Infeasibility of Instance Compression and Succinct PCPs for NP

Lance Fortnow
Department of Electrical Engineering and
Computer Science
Northwestern University
Evanston, Illinois, U.S.A
fortnow@eecs.northwestern.edu

Rahul Santhanam
Department of Computer Science
University of Toronto
Toronto, Canada
rsanthan@cs.toronto.edu

ABSTRACT

The OR-SAT problem asks, given Boolean formulae $\phi_1, \ldots, \phi_m$ each of size at most n, whether at least one of the ϕ_i's is satisfiable.

We show that there is no reduction from OR-SAT to any set A where the length of the output is bounded by a polynomial in n, unless $\mathsf{NP} \subseteq \mathsf{coNP}/\mathsf{poly}$, and the Polynomial-Time Hierarchy collapses. This result settles an open problem proposed by Bodlaender et. al. [4] and Harnik and Naor [15] and has a number of implications.

- A number of parametric NP problems, including Satisfiability, Clique, Dominating Set and Integer Programming, are not *instance compressible* or *polynomially kernelizable* unless $\mathsf{NP} \subseteq \mathsf{coNP}/\mathsf{poly}$.

- Satisfiability does not have PCPs of size polynomial in the number of variables unless $\mathsf{NP} \subseteq \mathsf{coNP}/\mathsf{poly}$.

- An approach of Harnik and Naor to constructing collision-resistant hash functions from one-way functions is unlikely to be viable in its present form.

- (Buhrman-Hitchcock) There are no subexponential-size hard sets for NP unless NP is in co-NP/poly.

We also study *probabilistic* variants of compression, and show various results about and connections between these variants. To this end, we introduce a new strong derandomization hypothesis, the Oracle Derandomization Hypothesis, and discuss how it relates to traditional derandomization assumptions.

Categories and Subject Descriptors

F.1.3 [**Computation by Abstract Devices**]: Complexity Measures and Classes—*Relations among Complexity Classes*

General Terms

Algorithms, Security, Theory

Keywords

1. INTRODUCTION

Bodlaender et. al. [4] and Harnik and Naor [15] raise the following question, which has relevance to a wide variety of areas, including parameterized complexity, cryptography, probabilistically checkable proofs and structural complexity. This question asks whether the OR-SAT problem (given a list of formulae, is at least one satisfiable) is compressible.

Question 1.1. *Is there a function f that, given as input m Boolean formula $\phi_1, \ldots, \phi_m$ where each ϕ_i has length at most n (possibly much less than m), has the following properties?*

- *f is computable in time polynomial in m and n,*

- *$f(\phi_1, \ldots, \phi_m)$ is satisfiable if and only if at least one of the ϕ_i are satisfiable, and*

- *$|f(\phi_1, \ldots, \phi_m)|$ is bounded by a polynomial in n.*

We essentially settle this question in the negative by showing that a positive answer to Question 1.1 implies that the polynomial-time hierarchy collapses. We actually show the following stronger statement, in which f is allowed to map to an arbitrary set.

Theorem 1.2. *If NP is not contained in $\mathsf{coNP}/\mathsf{poly}$ then there is no set A and function f such that given m Boolean formula $\phi_1, \ldots, \phi_m$ where each ϕ_i has length at most n, f has the following properties*

- *f is computable in time polynomial in m and n,*

- *$f(\phi_1, \ldots, \phi_m) \in A$ if and only if at least one of the ϕ_i are satisfiable, and*

- *$|f(\phi_1, \ldots, \phi_m)|$ is bounded by a polynomial in n.*

Bodlaender et. al. [4] arrive at Question 1.1 from the perspective of parameterized complexity. Parameterized complexity asks about the complexity of NP problems based on some inherent parameter, like the number of variables in a formula or clique size in a graph. In parameterized complexity, feasibility is identified with *fixed-parameter tractability*. A problem is fixed-parameter tractable (FPT) if it has an algorithm running in time $f(k)n^{O(1)}$ where n is the input size and $f(k)$ is an arbitrary function of the parameter k. One

technique to show fixed-parameter tractability is *kernelization*, where one reduces the solution of the given instance to the solution of an instance of size depending only on the parameter; this new instance is called a "problem kernel". Chen et. al. [7] show that a problem is FPT if and only if it has a kernelization. However, in general, the kernel can be arbitrarily long as a function of the parameter.

It is a fundamental question in parameterized complexity as to which problems have polynomial-size kernels [12, 14]. Bodlaender et. al. [4] develop a theory of polynomial kernelizability, and define a notion of *strong distillation* which is useful in this context. A problem L has a strong distillation function if there is a polynomial-time computable function f that takes inputs $x_1, \ldots, x_m$ and outputs a y with $|y|$ bounded by a polynomial in $\max_i |x_i|$, such that y is in L if and only if at least one of the x_i's are in L. Question 1.1 is equivalent to asking if SAT has a strong distillation. Bodlaender et. al. conjectured that the answer to Question 1.1 is negative, and under that conjecture showed that the parameterized versions of several NP-complete problems do not have polynomial kernelizations, including k-Path and k-Cycle. Theorem 1.1 confirms the conjecture of Bodlaender et. al. modulo a widely-believed complexity-theoretic assumption, a rare connection between parameterized complexity and a traditional complexity-theoretic hypothesis.

Harnik and Naor [15] arrived at essentially Question 1.1 with a very different motivation, *cryptographic* in nature. An NP language L is *instance compressible* if there is some polynomial-time computable function f and a set A in NP such that x is in L if and only if $(f(x), 1^{|x|})$ is in A, and $|f(x)|$ is bounded by a polynomial in the length of a *witness* for x.[1] They showed that if the Satisfiability problem is compressible then collision resistant hash functions can be constructed from one-way functions. If an even stronger compressibility assumption holds, then oblivious transfer protocols can be constructed from one-way functions. This would imply that public-key cryptography can be based on one-way functions, solving one of the outstanding open problems in theoretical cryptography.

The cryptographic reductions of Harnik and Naor also follow from a weaker assumption than compressibility of SAT, namely the compressibility of the OR-SAT problem. In the OR-SAT problem, a list of m formulae $\phi_1 \ldots \phi_m$ each of size at most n is given as input, and a "yes" instance is one in which at least one of these formulae is satisfiable. Note that the size of a witness for OR-SAT is bounded above by n, hence compressibility of OR-SAT implies a positive answer to Question 1.1. Harnik and Naor describe a hierarchy of problems including Satisfiability, Clique, Dominating Set and Integer Programming, the compression of any of which would imply the compression of the OR-SAT problem. Thus Theorem 1.2 shows that none of these problems are compressible unless the polynomial-time hierarchy collapses. From a cryptographic point of view, our result indicates that the approach of Harnik and Naor may not be viable in its current form. But rather than considering this as a purely negative result, we hope it stimulates research into whether weaker and more plausible conditions already suffice for their constructions to work.

