KNOWLEDGE COMPLEXITY VERSUS COMPUTATIONAL COMPLEXITY AND THE HARDNESS OF APPROXIMATIONS

Erez Petrank

KNOWLEDGE COMPLEXITY VERSUS COMPUTATIONAL COMPLEXITY AND THE HARDNESS OF APPROXIMATIONS

RESEARCH THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Science

Erez Petrank

Submitted to the Senate of the Technion - Israel Institute of Technology
Iyar 5755 H a i f a May 1995

This research was carried out in the faculty of Computer Science under the supervision of Prof. Oded Goldreich

I wish to thank Oded Goldreich for his invaluable guidance. I consider myself lucky to have been guided by Oded during my first steps into science and I hope I have absorbed some of his non-compromising deep search for the truth. I also thank Oded for his friendship and for his encouragement throughout this research.

I would like to thank Mihir Bellare, Mike Kearns, Joe Kilian, Seffi Naor, and Rafi Ostrovsky, with whom I had the opportunity to work closely. Each of them contributed in his own way to my knowledge and to my perception of science and research. In particular, I would like to thank Mihir and Rafi for collaborating with us in the research of knowledge complexity versus computational complexity (the second chapter in this thesis).

Last, I would like to thank all the people who were always willing to talk science (and non-science): Hagit Attiya, Amos Beimel, Shai Ben-David, Amir Ben-Dor, Ran Canetti, Benny Chor, Eli Dichterman, Guy Even, Shimon Even, Shai Halevi, Hugo Krawczyk, Eyal Kushilevitch, Silvio Micali, and Leonard Shulman.

To my parents and Yael.

Contents

Contents - (Continuation)

List of Figures

Abstract

In this D.Sc. dissertation we consider two areas in Computer Science: Interactive Proofs and Approximation Algorithms. The fascinating relation between these two areas was discovered in the last four years. However, in this dissertation we do not present results which regard the connection between the areas, but we study each of them separaely.

First, we study the knowledge complexity of interactive proofs. Our main result concerns the computational complexity of languages which have interactive proofs of logarithmic knowledge complexity. We show that all such languages can be recognized in $\mathcal{BPP}^{\mathcal{NP}}$. Prior to this work, for languages with greater-than-zero knowledge complexity (and specifically, even for knowledge complexity 1) only trivial computational complexity bounds (i.e., recognizability in $\mathcal{PSPACE} = \mathcal{IP}$) were known.

We recall that $\mathcal{BPP}^{\mathcal{NP}}$ is contained in the third level of the polynomial-time hierarchy ($\mathcal{PH}$). It is believed that $\mathcal{PH}$ is a proper subset of $\mathcal{PSPACE}$. Thus, assuming $\mathcal{PH} \subsetneq \mathcal{PSPACE}$, our result yields the first proof that there exist languages in $\mathcal{PSPACE}$ which cannot be proven by an interactive-proof that yields $O(\log n)$ bits of knowledge. In other words, there exist languages which do have interactive proofs but only interactive proofs with super-logarithmic knowledge-complexity.

Prior to our work, there was no solid indication that would contradict the possibility that all languages in $\mathcal{PSPACE}$ have interactive-proofs which yield only *one bit of knowledge*.

In the course of proving this result we have developed two tools which are important results on their own.

First, we develope a universal almost uniform generator. This is a probabilistic machine which is given a set $S \in \mathcal{NP}$ and a positive integer n and the machine samples "almost" uniformly in $S \cap \{0,1\}^n$ (almost means that the output distribution has exponentially small Norm-1 distance from the uniform distribution). We show that this generator can be run by an efficient machine that has access to an NP oracle.

A second result, which we develope and use in our proof, is a relation between statistical knowledge-complexity and perfect knowledge-complexity; specifically, we show that, for the honest verifier, these hierarchies coincide, up to a logarithmic additive term (i.e., $\mathcal{SKC}(k(\cdot)) \subseteq \mathcal{PKC}(k(\cdot) + O(\log(\cdot)))$).

In the second part of this dissertation, we study the hardness of approximation algorithms. We refine the complexity analysis of approximation problems by relating it to a new parameter called *gap location*. Many of the results obtained so far for approximations yield satisfactory analysis with respect to this refined parameter, but some known results (e.g., MAX-k-COLORABILITY, MAX 3-DIMENSIONAL MATCHING and MAX NOT-ALL-EQUAL 3SAT) fall short of doing so. As a second contribution, our work fills the gap in these cases by presenting new reductions.

Next, we present definitions and hardness results of new approximation versions of some NP-complete optimization problems. The problems we treat are VERTEX COVER (for which we define a different optimization problem from the one treated in [PY-91]), k-EDGE COLORING, and SET SPLITTING.

List of Symbols and Abbreviations

Symbol/Abbreviation	Meaning
$PKC(k(\cdot))$	The class of languages which have perfect knowledge complexity $k(\cdot)$.
$SKC(k(\cdot))$	The class of languages which have statistical knowledge complexity $k(\cdot)$.
$CKC(k(\cdot))$	The class of languages which have computational knowledge complexity $k(\cdot)$.
$\mathcal{AM}$	The class of languages which have a constant round Arthur Merlin protocols.
$\mathcal{IP}$	The class of languages which can be proven by an interactive proof.
$\mathcal{HP}$	The polynomial time hierarchy
$\mathcal{U}[B]$	The uniform distribution over a finite set B.
Negligible	A fraction which is asymptotically smaller than the reciprocal of any polynomial.
$OPT(I)$	The value of the optimal solution for a problem insance I.

Chapter 1

Introduction

In this dissertation, we present research conducted in two areas in Computer Science: Intercative Proofs and the Hardness of Approximations. Originally, the goal of this research was to explore the properties of languages according to the knowledge complexity of their interactive proofs. Indeed the first part of this dissertation describes the results we have obtained in this field. However, the fascinating conncetion recently discovered between interactive proofs and the hardness of approximations, which led to a solution of many long standing open problems related to the hardness of approximations, enticed us into exploring the connection between the two areas. Eventually, the results we have achieved in this area did not concern the connection between interactive proofs and the hardness of approximations, but were more specific to studying the hardness of approximations. This study appears in the second part of this dissertation.

Since the two different works are not directly connected, the rest of this dissertation is practically partitioned into two. In what follows we introduce each of the areas. The connection between the two areas is described while introducing the hardness of approximations (see Section 1.2).

1.1 Knowledge Complexity

The notion of knowledge-complexity was introduced in the seminal paper of Goldwasser Micali and Rackoff [GMR-85, GMR-89]. Knowledge-complexity (KC) is intended to measure the *computational advantage* gained by interaction. Satisfactory formulations of knowledge-complexity, for the case that it is not zero, have recently appeared in [GP-91]. A natural suggestion, made by Goldwasser, Micali and Rackoff, is to classify languages according to the knowledge-complexity of their interactive-proofs [GMR-89]. We feel that it is worthwhile to give this suggestion a fair try.

The lowest level of the knowledge-complexity hierarchy is the class of languages having interactive proofs of knowledge-complexity zero, better known as zero-knowledge. Actually, there are three hierarchies extending the three standard definitions of zero-knowledge; namely *perfect*, *statistical* and *computational*. Let us denote the corresponding hierarchies by $\mathcal{PKC}(\cdot)$, $\mathcal{SKC}(\cdot)$, and $\mathcal{CKC}(\cdot)$. Assuming the existence of one-way functions, the third hierarchy collapses, namely $\mathcal{CKC}(0) = \mathcal{IP} = \mathcal{CKC}(\mathit{poly})$ [GMW-86, IY-87, B+-88]. Put differently, the zero level of *computational* knowledge-complexity extends to the maximum possible. Anyhow, in the rest of this work we will be only interested in the other two hierarchies.

Previous works have provided information only concerning the zero level of these hierarchies. Fortnow has pioneered the attempts to investigate the computational complexity of (perfect/statistical) zero-knowledge [F-89], and was followed by Aiello and Hastad [AH-87]. Their results can be summarized by the following theorem that bounds the computational complexity of languages having zero-knowledge proofs.

Theorem [F-89, AH-87]:

$$\mathcal{SKC}(0) \subseteq \mathcal{AM} \bigcap \text{co-}\mathcal{AM}$$

Hence, languages having statistical zero-knowledge must lie in the second level of the polynomial-time hierarchy. Needless to say that $\mathcal{PKC}(k(\cdot)) \subseteq \mathcal{SKC}(k(\cdot))$, for any function k and in particular for $k \equiv 0$.

On the other hand, if we allow polynomial amount of knowledge to be revealed, then every language in $\mathcal{IP}$ can be proven.

Theorem [LFKN-90, Sha-90]:

$$\mathcal{PKC}(\mathit{poly}(\cdot)) = \mathcal{IP} = \mathcal{PSPACE}$$

As indicated in [GP-91], the first equality is a property of an adequate definition (of knowledge complexity) rather than a result.

In this work we study the class of languages that have interactive-proofs with logarithmic knowledge-complexity. In particular, we bound the computational complexity of such languages, showing that they can be recognized by probabilistic polynomial-time machines with access to an NP oracle.

Main Theorem:

$$\mathcal{SKC}(O(\log(\cdot))) \subseteq \mathcal{BPP}^{\mathcal{NP}}$$

We recall that $\mathcal{BPP}^{\mathcal{NP}}$ is contained in the third level of the polynomial-time hierarchy ($\mathcal{PH}$). It is believed that $\mathcal{PH}$ is a proper subset of $\mathcal{PSPACE}$. Thus, assuming $\mathcal{PH} \subsetneq \mathcal{PSPACE}$, our result yields the first proof that there exist languages in $\mathcal{PSPACE}$ which cannot be proven by an interactive-proof that yields $O(\log n)$ bits of knowledge. In other words, there exist languages which do have interactive proofs but only interactive proofs with super-logarithmic knowledge-complexity.

Prior to our work, there was no solid indication that would contradict the possibility that all languages in $\mathcal{PSPACE}$ have interactive-proofs which yield only *one bit of knowledge*.

Our proof of the Main Theorem consists of two parts. In the first part, we develope a procedure for recognizing languages having interactive proofs of logarithmic *perfect* knowledge complexity. To this end, we develope a tool called "the universal almost uniform generator". (We elaborate on this notion in Subsection 1.1.2 below). In the second part of our proof we transform interactive proofs of *statistical* knowledge complexity $k(n)$ into interactive proofs of *perfect* knowledge complexity $k(n) + O(\log n)$. This transformation refers only to knowledge-complexity with respect to the honest verifier, but this suffices since the first part of our proof applies to the knowledge-complexity with respect to the honest verifier. Yet, the transformation is interesting for its own sake, and a few words are in place.

1.1.1 The Realtion of Statistical to Perfect Knowledge Complexity

The question of whether statistical zero-knowledge equals perfect zero-knowledge is one of the better known open problems in this area. The question has been open also for the case of zero-knowledge with respect to the honest verifier. We show the following.

Theorem: For every poly-time computable function $k : \mathsf{N} \mapsto \mathsf{N}$ (and in particular for $k \equiv 0$)

$$\mathcal{SKC}(k(\cdot)) \subseteq \mathcal{PKC}(k(\cdot) + O(\log(\cdot)))$$

This result may be considered an indication that these two hierarchies may collide.

1.1.2 Efficient Almost Uniform Generation

A technique underlying our main result is to generate almost random elements of a set in probabilistic, polynomial time with an NP oracle. As this might be of independent interest, let us describe the result in more detail.

Let $S \subseteq \{0,1\}^*$ be a set verifiable in polynomial time (i.e., $S \in$ NP), and let $S_n = S \cap \{0,1\}^n$. The uniform generation problem is to generate, on input 1^n, an element of S_n distributed uniformly at random. Jerrum, Valiant, and Vazirani [JVV-86], using results of Stockmeyer [St-83] on approximate counting, showed that uniform generation can be done in probabilistic, polynomial time with a Σ_2^{P} oracle.

For our applications we would like a lower complexity than PPT with a Σ_2^{P} oracle. On the other hand, we could tolerate a slight deviation from uniform of the output distribution. Accordingly, we consider the problem of *almost* uniform generation: on input 1^n and $\delta > 0$ generate a random element from a distribution within distance δ of the uniform distribution on S_n (the distance between distributions E_1 and E_2 is defined as $\frac{1}{2}\sum_x |\operatorname{Pr}_{E_1}[x] - \operatorname{Pr}_{E_2}[x]|$).

If $\delta = n^{-c}$ for some fixed constant c then techniques from Impagliazzo, Levin and Luby [ILL-89] can be used to do almost uniform generation in probabilistic, polynomial (in n) time with an NP oracle. However (for applications in this work in particular) we would like to be able to achieve values of δ which are exponentially (in n) small, in the same complexity. We show that this can be done.

Theorem: Let $S \in$ NP. Then there is a probabilistic oracle machine A which on input 1^n and $\delta > 0$ runs in time polynomial in n and $\lg \delta^{-1}$, and has the property that the distribution $A^{NP}(1^n, \delta)$ is within distance δ of the uniform distribution on S_n.

The special case in which S is decidable in polynomial time (i.e. $S \in$ P) is important on its own and specifically, in this work we use the almost uniform generator only for sets in P.

In Theorem 2.5 we actually prove something a little stronger: the almost uniform generation is "universal" (in the sense that A does not depend on S but rather gets a description of S as an input).

This result is established by combining techniques from Jerrum, Valiant and Vazirani [JVV-86] and Stockmeyer [St-83] with Carter-Wegman universal hash function [CW-79] based techniques for estimating set sizes (Sipser [Si-83]). The details are in Section 2.2.1.

1.1.3 Motivation for studying KC

In addition to the self-evident fundamental appeal of knowledge complexity, we wish to point out some practical motivation for considering knowledge-complexity greater than zero. In particular, cryptographic protocols that release a small (i.e., logarithmic) amount of knowledge may be of practical value, especially if they are only applied once or if one can obtain sub-additive bounds on the knowledge complexity of their repeated executions. Note that typically, a (single application of a) sub-protocol leaking logarithmically many bits (of knowledge) does not compromise the security of the entire protocol. The reason being that these (logarithmically many) bits can be guessed with non-negligible probability, which in turn means that any attack due to the "leaked bits" can be simulated with non-negligible probability without them.

But why use low knowledge-complexity protocols when one can use zero-knowledge ones (see, [GMW-86, GMW-87])? The reason is that the non-zero-knowledge protocols may be more efficient and/or may require weaker computational assumptions (see, for example, [OVY-90]).

1.1.4 Remarks

A remark *concerning two definitions.* Throughout this work, $\mathcal{SKC}(k(\cdot))$ and $\mathcal{PKC}(k(\cdot))$ denote the classes of knowledge-complexity *with respect to the honest verifier.* Note that the Main Theorem is only strengthen by this, whereas the transformation from statistical to perfect knowledge complexity (mentioned above) is indeed weaker. Furthermore, by an interactive proof we mean one in which the *error probability is negligible* (i.e., smaller than any polynomial fraction). A few words of justification appear in Section 2.1.1.

A remark *concerning Fortnow's paper* [F-89]. In course of this research, we found out that the proof that $\mathcal{SKC}(0) \subseteq$ co-$\mathcal{AM}$ as it appears in [F-89] is not correct. In particular, there is a flaw in the AM-protocol presented in [F-89] for the complement language (see Section 2.6). However, the paper of Aiello and Hastad provides all the necessary machinery for proving Fortnow's result as well [AH-87, H-94]. Needless to say that the basic approach presented by Fortnow (i.e., looking at the "simulator-based prover") is valid and has inspired all subsequent works (e.g., [AH-87, BMO-90, Ost-91, BP-92a, OW-93]) as well as the current one.

1.2 The Hardness of Approximations

The importance of approximation algorithms became evident in the early 70's, when it was discovered that unless P=NP (which is very unlikely) many important optimization problems cannot be solved precisely in polynomial time [Coo-71, Kar-72]. The practical need to solve these problems led researchers to relax the precision requirement and try to find approximation algorithms to these problems. Let Π be a maximization (resp. minimization) problem, and let $OPT(I)$ be the precise solution of an instance I. We say that an approximation algorithm, A, approximates Π to within a ratio of $1-\epsilon$ (resp. $1+\epsilon$) if for any input, I, A outputs a number $A(I)$ which satisfies $(1-\epsilon)OPT(I) \leq A(I) \leq OPT(I)$ (respectively $OPT(I) \leq A(I) \leq (1+\epsilon)OPT(I)$ for minimization problems).

