we have to reduce all dist$\mathcal{NP}$ problems (i.e., pairs consisting of decision problems and simple distributions) to one single distributional problem (i.e., $(S_{\mathtt{u}}, U')$). Applying the aforementioned reductions, we end up with many distributional versions of $S_{\mathtt{u}}$, and furthermore the corresponding distributions are very different (and are not necessarily dominated by a single distribution).

Let us take a closer look at the aforementioned generic reduction, when applied to an arbitrary $(S, X) \in$ dist$\mathcal{NP}$. This reduction maps an instance x to a triple $(M_S, x, 1^{p_S(|x|)})$, where M_S is a machine verifying membership in S (while using adequate NP-witnesses) and p_S is an adequate polynomial. The problem is that x may have relatively large probability mass (i.e., it may be that $\Pr[X_{|x|} = x] \gg 2^{-|x|}$) while $(M_S, x, 1^{p_S(|x|)})$ has "uniform" probability mass (i.e., $\langle M_S, x, 1^{p_S(|x|)}\rangle$ has probability mass smaller than $2^{-|x|}$). This violates the domination condition (see Exercise 18), and thus an alternative reduction is required.

The key to the alternative reduction is an (efficiently computable) encoding of strings taken from an arbitrary *simple* distribution by strings that have a similar probability mass under the uniform distribution. This means that the encoding should shrink strings that have relatively large probability mass under the original distribution. Specifically, this encoding will map x (taken from the ensemble $\{X_n\}_{n\in\mathbb{N}}$) to a codeword x' of length that is upper-bounded by the logarithm of $1/\Pr[X_{|x|} = x]$, ensuring that $2^{-|x'|} \ge \Pr[X_{|x|} = x]$. Accordingly, the reduction will map x to a triple $(M_{S,X}, x', 1^{p'(|x|)})$, where $|x'| < O(1) + \log_2(1/\Pr[X_{|x|} = x])$ and $M_{S,X}$ is an algorithm that first verifies that x' is a proper encoding of x and next applies the standard verification (i.e., M_S) of the problem S. Such a reduction will be shown to satisfy all three conditions (i.e., efficiency, validity, and domination). Thus, instead of forcing the structure of the original distribution X on the target distribution U', the reduction will incorporate the structure of X in the reduced instance. A key ingredient in making this possible is the fact that X is simple (as per Definition 2).

With the foregoing motivation in mind, we now turn to the actual proof; that is, proving that any (S, X) is reducible to $(S_{\mathtt{u}}, U')$. The following technical lemma is the basis of the reduction. In this lemma as well as in the sequel, it will be convenient to consider the (accumulative) distribution function of the probability ensemble X. That is, we consider $\mu(x) \stackrel{\text{def}}{=} \Pr[X_{|x|} \le x]$, and note that $\mu : \{0,1\}^* \to [0,1]$ is polynomial-time computable (because X satisfies Definition 2).

Coding Lemma:[8] Let μ be a polynomial-time computable distribution function. Then there exist an encoding function C_μ satisfying the following three conditions.

[7]For example, we may encode $\langle M, x, 1^t\rangle$, where $M = \sigma_1 \cdots \sigma_k \in \{0,1\}^k$ and $x = \tau_1 \cdots \tau_\ell \in \{0,1\}^\ell$, by the string $\sigma_1\sigma_1 \cdots \sigma_k\sigma_k 01 \tau_1\tau_1 \cdots \tau_\ell\tau_\ell 01^t$.

[8]The lemma actually refers to $\{0,1\}^n$, for a fixed value of n, but the efficiency condition is stated more easily when allowing n to vary (and using the standard asymptotic analysis of algorithms). Furthermore, the lemma holds for any monotonically non-decreasing function that is efficiently computable, and its proof is less cumbersome when stated for functions defined over $\{0,1\}^*$. See further discussion in Exercise 19.

1. Compression: For every x it holds that $|C_\mu(x)| \leq 1 + \min\{|x|, \log_2(1/\mu'(x))\}$, where $\mu'(x) \overset{\text{def}}{=} \Pr[X_{|x|} = x]$.

2. Efficient Encoding: The function C_μ is computable in polynomial-time.

3. Unique Decoding: For every $n \in \mathbb{N}$, when restricted to $\{0,1\}^n$, the function C_μ is one-to-one (i.e., if $C_\mu(x) = C_\mu(x')$ and $|x| = |x'|$ then $x = x'$).

Proof: The function C_μ is defined as follows. If $\mu'(x) \leq 2^{-|x|}$ then $C_\mu(x) = 0x$ (i.e., in this case x serves as its own encoding). Otherwise (i.e., $\mu'(x) > 2^{-|x|}$) then $C_\mu(x) = 1z$, where z is chosen such that $|z| \leq \log_2(1/\mu'(x))$ and the mapping of n-bit strings to their encoding is one-to-one. Loosely speaking, z is selected to equal the shortest binary expansion of a number in the interval $(\mu(x) - \mu'(x), \mu(x)]$. Bearing in mind that this interval has length $\mu'(x)$ and that the different intervals are disjoint, we obtain the desired encoding. Details follows.

We focus on the case that $\mu'(x) > 2^{-|x|}$, and detail the way that z is selected (for the encoding $C_\mu(x) = 1z$). If $x > 0^{|x|}$ and $\mu(x) < 1$, then we let z be the longest common prefix of the binary expansions of $\mu(x-1)$ and $\mu(x)$, where $x - 1$ is the string preceding x in lexicographic order (e.g., if $\mu(1010) = 0.10010$ and $\mu(1011) = 0.10101111$ then $C_\mu(1011) = 1z$ with $z = 10$). Thus, in this case $0.z1$ is in the interval $(\mu(x-1), \mu(x)]$ (i.e., $\mu(x-1) < 0.z1 \leq \mu(x)$). For $x = 0^{|x|}$, we let z be the longest common prefix of the binary expansions of 0 and $\mu(x)$ and again $0.z1$ is in the relevant interval (i.e., $(0, \mu(x)]$). Finally, for x such that $\mu(x) = 1$ and $\mu(x-1) < 1$, we let z be the longest common prefix of the binary expansions of $\mu(x-1)$ and $1 - 2^{-|x|-1} < \mu(x)$ (and again $0.z1$ is in $(\mu(x-1), \mu(x)]$). Note that if $\mu(x) = \mu(x-1) = 1$ then $\mu'(x) = 0 < 2^{-|x|}$.

We now verify that the foregoing C_μ satisfies the conditions of the lemma. We start with the compression condition. Clearly, if $\mu'(x) \leq 2^{-|x|}$ then $|C_\mu(x)| = 1 + |x| \leq 1 + \log_2(1/\mu'(x))$. On the other hand, suppose that $\mu'(x) > 2^{-|x|}$ and let us focus on the sub-case that $x > 0^{|x|}$ and $\mu(x) < 1$. Let $z = z_1 \cdots z_\ell$ be the longest common prefix of the binary expansions of $\mu(x-1)$ and $\mu(x)$. Then, $\mu(x-1) = 0.z0u$ and $\mu(x) = 0.z1v$, where $u, v \in \{0,1\}^*$, and it follows that

$$\mu'(x) \;=\; \mu(x) - \mu(x-1) \;\leq\; \left(\sum_{i=1}^{\ell} 2^{-i} z_i + \sum_{i=\ell+1}^{\mathrm{poly}(|x|)} 2^{-i}\right) - \sum_{i=1}^{\ell} 2^{-i} z_i \;<\; 2^{-|z|}.$$

Thus, $|z| < \log_2(1/\mu'(x)) \leq |x|$ and it follows that $|C_\mu(x)| \leq 1 + \min(|x|, \log_2(1/\mu'(x)))$ holds in both cases. Clearly, C_μ can be computed in polynomial-time by computing $\mu(x-1)$ and $\mu(x)$. Finally, note that C_μ satisfies the unique decoding condition, by separately considering the two aforementioned cases (i.e., $C_\mu(x) = 0x$ and $C_\mu(x) = 1z$). Specifically, in the second case (i.e., $C_\mu(x) = 1z$), use the fact that $\mu(x-1) < 0.z1 \leq \mu(x)$. $\square$

To obtain an encoding that is one-to-one when applied to strings of different lengths we augment C_μ in the obvious manner; that is, we consider $C'_\mu(x) \overset{\text{def}}{=} (|x|, C_\mu(x))$, which may be implemented as $C'_\mu(x) = \sigma_1\sigma_1 \cdots \sigma_\ell\sigma_\ell 01 C_\mu(x)$ where $\sigma_1 \cdots \sigma_\ell$ is the binary expansion of $|x|$. Note that $|C'_\mu(x)| = O(\log|x|) + |C_\mu(x)|$ and that C'_μ is one-to-one.