Theorem 1.2 also has applications to probabilistically checkable proofs (PCPs), and to structural complexity.

The result is directly relevant to the question of whether there are succinct PCPs for NP, which has been raised recently by Kalai and Raz [17]. A succinct PCP for an NP language L is a probabilistically checkable proof for L where the size of the proof is polynomial in the witness size n rather than in the instance size m. Current proofs of the PCP theorem [3, 2, 9] do not yield such PCPs. Kalai and Raz state that the existence of succinct PCPs "would have important applications in complexity theory and cryptography, while a negative result would be extremely interesting from a theoretical point of view". We show such a negative result: unless NP $\subseteq$ coNP/poly and the Polynomial Hierarchy collapses, SAT does not have succinct PCPs, nor do problems like Clique and DominatingSet. On the other hand, polynomially kernelizable problems such as VertexCover do have succinct PCPs.

Buhrman and Hitchcock [5] use Theorem 1.2 to show implications of NP-complete problems reducing to subexponential-size sets. Mahaney [20] showed that if there are NP-complete sparse sets (polynomial number of strings at every length) then P=NP. Karp and Lipton [18] show that if NP has a Turing-reduction to sparse sets than NP is in P/poly. But we had no known polynomial-time consequences of reductions to larger sets. Buhrman and Hitchcock use our result to show that there are no NP-complete sets of size $2^{n^{o(1)}}$ unless NP is in coNP/poly. More generally they show that if NP is not in coNP/poly then for every set A of size $2^{n^{o(1)}}$, there is no reduction from Satisfiability to A using $n^{1-\epsilon}$ adaptive queries to A for any fixed $\epsilon > 0$.

All the results mentioned so far concern compressibility using an f that is computable in deterministic polynomial time or in probabilistic polynomial time with very small error. We also study the case of general probabilistic compression. Here we do not have strong negative results, but we do have negative results for restricted notions as well as connections between different notions. Under a strong derandomization assumption that we call the Oracle Derandomization Hypothesis, we extend our infeasibility result for succinct PCPs to the more general case of gap amplification without blowing up the parameter by more than a polynomial factor. This has applications to the theory of hardness of approximation in the context of parameterized complexity.

In Section 2 we formally define the model and the problems. In Section 3 we prove our main result Theorem 3.1. In Section 4 we describe some applications of our main result and techniques for succinct PCPs, kernelization and cryptography. In Section 5 we give results about probabilistic compression. We discuss the feasibility of the Oracle Derandomization Hypothesis in Section 5.3.

2. PRELIMINARIES

2.1 Basic Complexity Notions

For the definitions of basic complexity classes such as NP, P, E, BPP and AM, we refer the reader to the Complexity Zoo[2]. There are a number of good reference works on parameterized complexity[10, 12, 21].

---

[1]Harnik and Naor actually allow $|f(x)|$ to be bounded by a polynomial in both the length of the witness and $\log |x|$. This variation is actually equivalent to our formulation, as discussed in Section 2

[2]http://qwiki.caltech.edu/wiki/Complexity_Zoo

2.2 Instance Compression

Motivated by cryptographic applications, Harnik and Naor [15] introduced the notion of instance compression for languages in NP. Informally, a language $L \in$ NP is instance compressible if there is a polynomial time algorithm that, for each instance x, produces an instance $C(x)$ of length polynomial in the *witness size* for x, such that $C(x)$ preserves information about whether $x \in L$. When the witness size is significantly smaller than the instance size, this gives a significant *compression* of the original instance x with respect to membership in L.

Harnik and Naor actually allow the size of the compressed instance to be polynomial in the logarithm of the instance size m as well as in the witness size n, but this does not give rise to a more relaxed notion, for the following reason. There are two cases: either $n > \log(m)$, or $n \leqslant \log(m)$. In the first case, being polynomial in $(n + \log(m))$ is equivalent to being polynomial in n. In the second case, the instance can be decided in polynomial time in m just by brute-force search over witnesses. Given a compression algorithm f according to the Harnik-Naor notion, we can define a compression algorithm f' according to our notion which checks which case holds and supposing the second case holds, solves its instance in polynomial time and correspondingly outputs a constant-sized "yes" or "no" instance instead of the instance produced by f.

Definition 2.1. *A parametric problem is a subset of $\{\langle x, 1^n \rangle \mid x \in \{0,1\}^*, n \in \mathbb{N}\}$.*

We study parametric problems rather than languages, because there isn't a natural notion of "witness size" for an arbitrary language in NP, unless one also specifies a relation corresponding to the language.

Definition 2.2. *Let L be a parametric problem and $A \subseteq \{0,1\}^*$. L is said to be compressible within A if there is a polynomial $p(\cdot)$, and a polynomial-time computable function f, such that for each $x \in \{0,1\}^*$ and $n \in \mathbb{N}$, $|f(\langle x, 1^n \rangle)| \leqslant p(n)$ and $\langle x, 1^n \rangle \in L$ iff $f(\langle x, 1^n \rangle) \in A$. L is compressible if there is some A for which L is compressible within A. L is self-compressible if L is compressible within L.*

Harnik and Naor actually allow the size of the compressed instance to be polynomial in the logarithm of the instance size m as well as in the witness size n, but this does not give rise to a more relaxed notion, for the following reason. There are two cases: either $n > \log(m)$, or $n \leqslant \log(m)$. In the first case, being polynomial in $(n + \log(m))$ is equivalent to being polynomial in n. In the second case, the instance can be decided in polynomial time in m just by brute-force search over witnesses. Given a compression algorithm f according to the Harnik-Naor notion, we can define a compression algorithm f' according to our notion which checks which case holds and supposing the second case holds, solves its instance in polynomial time and correspondingly outputs a constant-sized "yes" or "no" instance instead of the instance produced by f.

We will also be interested in *probabilistic* compression.

Definition 2.3. *Let L be a parametric problem and $A \subseteq \{0,1\}^*$. L is said to be probabilistically compressible with error $\epsilon(n)$ within A if there is a probabilistic polynomial-time computable function f such that for each $x \in \{0,1\}^*$ and $n \in \mathbb{N}$, with probability at least $1 - \epsilon(|x|)$ we have:*

1. $|f(\langle x, 1^n \rangle)| \leqslant poly(n)$

2. $f(\langle x, 1^n \rangle) \in A$ iff $x \in L$

L *is probabilistically compressible if there is an A such that L is probabilistically compressible within A with error $< 1/3$. L is errorless compressible if there is an A such that L is probabilistically compressible within A with error 0.*

We say that a probabilistic compression function f has randomness complexity R if it uses at most R random bits.