It was soon discovered that some problems were easy to approximate. For problems as BIN PACKING and KNAPSACK it was even possible to construct a polynomial time approximation scheme (a polynomial time approximation scheme is a family of algorithms, one for each $\epsilon > 0$, such that A_ϵ approximates the optimization problem to within a ratio of $1-\epsilon$ for maximization problems and $1+\epsilon$ for minimization problems). For a second class of problems, like MAX-SAT, VERTEX COVER or metric TRAVELLING SALESMAN, it was possible to construct a polynomial time algorithm that approximated the correct solution to within some constant ratio, but no polynomial time approximation scheme was found for these problems. For the last set of problems, such as MAXIMUM CLIQUE or MINIMUM GRAPH COLORABILITY, even poor approximation algorithms could not be found.

Lower bounds on the possibility of constructing polynomial time algorithms (even assuming P$\neq$NP) were hardly known. Interesting such results concerning GRAPH COLORING, TRAVELLING SALESMAN (without the triangle inequality) and MAXIMUM CLIQUE can be found in [GJ-79] and [SG-76]. In a paper, that turned out to be somewhat prophetic, Papadimitriou and Yannakakis [PY-91] defined a class of optimization problems called MAX-SNP. They showed that all the problems in this class can be approximated to within some constant ratio. They also presented some very interesting problems that were *hard* for this class. That is, if a MAX-SNP-Hard problem has a polynomial time approximation scheme then all problems in MAX-SNP have polynomial time approximation schemes. The class MAX-SNP-Hard contains problems as MAX-3SAT-B, INDEPENDENT-SET-B, VERTEX-COVER-B, MAX-CUT, MAX-k-COLORABILITY [PY-91], metric TRAVELLING SALESMAN [PY-92], STEINER TREE [BP-89] SHORTEST SUPERSTRING[BJLTY-90], MUL-

TIWAY CUTS [DJPSY-92] and bounded 3-DIMENSIONAL MATCHING [Kan-91].

Meanwhile, with no apparent connection, the research in the area of interactive proofs was making a rapid progress. The research was initiated by Goldwasser Micali and Rackoff [GMR-89] and Babai [Bab-85]. The extended model of multi-provers interactive proofs was suggested by Ben-Or, Goldwasser, Kilian, and Wigderson [BGKW-88], and was shown equivalent to the model of transparent proofs [FRS-88]. It was then discovered that the languages, for which interactive proofs exist, are exactly the languages in PSPACE [LFKN-90, Sha-90] and that languages, for which multi-provers interactive proofs exist, are exactly the languages in NEXP-TIME.

Feige, Goldwasser, Lovasz, Safra, and Szegedi [FGLSS-91] were the first to observe the connection between transparent proofs and the hardness of approximations. They were able to show that if there exists a polynomial time algorithm which approximates MAXIMUM CLIQUE to within a factor $2^{\log^{1-\epsilon} n}$ then all languages in NP can be determined in almost polynomial deterministic time. Their result was improved by Arora and Safra [AS-92], who showed that if there exists a polynomial time algorithm that approximates the clique problem to within a ratio of $2^{\log n/(\log\log n)^{O(1)}}$ then P=NP. The following work, by Arora, Lund, Motwani, Sudan, and Szegedi [ALMSS-92], improved over this result by showing that no polynomial time algorithm can approximate MAXIMUM CLIQUE to within a ratio of n^ϵ for some $\epsilon > 0$ unless P=NP. They were also able to show (using the same techniques) another important result, namely, that unless P$\neq$NP there exists a constant ϵ such that no polynomial time algorithm can approximate MAX-3SAT to within a ratio of $1-\epsilon$. As this problem is a member of MAX-SNP, this result implies that no problem in MAX-SNP-Hard has a polynomial time approximation scheme unless P=NP. Lund and Yannakakis have shown two additional lower bounds. They showed that unless P=NP, no polynomial time algorithm can approximate MINIMUM GRAPH COLORABILITY to within a ratio of n^ϵ for some constant $\epsilon > 0$, and that unless NP$\subseteq$DTIME[$n^{poly\log n}$], no polynomial time algorithm approximates SET COVER to within a ratio of $c\log_2 N$ for any constant $0 < c < \frac{1}{4}$ (here N stands for the cardinality of the set that has to be covered). Recently, the second result was improved by Bellare, Goldwasser, Lund, and Russel [BGLR-93] by reducing the assumption to NP$\not\subseteq$DTIME[$n^{\log\log n}$].

In this work, we relate the study of approximation algorithms to a new parameter called *the gap location*. Let us begin by presenting a partition of the class of optimization problems into two categories. We later introduce the notion of gap location which refers to the second category.

1.2.1 Two categories of optimization problems

Generally, an optimization problem consists of a set of instances and a function $f : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ that assigns, for each instance I and each candidate solution σ, a real number $f(I,\sigma)$ called the value of the solution σ. The optimization task is to find a solution σ to a problem instance I such that $f(I,\sigma)$ is the largest possible over all $\sigma \in \{0,1\}^*$. We say that an algorithm A *approximates* a maximization (resp. minimization) problem Π to within $1-\epsilon$ (resp. $1+\epsilon$) if, for every instance I of Π whose optimal solution has value $OPT(I)$, the output of A on I satisfies $(1-\epsilon)OPT(I) \leq A(I) \leq OPT(I)$ (resp. $OPT(I) \leq A(I) \leq (1+\epsilon)OPT(I)$). Most natural optimization problems are associated with a decision problem in NP. We have a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ which is checkable in polynomial time (i.e., given (I,σ) it is possible to check in time polynomial in $|I|$ whether $(I,\sigma) \in R$), and we call σ a *valid solution* to an input I if $(I,\sigma) \in R$. The decision problem is whether or not there exists a valid solution to the input I. Two natural categories of corresponding optimization problems follow.

1. THE LARGEST SOLUTION.

   Here, we associate valid solutions with some natural "size" which we would like to maximize/minimize. More formally, we are trying to maximize the function

   $$f(I,\sigma) = \begin{cases} \text{size}(\sigma) & \text{if } (I,\sigma) \in R \\ -\infty & \text{otherwise} \end{cases}$$

   where size$(\cdot)$ is a function which depends on the problem and can be efficiently extracted from σ (usually, size(σ) is the number of elements encoded in the solution σ). We replace $-\infty$ by $+\infty$ when a minimization problem is involved. Two examples in this category are MAX CLIQUE, in which we are looking for the size of the largest clique in the input graph, and MIN COLORING, in which we would like to find the minimum number of colors required to color the input graph such that no two adjacent vertices have the same color.

2. THE QUALITY OF THE SOLUTION.

   Here, we assume that the condition $(I,\sigma) \in R$ contains a large (yet polynomially bounded) number of natural "sub-conditions" and our task is to find the maximum number of sub-conditions that can be satisfied by a single (i.e., the best) solution. In this category, we have MAX-SAT, in which we are trying to find the maximum number of clauses that can be satisfied by an assignment to the input formula, and MAX 3-COLORABILITY, in which we are trying to find the maximum number of consistent edges in a 3-coloring of the input graph. (We call an edge in a graph G *consistent* with respect to a 3-coloring of G if its two adjacent vertices are assigned different colors.) Note that, in this setting, the best solution of an instance I is not necessarily a valid solution, since there are instances that do not have a valid solution. Also, A solution σ is a valid solution of an instance I ($(I, \sigma) \in R$) iff σ satisfies all sub-conditions implied by I.

Approximation problems can be partitioned in the same manner and so can results concerning the difficulty of approximations. The first category contains problems such as approximating the size of the biggest clique in the input graph (the hardness of this problem was shown by [FGLSS-91, AS-92, ALMSS-92]), or the minimum number of colors needed to properly color the input graph (see [LY-92]). The second category includes problems such as approximating the maximum number of clauses that can be simultaneously satisfied in an input formula [ALMSS-92], or approximating the maximum number of consistent edges in a best 3-coloring of the input graph (Papadimitriou & Yannakakis 1991). (see [PY-91]).

The gap location parameter is a natural measure which arises when analyzing the hardness of approximation problems in the second category.

1.2.2 The gap-location parameter

Practically all researchers in the area noticed the connection between the hardness of approximating a problem and the existence of a "gap" that is hard to differentiate. For example, the hardness of approximating MAX-SAT was shown by proving that there exists a constant $\epsilon_0 > 0$ such that, unless P=NP, one can not distinguish in polynomial time between formulae for which all clauses can be satisfied and formulae for which only a fraction $1 - \epsilon_0$ of clauses can be satisfied [ALMSS-92]. The L-reductions that were used by Papadimitriou & Yannakakis [PY-91], preserve the existence of such gaps and thus, polynomial time inseparable gaps appear in all MAX-SNP-hard problems. This implies that these problems have no polynomial time approximation schemes. Loosely speaking, a hard gap for an optimization problem Π consists of two reals $0 \leq \alpha_0, \epsilon_0 \leq 1$ such that, given an instance I of Π, it is

NP-hard to tell whether I has a solution that satisfies at least a fraction α_0 of the sub-conditions posed by I or whether any solution of I satisfies at most a fraction $\alpha_0 - \epsilon_0$ of the sub-conditions posed by I. Such a hard gap is said to have location α_0. For example, the hard gap proven for MAX-SAT has location 1. Let us first argue that the "location" of the hard gap shown for MAX-SAT is the "right" one and then explain why there are some problems whose proofs of hardness are weaker in this respect.

Our main interest lies in the original question of satisfiability, i.e., in telling whether a formula φ has an assignment that satisfies all its clauses or not. It is therefore interesting to see that we cannot solve even the easier question of whether all the clauses of φ can be satisfied (simultaneously) or whether any assignment to φ satisfies at most a fraction $1 - \epsilon_0$ of its clauses. It would be somewhat artificial (and clearly of lesser interest) to show that it is impossible to tell whether a formula φ has an assignment that satisfies more than $2/3$ of its clauses or whether no assignment to φ satisfies more than $2/3 - \epsilon_0$ of its clauses, although showing this would still imply the hardness of approximating MAX-SAT (i.e., approximating the maximum number of simultaneously satisfied clauses). Furthermore, the intuition about the "right" location of the hard gap coincides with the power of such a result. Namely, for MAX-SAT, the existence of a hard gap at location 1 implies the existence of a hard gap at any location $1/2 < \alpha \leq 1$. That is, the fact that one cannot tell in polynomial time whether a formula has an assignment that satisfies all its clauses or whether all assignments to φ satisfy at most a fraction $1 - \epsilon_0$ of its clauses implies the fact that one cannot tell in polynomial time whether a formula has a satisfying assignment that satisfies more than a fraction α of its clauses or whether any assignment to φ satisfies at most a fraction $\alpha - \epsilon_\alpha$ of its clauses, where ϵ_α is a constant that depends only on ϵ_0 and α. (This can be shown by a padding argument. Namely, use enough new variables $y_1, \ldots, y_l$ and add the clauses (y_i) and $(\overline{y_i})$, $1 \leq i \leq l$, to the original formula.) There are no hard gaps at locations $0 \leq \alpha \leq 1/2$ since any formula has an assignment that satisfies at least half of its clauses. We conclude that, in this respect, the proof of the hardness of approximating MAX-SAT is the strongest possible.

We conjecture a generalization of this example. Suppose we have a "natural" optimization problem that seeks the quality of the best solution. We conjecture that showing a hard gap at location $\alpha_0 = 1$ can be used to prove the existence of hard gaps in all other locations (in which they exist).

In previous works, hard gaps are not always shown at location $\alpha_0 = 1$. For example, recall that in MAX k-COLORABILITY, we look for the maximum number of consistent edges in a k-coloring of the input graph. The interesting original problem (for $k \geq 3$) is to tell whether a graph $G(V, E)$ is k-colorable or not. Relaxing the precision requirement, we would like to know if it is easier to tell whether G has a k-coloring, for which all $|E|$ edges in G are consistent, or whether for all k-colorings of G at most $(1-\epsilon)|E|$ edges are consistent. Can this relaxation be determined in polynomial time for any constant ϵ? This question was not considered before. Instead, following the gap-preserving L-reductions used by Papadimitriou & Yannakakis [PY-91], we get that there are constants $0 < \epsilon_0, \alpha_0 < 1$ such that, unless P=NP, it is not possible to tell in polynomial time whether a given graph $G(V, E)$ has a k-coloring with more than $\alpha_0|E|$ consistent edges, or whether for any k-coloring of G at most $(\alpha_0 - \epsilon_0)|E|$ edges are consistent. Using this hard gap we can indeed say that there is no polynomial time approximation scheme for MAX k-COLORABILITY unless P=NP, but the hardness of the interesting gap (i.e., $\alpha_0 = 1$) remains open.

The importance of the gap at location $\alpha_0 = 1$ can also be expressed in terms of the analogous search problem. The implication of the result of [PY-91] is that given a graph G, we cannot find a k-coloring of G that is as close as desired to the optimal solution (unless P=NP). But, suppose we are given a k-colorable graph and we would like to color it such that as many edges as possible are consistent. There is no evidence given in [PY-91] to show that we cannot achieve this task in polynomial time such that the number of consistent edges is greater than $(1-\epsilon)|E|$ for any constant ϵ.

1.2.3 Summary of results

The hardness of finding a k-coloring that has "almost" as many consistent edges as possible, for k-colorable graphs, is implied by our showing a hard gap at location 1 for MAX k-COLORABILITY (for all $k \geq 3$). We thus settle the problem raised in the previous subsection. In our proof, we use a different reduction than the one in [PY-91]). Two other problems that were previously shown hard to approximate using a gap location different from 1 are 3-DIMENSIONAL MATCHING [Kan-91] and NOT-ALL-EQUAL-3SAT [PY-91]. We strengthen these results by showing that these problems indeed possess a hard gap at location 1.

Last, we define new approximation versions of some known NP-complete problems, in the spirit of approximating the quality of the best solution. In particular, we define approximation versions of:

- VERTEX COVER [Kar-72]. Note that the version that was treated by Papadimitriou &

Yannakakis [PY-91] is in the spirit of the largest solution. We treat the version that seeks the quality of the best solution.

- k-EDGE COLORABILITY (CHROMATIC INDEX) [Hoy-81, LG-83]. In this case, it is not hard to approximate the size of the smallest solution since there is a polynomial time algorithm that colors the edges of a graph of degree k with $k+1$ colors (Vizing's Theorem, see [Ber-73]). Again, we treat the version that seeks the quality of the best solution (i.e., a best k edge-coloring of a k-degree graph).

- SET SPLITTING [Lov-73].

We show that all these problems possess a hard gap at gap-location 1.

1.2.4 Summary of the motivation for proving hardness at gap-location 1

To complete the discussion, we summarize the arguments that motivate showing a hard gap at location 1. We provide four motivating arguments, as follows.

1. **The relation to the decision problem.** Recall that the original decision problem was to distinguish between instances for which all subconditions can be satisfied and instances for which not all subconditions can be satisfied. Showing a hard gap at location 1 implies the hardness of the relaxation of the original problem in which we have to distinguish between instances for which all subconditions can be satisfied and instances for which all solutions are "far from" satisfying all subconditions. This implication on the relaxation of the original problem does not follow from hard gaps in other locations. We believe that this implication is fundamental: the approximation task is initially meant to relax the hardness of the original task, therefore, we would expect that the hardness of approximation should imply the hardness of the relaxation of the original task.

2. **The relation to the search problem.** Suppose we are given instances which satisfy the original decision problem (i.e., there exists a solution that satisfies all their subconditions), and we would like to find a solution that satisfies as many subconditions as possible. For example, given a 3-colorable graph G, find a 3-coloring of G with as many as possible consistent edges. As discussed in Section 1.2.2, a hard gap at location

1 implies the NP-hardness of finding a solution which is even "close" to the optimal solution. A hard gap at any location other than 1 does not imply the hardness of this search task.