The machine associated with (S, X). Let μ be the accumulative probability function associated with the probability ensemble X and M_S be the polynomial-time machine that verifies membership in S while using adequate NP-witnesses (i.e., $x \in S$ if and only if there exists $y \in \{0,1\}^{\mathrm{poly}(|x|)}$ such that $M(x,y) = 1$). Using the encoding function C'_μ, we introduce an algorithm $M_{S,\mu}$ with the intension of reducing the distributional problem (S, X) to $(S_{\mathtt{u}}, U')$ such that all instances (of S) are mapped to triples in which the first element equals $M_{S,\mu}$. Machine $M_{S,\mu}$ is given an alleged encoding (under C'_μ) of an instance to S along with an alleged proof that the corresponding instance is in S, and verifies these claims in the obvious manner. That is, on input x' and $\langle x,y\rangle$, machine $M_{S,\mu}$ first verifies that $x' = C'_\mu(x)$, and next verifiers that $x \in S$ by running $M_S(x,y)$. Thus, $M_{S,\mu}$ verifies membership in the set $S' = \{C'_\mu(x) : x \in S\}$, while using proofs of the form $\langle x,y\rangle$ such that $M_S(x,y) = 1$ (for the instance $C'_\mu(x)$).[9]

The reduction. We maps an instance x (of S) to the triple $(M_{S,\mu}, C'_\mu(x), 1^{p(|x|)})$, where $p(n) \stackrel{\text{def}}{=} p_S(n) + p_C(n)$ such that p_S is a polynomial representing the running-time of M_S and p_C is a polynomial representing the running-time of the encoding algorithm. That is, on input (x,y), algorithm M_S makes at most $p_S(|x|)$ steps (and rejects (x,y) if $|y| > p(|x|) - |x|$).

Analyzing the reduction. Our goal is proving that *the foregoing mapping constitutes a reduction of* (S,X) *to* $(S_{\mathtt{u}}, U')$. We verify the corresponding three requirements (of Definition 3).

1. Using the fact that C_μ is polynomial-time computable (and noting that p is a polynomial), it follows that the foregoing mapping can be computed in polynomial-time.

2. Recall that, on input $(x', \langle x,y\rangle)$, machine $M_{D,\mu}$ accepts if and only if $x' = C'_\mu(x)$ and M_D accepts (x,y) within $p_S(|x|)$ steps. It follows that $x \in S$ if and only if there exists a string y of length at most $p(|x|)$ such that $M_{S,\mu}$ accepts $(C'_\mu(x), \langle x,y\rangle)$ in at most $p(|x|)$ steps. Thus, $x \in S$ if and only if $(M_{S,\mu}, C'_\mu(x), 1^{p(|x|)}) \in S_{\mathtt{u}}$, and the validity condition follows.

3. In order to verify the domination condition, we first note that the foregoing mapping is one-to-one (because the transformation $x \to C'_\mu(x)$ is one-to-one). Next, we note that it suffices to consider instances of $S_{\mathtt{u}}$ that have a preimage under the foregoing mapping (since instances with no preimage trivially satisfy the domination condition). Each of these instances (i.e., each image of this mapping) is a triple with the first element equal to $M_{S,\mu}$ and the second element being an encoding under C'_μ. By the definition of U', for every such image $\langle M_{S,\mu}, C'_\mu(x), 1^{p(|x|)}\rangle \in \{0,1\}^n$, it holds that

$$\begin{aligned} \mathsf{Pr}[U'_n = \langle M_{S,\mu}, C'_\mu(x), 1^{p(|x|)}\rangle] &= \binom{n}{2}^{-1} \cdot 2^{-(|M_{S,\mu}|+|C'_\mu(x)|)} \\ &> c \cdot n^{-2} \cdot 2^{-(|C_\mu(x)|+O(\log|x|))}, \end{aligned}$$

where $c = 2^{-|M_{S,\mu}|-1}$ is a constant depending only on S and μ (i.e., on the distributional problem (S,X)). Thus, for some positive polynomial p', we have

$$\mathsf{Pr}[U'_n = \langle M_{S,\mu}, C'_\mu(x), 1^{p(|x|)}\rangle] > p'(n)^{-1} \cdot 2^{-|C_\mu(x)|}. \qquad (2)$$

By virtue of the compression condition (of the Coding Lemma), we have $2^{-|C_\mu(x)|} \geq 2^{-1-\min(|x|,\log_2(1/\mu'(x)))}$. It follows that

$$2^{-|C_\mu(x)|} \geq \mathsf{Pr}[X_{|x|} = x]/2. \qquad (3)$$

Recalling that x is the only preimage that is mapped to $\langle M_{S,\mu}, C'_\mu(x), 1^{p(|x|)}\rangle$ and combining Eq. (2) & (3), we establish the domination condition.

The theorem follows. ■

[9]Note that $|y| = \text{poly}(|x|)$, but $|x| = \text{poly}(|C'_\mu(x)|)$ does not necessarily hold (and so S' is not necessarily in $\mathcal{NP}$). As we shall see, the latter point is immaterial.

Reflections. The proof of Theorem 4 demonstrates the fact that, unlike more advanced worst-case reductions, the generic reduction used in proving the existence of NP-complete problems does not introduce much structure in the reduced instances (i.e., does not reduce the original problem to a "highly structured special case" of the target problem). Put in other words, the latter reduction does not map "random" (i.e., uniformly distributed) instances to highly structured instances (which occur with negligible probability under the uniform distribution). Thus, this reduction suffices for reducing any distributional problem in dist$\mathcal{NP}$ to a distributional problem consisting of $S_{\tt u}$ coupled with *some* simple probability ensemble.[10]

However, Theorem 4 states more than the latter assertion. That is, it states that any distributional problem in dist$\mathcal{NP}$ is reducible to the same distributional version of $S_{\tt u}$. Indeed, the effort involved in proving Theorem 4 was due to the need for mapping instances taken from any simple probability ensemble (which may not be the uniform ensemble) to instances distributed in a manner that is dominated by a single probability ensemble (i.e., the quasi-uniform ensemble U').

Once we have established the existence of one dist$\mathcal{NP}$-complete problem, we may establish the dist$\mathcal{NP}$-completeness of other problems in dist$\mathcal{NP}$ by reducing any dist$\mathcal{NP}$-complete problem to them (and relying on the transitivity of reductions (see Exercise 17)). Thus, the difficulties encountered in the proof of Theorem 4 are no longer relevant. Unfortunately, a seemingly more severe difficulty arises: almost all know reductions in the theory of NP-completeness work by introducing much structure in the reduced instances (i.e., they actually reduce to highly structured special cases). Furthermore, this structure is too complex in the sense that the distribution of reduced instances does not seem simple (in the sense of Definition 2). Designing reductions that avoid the introduction of such structure has turned out to be quite difficult; still several such reductions are cited in [3].