Note that errorless compression is a distinct notion from deterministic compression.

We will mainly be discussing compressibility of parametric problems in NP. We next define some of the problems we will be studying.

Definition 2.4. *$SAT = \{\langle \phi, 1^n \rangle \mid \phi$ is satisfiable and ϕ has at most n variables$\}$.*

Definition 2.5. *$VertexCover = \{\langle G, 1^{k \log(m)} \rangle \mid G$ has a vertex cover of size at most $k\}$.*

Definition 2.6. *$Clique = \{\langle G, 1^{k \log(m)} \rangle \mid G$ has a clique of size at most $k\}$.*

Note that our representation of the inputs to VertexCover and Clique is slightly non-standard in that we specify $k \log(m)$ rather than just the size k of the object for which we're searching. This is because we would like the parameter to accurately reflect the *witness* size. We stress that this is only for consistency and does not affect our results. Similarly we define the parametric problems DominatingSet and IntegerProgramming in the natural way. The problem OR-SAT is central to our study.

Definition 2.7. *$OR\text{-}SAT = \{\langle \phi_1, \ldots, \phi_m, 1^n \rangle \mid$ At least one ϕ_i is satisfiable, and each ϕ_i has size at most $n\}$.*

First, we ask: are there any natural parametric versions of NP-complete problems which can be shown to be compressible? The answer is yes, for problems for which the parameter is polynomially related to the input length. For instance, in the problem 3-SAT, the size of the formula is polynomially related to the number of variables (after deleting repeated clauses), hence any non-redundant version of the formula is itself a compressed instance, by our definition.

There still remains the question of whether there are less trivial compression algorithms for natural parametric problems. Using a close relationship between compressibility and the technique of kernelization in parameterized complexity, Harnik and Naor show the following:

Theorem 2.8. *[10, 15] VertexCover is self-compressible.*

As for the remaining problems, all that was previously known were reductions between them. Harnik and Naor define a type of reduction called "W-reduction" which relates compressibility of various problems. We refer to their paper [15] for the exact definition, and restrict ourselves to describing the known relations between compressibility of natural NP problems.

Proposition 2.9. *[15] If OR-SAT is not compressible, then neither are SAT, Clique, DominatingSet and IntegerProgramming.*

3. INFEASIBILITY OF DETERMINISTIC COMPRESSION

In this section, we show that deterministic compression of SAT implies that the Polynomial Hierarchy collapses. In fact, the conclusion holds even under the milder assumption that OR-SAT is deterministically compressible. Theorem 3.1 below is just Theorem 1.2 re-phrased using the terminology of compression.

Theorem 3.1. *If OR-SAT is compressible, then* coNP $\subseteq$ NP$/poly$ *and the polynomial-time hierarchy collapses.*

PROOF. Let $\phi_1, \ldots, \phi_m$ each have size n. By the assumption on the compressibility of OR-SAT, there is a language A and a function f computable in deterministic time poly(m, n) such that $|f(\langle \phi_1, \ldots, \phi_m, 1^n \rangle)| \leqslant n^c$ for some constant c independent of m, and $f(\langle \phi_1, \ldots, \phi_m, 1^n \rangle) \in A$ if and only if at least one of the ϕ_i are satisfiable. We will fix $m = n^c$. By appropriate padding and since A and c are arbitrary we can assume without loss of generality that $|f(\langle \phi_1, \ldots, \phi_m, 1^n \rangle)| = n^c = m$.

Let $T = \Sigma^m - A$. Note we have that $f(\langle \phi_1, \ldots, \phi_m, 1^n \rangle)$ is in T if and only if all of the ϕ_i are unsatisfiable.

We say that a string $y \in T$ is ϕ-good if there exist $\phi_1, \ldots, \phi_m$ each of size n such that $\phi = \phi_i$ for some i and $f(\langle \phi_1, \ldots, \phi_m, 1^n \rangle) = y$.

We will create a set $C = \{y_1, \ldots, y_k\} \subseteq T$ for some $k = O(n)$ such that for every unsatisfiable ϕ of length n, there is some $y_i \in C$ such that y_i is ϕ-good. We show coNP is in NP$/poly$ by giving an NP algorithm with advice C for deciding unsatisfiability of a formula ϕ as follows: We guess $\phi_1, \ldots, \phi_m$ each of size n such that $\phi = \phi_i$ for some i, and check that $f(\langle \phi_1, \ldots, \phi_m, 1^n \rangle) \in C$, accepting if it does and rejecting if not. If a tuple containing ϕ maps to C, ϕ is unsatisfiable, since $C \subseteq T$. Conversely, C contains a y that is ϕ-good by assumption on C, thus there is some choice of $\phi_1, \ldots, \phi_m$ for which the check succeeds.

We define a stage-wise process for picking the strings in C. Let S_0 be the set of unsatisfiable formula of size n. $|S_0| < 2^n$.

Stage i: For each y in T let B_y be the set of ϕ in S_{i-1} such that y is ϕ-good. Pick y_i that maximizes $|B_{y_i}|$, and put y_i in C. Let $S_i = S_{i-1} - B_{y_i}$. If $S_i = \varnothing$ we stop and set $k = i$, otherwise we go to stage $i + 1$.

Let $S = S_{i-1}$. We will show that $|B_{y_i}| \geqslant |S|/2$ and thus $|S_i|$ drops by at least a factor 2 with each increment in i; therefore there are at most $n + 1$ stages before S_i is empty.

Let X_y be the set of tuples $(\phi_1, \ldots, \phi_m) \in S^m$ such that $f(\langle \phi_1, \ldots, \phi_m, 1^n \rangle) = y$. Note that $X_y \subseteq (B_y)^m$, because each ϕ_i in a tuple in X_y is y-good and belongs to $S = S_{i-1}$. Also, $\cup_{y \in T} X_y = S^m$. Now we have

$$|S|^m \leqslant \sum_{y \in T} |X_y| \leqslant \sum_{y \in T} |B_y|^m \leqslant |T||B_{y_i}|^m \leqslant 2^m |B_{y_i}|^m$$

since $T \subseteq \{0, 1\}^m$ and y_i maximizes $|B_{y_i}|$. Thus $|B_{y_i}| \geqslant |S|/2$.

Thus we get a polynomial-size C with the property we require, and as argued above, this implies coNP $\subseteq$ NP$/poly$, and hence the polynomial-time hierarchy collapses to the third level [24].