3. **The hardness of the complementary minimization problem.** Consider the complementary minimization problem in which we try to find the minimum (over all possible solutions) of the number of subconditions that are not satisfied. A hard gap at location 1 for our maximization problem implies that it is NP-hard to approximate this complementary minimization problem to within any ratio. This implication follows from the fact that it is NP-hard to tell between instances having a minimum of 0 subconditions not satisfied and instances having a minimum of a constant fraction of the subconditions not satisfied. This strong implication on the hardness of the complementary problem does not follow from the existence of a hard hard gap at any other location. (Note that this implication follows also from the NP-hardness of the decision problem, however, it is not implied by a hard gap at any location other than 1.)

4. **Strength of the result (given our conjecture).** We conjecture that for natural problems, showing a hard gap at location 1 implies a hard gap at all other possible locations. As will be explained in Section 3.1, the converse is not generally true. Namely, there exist problems having a hard gap at a location other than 1 which do not have a hard gap at location 1 at all. Therefore, given our conjecture, showing a hard gap at location 1 is the strongest possible result in this respect.

1.2.5 Related results

Quadratic equations over the rationals. A related problem possessing a hard gap at location 1 is MAX-QER: given a set of quadratic equations with rational coefficients, find the maximum number of equations satisfied, over all rational assignments to the variables. We include a simple proof of this fact which is due to Bellare & Petrank [BP-92b], Theorem 5.7. The same problem when the field is $GF(p)$ (p a prime) was considered by Håstad, Phillips, and Safra [HPS-93]; they showed that if P $\neq$ NP, then MAX-QE(p) cannot be approximated to within a factor of $\frac{1}{p-(1/poly)}$.

k-coloring. Most problems shown as having a hard gap in this paper have related positive results asserting that it is possible to approximate them to within some constant ratio in polynomial time. A different kind of positive result was given by Alon *et al.* [ADLRY-92]. They showed that for any $\epsilon > 0$, there is a polynomial time algorithm that, given k-colorable graphs having $|E| = \Omega(|V|^2)$, finds a k-coloring in which the number of consistent edges exceeds $(1-\epsilon)|E|$. This does not contradict our impossibility result for the general case and it is interesting to note that the graphs output by our reduction (which shows a hard gap for MAX k-COLORABILITY) have their number of edges linear in their number of vertices. The existence of hard gaps (or approximation schemes) for graphs having $|E|$ neither quadratic nor linear in $|V|$ remains open.

1.3 Organization

The rest of this dissertation goes as follows. In Chapter 2, we present our work on knowledge complexity. In Chapter 3, we present our work on the hardness of approximations.

Chapter 2: In Section 2.1 we present the preliminary definitions we need for this work. We follow in Section 2.2 with presenting a basic tool for the proof of our main result: We build the universal almost uniform generator and show how to use it to build the simulation based prover. In Section 2.3 we prove our main result for the case of perfect knwoledge complexity. In Section 2.4 we relate statistical knowledge complexity to perfect knowledge complexity. In particular, this relation implies the validity of our main result for statistical knowledge complexity. In Section 2.5 we consider interactive proofs with error probability which is not negligible. We show how our techniques yield non-trivial results also with respect to such error probabilities. In the next section (Section 2.6) we explain why the protocol given in [F-89] is not valid, and we explicitly give a counter example on which the protocol fails. In the last section of the chapter on knowledge complexity, (Section 2.7) we make some concluding remarks on the relation of knowledge compelxity to computational complexity and present a few interesting open problems.

Chapter 3: In Section 3.1 we define the gap location parameter, and follow by a discussion of its properties. In Section 3.2 we show a hard gap for k-COLORABILITY at gap location 1. In Section 3.3 we show a hard gap at location 1 for MAX NAE 3SAT and MAX 3DM. Last, in Section 3.4 we present approximation versions of NP-Hard problems, which were not considered before, and discuss the hardness of approximating them.

Chapter 2

Knowledge Complexity and Computational Complexity

2.1 Preliminaries

Let us state some of the definitions and conventions we use in this part of the dissertation. Throughout this work we use n to denote the length of the input x. A function $f : \mathsf{N} \to [0,1]$ is called *negligible* if for every polynomial p and all sufficiently large n's $f(n) < \frac{1}{p(n)}$.

2.1.1 Interactive proofs

Let us recall the concept of interactive proofs, presented by [GMR-89]. For formal definitions and motivating discussions the reader is referred to [GMR-89]. A protocol between a (computationally unbounded) *prover* P and a (probabilistic polynomial-time) *verifier* V constitutes an **interactive proof** for a language L if there exists a negligible function $\epsilon : \mathsf{N} \to [0,1]$ such that

1. **Completeness:** If $x \in L$ then

$$\Pr\left[(P,V)(x) \text{ accepts}\right] \;\geq\; 1 - \epsilon(n)$$

2. **Soundness:** If $x \notin L$ then for any prover P^*

$$\Pr\left[(P^*,V)(x) \text{ accepts}\right] \;\leq\; \epsilon(n)$$

Remark: Usually, the definition of interactive proofs is robust in the sense that setting the error probability to be bounded away from $\frac{1}{2}$ does not change their expressive power, since the error probability can be reduced by repetitions. However, this standard procedure is not applicable when knowledge-complexity is measured, since (even sequential) repetitions may increase the knowledge-complexity. The question is, thus, what is the *right* definition. The definition used above is quite standard and natural; it is certainly less arbitrary then setting the error to be some favorite constant (e.g., $\frac{1}{3}$) or function (e.g., 2^{-n}). Yet, our techniques yield non-trivial results also in case one defines interactive proofs with non-negligible error probability (e.g., constant error probability). For example, languages having interactive proofs with error probability 1/4 and perfect knowledge complexity 1 are also in $\mathcal{BPP}^{\mathcal{NP}}$. For more details see Section 2.5. Also note that we have allowed two-sided error probability; this strengthens our main result but weakens the statistical to perfect transformation[1].

2.1.2 Knowledge Complexity

Throughout the rest of the paper, we refer to knowledge-complexity *with respect to the honest verifier*; namely, the ability to simulate the (honest) verifier's view of its interaction with the prover. (In the stronger definition, one considers the ability to simulate the point of view of *any efficient verifier* while interacting with the prover.)

We let $(P,V)(x)$ denote the random variable that represents V's view of the interaction with P on common input x. The view contains the verifier's random tape as well as the sequence of messages exchanged between the parties.

We begin by briefly recalling the definitions of perfect and statistical zero-knowledge. A protocol (P,V) is *perfect zero-knowledge* (resp., *statistical zero-knowledge*) over a language L if there is a probabilistic polynomial time simulator M such that for every $x \in L$ the random variable $M(x)$ is distributed identically to $(P,V)(x)$ (resp., the statistical difference between $M(x)$ and $(P,V)(x)$ is a negligible function in $|x|$).

Next, we present the definitions of perfect (resp., statistical) knowledge-complexity which we use in the sequel. These definitions extend the definition of perfect (resp., statistical) zero-knowledge, in the sense that knowledge-complexity zero is exactly zero-knowledge. Actually, there are two alternative formulations of knowledge-complexity, called the *oracle version* and the *fraction version*. These formulations coincide at the zero level and differ by at most an

[1]Suppose you had a transformation for the one-sided case. Then, given a two-sided interactive proof of some statistical knowledge complexity you could have transformed it to a one-sided error proof of the same knowledge complexity (cf., [GMS-87]). Applying the transformation for the one-sided case would have yielded an even better result.

additive constant otherwise [GP-91]. For further intuition and motivation see [GP-91]. It will be convenient to use both definitions in this paper[2].

By the *oracle formulation*, the knowledge-complexity of a protocol (P,V) is the number of oracle (bit) queries that are needed to simulate the protocol efficiently.

Definition 2.1 (knowledge complexity — oracle version): *Let* $k\colon \mathsf{N}\to\mathsf{N}$. *We say that an interactive proof* (P,V) *for a language* L *has* perfect (resp., statistical) knowledge complexity $k(n)$ in the oracle sense *if there exists a probabilistic polynomial time oracle machine* M *and an oracle* A *such that:*

1. *On input* $x\in L$*, machine* M *queries the oracle* A *for at most* $k(|x|)$ *bits.*
2. *For each* $x\in L$*, machine* M^A *produces an output with probability at least* $\frac{1}{2}$*, and given that* M^A *halts with an output,* $M^A(x)$ *is* identically distributed (resp., statistically close) *to* $(P,V)(x)$.

In the *fraction formulation*, the simulator is not given any explicit help. Instead, one measures the density of the largest subspace of simulator's executions (i.e., coins) which is identical (resp., close) to the (P,V) distribution.

Definition 2.2 (knowledge complexity — fraction version): *Let* $\rho\colon \mathsf{N}\to(0,1]$. *We say that an interactive proof* (P,V) *for a language* L *has* perfect (resp., statistical) knowledge-complexity $\log_2(1/\rho(n))$ in the fraction sense *if there exists a probabilistic polynomial-time machine* M *with the following "good subspace" property. For any* $x\in L$ *there is a subset of* M*'s possible random tapes* S_x*, such that:*

1. *The set* S_x *contains at least a* $\rho(|x|)$ *fraction of the set of all possible coin tosses of* $M(x)$.
2. *Conditioned on the event that* $M(x)$*'s coins fall in* S_x*, the random variable* $M(x)$ *is* identically distributed (resp., statistically close) *to* $(P,V)(x)$. *Namely, for the perfect case this means that for every* $\bar{c}$

$$\mathrm{Prob}(M(x,\omega)=\bar{c}\,|\,\omega\in S_x) = \mathrm{Prob}((P,V)(x)=\bar{c})$$

where $M(x,\omega)$ *denotes the output of the simulator* M *on input* x *and coin tosses sequence* ω.

[2]The analysis of the [BP-92a] procedure is easier when using the fraction version, whereas the transformation from statistical to perfect is easier when using the oracle version.

As mentioned above, these two measures are almost equal.

Theorem [GP-91]: *The fraction measure and the oracle measure are equal up to an additive constant.*

Since none of our results is sensitive to a difference of an additive constant in the measure, we ignore this difference in the subsequent definition as well as in the statement of our results.

Definition 2.3 (knowledge complexity classes):

- $\mathcal{PKC}(k(\cdot))$ = *languages having interactive proofs of perfect knowledge complexity* $k(\cdot)$.
- $\mathcal{SKC}(k(\cdot))$ = *languages having interactive proofs of statistical knowledge complexity* $k(\cdot)$.

2.1.3 The simulation based prover

An important ingredient in our proof is the notion of a simulation based prover, introduced by Fortnow [F-89]. Consider a simulator M that outputs conversations of an interaction between a prover P and a verifier V. We define a new prover P^*, called *the simulation based prover*, which selects its messages according to the conditional probabilities induced by the simulation. Namely, on a partial history h of a conversation, P^* outputs a message α with probability

$$\text{Prob}(P^*(h) = \alpha) \stackrel{\text{def}}{=} \text{Prob}(M_{|h|+1} = h \circ \alpha \mid M_{|h|} = h)$$

where M_t denotes the distribution induced by M on t-long prefixes of conversations. (Here, the length of a prefix means the number of messages in it.)

It is important to note that the behavior of P^* is *not* necessarily close to the behavior of the original prover P. Specifically, if the knowledge complexity is greater than 0 and we consider the simulator guaranteed by the fraction definition, then P^* and P might have quite a different behavior. One of our main objectives will be to show that even in this case P^* still behaves in a manner from which we can benefit. Another main objective will be to show that this entity can be implemented by an efficient machine that has access to an NP oracle. This objective is addressed in the following section.

2.2 Universal Almost Uniform Generation

In this section, we present an important tool in the proof of our main result: the universal almost uniform generator. We believe that this generator is useful also in other settings, and in fact, it has already been used in papers that followed this work (see [BFK-95, ABV-95]). In this section, we show an efficient implementation of the universal almost uniform generator.

2.2.1 Definition and history

The generator samples "almost uniformly" from a given set $S \in \mathrm{NP}$. The generator is universal in the sense that it is not built for a specific set S, but is given a description of S in the input. The description of S is given to the sampler as an encoding $\langle M \rangle$ of a non-deterministic machine M that recognizes S. Alternatively, M is deterministic and has two inputs (r, w) and the set S is equal to:

$$\{r : \exists w \text{ s.t. } M \text{ accepts the input } (r, w)\}.$$

For any set S in NP there exists a machine M that runs in time polynomial in $|r|$ and corresponds to S in the above manner. We denote $M_n \stackrel{\mathrm{def}}{=} S \cap \{0,1\}^n$ and we denote by $\mathcal{U}[B]$ the uniform distribution over a finite set B. We use M_n for the notation rather than S_n as we are going to regard the set S in the sequel only by considering the machine M that recognizes it.

Definition 2.4 *A* universal almost uniform generator *is a (probabilistic) machine A which has the property that $A(1^n, \delta, \langle M \rangle)$ and $\mathcal{U}[M_n]$ are δ-close (in norm-1), for all $n \in \mathsf{N}, \delta > 0$ and Turing machines M.*

2.2.2 Our result and an overview of the proof

Our result is the following.

Theorem 2.5 *There is a probabilistic, oracle machine U such that U^{NP} is a universal almost uniform generator, and moreover, the running time of U^{NP} on inputs $1^n, \delta, \langle M \rangle$ is polynomial in $n, \lg \delta^{-1}$ and $T_M(n)$.*

Here $T_M(n)$ denotes the maximum, over all $r \in \{0,1\}^n$, $w \in \{0,1\}^*$, of the running time of M on input (r, w).

A very important property of the procedure of Theorem 2.5 is that its running time is polynomial in $\lg \delta^{-1}$ rather than polynomial in δ^{-1}, so that in time polynomial in n we can achieve exponentially small (in n) error.

We remark that even the special case in which we sample a set $S \in \mathrm{P}$ is important, and in fact, we use Theorem 2.5 throughout this paper only for sets which are decidable in polynomial time.

The proof of Theorem 2.5 is derived by combining techniques from [JVV-86, St-83, Si-83]. First, we build an approximate counter. The approximate counter of Stockmeyer [St-83] took polynomial time with a Σ_2^{P} oracle. We begin by relaxing the requirement on the approximate counter: we allow it to fail in approximating to within the required bounds with some small probability. We show that this relaxed version of approximate counting can be implemented in probabilistic polynomial time with an NP oracle. Our construction uses hashing techniques similar to the ones used by Sipser [Si-83]. Once we have this approximate counter, we use the probabilistic polynomial time reduction of Jerrum, Valiant and Vazirani [JVV-86] from uniform generation to approximate counting. The key observation now is the strength of the [JVV-86] reduction: it is capable of sampling distributions within an exponentially small distance from the uniform distribution, when given a primitive for estimating set sizes (namely approximate counting) that is only accurate to within the reciprocal of a polynomial. The reason that we do not achieve exactly the uniform distribution is that the approximate counter fails with some (very small, but greater than 0) probability. The details follow.

2.2.3 The construction

We use universal hash functions [CW-79]. Let $H_{k,b}$ denote the set of all affine transformations $h_{A,\alpha} : \{0,1\}^k \to \{0,1\}^b$ given by $h_{A,\alpha}(y) = Ay + \alpha$ where A is a b-by-k matrix over GF[2] and α is a b-vector over GF[2], and the arithmetic is over GF[2]. The following lemma follows from Lemma 4.1 in Aiello-Håstad [AH-87], which in turn is based on ideas of Sipser [Si-83] and Babai [Bab-85].

Lemma 2.6 *Let b, k be positive integers, $\delta > 0$ and $S \subseteq \{0,1\}^k$. Let $l = 16\lg(2\delta^{-1})$. Select uniformly and independently hash functions $h_1, \ldots, h_{2l} \in H_{k,b}$ and set $z \stackrel{\mathrm{def}}{=} 0^b$. Let the random variable X equal the number of indices i for which $h_i^{-1}(z) \cap S \neq \emptyset$. Then the probability that $X > l$ is*

(1) $\geq 1 - \delta$ *if* $|S| \geq 4 \cdot 2^b$

(2) $\leq \delta$ *if* $|S| \leq \frac{1}{4} \cdot 2^b$.