1.3 Probabilistic versions

The definitions in Section 1.1 can be easily extended to refer to randomized algorithms. For example, extending Definition 1, we have:

Definition 5 (the class tpc$\mathcal{BPP}$): *For a probabilistic algorithm A, a Boolean function B, and $t : \mathbb{N} \to \mathbb{N}$, we say that the string x is* t-bad for A with respect to B *if with probability exceeding $1/3$, on input x, either $A(x) \neq B(x)$ or A runs more that $t(|x|)$ steps. We say that A* typically solves $(S, \{X_n\}_{n\in\mathbb{N}})$ in probabilistic polynomial-time *if there exists a polynomial p such that the probability that X_n is p-bad for A with respect to the characteristic function of S is negligible. We denote by* tpc$\mathcal{BPP}$ *the class of distributional problems that are typically solvable in probabilistic polynomial-time.*

The definition of reductions can be similarly extended. This means that in Definition 3, both $M^T(x)$ and $Q(x)$ (mentioned in Items 2 and 3, respectively) are random variables rather than fixed objects. Furthermore, validity is required to hold (for every input) only with probability $2/3$, where the probability space refers only to the internal coin tosses of the reduction. Randomized reductions are closed under composition and preserve typical feasibility (see Exercise 20).

Randomized reductions allow the presentation of a dist$\mathcal{NP}$-complete problem that refers to the (perfectly) uniform ensemble. Recall that Theorem 4 establishes the dist$\mathcal{NP}$-completeness of $(S_{\tt u}, U')$, where U' is a quasi-uniform ensemble (i.e., $\mathsf{Pr}[U'_n = \langle M, x, 1^t\rangle] = 2^{-(|M|+|x|)}/\binom{n}{2}$, where $n = |\langle M, x, 1^t\rangle|$). We first note that $(S_{\tt u}, U')$ can be randomly reduced to $(S'_{\tt u}, U'')$, where $S'_{\tt u} =$

[10]Note that this cannot be said of most known Karp-reductions.

$\{\langle M, x, z\rangle : \langle M, x, 1^{|z|}\rangle \in S_{\mathtt{u}}\}$ and $\mathsf{Pr}[U''_n = \langle M, x, z\rangle] = 2^{-(|M|+|x|+|z|)}/\binom{n}{2}$ for every $\langle M, x, z\rangle \in \{0,1\}^n$. The randomized reduction consists of mapping $\langle M, x, 1^t\rangle$ to $\langle M, x, z\rangle$, where z is uniformly selected in $\{0,1\}^t$. Recalling that $U = \{U_n\}_{n\in\mathbb{N}}$ denotes the uniform probability ensemble (i.e., U_n is uniformly distributed on strings of length n) and using a suitable encoding we get.

Proposition 6 *There exists $S \in \mathcal{NP}$ such that every $(S', X') \in \text{dist}\mathcal{NP}$ is randomly reducible to (S, U).*

Proof Sketch: By the forgoing discussion, every $(S', X') \in \text{dist}\mathcal{NP}$ is randomly reducible to $(S'_{\mathtt{u}}, U'')$. Thus, we focus on reducing $(S'_{\mathtt{u}}, U'')$ to $(S''_{\mathtt{u}}, U)$, where $S''_{\mathtt{u}} \in \mathcal{NP}$ is defined as follows. The string $\alpha\beta uvw$ is in $S''_{\mathtt{u}}$ if $\langle u, v, w\rangle \in S'_{\mathtt{u}}$ and α (resp., β) represents the binary encoding of the integer $|u|$ (resp., $|v|$), where the encoding is padded with zeros to a total length of $\log_2|uvw|$. The reduction maps $\langle M, x, z\rangle$ to the string $\alpha\cdot\beta\cdot M\cdot x\cdot z$, where α (resp., β) represents the binary encoding of $|M|$ (resp., $|x|$) padded with zeros to a total length of $\log_2(|M|+|x|+|z|)$. Noting that this reduction satisfies all conditions of Definition 3, the proposition follows. $\square$

2 Ramifications

In our opinion, the most problematic aspect of the theory described in Section 1 is the definition of simple probability ensembles, which in turn restricts the definition of dist$\mathcal{NP}$ (Definition 2). This restriction strengthens the conjecture that dist$\mathcal{NP}$ is not contained in tpc$\mathcal{BPP}$, which means that it weakens conditional results that are based on this conjecture. An appealing extension of the class dist$\mathcal{NP}$ is presented in Section 2.2, where it is shown that if the extended class is not contained in tpc$\mathcal{BPP}$ then dist$\mathcal{NP}$ itself is not contained in tpc$\mathcal{BPP}$. Thus, dist$\mathcal{NP}$-complete problems enjoy the benefit of both being in the more restricted class (i.e., dist$\mathcal{NP}$) and being hard as long as some problems in the extended class is hard.

In Section 2.1, we extend the treatment from decision problems to search problems. This extension is motivated by the realization that search problem are actually of greater importance to real-life applications (cf. [5, Text 14]), and hence a theory motivated by real-life applications must address such problems, as we do next.

2.1 Search versus Decision

Indeed, as in the case of worst-case complexity, search problems are at least as important as decision problems. Thus, an average-case treatment of search problems is indeed called for. We first present distributional versions of $\mathcal{PF}$ and $\mathcal{PC}$ (cf. [5, Text 14]), following the underlying principles of the definitions of tpc$\mathcal{P}$ and dist$\mathcal{NP}$.

Definition 7 (the classes tpc$\mathcal{PF}$ and dist$\mathcal{PC}$): *As in [5, Text 14], we consider only polynomially bounded search problems; that is, binary relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$ such that for some polynomial q it holds that $(x, y) \in R$ implies $|y| \leq q(|x|)$. Recall that $R(x) \overset{\text{def}}{=} \{y : (x, y) \in R\}$.*

- *A* distributional search problem *consists of a polynomially bounded search problem coupled with a probability ensemble.*
- *The class* tpc$\mathcal{PF}$ *consists of all distributional search problems that are typically solvable in polynomial-time. That is, $(R, \{X_n\}_{n\in\mathbb{N}}) \in$ tpc$\mathcal{PF}$ if there exists an algorithm A and a polynomial p such that the probability that on input X_n algorithm A either errs or runs more that*

$p(n)$ steps is negligible, where A errs on x if $A(x) \notin R(x)$ in case $R(x) \neq \emptyset$ and $A(x) \neq \perp$ otherwise.

- *A distributional search problem (R, X) is in* dist$\mathcal{PC}$ *if $R \in \mathcal{PC}$ and X is simple* (as in Definition 2).

Likewise, the class tpc$\mathcal{BPPF}$ consists of all distributional search problems that are typically solvable in *probabilistic* polynomial-time (cf., Definition 5). The definitions of *reductions among distributional problems*, presented in the context of decision problem, extend to search problems.

Fortunately, as in the context of worst-case complexity, the study of distributional search problems "reduces" to the study of distributional decision problems.

Theorem 8 (reducing search to decision): dist$\mathcal{PC} \subseteq$ tpc$\mathcal{BPPF}$ *if and only if* dist$\mathcal{NP} \subseteq$ tpc$\mathcal{BPP}$. *Furthermore, every problem in* dist$\mathcal{NP}$ *is reducible to some problem in* dist$\mathcal{PC}$*, and every problem in* dist$\mathcal{PC}$ *is* randomly *reducible to some problem in* dist$\mathcal{NP}$.

Proof Sketch: The furthermore part is analogous to the actual contents of the proof of equivalence of the search and decision versions of the P-vs-NP Question. Indeed the reduction of $\mathcal{NP}$ to $\mathcal{PC}$ presented in that proof (cf. [5, Text 14]) extends to the current context. Specifically, for any $S \in \mathcal{NP}$, we consider a relation $R \in \mathcal{PC}$ such that $S = \{x : R(x) \neq \emptyset\}$, and note that, for any probability ensemble X, the identity transformation reduces (S, X) to (R, X).

A difficulty arises in the opposite direction. Recall that in the context of worst-case complexity we reduced the search problem of $R \in \mathcal{PC}$ to deciding membership in $S'_R \stackrel{\text{def}}{=} \{\langle x, y' \rangle : \exists y'' \text{ s.t. } (x, y'y'') \in R\} \in \mathcal{NP}$. The difficulty encountered here is that, on input x, this reduction makes queries of the form $\langle x, y' \rangle$, where y' is a prefix of some string in $R(x)$. These queries may induce a distribution that is not dominated by any simple distribution. Thus, we seek an alternative reduction.