Cai et. al. [6] show that if NP is in coNP$/poly$ then the polynomial-time hierarchy collapses to S_2^{NP}, currently the best known collapse from that assumption.

From Proposition 2.9, we also get consequences for other natural parametric problems.

Corollary 3.2. *If SAT is compressible, then* coNP $\subseteq$ NP$/poly$, *and* PH *collapses. The same conclusion holds if Clique, DominatingSet or IntegerProgramming are compressible.*

We next extend our infeasibility result to errorless compression and compression with very small error. This extension uses "Adleman's trick" [1] to embed a "good" random string in the advice, and then applies the argument for deterministic compression.

Theorem 3.3. *If OR-SAT is compressible with error* $<$ 2^{-m}, *where m is the instance size, then* NP $\subseteq$ coNP$/poly$ *and* PH *collapses.*

PROOF. The key observation is that compression with error $< 2^{-m}$ implies *non-uniform* compression. This is because, by a union bound, there must be some choice of randomness that yields a correct compressed instance for *each* instance of length m. Let z be such a random string. z is of size at most $poly(m)$, since the compression algorithm runs in probabilistic polynomial time. Now we just use the same argument as in the proof of Theorem 3.1 except that we also include z in the advice string when defining the proof system with advice for unsatisfiable formulae. In the earlier argument, the mapping from a tuple to a string could be performed deterministically; in the present case, we just perform it using z as the random string for the probabilistic compression function.

Our infeasibility result also extends to non-uniform compression, using the same proof as for Theorem 3.1.

Corollary 3.4. *If OR-SAT is non-uniformly compressible, then* NP $\subseteq$ coNP$/poly$ *and* PH *collapses.*

4. APPLICATIONS

In this section, we discuss the implications of our infeasibility result in various other research areas. First, we show a connection to succinct PCPs, which have attracted interest recently in the context of the study of short zero-knowledge proofs/arguments for NP. Second, we show that our result connects the theory of parameterized complexity to classical complexity theory, specifically with regard to the "kernelization" technique in parameterized complexity. Third, we discuss some implications for the cryptographic question of constructing collision-resistant hash functions from one-way functions.

4.1 Succinct PCPs

The PCP theorem [3, 2] is one of the landmark achievements in complexity theory. It states that any NP-complete language has polynomial-size proofs that can be verified probabilistically by reading only a *constant* number of bits of the proof.

Here "polynomial-size" means that the proof size is polynomial in the size of the input. It's natural to ask whether the proof size can be made polynomial in the size of the *witness* instead, giving more efficient proofs. The witness constitutes a proof which can be verified by reading all the bits of the proof - perhaps the number of bits read can be reduced to a constant by just blowing up the size of the proof polynomially? The known proofs of the PCP theorem [3, 9]

do not achieve this, since the size of the proof obtained is polynomial in the input size and not just in the witness size.

Kalai and Raz [17] raise this question in their paper on "interactive PCPs", and state that either a positive answer or a negative answer would be interesting. Here we give strong evidence for a negative answer by showing a strong connection between succinct PCPs and compressibility, and then invoking the results in the previous section.

We begin by defining "succinct PCPs", which are intuitively PCPs where the proof size depends polynomially on the witness length rather than the input length. In our framework of parametric problems, this corresponds to the proof size depending polynomially on the parameter rather than the input length.

Definition 4.1. *Let L be a parametric problem. L is said to have succinct PCPs with completeness c, soundness s, proof size S and query complexity q if there is a probabilistic polynomial-time oracle machine V such that the following holds for any instance $\langle x, 1^n \rangle$:*

1. *If $\langle x, 1^n \rangle \in L$, then there is a proof y of size $S(n)$ such that on input $\langle x, 1^n \rangle$ and with oracle access to y, V makes at most q queries to y on any computation path, and accepts with probability at least c.*

2. *If $\langle x, 1^n \rangle \notin L$, then there for any string y of size $S(n)$, on input $\langle x, 1^n \rangle$ and with oracle access to y, V makes at most q queries to y on any computation path, and accepts with probability at most s.*

L is said to have succinct PCPs if it has succinct PCPs with completeness 1, soundness $1/2$, proof size $poly(n)$ and query complexity $O(1)$.

For standard NP-complete problems such as SAT, CLIQUE and VERTEX-COVER, we abuse notation and say the problem has succinct PCPs if the natural parametric version of it does.

We show a near-equivalence between compressibility and the existence of succinct PCPs. In one direction, the existence of succinct PCPs implies compressibility with small error.

Theorem 4.2. *If SAT has succinct PCPs, then SAT is self-compressible with error less than 2^{-m}.*

PROOF. Assume that SAT has succinct PCPs with proof size n^C for some constant C, query size q (where q is a constant) and randomness complexity R. Since the verifier runs in probabilistic polynomial time, we have that $R = O(poly(m))$.

Our self-compression algorithm works as follows. It samples independently m^2 random strings $r_1, r_2 \ldots r_{m^2}$ each of length R. For each r_i, running the verifier for the succinct PCP with random string r_i corresponds to a function f_{r_i} on q bits of the proof such that the verifier accepts if and only if the function evaluates to 1 on those bits. Moreover, given the verifier, a canonical CNF formula of size $O(q2^q)$ for this function can be computed explicitly in time $poly(m)$. Note that since the proof has size at most n^C, there are at most $n^{Cq}2^{2^q}$ such functions. Here, the input formula is fixed, and hence the size of the input formula does not factor into the bound. The self-compression algorithm computes a canonical CNF formula of size at $O(q2^q)$ for each $f_{r_i}, i = 1 \ldots m^2$, and outputs the conjunction of these

formulae, omitting duplicate formulae. This is a SAT formula, and we need to show that the formula has size at most $poly(n)$, and that with error $< 2^{-m}$, the compressed formula is satisfiable if and only if the input formula is satisfiable.

The size bound follows from the upper bound on the total number of functions on q bits of the proof, together with the fact that we remove duplicates. For the error bound, it follows from the definition of PCPs that if the input formula is satisfiable, then the compressed formula is also satisfiable, since the verifier accepts with probability 1 on a correct proof. If the input formula is not satisfiable, then for any proof, the verifier accepts with probability at most $1/2$. In this case, the compressed formula we output may not be the conjunction of *all* possible formula corresponding to runs of the verifier, but we will argue that with very high probability, the total measure of r such that f_r is omitted is less than $1/2$, and hence that the compressed formula is still unsatisfiable even with the formulae corresponding to these f_r omitted from the conjunction. Indeed, if the measure of such r were greater than or equal to $1/2$, the probability that no such r is sampled as some r_i is at most $1/2^{m^2}$, since the sampling is done at random. Hence, with probability at least $1 - 1/2^{m^2}$, the compressed formula is unsatisfiable if the original formula is unsatisfiable. Thus the error is at most $1/2^{m^2}$.