We remark that the choice $z = 0^b$ is arbitrary and selecting z to be any constant string in $\{0,1\}^b$ will do as well. The notion of approximate counting to within a factor $(1 + \epsilon)$ with success probability at most $(1-\delta)$ is as follows (cf. [St-83]).

Definition 2.7 *A* universal approximate counter *is a (probabilistic) machine C which, on inputs $1^n, \epsilon > 0, \delta > 0, \langle M \rangle$ outputs an estimate v which, with probability $\geq 1-\delta$, satisfies $\frac{1}{1+\epsilon} \cdot |M_n| \leq v \leq (1+\epsilon) \cdot |M_n|$.*

Let us now establish the result on approximate counting that we need.

Theorem 2.8 *There exists a probabilistic, oracle machine C such that C^{NP} is a universal approximate counter, and moreover, the running time of C^{NP} on inputs $1^n, \epsilon, \delta, \langle M \rangle$ is polynomial in $n, \epsilon^{-1}, \lg \delta^{-1}$ and $T_M(n)$.*

Proof: The algorithm C is given in Figure 2.1. The idea behind it is as follows.

We first look at Lemma 2.6 and derive an algorithm to approximate a set S (given as $\langle M \rangle$) to within a constant factor with success probability of $1-\delta$. We note that the computational bottle neck in running this algorithm is having to decide, given z, h, and $\langle M \rangle$, whether $h_i^{-1}(z) \cap S \neq \emptyset$. A probabilistic polynomial time machine with an NP oracle can decide whether $h_i^{-1}(z) \cap S \neq \emptyset$ since an NP oracle can determine whether there exists a pair (r, w) for which $h(r) = z$ and M accepts (r, w). All other steps in the algorithm can be performed in polynomial time.

Using the setting of Lemma 2.6 the approximation algorithm is as follows. We enumerate over all possible $b = 1, 2, \ldots, n$, and select the first b for which the experiment defined in Lemma 2.6 fails. Namely, running the experiment with $b-1$ yielded the result $X > l$ whereas running it with b yielded $X \leq l$. We conclude that with probability at least $1 - n \cdot \delta$ it holds that

$$\frac{2^{b-1}}{4} \leq |S| \leq 4 \cdot 2^b. \tag{2.1}$$

This is true since we make at most n experiments and the error in each of them is at most δ.

Last, we have to improve the quality of the approximation. To this end, we define a machine N which runs M on t inputs and accepts iff M accepts all the t inputs. The set defined by the machine N is S^t which is also a set in NP. To be done, we note that approximating the size of S^t to within a factor of 8 implies an approximation of the size of S to within $8^{1/t}$. Thus, we can get an approximation for the size of S to within a factor of $1+\epsilon$ by setting t to be polynomial in ϵ and applying our constant-factor approximation on S^t (or on the machine N). ■

$C^{NP}(1^n, \epsilon, \delta, \langle M \rangle)$

Choose t such that $8^{1/t} \leq 1 + \epsilon$

Comment t will be polynomial in ϵ^{-1}

Let $k = nt$ and let $N = M^t$ be the machine defined by $N(y_1, \ldots, y_t) = 1$ iff $M(y_1) = \cdots = M(y_t) = 1$ and all the y_i have the same length.

Comment $N_k = M_n^t$

$b \leftarrow 0$; $l \leftarrow 16 \lg(2k\delta^{-1})$

`repeat`

  $b \leftarrow b + 1$

  Pick at random $h_1, \ldots, h_{2l} \in H_{k,b}$ and set $z = 0^b$. Use the NP oracle to determine, for each i, whether or not $h_i^{-1}(z) \cap N_k = \emptyset$, and let X be the number of indices i for which $h_i^{-1}(z) \cap N_k \neq \emptyset$.

`until` $(X \leq l$ or $b = k)$

Comment $\Pr[2^{b-1}/4 < |N_k| < 4 \cdot 2^b] \geq 1 - \delta$

Let $\alpha = 2^b$.

Comment $\Pr[|N_k|/4 < \alpha < 8|N_k|] \geq 1 - \delta$

Output $\alpha^{1/t}$

Comment $\Pr[|M_n|/(1+\epsilon) < \alpha < |M_n|(1+\epsilon)] \geq 1 - \delta$

Figure 2.1: Universal Approximate Counting in probabilistic polynomial time with an NP Oracle

Notice that the estimates from the above theorem are only accurate to within the reciprocal of a polynomial (this is since the running time of the algorithm is polynomial in ϵ^{-1}). We now use the reduction of [JVV-86] which is strong enough in this sense. Given an approximation to within a factor of $1+\epsilon$ (for an ϵ which is a reciprocal of a polynomial) it samples a distribution which is very close to the uniform distribution. Specifically, they show the following:

Theorem 2.9 [JVV-86]: *If there exists a universal approximate counter to within a factor of* $(1+\epsilon)$ *for any* $0 < \epsilon < 1$ *(that never fails) then there exists a universal (exact) uniform generator which, when given access to the approximate counter as an oracle, runs in polynomial time in* n *and* $\frac{1}{\epsilon}$.

For further reference, let $Q(n, \frac{1}{\epsilon})$ be a polynomial which bounds the number of oracle queries that the uniform generator (guaranteed in Theorem 2.9) makes to the approximate counter. We are going to run this generator with our universal approximate counter. The problem is that the generator expects an approximation procedure which is always correct, while our procedure fails to approximate the given set (to within $1+\epsilon$) with positive probability (i.e., bounded by δ).

We partition the analysis (of running the generator with our approximator) into two cases: In one case, our approximate counter succeeds in all $Q(n, \frac{1}{\epsilon})$ queries. This happens with probability at least $1 - Q(n, \frac{1}{\epsilon}) \cdot \delta$, and in this case, the generator outputs a uniform element in the set as required. The other possible case, is that the approximate counter makes an error in one of the queries. In this case, it is not possible to tell which distribution is output by the generator. However, since the later case happens with probability at most $Q(n, \frac{1}{\epsilon}) \cdot \delta$ we get that the distance between the the output distribution of the generator and the uniform distribution on the set M_n is at most $Q(n, \frac{1}{\epsilon}) \cdot \delta$. Since the running time of the approximation counter is polynomial in the logarithm of $\frac{1}{\delta}$, then the polynomial multiplicative factor $Q(n, \frac{1}{\epsilon})$, has no significant effect on the complexity of the procedure. Setting $\delta' \stackrel{\text{def}}{=} \frac{\delta}{Q(n, \frac{1}{\epsilon})}$ we get:

Corollary 2.10 *If there exists a universal approximate counter to within a factor of* $(1+\epsilon)$, C*, which runs in time polynomial in* $n, 1/\epsilon, \log \frac{1}{\delta'}$ *and* $T_M(n)$*, and which fails with probability at most* δ'*, then there exists a universal almost uniform generator,* U*, which uses* C *as an oracle and which runs in time polynomial in* $n, T_M(n)$ *and* $\log \frac{1}{\delta'}$.

Combining Corollary 2.10 with Theorem 2.8, we get Theorem 2.5 and we are done.

2.2.4 Application: An Efficient Implementation of the Simulation Based Prover

A basic ingredient in our proof is the construction of a new prover based on the simulator for the protocol, called a simulation based prover. We use the almost uniform generator to implement it efficiently. The simulation based prover was first used by Fortnow [F-89], and it was used in many works since [AH-87, BMO-90, Os-91].

Definition 2.11 *Let S be a simulator of a g rounds interactive proof with message length l. Let the output of S on input x and random tape r, $S(x,r)$, be $(R, \alpha_1\beta_1 \ldots \alpha_g\beta_g)$, where the α's are the messages of the verifier, the β's are the messages of the prover, and R is the random tape of the verifier. Then for each $t = 1, \ldots, g$ we define $S_t(x,r) = \alpha_1\beta_1 \ldots \alpha_{t-1}\beta_{t-1}\alpha_t$.*

Definition 2.12 *Let S be a simulator (of g rounds and message length l), and let p be the number of its coin tosses. We denote by P^* the probabilistic function whose output (next message to verifier), on input x (common input), and $\alpha_1\beta_1 \ldots \alpha_t$ (conversation so far), is given by the following experiment:*

(1) *Pick at random a string r from the set $\{ r \in \{0,1\}^{p(n)} : S_t(x,r) = \alpha_1\beta_1 \ldots \alpha_t \}$ (the set of random tapes of the simulator on which the simulator outputs a conversation having $\alpha_1\beta_1 \ldots \alpha_t$ as a prefix).*

(2) *Then compute $S(x,r)$ — this has the form*

$$(R, \alpha_1\beta_1 \ldots \alpha_t\beta'_t \ldots \alpha'_g\beta'_g)$$

for some $\beta'_t, \ldots, \alpha'_g, \beta'_g \in \{0,1\}^l$

(3) *Finally, output the string β'_t.*

We call P^* the simulation based prover *for the simulator S.*

Intuitively, if S is a (statistical) ZK P-simulator for V, then P^* is attempting to find P's replies in the interaction of P with V. He does this by imitating the behavior of the simulator S in the prover steps of the interaction. The properties of P^* in case the knowledge complexity is greater than zero, will be studied in Chapter 2.3.

We will denote the output of P^* on input x and $\alpha_1\beta_1 \ldots \alpha_t$ by $P^*(x, \alpha_1\beta_1 \ldots \alpha_t)$, but keep in mind the function is probabilistic so that $P^*(x, \alpha_1\beta_1 \ldots \alpha_t)$ is actually a distribution on l bit strings.

The complexity of computing P^* is the question we address next.

Exact computation of P^* is clearly possible in probabilistic polynomial space, and this is easily improved to probabilistic polynomial time with a Σ_2^{P} oracle by using the results of [JVV-86] on uniform generation.

Let us now turn to approximations: we will be interested in computing a distribution which is δ-close to P^*. If $\delta = n^{-c}$ for some constant c then, following [Os-91], this can be done in probabilistic polynomial time with an NP oracle by using techniques from [ILL-89]. That result will not, however, suffice for the applications in this paper: we need to be able to achieve values of δ which are exponentially small (in n). The following Theorem says we can do this, still in probabilistic polynomial time with an NP oracle.

Theorem 2.13 *Let S be a simulator (of g rounds and message length l). Then there exists a probabilistic oracle machine T with the following property. Suppose $x \in \{0,1\}^n$, $\alpha_1, \beta_1, \ldots, \alpha_t$ are $l(n)$ bit strings ($t \leq g$), and $\delta > 0$. Then the probability spaces $T^{NP}(x, \alpha_1\beta_1 \ldots \alpha_t, \delta)$ and $P^*(x, \alpha_1\beta_1 \ldots \alpha_t)$ are δ-close. Moreover, T runs in time polynomial in n and $\lg \delta^{-1}$.*
Note the running time of T is polynomial in $\lg \delta^{-1}$ (rather than δ^{-1}), which is why we can achieve exponentially small error.

Proof: The proof of this theorem is as follows (given the implementation of *universal almost uniform generation* in probabilistic polynomial time with an NP oracle, i.e., Theorem 2.5): follow the algorithm of Definition 2.12, using Theorem 2.5 to implement the first step. However, there is a technicality here that must be addressed. The set in which we would like to sample in is

$$A_{x,\alpha_1\beta_1\ldots\alpha_t} = \{\, r \in \{0,1\}^{p(n)} : S_t(x,r) = \alpha_1\beta_1 \ldots \alpha_t \,\}$$

This finite set doesn't seem to have a natural characterization as an intersection of $\{0,1\}^n$ with a language that some machine accepts, unless one considers a machine that accepts only this finite set. Of course a machine that recognizes a finite set can be made very efficient in the length of its inputs, but we should require more.

We need to have a fixed polynomial $P(\cdot)$ such that each such machine we build and for any r, x, and $\alpha_1\beta_1 \ldots \alpha_t$, the machine runs in time bounded by $P(|x|)$. Furthermore, since the running time of the uniform generator depends on the length of the encoding, we also demand that the length of the encoding of $M_{x,\alpha_1\beta_1\ldots\alpha_t}$ is also bounded by $P(|x|)$. Note that it is crucial that our machine will have x and $\alpha_1\beta_1 \ldots \alpha_t$ fixed within its description and not given as inputs, since we do not want to sample x nor $\alpha_1\beta_1 \ldots \alpha_t$. We only want to sample an appropriate r.

Consider first a machine M that on input $(x, r, \alpha_1\beta_1 \ldots \alpha_t)$ decides whether the simulator S on input x and random tape r outputs a conversation which has $\alpha_1\beta_1 \ldots \alpha_t$ as a prefix. This machine has a fixed encoding $< m >$ and it runs in polynomial time in $|x|$. Now consider a machine $M'_{x,\alpha_1\beta_1\ldots\alpha_t}$ which on input r runs M on $(x, r, \alpha_1\beta_1 \ldots \alpha_t)$. Machine M' is the machine we would like to give to our uniform generator and it can easily be implemented such that its running time is polynomial in $|x|$ (which is a constant time with respect to M'). We would like to claim that it is possible to encode such a machine M' by an encoding which is also of size polynomial in $|x|$. A possible implementation of M' is a machine which writes $(< M >, x, r, \alpha_1\beta_1 \ldots \alpha_t)$ on its work tape (where $< M >, x$, and $\alpha_1\beta_1 \ldots \alpha_t$ are constant strings and r is copied from the input tape). Then, machine M' runs the universal Turing machine on its work tape. Such a machine M' has an encoding which is polynomial in $|x|$.

To summarize this technical discussion, our simulation based prover runs in time polynomial in $|x|$ since it feeds the uniform generator with inputs that make it run in time polynomial in $|x|$ (given access to an NP-oracle), and we are done. ■

2.3 The Proof of the Main Result for the Perfect Case

In this section we prove that the Main Theorem holds for the special case of *perfect* knowledge complexity. Combining this result with the transformation (Theorem 2.17) of the subsequent section, we get the Main Theorem.

Theorem 2.14 $\mathcal{PKC}(O(\log n)) \subseteq \mathcal{BPP}^{\mathcal{NP}}$

Suppose that (P, V) is an interactive proof of perfect knowledge complexity $k(\cdot) = O(\log n)$ for the languages L, and let M be the simulator guaranteed by the fraction formulation (i.e., Definition 2.2). We consider the conversations of the original verifier V with the simulation-based-prover P^* (see definition in Section 2.1.3). We are going to show that the probability that the interaction (P^*, V) is accepting is negligible if $x \notin L$ and greater than a polynomial fraction if $x \in L$. This separation between the cases $x \notin L$ and $x \in L$ can be amplified by sequential repetitions of the protocol (P^*, V). So it remains to observe that we can sample the (P^*, V) interactions in probabilistic polynomial-time having access to an NP oracle. This observation is justified as follows. Clearly, V's part of the interaction can be produced in polynomial-time. By Theorem 2.13, we can implement P^* by a probabilistic polynomial time machine that has access to an NP oracle. Actually, the implementation deviates from the simulation based prover by an exponentially small fraction, but this does not matter. Thus, it remains only to prove the following lemma.

Lemma 2.15

1. *If $x \in L$ then the probability that (P^*, V) outputs an accepting conversation is at least $\frac{1}{2} \cdot 2^{-k}$.*
2. *If $x \notin L$ then the probability that (P^*, V) outputs an accepting conversation is negligible.*

proof: The second part of the lemma follows from the soundness property as before. We thus concentrate on the first part. We fix an arbitrary $x \in L$ for the rest of the proof and allow ourselves not to mention it in the sequel discussion and notation. Let $k = k(|x|)$ and q be the number of coin tosses made by M. We denote by $\Omega \stackrel{\text{def}}{=} \{0,1\}^q$ the set of all possible coin tosses, and by S the "good subspace" of M (i.e., S has density 2^{-k} in Ω and for ω chosen uniformly in S the simulator outputs exactly the distribution of the interaction (P, V)).

Consider the conversations that are output by the simulator on $\omega \in S$. The probability to get such a conversation when the simulator is run on ω uniformly selected in Ω, is at least 2^{-k}. We claim that the probability to get these conversations in the interaction (P^*, V) is also at least 2^{-k}. This is not obvious, since the distribution produced by (P^*, V) may not be identical to the distribution produced by M on a uniformly selected $\omega \in \Omega$. Nor is it necessarily identical to the distribution produced by M on a uniformly selected $\omega \in S$. However, the prover's moves in (P^*, V) are distributed as in the case that the simulator selects ω uniformly in Ω, whereas the verifier's moves (in (P^*, V)) are distributed as in the case that the simulator selects ω uniformly in S. Thus, it should not be too surprising that the above claim can be proven.