As a warm-up, let us assume for a moment that R has unique solutions; that is, for every x it holds that $|R(x)| \leq 1$. In this case we may easily reduce the search problem of $R \in \mathcal{PC}$ to deciding membership in $S''_R \in \mathcal{NP}$, where $\langle x, i, \sigma \rangle \in S''_R$ if and only if $R(x)$ contains a string in which the i^{th} bit equals σ. Specifically, on input x, the reduction issues the queries $\langle x, i, \sigma \rangle$, where $i \in [\ell]$ (with $\ell = \text{poly}(|x|)$) and $\sigma \in \{0, 1\}$, which allows for determining the single string in $R(x)$ (whenever such a string exists). The point is that this reduction can be used to reduce any $(R, X) \in$ dist$\mathcal{PC}$ (having unique solutions) to $(S''_R, X'') \in$ dist$\mathcal{NP}$, where X'' equally distributes the probability mass of x (under X) to all the tuples $\langle x, i, \sigma \rangle$; that is, for every $i \in [\ell]$ and $\sigma \in \{0, 1\}$, it holds that $\Pr[X''_{|\langle x,i,\sigma \rangle|} = \langle x, i, \sigma \rangle]$ equals $\Pr[X_{|x|} = x]/2\ell$.

Unfortunately, in the general case, R may not have unique solutions. Nevertheless, applying the main idea that underlies the proof of the NP-hardness of solving unique solution problems (see [5, Text 15]), this difficulty can be overcome. We first note that the foregoing mapping of instances of the distributional problem $(R, X) \in$ dist$\mathcal{PC}$ to instances of $(S''_R, X'') \in$ dist$\mathcal{NP}$ satisfies the efficiency and domination condition even in the case that R does not have unique solutions. What may possibly fail (in the general case) is the validity condition (i.e., if $|R(x)| > 1$ then we may fail to recover any element of $R(x)$).

Recall that the main part of the proof of the NP-hardness of solving unique solution problems is a randomized reduction that maps instances of R to triples of the form (x, m, h), where m is an integer and h is a hashing function that are uniformly distributed in some adequate sets $[\ell]$ and H^m_ℓ, where $\ell = \text{poly}(|x|)$ and H^m_ℓ is as in [5, Text 15]. Furthermore, if $R(x) \neq \emptyset$ then, with probability $\Omega(1/\ell)$ over yje choices of $m \in [\ell]$ and $h \in H^m_\ell$, there exists a unique $y \in R(x)$ such that $h(y) = 0^m$. Defining $R'(x,m,h) \stackrel{\rm def}{=} \{y \in R : h(y) = 0^m\}$, this yields a randomized reduction of the search problem of R to the search problem of R' such that with noticeable probability[11] the reduction maps instances that have solutions to instances having a unique solution. Furthermore, this reduction can be used to reduce any $(R,X) \in {\rm dist}\mathcal{PC}$ to $(R',X') \in {\rm dist}\mathcal{PC}$, where X' distributes the probability mass of x (under X) to all the triples (x,m,h) such that for every $m \in [\ell]$ and $h \in H_\ell^m$ it holds that ${\sf Pr}[X''_{|(x,m,h)|} = (x,m,h)]$ equals ${\sf Pr}[X_{|x|} = x]/(\ell \cdot |H_\ell^m|)$. (Note that with a suitable encoding, X' is indeed simple.)

The theorem follows by combining the two aforementioned reductions. That is, we first apply the randomized reduction of (R,X) to (R',X'), and next reduce the resulting instance to an instance of the corresponding decision problem $(S''_{R'}, X'')$. The combined randomized mapping satisfies the efficiency and domination conditions, and is valid with noticeable probability. The error probability can be made negligible by straightforward amplification (see Exercise 20). $\square$

2.2 Simple versus sampleable distributions

Recall that the definition of simple probability ensembles (underlying Definition 2) requires that the accumulating distribution function (as defined in the proof of Theorem 4) is polynomial-time computable. Recall that $\mu : \{0,1\}^* \to [0,1]$ is called the accumulating distribution function of $X = \{X_n\}_{n\in\mathbb{N}}$ if for every $n \in \mathbb{N}$ and $x \in \{0,1\}^n$ it holds that $\mu(x) \stackrel{\rm def}{=} {\sf Pr}[X_n \leq x]$, where the inequality refers to the standard lexicographic order of n-bit strings.

As argued in Section 1.1, the requirement that the accumulating distribution function is polynomial-time computable imposes severe restrictions on the set of admissible ensembles. Furthermore, it seems that these simple ensembles are indeed "simple" in some intuitive sense and hence represent distributions that may occur in practice. However, a more robust definition of the latter is offered by the notion of polynomial-time sampleable ensembles (underlying Definition 9). We believe that the class of such ensembles contains all distributions that may occur in practice, because we believe that the real world should be modeled as a *feasible* (rather than an arbitrary) randomized process

Definition 9 (sampleable ensembles and the class samp$\mathcal{NP}$): *We say that a probability ensemble $X = \{X_n\}_{n\in\mathbb{N}}$ is* (polynomial-time) sampleable *if there exists a probabilistic polynomial-time algorithm A such that for every $x \in \{0,1\}^*$ it holds that* ${\sf Pr}[A(1^{|x|}) = x] = {\sf Pr}[X_{|x|} = x]$. *We denote by* samp$\mathcal{NP}$ *the class of distributional problems consisting of decision problems in $\mathcal{NP}$ coupled with sampleable probability ensembles.*

We first note that all simple probability ensembles are indeed sampleable (see Exercise 21), and thus dist$\mathcal{NP} \subseteq$ samp$\mathcal{NP}$. On the other hand, it seems that there are sampleable probability ensembles that are not simple (see Exercise 22). In fact, extending the scope of distributional problems (from dist$\mathcal{NP}$ to samp$\mathcal{NP}$) allows proving that every NP-complete problem has a distributional version in samp$\mathcal{NP}$ that is dist$\mathcal{NP}$-hard (see Exercise 23). Furthermore, it is possible to prove that all natural NP-complete problem have distributional versions that are samp$\mathcal{NP}$-complete.

Theorem 10 (samp$\mathcal{NP}$-completeness): *Suppose that $S \in \mathcal{NP}$ and that every set in $\mathcal{NP}$ is reducible to S by a Karp-reduction that does not shrink the input. Then there exists a polynomial-time sampleable ensemble X such that any problem in* samp$\mathcal{NP}$ *is reducible to (S,X)*

[11] Recall that the probability of an event is said to be noticeable (in a relevant parameter) if it is greater than the reciprocal of some positive polynomial. In the context of randomized reductions, the relevant parameter is the length of the input to the reduction.

The proof of Theorem 10 is based on the observation that there exists a polynomial-time sampleable ensemble that dominates all polynomial-time sampleable ensembles. The existence of this ensemble is based on the notion of a universal (sampling) machine. For further details see Exercise 24. (Recall that when proving Theorem 4, we did not establish an analogous result for simple ensembles (but rather capitalized on the universal nature of $S_{\mathbf{u}}$).)

Theorem 10 establishes a rich theory of samp$\mathcal{NP}$-completeness, but does not relate this theory to the previously presented theory of dist$\mathcal{NP}$-completeness. This is done in the next theorem, which asserts that the existence of typically hard problems in samp$\mathcal{NP}$ implies their existence in dist$\mathcal{NP}$.

Theorem 11 (samp$\mathcal{NP}$-completeness versus dist$\mathcal{NP}$-completeness): *If* samp$\mathcal{NP}$ *is not contained in* tpc$\mathcal{BPP}$ *then* dist$\mathcal{NP}$ *is not contained in* tpc$\mathcal{BPP}$.