A connection between probabilistic compression and parametric gap amplification (which is a more relaxed notion than succinct PCPs) was made in the work of Harnik and Naor [15]. Since we don't know how to rule out probabilistic compression in general, this connection does not suffice for our result. Instead, we show a connection between succinct PCPs and probabilistic compression with negligible error, which we do know how to rule out based on a standard assumption.

Using a more refined approach, we can show the infeasibility of PCPs with non-negligible gap between soundness and completeness, not just of PCPs with completeness 1. We provide a sketch of the proof.

Theorem 4.3. *If SAT has succinct PCPs with completeness c, soundness s, proof size $poly(n)$ and query complexity $O(1)$, where $c + s$ is computable in time $poly(m)$ and $c - s \geqslant 1/poly(n)$, then SAT is self-compressible.*

Proof Sketch. We follow the basic framework of the proof of Theorem 4.2. We compute different functions corresponding to independent runs of the verifier, and then take the conjunction of constraints derived from these functions in order to produce a compressed instance. The compressed instance will not directly be a SAT instance, however it will be an instance of an NP language, hence we can reduce it to a SAT instance of polynomial size.

The fact that the verifier does not have completeness 1 does complicate the argument. Let us call a function on $O(1)$ bits of the proof corresponding to some run of the verifier a "test function". Now we also need a notion of the "weight" of a test function, which is essentially the measure of random strings on which the verifier run corresponds to the function. Our compressed formula will consist of the conjunction of constraints derived from test functions. Each constraint is on $O(1)$ variables, and we pick an arbitrary representation for these constraints (say, a list consisting of the names of variables involved, together with the truth table

of the test function). The constraints will be present with multiplicity, where the multiplicity of a constraint is approximately the weight of its test function. This is designed to ensure that in the case of satisfiable input formulae, approximately a fraction c of constraints are satisfiable, and in the case of unsatisfiable ones, approximately a fraction s of them are satisfiable. We sample enough test functions and choose multiplicities with enough granularity (while still keeping the multiplicity of any one constraint polynomial in n) so that with probability more than $1 - 2^{-m}$, at least $(c+s)/2$ constraints are satisfiable if the input formula is satisfiable, and fewer than $(c+s)/2$ are satisfiable if the input formula is not satisfiable. This is argued with Chernoff bounds, using the facts that there is a non-negligible separation between c and s and that we sample enough. When we sample test functions, we also update estimates of their weights, sampling enough to ensure that the estimates are very accurate. The estimates are normalized and truncated to their $O(\log(n))$ high-order bits to ensure that multiplicities are at most polynomial. The compressed instance produced in this way is still of size polynomial in n, and is an instance of a general constraint satisfaction problem in NP, where we are given a set of constraints explicitly together with a parameter k and asked whether at least k of those constraints are satisfiable. Here the parameter $k = (c+s)/2$, which by our assumption can be computed explicitly in time $poly(m)$. This already establishes compressibility of SAT with error $< 2^{-m}$ from the assumption on succinct PCPs; in order to get self-compressibility, we reduce the constraint satisfaction instance to a SAT instance using the Cook-Levin reduction [8], which preserves the instance size up to a polynomial factor.

Note that it is actually sufficient to obtain a good approximation (i.e., an approximation to within an arbitrary $1/poly(n)$ additive term) of $c + s$ or even of just one of c and s, in order to obtain our result. But we do not see how to carry the compression through without access to *any* information about c or s apart from the fact that they are non-negligibly separated. □

Using Corollary 3.2, we get the following.

Corollary 4.4. *SAT does not have succinct PCPs unless* NP $\subseteq$ coNP$/poly$ *and* PH *collapses.*

In the other direction, self-compressibility implies the existence of succinct PCPs.

Theorem 4.5. *If SAT is self-compressible, then SAT has succinct PCPs.*

PROOF. Assume SAT is self-compressible via a compression function f. Given a formula ϕ of size m on n variables, $f(\phi)$ is a formula of size $poly(n)$ such that $f(\phi)$ is satisfiable if and only if ϕ is satisfiable. The PCP theorem states that there are PCPs for SAT with polynomial proof size, completeness 1, soundness $1/2$ and $O(1)$ queries.

We modify the verifier V in the PCP theorem to a verifier V' that does the following: It first applies the compression function f to the input formula ϕ and obtains a formula $f(\phi)$ which is satisfiable if and only if ϕ is satisfiable. Then it treats the proof as a proof that $f(\phi)$ is satisfiable and runs V with input $f(\phi)$ and oracle access to the proof. The correctness of V' follows from the self-compressibility property, and V' inherits its parameters from V. The proof size

is polynomial in the size of $f(\phi)$, i.e., of size $poly(n)$. Thus the PCP with verifier V' is a succinct PCP.

It's straightforward to prove a more general version of Theorem 4.5.

Theorem 4.6. *Let L be any parametric problem in* NP*. If L is compressible within* NP*, then L has succinct PCPs.*

Using Theorem 2.8, we derive the following corollary.

Corollary 4.7. *VertexCover has succinct PCPs.*

4.2 Kernelization

The notion of compression is very closely related to the notion of kernelization in parameterized complexity. Indeed, a parametric problem has a polynomial-size kernel if and only if it is self-compressible. Thus, Theorem 2.8 is just a reflection of the well-known fact that VertexCover has a polynomial-size kernel, in fact a linear-size kernel. This raises the question of whether the parameterized versions of other NP-complete problems such as SAT and Clique are polynomially kernelizable. The question of whether the parametric problems SAT has a polynomial kernelization is explicitly posed as an open problem by Flum and Grohe [12]. In their survey on kernelization, Guo and Niedermeier [14] state, "In terms of computational complexity theory, the derivation of lower bounds is of the utmost importance."

Bodlaender et. al. [4] develop a framework for studying polynomial kernelizability of natural parametric problems. The central question in their framework is Question 1.1, and they show that a negative answer to this question implies that various natural parametric problems are not polynomially kernelizable. Proposition 2.9 also gives that a negative answer to Question 1.1 implies that SAT is not polynomially kernelizable. Thus, from Theorem 3.1, we immediately get:

Corollary 4.8. *SAT is not polynomially kernelizable unless* PH *collapses.*

Proposition 2.9 also implies that various other parametric problems are unlikely to be polynomially kernelizable.