However, we need more than the above claim: It is not enough that the (P^*, V) conversations have an origin in S, they must be *accepting* as well. (Note that this is not obvious since M simulates an interactive proof that may have two-sided error.) Again, the density of the accepting conversations in the "good subspace" of M is high (i.e., $\geq 1 - \epsilon$), yet we need to show that this is the case also for the (P^*, V) interaction. Actually, we will show that the probability than an (P^*, V) conversation is accepting and "has an origin" in S is at least $\frac{1}{2} \cdot 2^{-k}$.

Let us begin the formal argument with some notations. For each possible history of the interaction, h, we define subsets of the random tapes of the simulator (i.e., subsets of Ω) as follows. Ω_h is the set of $\omega \in \Omega$ which cause the simulator to output a conversation with prefix h. S_h is the subset of ω's in Ω_h which are also in S. A_h is the set of ω's in S_h which are also accepting.

Thus, letting $M_t(\omega)$ denote the t-message long prefix output by the simulator M on coins ω, we get

$$\begin{aligned}
\Omega_h &\stackrel{\text{def}}{=} \{\omega : M_{|h|}(\omega) = h\} \\
S_h &\stackrel{\text{def}}{=} \Omega_h \cap S \\
A_h &\stackrel{\text{def}}{=} \{\omega \in S_h : M(\omega) \text{ is accepting}\}
\end{aligned}$$

Let C be a random variable representing the (P^*, V) interaction, and χ be an indicator so that $\chi(\bar{c}) = 1$ if the conversation $\bar{c}$ is accepting and $\chi(\bar{c}) = 0$ otherwise. Our aim is to prove that $\text{Prob}(\chi(C) = 1) \geq \frac{1}{2} \cdot 2^{-k}$. Note that

$$\begin{aligned}
\text{Prob}(\chi(C) = 1) &= \sum_{\bar{c}} \text{Prob}(C = \bar{c}) \cdot \chi(\bar{c}) \\
&\geq \sum_{\bar{c}} \text{Prob}(C = \bar{c}) \cdot \frac{|A_{\bar{c}}|}{|\Omega_{\bar{c}}|}
\end{aligned}$$

The above expression is exactly the expectation value of $\frac{|A_c|}{|\Omega_c|}$. Thus, we need to show that:

$$\text{Exp}_{\bar{c}}\left(\frac{|A_{\bar{c}}|}{|\Omega_{\bar{c}}|}\right) > \frac{1}{2} \cdot 2^{-k} \tag{2.2}$$

where the expectation is over the possible conversations $\bar{c}$ as produced by the interaction (P^*, V). Once Equation (2.2) is proven, we are done. Denote the empty history by λ. To prove Equation (2.2) it suffices to prove that

$$\text{Exp}_{\bar{c}}\left(\frac{|A_{\bar{c}}|}{|\Omega_{\bar{c}}|} \cdot \frac{|A_{\bar{c}}|}{|S_{\bar{c}}|}\right) \geq \frac{|A_\lambda|}{|\Omega_\lambda|} \cdot \frac{|A_\lambda|}{|S_\lambda|} \tag{2.3}$$

since using $\frac{|A_\lambda|}{|S_\lambda|} > \sqrt{\frac{1}{2}}$ and $\frac{|S_\lambda|}{|\Omega_\lambda|} \geq 2^{-k}$, we get

$$\begin{aligned}
\text{Exp}_{\bar{c}}\left(\frac{|A_{\bar{c}}|}{|\Omega_{\bar{c}}|}\right) &\geq \frac{|A_\lambda|}{|\Omega_\lambda|} \cdot \frac{|A_\lambda|}{|S_\lambda|} \\
&= \left(\frac{|A_\lambda|}{|S_\lambda|}\right)^2 \cdot \frac{|S_\lambda|}{|\Omega_\lambda|} \\
&\geq \frac{1}{2} \cdot 2^{-k}
\end{aligned}$$

The proof of Equation (2.3) is by induction on the number of rounds. Namely, for each round i, we show that the expected value of $\frac{|A_h|}{|\Omega_h|} \cdot \frac{|A_h|}{|S_h|}$ over all possible histories h of i rounds (i.e., length i) is greater or equal to the expected value of this expression over all histories h' of $i-1$ rounds. In order to show the induction step we consider two cases:

1. the current step is by the prover (i.e., P^*); and
2. the current step is by the verifier (i.e., V).

In both cases we show, for any history h,

$$\mathrm{Exp}_m\left(\frac{|A_{hom}|}{|\Omega_{hom}|} \cdot \frac{|A_{hom}|}{|S_{hom}|}\right) \geq \frac{|A_h|}{|\Omega_h|} \cdot \frac{|A_h|}{|S_h|} \tag{2.4}$$

where the expectation is over the possible current moves m, given history h, as produced by the interaction (P^*, V).

Technical Claim

The following technical claim is used for deriving the inequalities in both cases.

Claim 2.16 *Let x_i, y_i, $1 \leq i \leq n$ be positive reals. Then,*

$$\sum_{i=1}^{n} \frac{{x_i}^2}{y_i} \;\geq\; \frac{(\sum_{i=1}^{n} x_i)^2}{\sum_{i=1}^{n} y_i}$$

Proof: The Cauchy-Schwartz Inequality asserts:

$$\left(\sum_{i=1}^{n} {a_i}^2\right) \cdot \left(\sum_{i=1}^{n} {b_i}^2\right) \;\geq\; \left(\sum_{i=1}^{n} a_i \cdot b_i\right)^2$$

Setting $a_i \stackrel{\text{def}}{=} \sqrt{y_i}$ (we can do this since y_i is positive) and $b_i \stackrel{\text{def}}{=} \frac{x_i}{a_i}$, and rearranging the terms, we get the desired inequality. □

Prover Step – denoted α

Given history h, the prover P^* sends α as its next message with probability $\frac{|\Omega_{ho\alpha}|}{|\Omega_h|}$. Thus,

$$\begin{aligned}
\mathrm{Exp}_\alpha\left(\frac{|A_{ho\alpha}|}{|\Omega_{ho\alpha}|} \cdot \frac{|A_{ho\alpha}|}{|S_{ho\alpha}|}\right) &= \sum_\alpha \frac{|\Omega_{ho\alpha}|}{|\Omega_h|} \cdot \frac{|A_{ho\alpha}|}{|\Omega_{ho\alpha}|} \cdot \frac{|A_{ho\alpha}|}{|S_{ho\alpha}|} \\
&= \frac{1}{|\Omega_h|} \cdot \sum_\alpha \frac{|A_{ho\alpha}|^2}{|S_{ho\alpha}|} \\
&\geq \frac{|A_h|}{|\Omega_h|} \cdot \frac{|A_h|}{|S_h|}
\end{aligned}$$

The inequality is justified by using the Technical Claim and noting that $\sum_\alpha |A_{ho\alpha}| = |A_h|$ and $\sum_\alpha |S_{ho\alpha}| = |S_h|$.

Verifier Step – denoted β

By the perfectness of the simulation, when restricted to the good subspace S, we know that given history h, the verifier V sends β as its next message with probability $\frac{|S_{h\circ\beta}|}{|S_h|}$. Thus,

$$\begin{aligned}
\mathrm{Exp}_\beta\left(\frac{|A_{h\circ\beta}|}{|\Omega_{h\circ\beta}|}\cdot\frac{|A_{h\circ\beta}|}{|S_{h\circ\beta}|}\right) &= \sum_\beta \frac{|S_{h\circ\beta}|}{|S_h|}\cdot\frac{|A_{h\circ\beta}|}{|\Omega_{h\circ\beta}|}\cdot\frac{|A_{h\circ\beta}|}{|S_{h\circ\beta}|} \\
&= \frac{1}{|S_h|}\cdot\sum_\beta \frac{|A_{h\circ\beta}|^2}{|\Omega_{h\circ\beta}|} \\
&\geq \frac{|A_h|}{|\Omega_h|}\cdot\frac{|A_h|}{|S_h|}
\end{aligned}$$

The inequality is justified by using the Technical Claim and noting that $\sum_\beta |A_{h\circ\beta}| = |A_h|$ and $\sum_\beta |\Omega_{h\circ\beta}| = |\Omega_h|$.

Having proven Equation (2.4) for both cases, Equation (2.3) follows and so does the lemma. $\square$

2.4 Relating Statistical to Perfect Knowledge Compelxity

In this section we show how to transform *statistical* knowledge complexity into *perfect* knowledge complexity, incurring only a logarithmic additive term. This transformation combined with Theorem 2.14 yields the Main Theorem.

Theorem 2.17 *For every (poly-time computable) $k : \mathsf{N} \mapsto \mathsf{N}$,*

$$\mathcal{SKC}\,(k(\cdot)) \subseteq \mathcal{PKC}\,(k(\cdot) + O(\log(\cdot)))$$

We stress again that these knowledge complexity classes refer to the honest verifier and that we don't know whether such a result holds for the analogous knowledge complexity classes referring to arbitrary (poly-time) verifiers.

proof: Here we use the oracle formulation of knowledge complexity (see Definition 2.1). We start with an overview of the proof. Suppose we are given a simulator M which produces output that is *statistically close* to the real prover-verifier interaction. We change both the interactive proof and its simulation so that they produce exactly the same distribution space. We will take advantage of the fact that the prover in the interactive proof and the oracle that "assists" the simulator are both infinitely powerful. Thus, the modification to the prover's program and the augmentation to the oracle need not be efficiently computable. We stress that the modification to the simulator itself will be efficiently computable. Also, we maintain the original verifier (of the interactive proof), and thus the resulting interactive proof is still sound. Furthermore, the resulting interaction will be statistically close to the original one (on any $x \in L$) and therefore the completeness property of the original interactive proof is maintained (although the error probability here may increase by a negligible amount).

Preliminaries

Let $L \in \mathcal{SKC}(k(\cdot))$, and (P, V) be the guaranteed interactive proof. Without loss of generality, we may assume that all messages are of length 1. This message-length convention is merely a matter of encoding.

Recall that Definition 2.1 only guarantees that the simulator produces output with probability $\geq \frac{1}{2}$. Yet, employing Proposition 3.8 in [GP-91], we get that there exists an oracle machine M, that after asking $k(n) + 2\log\log n$ queries, *always* produces an output so that the output is statistically close to the interaction of (P, V). Let A denote the associated oracle, and let $M' \stackrel{\text{def}}{=} M^A$ and P' and V' be the simulation-based prover and verifier[3] induced by M' (i.e., $(P', V') = M'$).

In the rest of the presentation, we fix a generic input $x \in L$ and omit it from the notation.

notations: Let $[A, B]_i$ be a random variable representing the i-message (i-bit) long prefix of the interaction between A and B (the common input x is implicit in the notation). We denote by $A(h)$ the random variable representing the message sent by A after interaction-history h. Thus, if the i^{th} message is sent by A, we can write $[A, B]_{i-1} \circ A([A, B]_{i-1}) = [A, B]_i$. By $X \stackrel{\text{s}}{=} Y$ we denote the fact that the random variables X and Y are statistically close.

Using these notations we may write for every $h \in \{0, 1\}^i$ and $\sigma \in \{0, 1\}$:

$$\text{Prob}(P'(h) = \sigma) \;=\; \text{Prob}\left([M']_{i+1} = h \circ \sigma | [M']_i = h\right)$$

and similarly,

$$\text{Prob}(V'(h) = \sigma) \;=\; \text{Prob}\left([M']_{i+1} = h \circ \sigma | [M']_i = h\right).$$

[3]A simulator-based verifier is defined analogously to the simulator-based prover. It is a fictitious entity which does not necessarily coincide with V.

Claim 2.18 *The distribution induced by (P',V) is statistically close to the distributions induced by both $M'=(P',V')$ and (P,V).*

proof: By definition, the distributions produced by $M'=(P',V')$ and (P,V) are statistically close. Thus, we have

$$[P,V]_i \stackrel{\mathrm{s}}{=} [P',V']_i, \quad \text{for every } i \tag{2.5}$$

We prove that $[P',V]$ is statistically close to $[P',V']$ by induction on the length of the interaction. Assuming that $[P',V]_i \stackrel{\mathrm{s}}{=} [P',V']_i$, we wish to prove it for $i+1$. We distinguish two cases. In case the $i+1^{\mathrm{st}}$ move is by the prover, we get

$$\begin{aligned} [P',V]_{i+1} &= [P',V]_i \circ P'([P',V]_i) \\ &\stackrel{\mathrm{s}}{=} [P',V']_i \circ P'([P',V']_i) \\ &= [P',V']_{i+1} \end{aligned}$$

(use the induction hypothesis for $\stackrel{\mathrm{s}}{=}$). In case the $i+1^{\mathrm{st}}$ move is by the verifier, we get

$$\begin{aligned} [P',V]_{i+1} &= [P',V]_i \circ V([P',V]_i) \\ &\stackrel{\mathrm{s}}{=} [P',V']_i \circ V([P',V']_i) \\ &\stackrel{\mathrm{s}}{=} [P,V]_i \circ V([P,V]_i) \\ &= [P,V]_{i+1} \\ &\stackrel{\mathrm{s}}{=} [P',V']_{i+1} \end{aligned}$$

where the first $\stackrel{\mathrm{s}}{=}$ is justified by the induction hypothesis and the two others by Eq. (2.5). We stress that since the induction hypothesis is used only once in the induction step, the statistical distance is linear in the number of induction steps (rather than exponential). □

Motivating discussion: Note that the statistical difference between the interaction (P',V) and the simulation $M'=(P',V')$ is due solely to the difference between the proper verifier (i.e., V) and the verifier induced by the simulator (i.e., V'). This difference is due to V' putting too much probability weight on certain moves and thus also too little weight on their sibling messages (recall that a message in the interaction contains one bit). In what follows we deal with two cases.

The first case is when this difference between the behavior of V' (induced by M') and the behavior of the verifier V is "more than tiny". This case receives most of our attention.

We are going to use the oracle in order to move weight from a verifier message β that gets too much weight (after a history h) to its sibling message $\beta \oplus 1$ that gets too little weight (after the history h) in the simulation. Specifically, when the new simulator M'' invokes M' and comes up with a conversation that has $h \circ \beta$ as a prefix, the simulator M'' (with the help of the oracle) will output (a different) conversation with the prefix $h \circ (\beta \oplus 1)$ instead of outputting the original conversation. The simulator M'' will do this with probability that exactly compensates for the difference between V' and V. This leaves one problem. How does the new simulator M'' come up with a conversation that has a prefix $h \circ (\beta \oplus 1)$? The cost of letting the oracle supply the rest of the conversation (after the known prefix $h \circ (\beta \oplus 1)$) is too high. We adopt a "brutal" solution in which we truncate all conversations that have $h \circ (\beta \oplus 1)$ as a prefix. The truncation takes place both in the interaction (P'', V), where P'' stops the conversation after $\beta \oplus 1$ (with a special STOP message) and in the simulation where the oracle recognizes cases in which the simulator M'' should output a truncated conversation. These changes make M'' and V behave exactly the same on messages for which the difference between V' and V is more than tiny. Naturally, V immediately rejects when P'' stops the interaction abruptly, so we have to make sure that this change does not foil the ability of P'' to convince V on an input $x \in L$. It turns out that these truncations happen with negligible probability since such truncation is needed only when the difference between V and V' is more than tiny. Thus, P'' convinces V on $x \in L$ almost with the same probability as P' does.

The second possible case is that the difference between the behavior of V and V' is tiny. In this case, looking at a full conversation $\bar{c}$, we get that the tiny differences sum up to a small difference between the probability of $\bar{c}$ in the distributions of M' and in the distribution of (P', V). We correct these differences by lowering the probabilities of all conversations in the new simulator. The probability of each conversation is lowered so that its relative weight (relatively to all other conversations) is equal to its relative weight in the interaction (P'', V). Technically, this is done by M'' not producing an output in certain cases that M' did produce an output.