Thus, the two "typical-case complexity" versions of the P-vs-NP Question are equivalent. That is, if some "sampleable distribution" versions of NP are not typically feasible then some "simple distribution" versions of NP are not typically feasible. In particular, if samp$\mathcal{NP}$-complete problems are not in tpc$\mathcal{BPP}$ then dist$\mathcal{NP}$-complete problems are not in tpc$\mathcal{BPP}$.

The foregoing assertions would all follow if samp$\mathcal{NP}$ were (randomly) reducible to dist$\mathcal{NP}$ (i.e., if every problem in samp$\mathcal{NP}$ were reducible (under a randomized version of Definition 3) to some problem in dist$\mathcal{NP}$); but, unfortunately, we do not know whether such reductions exist. Yet, underlying the proof of Theorem 11 is a more liberal notion of a reduction among distributional problem.

Proof Sketch: We shall prove that if dist$\mathcal{NP}$ is contained in tpc$\mathcal{BPP}$ then the same holds for samp$\mathcal{NP}$ (i.e., samp$\mathcal{NP}$ is contained in tpc$\mathcal{BPP}$). Actually, we shall show that if dist$\mathcal{PC}$ is contained in tpc$\mathcal{BPPF}$ then the sampleable version of dist$\mathcal{PC}$, denoted samp$\mathcal{PC}$, is contained in tpc$\mathcal{BPPF}$ (and refer to Exercise 25). Specifically, we shall show that under a relaxed notion of a randomized reduction, every problem in samp$\mathcal{PC}$ is reduced to some problem in dist$\mathcal{PC}$. Loosely speaking, this relaxed notion of a randomized reduction requires only a noticeable fraction of the probability space of the reduction to satisfies the validity and domination conditions (of Definition 3, when adapted to randomized reductions). We start by formulating this notion, when referring to distributional *search* problems.

A relaxed reduction of the distributional problem (R, X) to the distributional problem (T, Y) is a probabilistic polynomial-time oracle machine M that satisfies the following conditions:

Notation: For every $x \in \{0,1\}^*$, we denote by $m(|x|) = \text{poly}(|x|)$ the number of internal coin tosses of M on input x, and denote by $M^T(x, r)$ the execution of M on input x, internal coins $r \in \{0,1\}^m$, and oracle access to T.

Validity: For every $x \in \{0,1\}^*$, there exists a set $\Omega_x \subseteq \{0,1\}^{m(|x|)}$ of size at least $\rho(|x|) \cdot 2m(|x|)$, where $\rho(|x|) > 1/\text{poly}(|x|)$ such that for every $r \in \Omega_x$ the reduction yields a correct answer (i.e., $M^T(x, r) \in R(x)$ if $R(x) \neq \emptyset$ and $M^T(x, r) = \perp$ otherwise).

Domination: There exists a positive polynomial p such that, for every $y \in \{0,1\}^*$, it holds that

$$\max_{n \in \mathbb{N}} \{\mathsf{Pr}[Q'(X_n) \ni y]\} \leq p(|y|) \cdot \mathsf{Pr}[Y_{|y|} = y], \tag{4}$$

where $Q'(x)$ is a random variable, defined over the set Ω_x (as in the validity condition), representing the set of queries made by M on input x and oracle access to T. That is, $Q'(x)$ is defined by uniformly selecting $r \in \Omega_x$ and considering the set of queries made by M on input x, internal coins r, and oracle access to T.

The reader may verify that this relaxed notion of a reduction preserves typical feasibility; that is, for $R \in \mathcal{PC}$, if there exists a relaxed reduction of (R,X) to (T,Y) and (T,Y) is in tpc$\mathcal{BPPF}$ then (R,X) is in tpc$\mathcal{BPPF}$. The key observation is that the analysis may discard the case that, on input x, the reduction selects coins not in Ω_x. Indeed, the queries made in that case may be untypical and the answers received may be wrong, but this is immaterial (because correct solutions can be recognized using $R \in \mathcal{PC}$). That is, if x has a solution then with noticeable probability the reduction will find one and output it, whereas the reduction will never output a wrong solution.

Our goal is presenting, for every $(R,X) \in$ samp$\mathcal{PC}$, a relaxed reduction of (R,X) to a related problem $(R',X') \in$ dist$\mathcal{PC}$. (As usual, let $X = \{X_n\}_{n\in\mathbb{N}}$ and similarly for X'.) For starters, *suppose that X_n is uniformly distributed on some set $S_n \subseteq \{0,1\}^n$ and that there is a polynomial-time computable and invertible mapping μ of S_n to $\{0,1\}^{\ell(n)}$, where $\ell(n) = \log_2 |S_n|$.* Then, mapping x to $1^{|x|-\ell(|x|)}0\mu(x)$, we obtain a reduction of (R,X) to (R',X'), where X'_{n+1} is uniform over $\{1^{n-\ell(n)}0v : v \in \{0,1\}^{\ell(n)}\}$ and $R'(1^{n-\ell(n)}0v) = R(\mu^{-1}(v))$ (or, equivalently, $R(x) = R'(1^{|x|-\ell(|x|)}0\mu(x))$). Note that X' is a simple ensemble and $R' \in \mathcal{PC}$; hence, $(R',X') \in$ dist$\mathcal{PC}$. Also note that the foregoing mapping is indeed a valid reduction (i.e., it satisfies the efficiency, validity, and domination conditions). Thus, (R,X) is reduced to a problem in dist$\mathcal{PC}$ (and indeed the relaxation was not used here).

Next, we drop the assumption that there is a polynomial-time computable and invertible mapping μ of S_n to $\{0,1\}^{\ell(n)}$, but maintain the assumption that X_n is uniform on some set $S_n \subseteq \{0,1\}^n$ and assume that $|S_n| = 2^{\ell(n)}$ is easily computable (from n). In this case, we may map $x \in \{0,1\}^n$ to its image under a suitable randomly chosen hashing function h, which in particular maps n-bit strings to $\ell(n)$-bit strings. That is, we randomly map x to $(h, 1^{n-\ell(n)}0h(x))$, where h is uniformly selected in a set $H_n^{\ell(n)}$ of suitable hash functions (see [5, Text 15]). This calls for redefining R' such that $R'(h, 1^{n-\ell(n)}0v)$ corresponds to the preimages of v under h that are in S_n. Assuming that h is a 1-1 mapping of S_n to $\{0,1\}^{\ell(n)}$, we may define $R'(h, 1^{n-\ell(n)}0v) = R(x)$ where x is the unique string satisfying $x \in S_n$ and $h(x) = v$, where the condition $x \in S_n$ may be *verified by providing the internal coins of the sampling procedure that generate x*. Denoting the sampling procedure of X by S, and letting $S(1^n, r)$ denote the output of S on input 1^n and internal coins r, we actually redefine R' as

$$R'(h, 1^{n-\ell(n)}0v) = \{\langle r,y\rangle : h(S(1^n,r)) = v \wedge y \in R(S(1^n,r))\}. \tag{5}$$

We note that $\langle r,y\rangle \in R'(h, 1^{|x|-\ell(|x|)}0h(x))$ yields a solution $y \in R(x)$ if $S(1^{|x|},r) = x$, but otherwise "all bets are off" (as y will be a solution for $S(1^{|x|},r) \neq x$). Now, although typically h will not be a 1-1 mapping of S_n to $\{0,1\}^{\ell(n)}$, for each $x \in S_n$, with constant probability over the choice of h, it holds that $h(x)$ has a unique preimage in S_n under h. In this case $\langle r,y\rangle \in R'(h, 1^{|x|-\ell(|x|)}0h(x))$ implies $S(1^{|x|},r) = x$ (which, in turn, implies $y \in R(x)$). We claim that *the randomized mapping of x to $(h, 1^{n-\ell(n)}0h(x))$, where h is uniformly selected in $H_{|x|}^{\ell(|x|)}$, yields a relaxed reduction of (R,X) to (R',X'), where $X'_{n'}$ is uniform over $H_n^{\ell(n)} \times \{1^{n-\ell(n)}0v : v \in \{0,1\}^{\ell(n)}\}$*. (Needless to say, the claim refers to the reduction that makes the query $(h, 1^{n-\ell(n)}0h(x))$ and returns y if the oracle answer equals $\langle r,y\rangle$ and $y \in R(x)$.)