Corollary 4.9. *The following parametric problems are not polynomially kernelizable unless PH collapses: Clique, DominatingSet, IntegerProgramming.*

Note that as per our definitions, all the problems considered above are in fact fixed-parameter tractable, since they can be solved in time $O(2^n)poly(m)$. However, except for Corollary 4.8, this is true for a parameterization of the problem that is different from the standard parameterization. Corollary 4.9 is of philosophical interest, showing a novel link between a classical complexity assumption and a parameterized complexity assumption. Bodlaender et. al. [4] show separations between fixed-parameter tractability and polynomial kernelizability for various parametric problems more widely considered in practice.

4.3 Cryptographic Reductions

The main motivation of Harnik and Naor [15] for studying instance compressibility was the wealth of cryptographic applications. They showed that compressibility of OR-SAT would imply constructions of collision-resistant hash functions (CRHs) from one-way functions (OWFs), which would solve a long-standing open problem in cryptography. They

also showed that the compressibility assumption would have interesting implications for the active research area of "everlasting security" in the bounded storage model - any hybrid scheme could be broken. Harnik and Naor also discuss a stronger compressibility assumption, *witness-retrievable compressibility*, which implies construction of oblivious transfer protocols (OT) based on OWFs, but they also give evidence against their assumption, so we do not discuss it here.

Our results indicate that the approach of Harnik and Naor to cryptographic constructions needs to be modified in order to become feasible. An interesting direction is to find weaker conditions than compression of OR-SAT under which their reductions work - conditions which there is less evidence against, and which may even be feasible. One advantage of the Harnik-Naor is that it's non-black-box, which is essential for any construction of CRHs from OWFs [23], so it may be worthwhile trying to salvage it.

Independently, Dubrov and Ishai [11] used strong *incompressibility* assumptions to construct "non-Boolean" pseudo-random generators (PRGs), namely pseudo-random generators that fool non-Boolean tests. They show how to use such assumptions to reduce the randomness complexity of sampling algorithms, so that the randomness used is only marginally larger than the output length of the sampler. It would be interesting to derive such generators based on a standard complexity-theoretic assumption, rather than an incompressibility assumption. Theorem 3.1 makes some progress towards this goal, however a stronger theorem which rules out *average-case* compressibility of *polynomial-time* functions to a fixed polynomial size under a natural complexity-theoretic assumption is required, in order to obtain direct applications to the Dubrov-Ishai setting.

We expand on the above points in the full version of our paper [13].

5. PROBABILISTIC COMPRESSION

In this section, we study probabilistic compression. We are unable to show that probabilistic compression of NP-complete problems is unlikely under a standard complexity-theoretic assumption, but we do have results in settings where the number of random bits used is restricted. We also have negative results for the *implicit* case where the compression algorithm operates in time poly(n) when given random access to its input, but these negative results are under a somewhat non-traditional strong derandomization assumption. The implicit case is relevant to the question of parametric hardness of approximation, where we ask if the classical machinery for hardness-of-approximation results can be extended to the parameterized setting.

5.1 Negative Results in Restricted Settings

Theorem 3.3 applies to the case where the error is less than 2^{-m}, so here we will be concerned with the case where the error is larger. The techniques of Theorem 3.1 and 3.3 do not seem to apply here. But we do obtain various reductions between different versions of the problem, as well as results for restricted settings.

First, we note that the correctness probability can be amplified to $1 - 2^{-\text{poly}(n)}$ using standard techniques.

Proposition 5.1. *If L is probabilistically compressible with error $1/3$, then L is probabilistically compressible with error $2^{-\text{poly}(n)}$.*

PROOF. Let A be a language such that L is probabilistically compressible within A. Let f be the compression function compressing inputs with length m and parameter n with high probability to length at most $t(n)$ which is polynomial in n. We fix a polynomial $p(\cdot)$ and define a new language A' and compression function f' such that L is probabilistically compressible within A' via f'. An input $\langle x_1, x_2 \ldots x_k \rangle$ is in A' iff the majority of x_i's are in A. All inputs not of this form are excluded from A'. The probabilistic compression function f', given input x of length m, simulates f independently $p(n)$ times on x to obtain $x_1 \ldots x_p(n)$. It discards all strings of length $> t(n)$, re-indexes the strings to obtain $x_1, x_2 \ldots x_k$ and then outputs $\langle x_1, x_2 \ldots x_k \rangle$. Using Chernoff bounds, one gets that if f has error at most $1/3$, then f' has error at most $2^{-\Omega(p(n))}$.

Note that the new error is not sufficiently small to use Adleman's trick as in the proof of Theorem 3.3, since the instance size is m.

Next, we show some results where the randomness complexity of the compression function is restricted.

Lemma 5.2. *If L is probabilistically compressible with randomness complexity $O(\log(n))$, then L is deterministically compressible.*

PROOF. The proof is similar to the proof of Proposition 5.1. Let L be probabilistically compressible within A via a compression function f with randomness complexity $O(\log(n))$. Let c be a constant such that for any x, f compresses $\langle x, 1^n \rangle$ to a string of length at most n^c, with high probability. We define a new compression function f' computable in deterministic polynomial time, and a set A', such that L is compressible within A' via f'. Given input x, f' simulates f on x with every possible random string of length $O(\log(n))$ to obtain $x_1, x_2 \ldots x_t$, where $t = \text{poly}(n)$. It discards all strings with length $> n^c$ and then re-indexes the strings to obtain $x_1 \ldots x_k$. It outputs $\langle x_1, x_2 \ldots x_k \rangle$. The set A' is just the set of inputs of the form $\langle x_1, x_2 \ldots x_k \rangle$ such that the majority of x_i's are in A.

As a corollary, we can extend the negative result in Theorem 3.1 to the case of probabilistic compression with small randomness complexity.

Corollary 5.3. *If OR-SAT is probabilistically compressible with randomness complexity $O(\log(n))$, then PH collapses.*

We next show that if we could extend the negative result slightly to rule out probabilistic compression with randomness complexity $O(\log(m))$ (using advice), then it would also rule out probabilistic compression in the general case.