Technical remark: The oracle can be used to allow the simulator to toss bias coins when the simulator does not "know" the bias. Suppose that the simulator needs to toss a coin so that it comes-up `head` with probability $\frac{N}{2^m}$, where $N < 2^m$ and both N and m are integers. The simulator supplies the oracle with a uniformly chosen $r \in \{0,1\}^m$ and the oracle answers `head` if r is among the first N strings in $\{0,1\}^m$ and `tail` otherwise. A similar procedure is applicable for implementing a lottery with more than two a-priori known values. Using this procedure, we can get extremely good approximations of probability spaces at a cost related to an a-priori known upper bound on the size of the support (i.e., the oracle answer is logarithmic in the size of the support).

Definition: Let $\epsilon \stackrel{\text{def}}{=} \frac{1}{O(t)}$ where t is the number of rounds in the interaction (P, V).

- Let h be a partial history of the interaction and β be a possible next move by the verifier. We say that β is *weak* with respect to h if

$$\text{Prob}(V'(h) = \beta) < (1-\epsilon) \cdot \text{Prob}(V(h) = \beta)$$

- A conversation $\bar{c} = (c_1, ..., c_t)$ is i-weak if c_i is weak with respect to $(c_1, ..., c_{i-1})$, otherwise it is *i-good.* (Note that a conversation can be i-weak only if the i^{th} move is a verifier move.)
- A conversation $\bar{c} = (c_1, ..., c_t)$ is *i-critical* if it is i-weak but j-good for every $j < i$. A conversation $\bar{c} = (c_1, ..., c_t)$ is *i-co-critical* if the conversation obtained from $\bar{c}$, by complementing (only) the i^{th} bit, is i-critical. (Note that a conversation can be i-critical only for a single i, yet it may be i-co-critical for many i's.)
- A conversation is *weak* if it is i-weak for some i, otherwise it is *good.*

Claim 2.19 *(P', V) outputs weak conversations with negligible probability.*

proof: Recall that $[P', V] \stackrel{\text{s}}{=} [P', V']$ and that the same holds also for prefixes of the conversations. Namely, for any $1 \le i \le t$, $[P', V]_i \stackrel{\text{s}}{=} [P', V']_i$. Let us define a prefix $h \in \{0,1\}^i$ of a conversation to be *bad* if either

$$\text{Prob}([P', V']_i = h) < \left(1 - \frac{\epsilon}{2}\right) \cdot \text{Prob}([P', V]_i = h)$$

or

$$\text{Prob}([P', V']_i = h) > \left(1 + \frac{\epsilon}{2}\right) \cdot \text{Prob}([P', V]_i = h)$$

The claim follows by combining two facts.

Fact 2.20 *The probability that (P', V) outputs a conversation with a bad prefix is negligible.*

proof: Define B_i to be the set of bad prefixes of length i. By the statistical closeness of $[P', V]_i$ and $[P', V']_i$, we get that

$$\Delta \stackrel{\mathrm{def}}{=} \sum_{h \in B_i} |\mathrm{Prob}([P', V]_i = h) - \mathrm{Prob}([P', V']_i = h)| \leq \gamma$$

for some negligible fraction γ. On the other hand, Δ can be bounded from bellow by

$$\sum_{h \in B_i} \mathrm{Prob}([P', V]_i = h) \cdot \left| 1 - \frac{\mathrm{Prob}([P', V']_i = h)}{\mathrm{Prob}([P', V]_i = h)} \right|$$

which by definition of B_i is at least

$$\mathrm{Prob}([P', V]_i \in B_i) \cdot \left| \pm \frac{\epsilon}{2} \right|$$

Thus, $\mathrm{Prob}([P', V]_i \in B_i) \leq \frac{2\gamma}{\epsilon}$ and the fact follows. $\Box$

Fact 2.21 *If a conversation $\bar{c} = (c_1, ..., c_t)$ is weak then it contains a bad prefix.*

proof: Suppose that $\beta \stackrel{\mathrm{def}}{=} c_{i+1}$ is weak with respect $h \stackrel{\mathrm{def}}{=} (c_1, ..., c_i)$. If h is a bad prefix then we are done. Otherwise it holds that

$$\mathrm{Prob}([P', V']_i = h) < \left(1 + \frac{\epsilon}{2}\right) \cdot \mathrm{Prob}([P', V]_i = h)$$

Using the fact that β is weak with respect to h, we get

$$\begin{aligned} \mathrm{Prob}([P', V']_{i+1} = h \circ \beta) &< \left(1 + \frac{\epsilon}{2}\right) \cdot (1 - \epsilon) \cdot \mathrm{Prob}([P', V]_{i+1} = h \circ \beta) \\ &< \left(1 - \frac{\epsilon}{2}\right) \cdot \mathrm{Prob}([P', V]_{i+1} = h \circ \beta) \end{aligned}$$

which implies that $h \circ \beta$ is a bad prefix of $\bar{c}$. $\Box$

Combining Facts 2.20 and 2.21, Claim 2.19 follows. $\Box$

Claim 2.22 *Suppose that $\bar{c} = (c_1, ..., c_t)$ is a good conversation. Then, the probability that $\bar{c}$ is output by M' is at least $(1 - \epsilon)^{\lceil t/2 \rceil} \cdot \mathrm{Prob}([P', V] = \bar{c})$. Furthermore, for $l < k$, if $\bar{c} = (c_1, ..., c_t)$ is i-good for every $i \in \{l + 1, ..., k\}$, then*

$$\mathrm{Prob}\left([M']_k = \gamma \,|\, [M']_l = h\right) \geq (1 - \epsilon)^{\lceil \frac{k-l}{2} \rceil} \cdot \mathrm{Prob}\left([P', V]_k = \gamma \,|\, [P', V]_l = h\right)$$

where $\gamma \stackrel{\mathrm{def}}{=} (c_1, ..., c_k)$ and $h \stackrel{\mathrm{def}}{=} (c_1, ..., c_l)$

proof: To see that this is the case, we write the probabilities step by step conditioned on the history so far. We note that the prover's steps happen with equal probabilities in both sides of the inequality, and therefore can be reduced. Since the relevant verifier's steps are not weak, we get the mentioned inequality. The actual proof proceeds by induction on $k-l$. Clearly, if $k-l=0$ the claim holds. We note that if $k-l=1$ the claim also holds since step k in the conversation is either a prover step or a k-good verifier step.

To show the induction step we use the induction hypothesis for $k-l-2$. Namely,

$$\begin{aligned} &\mathrm{Prob}\left([M']_{k-2}=(c_1,\ldots,c_{k-2}) \mid [M']_l=(c_1,\ldots,c_l)\right) \\ &\quad \geq\ (1-\epsilon)^{\lceil\frac{k-l}{2}\rceil-1}\cdot \mathrm{Prob}\left([P',V]_{k-2}=(c_1,\ldots,c_{k-2}) \mid [P',V]_l=(c_1,\ldots,c_l)\right) \end{aligned} \tag{2.6}$$

Steps $k-1$ and k include one prover message and one verifier message. Assume, without loss of generality, that the prover step is $k-1$. Since P' is the simulator based prover, we get

$$\begin{aligned} &\mathrm{Prob}\left([M']_{k-1}=(c_1,\ldots,c_{k-1}) \mid [M']_{k-2}=(c_1,\ldots,c_{k-2})\right) \\ &\quad =\ \mathrm{Prob}\left([P',V]_{k-1}=(c_1,\ldots,c_{k-1}) \mid [P',V]_{k-2}=(c_1,\ldots,c_{k-2})\right) \end{aligned} \tag{2.7}$$

Since step k of the verifier is good, we also have:

$$\begin{aligned} &\mathrm{Prob}\left([M']_{k}=(c_1,\ldots,c_{k}) \mid [M']_{k-1}=(c_1,\ldots,c_{k-1})\right) \\ &\quad \geq\ (1-\epsilon)\cdot \mathrm{Prob}\left([P',V]_{k}=(c_1,\ldots,c_{k} \mid [P',V]_{k-1}=(c_1,\ldots,c_{k-1})\right) \end{aligned} \tag{2.8}$$

Combining Equations 2.6, 2.7, and 2.8, the induction step follows and we are done. □

Dealing with weak conversations

We start by modifying the prover P', resulting in a modified prover, denoted P'', that stops once it gets a verifier message which is weak with respect to the current history; otherwise, P'' behaves as P'. Namely,

Definition (modified prover - P''): For any $h\in\{0,1\}^*$ and $\beta\in\{0,1\}$,

$$P''(h\circ\beta)=\begin{cases} STOP & \text{if } \beta \text{ is weak with respect to } h.\\ P'(h\circ\beta) & \text{Otherwise}\end{cases}$$

We assume that the verifier V stops and rejects immediately upon receiving an illegal message from the prover (and in particular upon receiving this STOP message).

Next, we modify the simulator so that it outputs either good conversations or truncated conversations which are originally i-critical. Jumping ahead, we stress that such truncated i-critical conversations will be generated from both i-critical and i-co-critical conversations. The modified simulator, denoted M'', proceeds as follows[4]. First, it invokes M' and obtains a conversation $\bar{c} = (c_1, ..., c_t)$. Next, it queries the augmented oracle on $\bar{c}$. The oracle answers probabilistically and its answers are of the form (i, σ), where $i \in \{1, ..., t\}$ and $\sigma \in \{0, 1\}$. The probability distribution will be specified below, at this point we only wish to remark that the oracle only returns pairs (i, σ) for which one of the following three conditions holds

1. $\bar{c}$ is good, $i = t$ and $\sigma = 0$ (if $\bar{c}$ is good and is not i-co-critical for any i's then the oracle always answers this way);
2. $\bar{c}$ is i-critical and $\sigma = 0$;
3. $\bar{c}$ is i-co-critical and $\sigma = 1$.

Finally, the new simulator (M'') halts outputting $(c_1, ..., c_{i-1}, c_i \oplus \sigma)$, which in case $\sigma = 1$ is not a prefix of $\bar{c}$. Note that i may be smaller than t, in which case M'' outputs a truncated conversation which is always i-critical; otherwise, M'' outputs a non-truncated conversation. Note that this oracle message contains at most $1 + \log t$ bits where t is the length of the interaction between P' and V. It remains to specify the oracle's answer distribution.

Let us start by considering two special cases. In the first case, the conversation generated by M' is i-critical, for some i, but is not j-co-critical for any $j < i$. In this case the oracle always answers $(i, 0)$ and consequently the simulator always outputs the i-bit long prefix. However, this prefix is still being output with too low probability. This will be corrected by the second case hereby described. In this ("second") case, the conversation $\bar{c}$ generated by M' is good and i-co-critical for a single i. This means that the i-bit long prefix is given too much probability weight whereas the prefix obtained by complimenting the i^{th} bit gets too little weight. To correct this, the oracle outputs $(i, 1)$ with probability q and $(t, 0)$ otherwise, where q will be specified. What happens is that the M'' will output the "i-complimented prefix" with higher probability than with which it has appeared in M'. The value of q is determined as follows. Denote $p \stackrel{\text{def}}{=} \text{Prob}(V(c_1, ..., c_{i-1}) = c_i \oplus 1)$ and $p' \stackrel{\text{def}}{=} \text{Prob}(V'(c_1, ..., c_{i-1}) = c_i \oplus 1)$. Then, setting q so that $p' + (1 - p') \cdot q = p$ (i.e., $q = \frac{p - p'}{1 - p'}$) allows the simulator to output the prefix $(c_1, ..., c_{i-1}, c_i \oplus 1)$ with the right probability.

In the general case, the conversation generated by M' may be i-co-critical for many

[4] We stress that P'' is not necessarily the simulator-based prover of M''.

i's as well as j-critical for some (*single*) j. In case it is j-critical, it can be i-co-critical only for $i < j$. Let us consider the sequence of indices, $(i_1, ..., i_l)$, for which the generated conversation is critical or co-critical (i.e., the conversation is i_k-co-critical for all $k < l$ and is either i_l-critical or i_l-co-critical). We consider two cases. In both cases the q_k's are set as in the above example; namely, $q_k = \frac{p_k - p'_k}{1-p'_k}$, where $p_k \stackrel{\text{def}}{=} \text{Prob}(V(c_1, ..., c_{i_k-1}) = c_{i_k} \oplus 1)$ and $p'_k \stackrel{\text{def}}{=} \text{Prob}(V'(c_1, ..., c_{i_k-1}) = c_{i_k} \oplus 1)$.

1. The generated conversation, $\bar{c} = (c_1, ..., c_t)$, is i_k-co-critical for every $k < l$ and is i_l-critical. In this case, the distribution of the oracle answers is as follows. For every $k < l$, the pair $(i_k, 1)$ is returned with probability $(\prod_{j<k}(1 - q_j)) \cdot q_k$; whereas the pair $(i_l, 0)$ appears with probability $\prod_{j<l}(1 - q_j)$. We stress that no other pair appears in this distribution.[5]

2. The generated conversation, $\bar{c} = (c_1, ..., c_t)$, is i_k-co-critical for every $k \leq l$. In this case, the distribution of the oracle answers is as follows. For every $k \leq l$, the pair $(i_k, 1)$ is returned with probability $(\prod_{j<k}(1 - q_j)) \cdot q_k$; whereas the pair $(t, 0)$ appears with probability $\prod_{j\leq l}(1 - q_j)$. Again, no other pair appears in this distribution.

Claim 2.23

1. $[P'', V] \stackrel{\text{s}}{=} [P', V]$;

2. Each conversation of (P'', V)*, be it a complete* (P', V)*-conversation or a truncated (i.e., critical) one, is output by* M'' *with probability that is at least a* $(1 - \epsilon)^t > \frac{3}{4}$ *fraction of the probability that it appears in* $[P'', V]$.

proof: The weak conversations are negligible in the output distribution of (P', V) (see Claim 2.19). The only difference between $[P'', V]$ and $[P', V]$ originates from a different behavior of P'' on weak conversations, specifically P'' truncates them while P' does not. Yet, the distribution on the good conversations remains unchanged. Therefore the distribution of $[P'', V]$ is statistically close to the distribution of $[P', V]$, and we are done with Part (1).

For Part (2) let us start with an intuitive discussion which may help reading through the formal proof that follows. First, we recall that the behavior of the simulation M' in prover steps is identical to the behavior of the interaction (P', V) in prover's steps. This follows

[5] Indeed the reader can easily verify that these probabilities sum up to 1.

simply from the fact that P' is the simulation based prover of M'. We will show that this property still holds for the new interaction (P'', V) and the new simulation M''. We will do this by noting two different cases. In one case, the prover step is conducted by P'' exactly as it is done by P' and then M'' behaves exactly as M'. The second possible case is that the prover step contains the special message STOP. We shall note that this occurs with exactly the same probability in the distribution (P'', V) and in the distribution of M''.

Next, we consider the verifier steps. In the construction of M'' and P'' we considered the behavior of M' and V on verifier steps and made changes when these differences were not "tiny". We called a message β weak with respect to a history h, if the simulator assigns the message β (after outputting h) a probability which is smaller by a factor of more than $(1-\epsilon)$ from the probability that the verifier V outputs the message β on history h. We did not make changes in messages whose difference in weight (between the simulation M' and the interaction (P', V)) were smaller than that. In the proof, we consider two cases. First, the message β is weak with respect to the history h. Clearly, the sibling message $\beta \oplus 1$ is getting too much weight in the simulation M'. So in the definition of M'' we made adjustments to move weight from the prefix $h \circ (\beta \oplus 1)$ to the prefix $h \circ \beta$. We will show that this transfer of weight exactly cancels the difference between the behavior of V and the behavior of M'. Namely, the weak messages (and their siblings) are assigned exactly the same probability both in M'' and by V. Thus, we show that when a weak step is involved, the behavior of (P'', V) and the behavior of M'' are exactly equivalent. It remains to deal with messages for which the difference between the conditional behavior of V and M' is "tiny" and was not considered so far. In this case, M'' behaves like M'. However, since the difference is so tiny, we get that even if we accumulate the differences throughout the conversation, they sum up to at most the multiplicative factor $3/4$ stated in the claim.