The claim is proved by considering the set Ω_x of choices of $h \in H_{|x|}^{\ell(|x|)}$ for which $x \in S_n$ is the only preimage of $h(x)$ under h that resides in S_n (i.e., $|\{x' \in S_n : h(x') = h(x)\}| = 1$). In this case (i.e., $h \in \Omega_x$) it holds that $\langle r,y\rangle \in R'(h, 1^{|x|-\ell(|x|)}0h(x))$ implies that $S(1^{|x|},r) = x$ and $y \in R(x)$, and the (relaxed) validity condition follows. The (relaxed) domination condition follows by noting that $\mathsf{Pr}[X_n = x] \approx 2^{-\ell(|x|)}$, that x is mapped to $(h, 1^{|x|-\ell(|x|)}0h(x))$ with probability $1/|H_{|x|}^{\ell(|x|)}|$, and that x is the only preimage of $(h, 1^{|x|-\ell(|x|)}0h(x))$ under the mapping (among $x' \in S_n$ such that

$\Omega_{x'} \ni h$).

Before going any further, let us highlight the importance of hashing X_n to $\ell(n)$-bit strings. On one hand, this mapping is "typically" one-to-one, and thus (with constant probability) the solution provided for the hashed instance (i.e., $h(x)$) yield a solution for the original instance (i.e., x). This guarantees the validity of the reduction. On the other hand, for a typical h, the mapping of X_n to $h(X_n)$ covers the relevant range almost uniformly. This guarantees that the reduction satisfies the domination condition. Note that these two phenomena impose conflicting requirements that are both met at the correct value of ℓ; that is, the one-to-one condition requires $\ell(n) \geq \log_2 |S_n|$, whereas an almost uniform cover requires $\ell(n) \leq \log_2 |S_n|$. Also note that $\ell(n) = \log_2(1/\mathsf{Pr}[X_n = x])$ for every x in the support of X_n; the latter quantity will be in our focus in the general case.

Finally, we need to get rid of the assumption that X_n is *uniformly distributed* over some subset of $\{0,1\}^n$. All that we know is that there exists a probabilistic polynomial-time ("sampling") algorithm S such that $S(1^n)$ is distributed identically to X_n. In this (general) case, we map instances of (R, X) according to their probability mass such that x is mapped to an instance (of R') that consists of $(h, h(x))$ and additional information, where h is a random hash function mapping n-bit long string to strings of length $\ell_x \stackrel{\text{def}}{=} \lceil \log_2(1/\mathsf{Pr}[X_{|x|} = x]) \rceil$. Since (in the general case) there may be more than 2^{ℓ_x} strings in the support of X_n, we need to augment the reduced instance in order to ensure that it is uniquely associated with x. The basic idea is augmenting the mapping of x to $(h, h(x))$ with additional information that restricts X_n to strings that occur with probability at least $2^{-\ell_x}$.

Let $q(n)$ denote the randomness complexity of S and $S(1^n, r)$ denote the output of S on input 1^n and internal coin tosses $r \in \{0,1\}^{q(n)}$. Then, we randomly map x to $(h, h(x), h', v')$, where $h : \{0,1\}^{|x|} \to \{0,1\}^{\ell_x}$ and $h' : \{0,1\}^{q(|x|)} \to \{0,1\}^{q(|x|)-\ell_x}$ are random hash functions and $v' \in \{0,1\}^{q(|x|)-\ell_x}$ is uniformly distributed. The instance (h, v, h', v') of R' has solutions that consists of pairs $\langle r, y \rangle$ such that $h(S(1^n, r)) = v \wedge h'(r) = v'$ and $y \in R(S(1^n, r))$. As we shall see, this augmentation guarantees that, with constant probability (over the choice of h, h', v'), the solutions to the reduced instance $(h, h(x), h', v')$ correspond to the solutions to the original instance x.

The foregoing description assumes that, on input x, we can determine ℓ_x, which is an assumption that we cannot justify. Instead, we may just select ℓ uniformly in $\{0, 1, ..., q(|x|)\}$ and be correct with noticeable probability (i.e., $\mathsf{Pr}[\ell = \ell_x] = 1/(q(|x|) + 1) = 1/\text{poly}(|x|)$). Furthermore, for clarity, we make n and ℓ explicit in the reduced instance. Specifically, we randomly map $x \in \{0,1\}^n$, to $(1^n, 1^\ell, h, h(x), h', v') \in \{0,1\}^{n'}$, where $\ell \in \{0, 1, ..., q(n)\}$, $h \in H_n^\ell$, $h' \in H_{q(n)}^{q(n)-\ell}$, and $v' \in \{0,1\}^{q(n)-\ell}$ are uniformly distributed.[12] This mapping will be used to reduce (R, X) to (R', X'), where

$$R'(1^n, 1^\ell, h, v, h', v') = \{\langle r, y \rangle : h(S(1^n, r)) = v \wedge h'(r) = v' \wedge y \in R(S(1^n, r))\} \tag{6}$$

and $X'_{n'}$ assigns equal probability to each $X_{n',\ell}$ (for $\ell \in \{0, 1, ..., n\}$), and each $X_{n',\ell}$ is isomorphic to the uniform distribution over $H_n^\ell \times \{0,1\}^\ell \times H_{q(n)}^{q(n)-\ell} \times \{0,1\}^{q(n)-\ell}$. Note that indeed $(R', X') \in \text{dist}\mathcal{PC}$.

The aforementioned randomized mapping is analyzed by considering the correct choice for ℓ; that is, on input x, we focus on the choice $\ell = \ell_x$. Under this conditioning (as we shall show), *with constant probability over the choice of h, h' and v', the instance x is the only value x'* (of X_n) *that is mapped to $(1^n, 1^\ell, h, h(x), h', v')$ such that there exists r that satisfies $S(1^n, r) = x'$*

[12] As in other places, a suitable encoding will be used such that the reduction maps strings of the same length to strings of the same length (i.e., n-bit string are mapped to n'-bit strings, for $n' = \text{poly}(n)$). For example, we may encode $\langle 1^n, 1^\ell, h, h(x), h', v' \rangle$ as $1^n 0 1^\ell 0 1^{q(n)-\ell} 0 \langle h \rangle \langle h(x) \rangle \langle h' \rangle \langle v' \rangle$, where each $\langle w \rangle$ denotes an encoding of w by a string of length $(n' - (n + q(n) + 3))/4$.

and $h'(r) = v'$. It follows that (for such h, h' and v') any solution $\langle r, y\rangle \in R'(1^n, 1^\ell, h, h(x), h', v')$ satisfies $S(1^n, r) = x$ and thus $y \in R(x)$, which means that the validity condition is satisfied. The domination condition is satisfied too, because (for such h, h' and v') the probability that X_n is mapped to $(1^n, 1^\ell, h, h(x), h', v')$ approximately equals $\mathsf{Pr}[X'_{n',\ell} = (1^n, 1^\ell, h, h(x), h', v')]$.