Theorem 5.4. *If L is probabilistically compressible, then L is probabilistically compressible with randomness complexity $O(\log(m))$ and $\text{poly}(m)$ advice.*

Proof Sketch. The idea is to try to derandomize the probabilistic reduction f for L by using a discrepancy set of size $\text{poly}(m)$. The probabilistic reduction can be viewed as a deterministic function taking x and the random string as input and yielding a string of size $\text{poly}(n)$ which is in L iff $x \in L$, with high probability over the choice of random string. We can get by using only polynomially many pseudo-random strings rather than all possible random strings if the "test" that the output string is in L iff $x \in L$ is satisfied

with approximately the same probability (say, with absolute difference at most $1/12$ between the probabilities) over pseudo-random strings as over random strings. The test can be encoded by a polynomial-size Boolean circuit with oracle access to L, hence a set of $\text{poly}(m)$ strings chosen at random will "fool" all such circuits with high probability. Thus there must exist such a set S of strings - we specify S explicitly in the advice string using $\text{poly}(m)$ bits. The new compression function f' for L with randomness complexity $O(\log(m))$ just uses its random string as an index into the discrepancy set encoded by the advice string, and simulates f using the corresponding element of the discrepancy set as its "random choice" rather than using a purely random string. The defining property of the discrepancy set ensured that this is still a valid probabilistic compression algorithm for L. □

5.2 Implicit Compression and Parametric Hardness of Approximation

Here we study *implicit* probabilistic compression, where the compression function is required to be computable in time polynomial in the size of the parameter. This sub-case of probabilistic compression is relevant to the question of whether parametric problems are as hard to approximate as they are to compute.

Definition 5.5. *A parametric problem L is implicitly probabilistically compressible if it is probabilistically compressible via a compression function f that, given an input $\langle x, 1^n \rangle$, can be computed in time $\text{poly}(n)$ with oracle access to the input.*

We show that if OR-SAT is implicitly probabilistically compressible, then either PH collapses or that a certain strong derandomization hypothesis fails. We do not have strong evidence for the *truth* of our derandomization assumption, but we do believe it is hard to *refute*, thus our result provides evidence that implicit probabilistic compression may be hard to obtain using known techniques. We discuss our derandomization hypothesis in more detail in Subsection 5.3.

The assumption we use states that given a string x, we can compute from x in polynomial time the truth table of a function f on $< \lfloor \log(x) \rfloor$ bits such that f requires exponential-size non-deterministic circuits even when the circuits have oracle access to x. The assumption can be interpreted as a strong diagonalization assumption - given an oracle in explicit form, the assumption states that it's possible to compute efficiently a function that diagonalizes against small circuits accessing that oracle.

Hypothesis 5.6. *(Oracle Derandomization Hypothesis) Let m be polynomially bounded as a function of n, such that $m(n)$ is both computable and invertible in polynomial time. There is a family of functions $G = G_n$, $G_n : \{0,1\}^m \to \{0,1\}^n$ computable in time $\text{poly}(m)$ and a constant $\epsilon > 0$ such that for any x, $G(x)$, when interpreted as the truth table of a function on $\lfloor \log(x) \rfloor$ bits, requires non-deterministic circuits of size n^ϵ even when the circuits are given oracle access to x. If the condition holds for $x \in A$ for some set $A \subseteq \{0,1\}^*$, but not necessarily for all x, we say that the Oracle Derandomization Hypothesis holds on A.*

We use Hypothesis 5.6 by combining it with a pseudo-random generator of Shaltiel and Umans [22] to derandom-

ize certain kinds of sampling algorithms using a smaller seed size than we would require if we carried out the derandomization naively. The obvious point of reference is Theorem 5.4, where we reduced the randomness to $O(\log(m))$ bits by embedding an appropriate discrepancy set in the advice string. We could not completely derandomize Theorem 5.4, i.e., convert it to a deterministic compression algorithm, because running over all possible random strings of length $O(\log(m))$ and defining the deterministic compressed string to correspond to the majority value of the probabilistic compressed strings would blow the size of the compressed string up to $\text{poly}(m)$, which makes the resulting deterministic compression trivial. Thus it is essential for us that the randomness is reduced to size $O(\log(n))$. If the probabilistic compression is implicit, we can use Hypothesis 5.6 to achieve this. We state the result we obtain, and refer to the full version of our paper [13] for details of the proof.

Theorem 5.7. *If OR-SAT is implicitly probabilistically compressible within NP, then either PH collapses or the Oracle Derandomization Hypothesis fails.*

The main reason for studying implicit compression is to extend our negative results on succinct PCPs to results on reductions from parametric problems to gap versions thereof. Unlike in the unparameterized setting, reduction of the parameterized problem SAT to its gap version is not equivalent to the existence of succinct PCPs - we know that succinct PCPs imply a reduction to the gap version, but not the converse. Theorem 5.7 has implications for the question of whether parameterized SAT reduces to its gap version, and in general for the question of whether a theory of parametric hardness of approximation can be developed along lines analogous to the classical theory based on PCPs - we refer to the full version of our paper [13] for details.

5.3 The Oracle Derandomization Hypothesis

In this subsection we discuss the Oracle Derandomization Hypothesis used in the previous subsection. We comment on how this hypothesis relates to traditional derandomization hypotheses, and show that *disproving* it as hard as separating P and NP.

In our opinion, quite apart from its relevance to compressibility, the Oracle Derandomization Hypothesis is interesting in its own right because it tests our intuitions of which kinds of derandomization are plausible and which are not. The hypothesis that E requires linear exponential size circuits, which was used by Impagliazzo and Wigderson [16] to completely derandomize BPP, now seems widely accepted by complexity theorists, so too the assumption that E requires linear exponential size *non-deterministic* circuits, used by Klivans and van Melkebeek [19] to derandomize AM. Klivans and van Melkebeek [19] use even stronger assumptions for other results on derandomization in a relativized context, such as the result that $\mathsf{PH} \subseteq \mathsf{P}^{\oplus \mathsf{P}}$ if E does not have sub-exponential size circuits with oracle access to $\oplus \mathsf{P}$. First we make some observations about how the Hypothesis generalizes traditional derandomization hypotheses.

Proposition 5.8. *There is an $\epsilon > 0$ such that $\mathsf{E} \not\subseteq i.o.\mathsf{NSIZE}(2^{\epsilon n})$ iff the Oracle Derandomization Hypothesis holds on 0^*.*

PROOF. We prove the "if" direction first. If the Hypothesis holds on 0^*, then we define a function f in E which doesn't have non-deterministic circuits of size $2^{\epsilon n}$ for some

$\epsilon > 0$. On an input x of length n, we do the following to compute $f(x)$: we simulate G_{2^n} on $0^{m(2^n)}$ to obtain a string X_n of length 2^n. X_n will be interpreted as the truth table of f on inputs of length n. $f(x)$ is computed by just reading off the appropriate value in the bit string X_n.

To prove the hardness of f, let ϵ be the constant in the statement of Hypothesis 5.6. Assume contrariwise that f has non-deterministic circuits of size 2^ϵ infinitely often. Then the same circuits also contradict the Oracle Derandomization Hypothesis on 0^*.