Let us begin the formal proof by writing again the probability that (P'', V) outputs $\bar{c}$ as the product of the conditional probabilities of the t steps. Namely,

$$\prod_{i=1}^{t} \mathrm{Prob}\left([P'', V]_{i+1} = h_i \circ c_{i+1} \mid [P'', V]_i = h_i\right)$$

where $h_i \stackrel{\mathrm{def}}{=} (c_1, \ldots, c_i)$. We do the same for the probability that M'' outputs a conversation $\bar{c}$. We will show by induction that each step of any conversation is produced by M'' with at least $(1-\epsilon)$ times the probability of the same step in the (P'', V)-interaction. Once we have shown this, we are done. Clearly this claim holds for the null prefix. To prove the induction step, we consider the two possibilities for the party making the $i+1^{\text{st}}$ step.

$i+1^{st}$ step is by the prover: Consider the conditional behavior of M'' given the history so far. We will show that this behavior is identical to the behavior of P'' on the same partial history.

A delicate point to note here is that we may talk about the behavior of M'' on a prefix h_i only if this prefix appears with positive probability in the output distribution $[M'']_i$. However, by the induction hypothesis any prefix that is output by $[P'', V]_i$ appears with positive probability in $[M'']_i$.

We partition the analysis into two cases.

1. First, we consider the case in which the last message of the verifier is weak with respect to the history that precedes it. Namely, $h = h' \circ \beta$ and β is weak with respect to h'. In this case, both in the interaction (P'', V) and in the simulation M'', the next message of the prover is set to STOP with probability 1. Namely,

$$\begin{aligned} \text{Prob}\left(M'' = h \circ \text{STOP} \mid [M'']_i = h\right) &= 1 \\ &= \text{Prob}\left(P''(h) = \text{STOP}\right) \end{aligned}$$

2. The other possible case is that the last message of the verifier is not weak with respect to its preceding history. In this case, the simulator M'' behaves like M' and the prover P'' behaves like P'. (Note that the changes in critical and co-critical steps apply only to verifier steps.) Thus,

$$\begin{aligned} \text{Prob}\left([M'']_{i+1} = h \circ \alpha \mid [M'']_i = h\right) &= \text{Prob}\left([M']_{i+1} = h \circ \alpha \mid [M']_i = h\right) \\ &= \text{Prob}\left(P'(h) = \alpha\right) \\ &= \text{Prob}\left(P''(h) = \alpha\right) \end{aligned}$$

To summarize, the conditional behavior of M'' in the prover steps and the conditional behavior of P'' are exactly equal.

$i+1^{st}$ step is by the verifier: Again, we consider the conditional behavior of M'' given the history so far. Let us recall the second modification applied to M' when deriving M''. This modification changes the conditional probability of the verifier steps in the distribution of M' in order to add weight to steps having low probability in the simulation. We note that this modification is made only in critical or co-critical steps of the verifier. Consider a history h_i which might appear in the interaction (P'', V) and a possible response β of V to h_i. Again, by the induction hypothesis, h_i has a positive probability to be output by the simulation

M'' and therefore we may consider the conditional behavior of M'' on this history h_i. There are three cases to be considered, corresponding to whether either β or $\beta \oplus 1$ or none is weak with respect to h_i.

We start with the simplest case in which neither β nor $\beta \oplus 1$ is weak (w.r.t. h_i). In this case, the behavior of M'' is identical to the behavior of M' since the oracle never sends the message $(i+1, \sigma)$ in this case. However, by the fact that β is not weak, we get that

$$\begin{aligned}(1-\epsilon) \cdot \text{Prob}(V(h) = \beta) &\leq \text{Prob}\left([M']_{i+1} = h \circ \beta \mid [M']_i = h\right) \\ &= \text{Prob}\left([M'']_{i+1} = h \circ \beta \mid [M'']_i = h\right)\end{aligned}$$

and we are done with this simple case.

We now turn to the case in which β is weak (w.r.t. h_i). In this case, given that M'' has produced the prefix h_i, it produces $h_i \circ \beta$ whenever M' produces the prefix $h_i \circ \beta$. Furthermore, with conditional probability q (as defined above), M'' produces the prefix $h_i \circ \beta$ also in case M' produces the prefix $h_i \circ (\beta \oplus 1)$. As above, we define

$$\begin{aligned} p &\stackrel{\text{def}}{=} \text{Prob}\left(V(h_i) = \beta\right) \\ p' &\stackrel{\text{def}}{=} \text{Prob}\left(V'(h_i) = \beta\right)\end{aligned}$$

Since V' is the simulation (M') based verifier, we may also write

$$p' = \text{Prob}\left([M']_{i+1} = h_i \circ \beta \mid [M']_i = h_i\right) \tag{2.9}$$

Also, recall that q was defined as $\frac{p-p'}{1-p'}$. Now, using these notations:

$$\begin{aligned}\text{Prob}\left([M'']_{i+1} = h_i \circ \beta \mid [M'']_i = h_i\right) &= \text{Prob}\left([M']_{i+1} = h_i \circ \beta \mid [M']_i = h_i\right) \\ &\quad + \frac{p-p'}{1-p'} \cdot \text{Prob}\left([M']_{i+1} = h_i \circ (\beta \oplus 1) \mid [M']_i = h_i\right)\end{aligned}$$

Using Equation (2.9), we get

$$\begin{aligned} &= p' + \frac{p-p'}{1-p'} \cdot (1-p') \\ &= p \\ &= \text{Prob}\left(V(h) = \beta\right)\end{aligned}$$

Finally, we turn to the case in which $\beta \oplus 1$ is weak (w.r.t. h_i). Again, this means that β is co-critical in $\bar{c}$. Given that M'' has produced the prefix h_i, it produces $h_i \circ \beta$ only when M' produces the prefix $h_i \circ \beta$, and furthermore, M'' does so only with probability $1-q$ (where

q is again as defined above). We denote p and p', with respect to the critical message $\beta \oplus 1$. Namely,

$$
\begin{aligned}
p &\stackrel{\text{def}}{=} \text{Prob}\,(V(h_i) = \beta \oplus 1) \\
p' &\stackrel{\text{def}}{=} \text{Prob}\,(V'(h_i) = \beta \oplus 1) \\
&= \text{Prob}\,([M']_{i+1} = h_i \circ (\beta \oplus 1) \mid [M']_i = h_i)
\end{aligned}
$$

Thus, recalling that $q = \frac{p-p'}{1-p'}$, we get

$$
\begin{aligned}
\text{Prob}\,([M'']_{i+1} = h_i \circ \beta \,|[M'']_i = h_i) &= (1 - \frac{p-p'}{1-p'}) \cdot \text{Prob}\,([M']_{i+1} = h_i \circ \beta \,|[M']_i = h_i) \\
&= \frac{1-p}{1-p'} \cdot (1-p') \\
&= 1-p \\
&= \text{Prob}\,(V(h_i) = \beta)
\end{aligned}
$$

This completes the proof of Claim 2.23. $\square$

Lowering the probability of some simulator outputs

After handling the differences between M' and (P', V) which are not tiny, we make the last modification, in which we deal with tiny differences. We do that by lowering the probability that the simulator outputs a conversation, in case it outputs this conversation more frequently than it appears in (P'', V). The modified simulator, denoted M''', runs M'' to obtain a conversation $\bar{c}$. (Note that M'' always produces output.) Using the further-augmented oracle, M''' outputs $\bar{c}$ with probability

$$
p_{\bar{c}} \stackrel{\text{def}}{=} \frac{3}{4} \cdot \frac{\text{Prob}([P'', V] = \bar{c})}{\text{Prob}([M''] = \bar{c})}
$$

Note that $p_{\bar{c}} \leq 1$ holds due to Part 2 of Claim 2.23.

Claim 2.24

1. *M''' produces output with probability $\frac{3}{4}$;*
2. *The output distribution of M''' (i.e., in case it has output) is identical to the distribution $[P'', V]$.*

proof: The probability that M''' produces an output is exactly:

$$
\sum_{\bar{c}} \text{Prob}\,([M''] = \bar{c}) \cdot p_{\bar{c}} = \frac{3}{4}
$$

As for part (2), we note that the probability that a conversation $\bar{c}$ is output by M''' is exactly $\frac{3}{4} \cdot \text{Prob}([P'', V] = \bar{c})$. Since the simulator halts with an output with probability exactly $\frac{3}{4}$, we get that given that M''' halts with an output, it outputs $\bar{c}$ with probability exactly $\text{Prob}([P'', V] = \bar{c})$ and we are done. $\square$

An important point not explicitly addressed so far is whether all the modifications applied to the simulator preserve its ability to be implemented by a probabilistic polynomial-time with bounded access to an oracle. Clearly, this is the case with respect to M'' (at the expense of additional $1 + \log_2 t = O(\log n)$ oracle queries). Yet, regarding the last modification there is a subtle points which needs to be addressed. Specifically, we need to verify that the definition of M''' is implementable; namely, that M''' can (with help of an augmented oracle) "sieve" conversations with *exactly* the desired probability. Note that the method presented above (in the "technical remark") may yield exponentially small deviation from the desired probability. This will get very close to a perfect simulation, but yet will not achieve it.

To this end, we modify the "sieving process" suggested in the technical remark to deal with the specific case we have here. But first we modify P'' so that it makes its random choices (in case it has any) by flipping a polynomial number of unbiased coins.[6] This rounding does change a bit the behavior of P'', but the deviation can be made so small that the above assertions (specifically Claim 2.23) still hold.

Consider the specific sieving probability we need here. Namely: $p_{\bar{c}} = \frac{3}{4} \cdot \frac{a/b}{c/d}$, where $\frac{a}{b} = \text{Prob}([P'', V] = \bar{c})$ and $\frac{c}{d} = \text{Prob}([M''] = \bar{c})$. A key observation is that c is the number of coin tosses which lead M'' to output $\bar{c}$ (i.e., using the notation of the previous section, $c = |\Omega_{\bar{c}}|$). Observing that b is the size of probability space for $[P'', V]$ and using the above modification to P'', we rewrite $p_{\bar{c}}$ as $\frac{3ad}{4b} \cdot \frac{1}{c} = \frac{e}{c2^f}$, where e and $f = \mathit{poly}(n)$ are some non-negative integers.

We now note, that the oracle can allow the simulator to sieve conversations with probability $\frac{e}{c}$ ($f = 0$), for any $0 \leq e \leq c$ in the following way. M''' sends to the oracle the random tape ω that it has tossed for M'', and the oracle sieves only e out of the possible c random tapes which lead M'' to output $\bar{c}$. The general case of $p_{\bar{c}} = \frac{e}{c2^f}$ is deal by writing $p_{\bar{c}} = \frac{q}{c} + \frac{r}{c2^f}$, where $q = \lfloor e/2^f \rfloor$ and $r = e - q2^f < 2^f$. To implement this sieve, M''' supplies

[6] The implementation of P'' was not discussed explicitly. It is possible that P'' uses an infinite number of coin tosses to select its next message (either 0 or 1). However, an infinite number of coin tosses is not really needed since rounding the probabilities so that a polynomial number of coins suffices, causes only exponentially small rounding errors.

the oracle with a uniformly chosen f-bit long string (in addition to ω). The oracle sieves out q random-tapes (of M'') as before, and uses the extra bits in order to decide on the sieve in case ω equals a specific (different) random-tape.

Combining Claims 2.18, 2.23 (part 1), and 2.24, we conclude that (P'', V) is an interactive proof system of *perfect* knowledge complexity $k(n) + O(\log n)$ for L. This completes the proof of Theorem 2.17. ■

2.5 Applying our Techniques for Non-Negligible Error Probabilities

As explained in Section 2.1.1, the notion of an interactive proof with bounded knowledge complexity is not robust under changes in the allowed error probability. Throughout this work, we use the natural definition of interactive proofs in which the error probability is negligible. However, our techniques yield non-trivial results also in the case one defines interactive proofs with some specific non-negligible error probability. In this section we explain how such assertions may be obtained, and state such results for two special cases.

Denote by $\epsilon_c(n)$ (an upper bound on) the probability that the verifier rejects an input x although $x \in L$ and the prover plays honestly. This is the error probability related to the completeness condition. Similarly, denote by $\epsilon_s(n)$ (an upper bound on) the probability that the verifier accepts $x \notin L$ when the prover follows its optimal strategy (not necessarily following the protocol). This is the error probability related to the soundness condition. We say that an interactive proof has error probabilities (ϵ_s, ϵ_c) if its error probability in the soundness condition is bounded by ϵ_s and its error probability in the completeness condition is bounded by ϵ_c.

2.5.1 The perfect case

In this subsection, we consider the more restricted case of *perfect* knowledge complexity, and derive Theorem 2.26 which is the analogue of Theorem 2.14 for the case that the error probabilities are not negligible. Following the definitions in Section 2.3, we denote the simulation based prover by P^*.

Let us follows the steps of the proof of our main theorem and observe which assertions hold for the case of non-negligible error probability. We begin by observing that the following generalization of Lemma 2.15 holds:

Lemma 2.25 *Let (P,V) be an interactive proof for L with error probabilities $(\epsilon_s(n),\epsilon_c(n))$ and with knowledge complexity $k(n)$, then*

1. *If $x \in L$ then the probability that (P^*,V) outputs an accepting conversation is at least $(1-\epsilon_c(n))^2 \cdot 2^{-k(n)}$, where $n=|x|$.*
2. *If $x \notin L$ then the probability that (P^*,V) outputs an accepting conversation is at most $\epsilon_s(n)$, where $n=|x|$.*

The proof of this lemma is identical to the proof of Lemma 2.15, except that here $\frac{|A_\lambda|}{|S_\lambda|} = 1-\epsilon_c(n)$. As explained in Section 2.3, an efficient machine with access to an NP oracle can sample conversations in (P^*,V). By Lemma 2.25, this would yield an accepting conversation with probability at most $\epsilon_s(n)$ in the case $x \notin L$ and at least $(1-\epsilon_c(n))^2 \cdot 2^{-k(n)}$ when $x \in L$. In case these two probabilities differ sufficiently (i.e., by more then a polynomial fraction), we can use standard amplification techniques to get a probabilistic algorithm that determines whether $x \in L$ with error probability less than 1/3 (or negligible, or 2^{-n}). To summarize, we get the following theorem for perfect knowledge complexity.

Theorem 2.26 *If a language L has an interactive proof with perfect knowledge complexity $k(n)$ and error probabilities (ϵ_s,ϵ_c) and if there exists a polynomial $p(n)$ such that*

$$(1-\epsilon_c(n))^2 \cdot 2^{-k(n)} \; > \; \epsilon_s(n) + \frac{1}{p(n)}$$

then $L \in \mathcal{BPP}^{\mathcal{NP}}$.

Examples: Theorem 2.26 implies, for example, that if a language L has an interactive proof of knowledge complexity 1 and error probability 1/4 (both in the soundness condition and in the completeness condition), then L is in $\mathcal{BPP}^{\mathcal{NP}}$. Another interesting example is the case of one-sided error (i.e., $\epsilon_c = 0$). Theorem 2.26 implies that for any polynomial $p(\cdot)$, if a language L has a one-sided error interactive proof (P,V) of knowledge complexity at most $\log_2\left(\frac{p(\cdot)}{2}\right)$ and error probability $\epsilon_s \leq \frac{1}{p(\cdot)}$, then L is in $\mathcal{BPP}^{\mathcal{NP}}$.

2.5.2 The general (statistical) case

Unfortunately, the analogue result for statistical knowledge complexity is not as clean, and has various different formulations according to possible properties of the error probabilities.

Let us explain how such a result can be obtain, and give a specific example for the special case in which $\epsilon_c = 0$, i.e., the original interaction has one-sided error.

Recall that the proof for the negligible error-probability case uses the transformation from statistical to perfect knowledge complexity and then uses Theorem 2.14. This transformation increases the knowledge complexity by a logarithmic additive term. In view of Lemma 2.25, it is desirable not to increase the knowledge complexity without concurrently decreasing the error probability. Thus, before applying the transformation, we reduce the error probability by iterating the protocol as many times as possible while maintaining logarithmic knowledge complexity.