We now turn to analyze the probability, over the choice of h, h' and v', that the instance x is the only value x' (of X_n) that is mapped to $(1^n, 1^{\ell_x}, h, h(x), h', v')$ such that there exists r that satisfies $S(1^n, r) = x'$ and $h'(r) = v'$. Firstly, we note that $|\{r : S(1^n, r) = x\}| \geq 2^{q(n)-\ell_x}$, and thus, with constant probability over the choice of $h' \in H_{q(n)}^{q(n)-\ell_x}$ and $v' \in \{0,1\}^{q(n)-\ell_x}$, there exists r that satisfies $S(1^n, r) = x$ and $h'(r) = v'$. Next, we note that, with constant probability over the choice of $h \in H_n^{\ell_x}$, it holds that x is the only string having probability mass at least $2^{-\ell_x}$ (under X_n) that is mapped to $h(x)$ under h. Finally, we prove that, with constant probability over the choice of $h \in H_n^{\ell_x}$ and $h' \in H_{q(n)}^{q(n)-\ell_x}$ (and even when conditioning on the previous items), the mapping $r \mapsto (h(S(1^n, r)), h'(r))$ maps the set $\{r : \mathsf{Pr}[X_n = S(1^n, r)] \leq 2^{-\ell_x}\}$ almost uniformly to $\{0,1\}^{q(n)}$. Specifically, with constant probability, no other r is mapped to the aforementioned pair $(h(x), v')$. Thus, the claim follows and so does the theorem. $\square$

Reflection. Theorem 11 implies that if samp$\mathcal{NP}$ is not contained in tpc$\mathcal{BPP}$ then every dist$\mathcal{NP}$-complete problem is not in tpc$\mathcal{BPP}$. This means that the hardness of some distributional problems that refer to sampleable distributions implies the hardness of some distributional problems that refer to simple distributions. Furthermore, by Proposition 6, this implies the hardness of distributional problems that refer to the uniform distribution. Thus, hardness with respect to some distribution in an utmost wide class (which arguably captures all distributions that may occur in practice) implies hardness with respect to a single simple distribution (which arguably is the simplest one).

Relation to one-way functions. We note that the existence of one-way functions (see [4]) implies the existence of problems in samp$\mathcal{PC}$ that are not in tpc$\mathcal{BPPF}$ (which in turn implies the existence of such problems in dist$\mathcal{PC}$). Specifically, for a length-preserving one-way function f, consider the distributional search problem $(R_f, \{f(U_n)\}_{n\in\mathbb{N}})$, where $R_f = \{(f(r), r) : r \in \{0,1\}^*\}$.[13] On the other hand, it is not known whether the existence of a problem in samp$\mathcal{PC} \setminus$ tpc$\mathcal{BPPF}$ implies the existence of one-way functions. In particular, the existence of a problem (R, X) in samp$\mathcal{PC} \setminus$ tpc$\mathcal{BPPF}$ represents the feasibility of generating hard instances for the search problem R, whereas the existence of one-way function represents the feasibility of generating instance-solution pairs such that the instances are hard to solve. Indeed, the gap refers to whether or not *hard instances can be efficiently generated together with corresponding solutions.* Our world view is thus depicted in Figure 1, where lower levels indicate seemingly weaker assumptions.

Notes

The theory of average-case complexity was initiated by Levin [8], who in particular proved Theorem 4. In light of the laconic nature of the original text [8], we refer the interested reader to a survey [3], which provides a more detailed exposition of the definitions suggested by Levin as well as a discussion of the considerations underlying these suggestions. (This survey [3] provides also a brief account of further developments.)

[13]Note that the distribution $f(U_n)$ is uniform in the special case that f is a permutation over $\{0,1\}^n$.

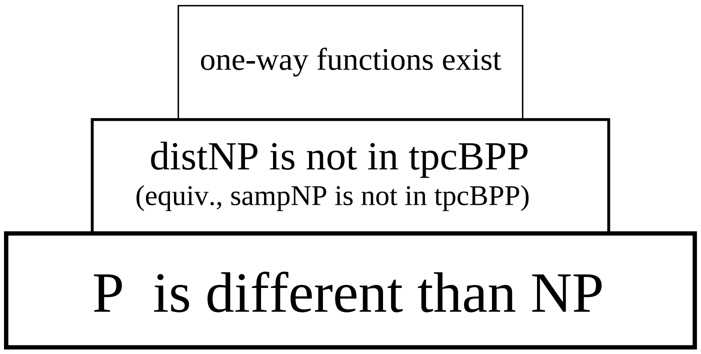

Figure 1: Worst-case vs average-case assumptions

As noted in Section 1.1, the current text uses a variant of the original definitions. In particular, our definition of "typical feasibility" differs from the original definition of "average feasibility" in totally discarding exceptional instances and in even allowing the algorithm to fail on them (and not merely run for an excessive amount of time). The alternative definition was suggested by several researchers and appears as a special case of the general treatment provided in [2].

Section 2 is based on [1, 7]. Specifically, Theorem 8 (or rather the reduction of search to decision) is due to [1] and so is the introduction of the class samp$\mathcal{NP}$. A version of Theorem 11 was proven in [7], and our proof follows their ideas, which in turn are closely related to the ideas underlying the construction of pseudoradom generators based on any one-way functions (proved in [6]).

Recall that we know of the existence of problems in dist$\mathcal{NP}$ that are hard provided samp$\mathcal{NP}$ contains hard problems. However, these problems refer to somewhat generic decision problems such as $S_{\mathtt{u}}$. The presetation of dist$\mathcal{NP}$-complete problems that combine a more natural decision problem (like `SAT` or `Clique`) with a simple probability ensemble is an open problem.

Exercises

Exercise 12 (an equivalent definition of tpc$\mathcal{P}$**)** Prove that $(S, X) \in$ tpc$\mathcal{P}$ if and only if there exists a polynomial-time algorithm A such that the probability that $A(X_n)$ errs (in determining membership in S) is a negligible function in n.

Exercise 13 (tpc$\mathcal{P}$ **versus** $\mathcal{P}$ **– Part 1)** Prove that tpc$\mathcal{P}$ contains a problem (S, X) such that S is not even recursive. Furthermore, use $X = U$.

Guideline: Let $S = \{0^{|x|}x : x \in S'\}$, where S' is an arbitrary (non-recursive) set.

Exercise 14 (tpc$\mathcal{P}$ **versus** $\mathcal{P}$ **– Part 2)** Prove that there exists a distributional problem (S, X) such that $S \notin \mathcal{P}$ and yet there exists an algorithm solving S (correctly on all inputs) in time that is typically polynomial with respect to X. Furthermore, use $X = U$.

Guideline: For any time-constructible function $t : \mathbb{N} \to \mathbb{N}$ that is super-polynomial and sub-exponential, use $S = \{0^{|x|}x : x \in S'\}$ for any $S' \in \mathrm{DTIME}(t) \setminus \mathcal{P}$.

Exercise 15 (reductions preserve typical polynomial-time solveability) Prove that if the distributional problem (S, X) is reducible to the distributional problem (S', X') and $(S', X') \in$ tpc$\mathcal{P}$, then (S, X) is in tpc$\mathcal{P}$.

Guideline: Let B' denote the set of exceptional instances for the distributional problem (S', X') (i.e., the set of instances on which the solver in the hypothesis errs or exceeds the typical running-time). Prove that $\Pr[Q(X_n) \cap B' \neq \emptyset]$ is a negligible function (in n), using both $\Pr[y \in Q(X_n)] \leq p(|y|) \cdot \Pr[X'_{|y|} = y]$ and $|x| \leq p(|y|)$ for every $y \in Q(x)$. Specifically, use the latter condition for inferring that $\sum_{y \in B'} \Pr[y \in Q(X_n)]$ equals $\sum_{y \in \{y' \in B' : p(|y'|) \geq n\}} \Pr[y \in Q(X_n)]$, which guarantees that a negligible function in $|y|$ for any $y \in Q(X_n)$ is negligible in n.

Exercise 16 (reductions preserve error-less solveability) In continuation to Exercise 15, prove that reductions preserve error-less solveability (i.e., solveability by algorithms that never err and typically run in polynomial-time).

Exercise 17 (transitivity of reductions) Prove that reductions among distributional problems (as in Definition 3) are transitive.

Guideline: The point is establishing the domination property of the composed reduction. The hypothesis that reductions do not make too short queries is instrumental here.

Exercise 18 For any $S \in \mathcal{NP}$ present a simple probability ensemble X such that the generic reduction used in the proof of the existence of NP-complete problems violates the domination condition regarding a possible reduction of (S, X) to $(S_{\mathtt{u}}, U')$.