For the reverse direction, let f be a function without non-deterministic circuits of size $2^{\epsilon n}$. We define G as follows: on input $0^{m(n)}$, it outputs the truth table of f on $\lfloor \log(n) \rfloor$ bits, followed by a string of 0's. Now suppose the Oracle Derandomization Hypothesis were false on 0^*. Then, for infinitely many n, $G(0^{m(n)})$ has non-deterministic circuits of size $2^{\epsilon n}$ with oracle access to $0^{m(n)}$. Now, if we replace each oracle gate with the input "0", these are non-deterministic circuits of size $2^{\epsilon n}$ deciding f, contradicting the assumption on f.

Basically the same proof gives that the condition on the set A on which the Hypothesis holds can be made a little weaker.

Proposition 5.9. *let $\delta > 0$ be any constant, and $m(n)$ be a polynomially bounded function as in the statement of Hypothesis 5.6. Let A_m be the set of strings in $\{0,1\}^{m(n)}$ which, when interpreted as the truth table of a function on $\lfloor \log(m) \rfloor$ bits, have circuits of size $2^{n^{1-\delta}}$. Let A be the union of $A_{m(n)}$ over all n. Then there exists an $\epsilon > 0$ such that $\mathsf{E} \not\subseteq \mathsf{NSIZE}(2^{\epsilon n})$ iff the Oracle Derandomization Hypothesis holds on A.*

Proposition 5.9 states that the traditional derandomization hypothesis used to derandomize AM is equivalent to the Oracle Derandomization Hypothesis holding on all "succinct" strings, where "succinct" means that the string can be described by a circuit of size sub-polynomial in the length of the string.

Next, we ask the question, is there any heuristic evidence the Oracle Derandomization Hypothesis is true? Often, if there is a probabilistic process which generates an object of a certain type with high probability, then it is reasonable to conjecture that there is a deterministic process of similar complexity producing an object of that type. For instance, a function with specified input and output lengths which is chosen uniformly at random is usually a pseudo-random generator with strong properties. However, in our case, a function G chosen uniformly at random does not satisfy the required condition with high probability, but only with non-zero probability. This is not necessarily evidence *against* Hypothesis 5.6. First, there might be some non-trivial probabilistic process sampling G's that satisfy the condition in Hypothesis 5.6 with high probability and it might be reasonable to conjecture that this non-trivial process can be derandomized. Second, there are natural examples of probabilistic constructions (proved to be correct using the Lovasz Local Lemma) which are only known to work with non-zero probability, and can still be derandomized.

We turn the question on its head and ask if it might be possible to *unconditionally* disprove the Oracle Derandomization Hypothesis. This seems hard, since it would imply $\mathsf{P} \neq \mathsf{NP}$.

Theorem 5.10. *If the Oracle Derandomization Hypothesis is false, then $\mathsf{P} \neq \mathsf{NP}$.*

PROOF. Assuming $\mathsf{P} = \mathsf{NP}$, we construct a G satisfying the required condition. Fix some ϵ such that $0 < \epsilon < 1$. We observe that for any x of length m, there *does* exist a string of length n, which when interpreted as a function on $\lfloor \log(n) \rfloor$ bits, doesn't have non-deterministic circuits of size n^ϵ with oracle access to x, just by a counting argument. Indeed, we can find the lexicographically first such string $y(x)$ in $\mathsf{P}^{\Sigma_3^\mathsf{P}}$. By assumption, $\mathsf{P} = \mathsf{NP}$ and hence $\mathsf{P}^{\Sigma_3^\mathsf{P}} = \mathsf{P}$, thus we can carry out the search in polynomial time and output y.

6. QUESTIONS

We highlight the two main technical questions that arise from this work.

The first is whether some negative results can be shown under a standard assumption for the probabilistic or closely-related average-case version of compression. Such results would have relevance to parametric hardness of approximation, and provide further evidence for the inviability of certain approaches to cryptographic constructions.

The second is the general question of characterizing for which functions f, the compressibility of f-SAT implies collapse of PH. Here f-SAT is the natural generalization of the OR-SAT problem to Boolean functions other than OR. This question basically reduces to the question of whether AND-SAT is compressible, since the compression of f-SAT for non-monotone f directly implies $\mathsf{NP} \subseteq \mathsf{coNP}/\mathrm{poly}$, and every monotone f that depends on all its m variables embeds either a $\sqrt{m}$ sized OR or AND. Thus if we can show that (assuming NP not in $\mathsf{coNP}/\mathrm{poly}$) AND-SAT is not compressible, then under that same assumption f-SAT is not compressible for any f that has no useless variables.

7. REFERENCES

[1] L. Adleman. Two theorems on random polynomial time. In *Proceedings of the 20th Annual IEEE Symposium on the Foundations of Computer Science*, pages 75–83, 1978.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[3] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[4] H. Bodlaender, R. Downey, M. Fellows, and D. Hermelin. On problems without polynomial kernels. Manuscript, 2007.

[5] H. Buhrman and J. Hitchcock. NP-complete sets are exponentially dense unless NP $\subseteq$ co-NP/poly. In

Proceedings of 23rd Annual IEEE Conference on Computational Complexity, 2008. To appear.

[6] J.-Y. Cai, V. Chakaravarthy, L. Hemaspaandra, and M. Ogihara. Competing provers yield improved Karp–Lipton collapse results. *Information and Computation*, 198(1):1–23, 2005.

[7] L. Cai, J. Chen, R. Downey, and M. Fellows. Advice classes of parameterized tractability. *Annals of Pure and Applied logic*, 84(1):119–138, 1997.

[8] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[9] I. Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3), 2007.

[10] R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[11] B. Dubrov and Y. Ishai. On the randomness complexity of efficient sampling. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 711–720, 2006.

[12] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[13] L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(96), 2007.

[14] J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007.

[15] D. Harnik and M. Naor. On the compressibility of NP instances and cryptographic applications. In *Proceedings if the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 719–728, 2006.

[16] R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 220–229, 1997.

[17] Y. T. Kalai and R. Raz. Interactive PCP. *Electronic Colloquium on Computational Complexity*, 7(31), 2007.

[18] R. Karp and R. Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28(2):191–209, 1982.

[19] A. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial hierarchy collapses. *SIAM Journal of Computing*, 31(5):1501–1526, 2002.

[20] S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of berman and hartmanis. *Journal of Computer and System Sciences*, 25(2):130–143, 1982.

[21] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[22] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, 2001.

[23] D. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? *Advances in Cryptology, EUROCRYPT 1998, Lecture Notes in Computer Science*, 1403:334–345, 1998.

[24] C.-K. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26:287–300, 1983.