Guideline: Consider $X = \{X_n\}_{n \in \mathbb{N}}$ such that X_n is uniform over $\{0^{n/2}x' : x' \in \{0,1\}^{n/2}\}$.

Exercise 19 (variants of the Coding Lemma) Prove the following two variants of the Coding Lemma (which is stated in the proof of Theorem 4).

1. A variant that refers to any monotonically non-decreasing function $\mu : \{0,1\}^* \to [0,1]$ that is efficiently computable, where here we refer to the lexicographic order over $\{0,1\}^*$.

2. As in Part 1, except that in this variant the function μ is strictly increasing and the compression condition requires that $|C_\mu(x)| \leq \log_2(1/\mu'(x))$ rather than $|C_\mu(x)| \leq 1 + \min\{|x|, \log_2(1/\mu'(x))\}$, where $\mu'(x) \stackrel{\text{def}}{=} \mu(x) - \mu(x-1)$.

Guideline: In both cases, the proof is less cumbersome than the one presented in the main text.

Exercise 20 (randomized reductions) Following the outline in Section 1.3, provide a definition of randomized reductions among distributional problems.

1. Prove that randomized reductions are transitive (cf. Exercise 17).

2. In analogy to Exercise 15, prove that randomized reductions preserve feasible solveability (i.e., typical solveability in probabilistic polynomial-time). That is, if the distributional problem (S, X) is randomly reducible to the distributional problem (S', X') and $(S', X') \in \text{tpc}\mathcal{BPP}$, then (S, X) is in $\text{tpc}\mathcal{BPP}$.

3. In analogy to Exercise 16, prove that randomized reductions preserve solveability by probabilistic algorithms that err with probability at most $1/3$ on each input and typically run in polynomial-time.

4. Show that the error probability of such reductions can be reduced (while preserving the domination condition).

Extend the foregoing to reductions that involve distributional *search* problems.

Exercise 21 (simple vs sampleable ensembles – Part 1) Prove that any simple probability ensemble is polynomial-time sampleable.

Guideline: Let μ be the accumulating distribution function of $X = \{X_n\}_{n\in\mathbb{N}}$ (i.e., $\mu(x) = \mathsf{Pr}[X_{|x|} \leq x]$), and let p be a polynomial such that (without loss of generality) $|\mu(x)| = p(|x|)$ for every x. Consider the algorithm that, on input 1^n, uniformly selects $i \in [2^{p(n)}]$ and outputs x if and only if x is the lexicographically first string such that $i \leq \mu(x) \cdot 2^{p(n)}$. Note that this x can be found by binary search, using the fact that μ is polynomial-time computable.

Exercise 22 (simple vs sampleable ensembles – Part 2) Assuming that $\#\mathcal{P}$ contains functions that are not computable in polynomial-time, prove that there exists polynomial-time sampleable ensembles that are not simple.

Guideline: Consider any $R \in \mathcal{PC}$ and suppose that p is a polynomial such that $(x,y) \in R$ implies $|y| = p(|x|)$. Then consider the sampling algorithm A that, on input 1^n, uniformly selects $(x,y) \in \{0,1\}^{n-1} \times \{0,1\}^{p(n-1)}$ and outputs $x1$ if $(x,y) \in R$ and $x0$ otherwise. Note that $\#R(x) = 2^{p(|x|-1)} \cdot \mathsf{Pr}[A(1^{|x|-1}){=}x1]$.

Exercise 23 (distributional versions of NPC problems – Part 1 [1]) Prove that for any NP-complete problem S there exists a polynomial-time sampleable ensemble X such that any problem in dist$\mathcal{NP}$ is reducible to (S,X). We actually assume that the many-to-one reductions establishing the NP-completeness of S do not shrink the length of the input.

Guideline: Prove that the guaranteed reduction (of $S_{\mathtt{u}}$ to S) also reduces $(S_{\mathtt{u}},U')$ to (S,X), for some sampleable probability ensemble X. Specifically, note that U' is sampleable (by Exercise 21) and prove that the standard reduction of $S_{\mathtt{u}}$ to S, when applied to a sampleable probability ensemble, induces a sampleable distribution on the instances of S. Consider first the case that the standard reduction is length preserving, and next extend the treatment to the general case.

Exercise 24 (distributional versions of NPC problems – Part 2 [1]) Prove Theorem 10 (i.e., for any NP-complete problem S there exists a polynomial-time sampleable ensemble X such that any problem in samp$\mathcal{NP}$ is reducible to (S,X)). As in Exercise 23, we actually assume that the many-to-one reductions establishing the NP-completeness of S do not shrink the length of the input.

Guideline: We establish the result for $S_{\mathtt{u}}$, and the rest follows as in Exercise 23. Thus, we focus on showing that, for a fixed sampleable X, we can reduce any $(S',X') \in$ samp$\mathcal{NP}$ to $(S_{\mathtt{u}},X)$. Loosely speaking, X will be an adequate convex combination of all sampleable distributions (and thus X will not equal U' or U). Specifically, $X = \{X_n\}_{n\in\mathbb{N}}$ is defined such that X_n selects $i \in [n]$ with probability $\approx 1/i^2$, emulate the execution of the i^{th} algorithm (in lexicographic order) on input 1^n for n^3 steps,[14] and outputs whatever the latter has output (or 0^n in case the said algorithm has not halted within n^3 steps). Prove that, for any $(S'',X'') \in$ samp$\mathcal{NP}$ such that X'' is sampleable in cubic time, the standard reduction of S'' to $S_{\mathtt{u}}$ reduces (S'',X'') to $(S_{\mathtt{u}},X)$ (as per Definition 2; i.e., in particular, it satisfies the domination condition).[15] Finally, using adequate padding, reduce any $(S',X') \in$ samp$\mathcal{NP}$ to some $(S'',X'') \in$ samp$\mathcal{NP}$ such that X'' is sampleable in cubic time.

[14]Needless to say, the choice to consider n algorithms in the definition of X_n is quite arbitrary. Any other unbounded function of n that is at most a polynomial (and is computable in polynomial-time) will do. Likewise, the choice to emulate each algorithm for a cubic number of steps (rather some other polynomial number of steps) is quite arbitrary.

[15]Note that applying this reduction, denoted f, to X'' yields an ensembles that is also sampleable in cubic time. This uses the fact that the standard reduction runs in time that is almost linear in its output, which in turn is longer than the input.

Exercise 25 (search vs decision in the context of sampleable ensembles) Prove that every problem in samp$\mathcal{NP}$ is reducible to some problem in samp$\mathcal{PC}$, and every problem in samp$\mathcal{PC}$ is *randomly* reducible to some problem in samp$\mathcal{NP}$.

Guideline: See proof of Theorem 8.

References

[1] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the Theory of Average Case Complexity. *Journal of Computer and System Science*, Vol. 44 (2), pages 193–219, 1992.

[2] A. Bogdanov and L. Trevisan. Average-case complexity: a survey. In preparation, 2005.

[3] O. Goldreich. Notes on Levin's Theory of Average-Case Complexity. *ECCC*, TR97-058, Dec. 1997.

[4] O. Goldreich. *Foundation of Cryptography: Basic Tools.* Cambridge University Press, 2001.

[5] O. Goldreich. Expositions in Complexity Theory (various texts). Unpublished notes, December 2005. Availabe from the webpage `http://www.wisdom.weizmann.ac.il/~oded/cc-texts.html`

[6] J. Håstad, R. Impagliazzo, L.A. Levin and M. Luby. A Pseudorandom Generator from any One-way Function. *SIAM Journal on Computing*, Volume 28, Number 4, pages 1364–1396, 1999. Preliminary versions by Impagliazzo *et. al.* in *21st STOC* (1989) and Håstad in *22nd STOC* (1990).

[7] R. Impagliazzo and L.A. Levin. No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random. In *31st IEEE Symposium on Foundations of Computer Science*, pages 812–821, 1990.

[8] L.A. Levin. Average Case Complete Problems. *SIAM Journal on Computing*, Vol. 15, pages 285–286, 1986.