Specifically, denote the length of the interaction by $l(n)$. Also, fix an input x of length n, and let $l = l(n)$, $k = k(n)$, $\epsilon_s = \epsilon_s(n)$ and $\epsilon_c = \epsilon_c(n)$. The transformation from statistical to perfect knowledge complexity (as described in Section 2.4) increases the knowledge complexity by $1 + \log_2 l$. We begin by running the original protocol (P, V) sequentially $t \stackrel{\rm def}{=} \lceil (\log_2 l)/k \rceil$ times. These repetitions yield a new protocol (P', V') whose length is $t \cdot l$, its knowledge complexity is bounded by $t \cdot k < (k-1) + \log_2 l$, and its error probability decreases. To compute the decrease in the error probabilities, we partition the analysis into two cases according to whether the original interaction has one sided error or not.

If the original interaction has one sided error, i.e., the verifier always accepts when $x \in L$, then the new verifier V' accepts only if all repetitions of the original protocols end accepting. The error probabilities in this case decrease from $(\epsilon_s, 0)$ to $(\epsilon_s^t, 0)$. In the case where the original interactive proof was not one sided, the verifier counts the number of original interactions that end with the original verifier accepting. The new verifier accepts if this number is greater than $\frac{\epsilon_s + (1-\epsilon_c)}{2} \cdot t$. In order to compute the new error probabilities we may apply the Chernoff bound and get an upper bound on the new error probabilities which depends on t, on the difference between $1 - \epsilon_c$ and ϵ_s, and of-course on ϵ_s and ϵ_c themselves.

Next, we apply the transformation of Section 2.4 ("from statistical to perfect knowledge complexity") and get a new interactive proof (P'', V'') for L which has knowledge complexity $k - 1 + \log_2 l + 1 + \lceil \log_2(l \cdot t) \rceil$, where the additional $1 + \lceil \log_2(l \cdot t) \rceil$ term comes from the transformation. Finally, if the resulting parameters of (P'', V'') satisfy the conditions stated in Theorem 2.26, then we get that the language L is in $\mathcal{BPP}^{\mathcal{NP}}$. Let us provide full details for the special (yet important) case of one sided error (i.e., $\epsilon_c = 0$).

In the special case of one-sided error, we end up using Theorem 2.26 for an interactive proof with knowledge complexity $k + \log_2 l + \lceil \log_2(l \cdot t) \rceil$ and (one-sided) error probability

${\epsilon_s}^t$. Thus, we get the following theorem for statistical knowledge complexity:

Theorem 2.27 *Suppose that a language L has an interactive proof of statistical knowledge complexity $k(n)$, one-sided error probability $\epsilon_s(n)$, and with length $l(n)$ so that there exists a polynomial $p(n)$ for which the following inequality holds*

$$\frac{1}{2 \cdot 2^{k(n)} \cdot l(n)^2 \cdot \left\lceil \frac{\log_2 l(n)}{k(n)} \right\rceil} \geq \epsilon_s(n)^{\lceil (\log_2 l(n))/k(n) \rceil} + \frac{1}{p(n)}$$

Then $L \in \mathcal{BPP}^{\mathcal{NP}}$.

2.6 A Flaw in [F-89]

As remarked in the introduction, In course of this research, we found out that the proof that $\mathcal{SKC}(0) \subseteq$ co-$\mathcal{AM}$ as it appears in [F-89] is not correct. In particular, there is a flaw in the AM-protocol presented in [F-89] for the complement language. In this section, we explain the problem in the proof there and present a counter-example.

In [F-89], Fortnow presents a constructive method for proving that $\mathcal{SZK} \stackrel{\text{def}}{=} \mathcal{SKC}(0)$ is contained in co-$\mathcal{AM}$. Given an interactive proof (P, V) for a languages L and a (statistical) zero-knowledge simulator M (for the honest verifier V), he constructs a two-round protocol (P', V'). This protocol was claimed to constitute an interactive proof system for $\overline{L}$. This claim, as we are going to show, is wrong. Yet, the result $\mathcal{SZK} \subseteq$ co-$\mathcal{AM}$ does hold, since the work of Aiello and Hastad contains the necessary refinements which enable to present a modified AM-protocol for $\overline{L}$ (see [AH-87, H-94]). Furthermore, Fortnow's basic approach is valid, and indeed it was used in subsequent works (e.g., [AH-87, BMO-90, Ost-91, BP-92a, OW-93]).

Fortnow's basic approach starts with the observation that the simulator M must behave differently on $x \in L$ and $x \notin L$. Clearly, the difference cannot be recognized in polynomial-time, unless $L \in \mathcal{BPP}$. Yet, stronger recognition devices, such as interactive proofs should be able to tell the difference. Fortnow suggests a characterization of the simulator's behavior on $x \in L$ and uses this characterization in his protocol for $\overline{L}$, yet this characterization is wrong. Aiello and Hastad present a refinement of Fortnow's characterization [AH-87], their characterization is correct and can be used to show that $\mathcal{SZK} \subseteq \mathcal{AM}$ (which is the goal of their paper) as well as $\mathcal{SZK} \subseteq$ co-$\mathcal{AM}$.

Fortnow's characterization

Given an interactive proof (P, V) for L and a simulator M, and fixing a common input $x \in \{0,1\}^*$, the following sets are defined. Let us denote by t the number of random bits that the verifier V uses on input x, and by q the number of random bits used by the simulator M. For every conversation prefix, h, we consider the set of the verifier's coin tosses which are consistent with h (the conversation so far). We denote this set by R_1^h. Namely, for $h = (\alpha_1, \beta_1, ..., \alpha_i, \beta_i)$ (or $h = (\alpha_1, \beta_1, ..., \alpha_i, \beta_i, \alpha_{i+1})$), $r \in R_1^h$ iff $V(x, r, \alpha_1, ..., \alpha_j) = \beta_j$ for every $j \leq i$, where $V(x, r, \bar{\alpha})$ denotes the message sent by V on input x random-tape r and prover message-sequence $\bar{\alpha}$. The set R_1^h depends only on the verifier V. Next, we consider sets R_2^h which are subsets of the corresponding R_1^h's. Specifically, they contain only r's that can appear with h in an accepting conversation output by the simulator M. Namely, $r \in R_2^h$ iff $r \in R_1^h$ and there exists $\omega \in \{0,1\}^q$ so that $M(x, \omega)$ is an accepting conversation with prefix h. (Here $M(x, \omega)$ denotes the conversation output by M on input x and simulator-random-tape ω.)

Motivation: For simplicity, suppose that the simulation is perfect (i.e., M witnesses that (P, V) is *perfect* zero-knowledge) and that (P, V) has one sided error (i.e., "perfect completeness"). Then, for every $x \in L$ and every possible h, we must have $R_2^h = R_1^h$ (otherwise the simulation is not perfect). However, if $x \notin L$ then there must exist h's so that R_2^h is much smaller than R_1^h. Otherwise the simulator-based prover (for M) will always convince V to accept x, thus violating the soundness condition of (P, V). The problem with the above dichotomy is that it is "too existential" and thus it is not clear how to use it. Instead Fortnow claimed a dichotomy which is more quantitative.

A False Characterization: Let $\text{pref}(\bar{c})$ denote the set of all message-subsequences in the conversation $\bar{c}$.

- if $x \in L$ then

$$\text{Prob}_\omega(\forall h \in \text{pref}(M(x, \omega)) \ \left|R_2^h\right| \approx_1 \left|R_1^h\right|) > \frac{3}{4}$$

- if $x \notin L$ then

$$\text{Prob}_\omega(\forall h \in \text{pref}(M(x, \omega)) \ \left|R_2^h\right| \approx_2 \left|R_1^h\right|) < \frac{1}{4}$$

where the probability (in both cases) is taken uniformly over $\omega \in \{0,1\}^q$. We did not specify what is meant by $\approx_i$. One may substitute $\alpha \approx_1 \beta$ by $\alpha \geq \frac{1}{2} \cdot \beta$, and $\alpha \approx_2 \beta$ by $\alpha \geq \frac{1}{4} \cdot \beta$. The gap between the two is needed for the approximate lower/upper bound protocols.

A Counterexample

The mistake is in the second item of the characterization. The false argument given in [F-89] confuses between the probability distribution of conversations output by the simulator and the probability distribution of the conversations between a simulator-based prover (denote P^*) and the verifier. These distributions are not necessarily the same (note that we are in case $x \notin L$). Consequently, the probability that "good" conversations (i.e., conversations for which $|R_2| \approx |R_1|$ for all prefixes) occur in the (P^*, V) interaction is not the same as the probability that the simulator outputs "good" conversations. This point is ignored in [F-89] and leads there to the false conclusion that the characterization holds. Bellow, we present an interactive proof (P, V) and a (perfect) zero-knowledge simulator for which the characterization fails.

The interactive proof that we present is for the empty language Φ. This interactive proof is perfect zero knowledge for the trivial reason that the requirement is vacuous. Yet, we present a simulator for this interactive proof which, for every $x \in \{0,1\}^* = \overline{\Phi}$, outputs "good" conversation with probability close to 1. Thus, the characterization fails.

The interactive proof (from the verifier's point of view – input $x \in \{0,1\}^n$):

- The verifier uniformly selects $\alpha \in \{0,1\}^n$ and sends α to the prover.
- The verifier waits for the prover's message $\beta \in \{0,1\}^n$.
- Next, the verifier uniformly selects $\gamma \in \{0,1\}^n$ and sends γ to the prover.
- The verifier accepts iff either $\alpha = 0^n$ or $\beta = \gamma$.

Regardless of the prover's strategy, the verifier accepts each $x \in \{0,1\}^n$ with negligible probability; specifically $2^{-n} + (1 - 2^{-n}) \cdot 2^{-n}$. Thus, the above protocol indeed constitutes an interactive proof for the empty language Φ.

The simulator operates as follows (on input $x \in \{0,1\}^n$):

- With probability $1 - \epsilon$, the simulator M outputs a conversation uniformly distributed in $0^n \times \{0,1\}^{2n}$. (ϵ is negligible, say $\epsilon = 2^{-n}$)
- With probability ϵ, the simulator M outputs a conversation uniformly distributed in $(\{0,1\}^n - 0^n) \times \{0,1\}^{2n}$.

Claim: In contradiction to the characterization, for every $x \in \{0,1\}^* = \overline{\Phi}$,

$$\mathrm{Prob}_\omega(\forall h \in \mathrm{pref}(M(x,\omega)) \ \left|R_2^h\right| = \left|R_1^h\right|) \geq 1 - \epsilon$$

Proof: It suffices to show that every conversation of the form $0^n\beta\gamma$ satisfies $R_2 = R_1$ for all its prefixes. First observe that $R_1^\lambda = \{0,1\}^{2n} = R_2^\lambda$, since for every $\alpha\gamma \in \{0,1\}^{2n}$ the simulator outputs the accepting conversation $\alpha\gamma\gamma$ with non-zero probability. Similarly, $R_1^{0^n} = 0^n\{0,1\}^n = R_2^{0^n}$. Next, for every $\beta \in \{0,1\}^n$, we have $R_1^{0^n\beta} = 0^n\{0,1\}^n = R_2^{0^n\beta}$, since for every $\gamma \in \{0,1\}^n$ the simulator outputs the accepting conversation $0^n\beta\gamma$ with non-zero probability. (Here we use the fact that the verifier always accepts when $\alpha = 0^n$.) Similarly, $R_1^{0^n\beta\gamma} = 0^n\gamma = R_2^{0^n\beta\gamma}$. $\square$

Conclusion

The source of trouble is that the definition of the sets R_2^h's does not take into account the probability weight assigned by the simulator to ω's that witness the assertion "the simulator outputs an accepting conversation that starts with h". Indeed, this is exactly the nature of the refinement suggested by Aiello and Hastad [AH-87].

2.7 Concluding Remarks

We consider our main result as a very first step towards a classification of languages according to the knowledge complexity of their interactive proof systems. Indeed there is much to be known. Below we first mention two questions which do not seem too ambitious. The first is to try to provide evidence that NP-complete languages cannot be proven within low (say logarithmic or even constant) knowledge complexity. A possible avenue for proving this conjecture is to show that languages having logarithmic knowledge complexity are in co-$\mathcal{AM}$, rather than in $\mathcal{BPP}^{\mathcal{NP}}$ (recall that NP is unlikely to be in co-$\mathcal{AM}$ - see also [BHZ-87]). The second suggestion is to try to provide indications that there are languages in $\mathcal{PSPACE}$ which do not have interactive proofs of linear (rather than logarithmic) knowledge complexity. The reader can easily envision more moderate and more ambitious challenges in this direction.

Another interesting question is whether all levels greater then zero of the knowledge-complexity hierarchy contain strictly more languages than previous levels, or if some partial collapse occurs. For example, it is open whether constant or even logarithmic knowledge complexity classes do not collapse to the zero level.

Regarding our transformation of statistical knowledge complexity into perfect knowledge complexity (i.e., Theorem 2.17), a few interesting questions arise. Firstly, can the cost of the transformation be reduced to bellow $O(\log n)$ bits of knowledge? A result for the special case of statistical zero-knowledge will be almost as interesting. Secondly, can one present an analogous transformation that preserves *one-sided error probability* of the interactive proof? (Note that our transformation introduces a negligible error probability into the completeness condition.) Finally, can one present an analogous transformation that applies to knowledge complexity *with respect to arbitrary verifiers*? (Our transformation applies only to knowledge complexity *with respect to the honest verifier*.)

Chapter 3

The Hardness of Approximations: Gap Location

3.1 The Gap Location Parameter

In this section, we introduce the definitions concerning hard gaps. We follow the definitions with an informal discussion. Let us make a distinction between optimization problems that seek the quality of the best solution and optimization problems that seek the size of the largest solution (see Subsection 1.2.1). We concentrate on the first class of problems, for which the gap-location parameter is more interesting. However, it should be noted that a straightforward modification of the definition below makes it suitable for all optimization problems. Denote by $N(I)$ the number of sub-conditions posed by the instance I. (To make this notation suitable for all optimization problems, one may interpret $N(I)$ as a polynomial time computable bound on the optimum value of I.) $OPT(I)$ denotes the number of sub-conditions which are satisfied by the best solution of I (i.e., the solution that maximizes the number of satisfied sub-conditions).

Definition 3.1 (The parameters of a hard gap). *Consider an optimization problem in which we are looking for the quality of the best solution. Suppose that it is NP-hard to tell whether $OPT(I) \leq \alpha_1 \cdot N(I)$ or $OPT(I) \geq \alpha_0 \cdot N(I)$. Then we say that there exists a* hard gap *for the problem P, at* location α_0, *with* width $\alpha_0 - \alpha_1$.

In this work, we are only concerned with gaps of constant width so we omit the width parameter in what follows. Throughout the paper, we show that problems have hard gaps. Let us clearly state the implication of hard gaps on the hardness of approximations.

Remark 3.2 *If a maximization (or a minimization) problem P has a hard gap at any location $0 < \alpha \leq 1$, then there exists a constant $\epsilon > 0$ such that, unless P=NP, there is no polynomial time algorithm that approximates P to within $1 - \epsilon$ (or $1 + \epsilon$), respectively.*

The instances of an optimization problem always satisfy $0 \leq OPT(I) \leq N(I)$, but sometimes the solution interval is more restricted.

Definition 3.3 (THE SOLUTION INTERVAL). *Consider an optimization problem P in which we seek the quality of the best solution. We say that P has solution interval $[\beta_1, \beta_2]$ if $\liminf_{|I| \to \infty} \frac{OPT(I)}{N(I)} = \beta_1$ and $\limsup_{|I| \to \infty} \frac{OPT(I)}{N(I)} = \beta_2$.*

Namely, neglecting a finite number of instances, $OPT(I)$ satisfies $\beta_1 \cdot N(I) \leq OPT(I) \leq \beta_2 \cdot N(I)$. Usually, $\beta_2 = 1$ in natural problems.

Informal discussion. Let us restrict the discussion to natural problems that seek the quality of the best solution and that are hard to approximate (i.e., have a hard gap somewhere in their solution interval). Also, let the solution interval of P be $[\beta_P, 1]$. We would like to state two beliefs we have concerning the issue of gap-location.

The first belief is that if there exists a hard gap at any location in the interval solution, then there exists hard gaps at all points in the open interval $(\beta_P, 1)$. Note that this implies a difference between proving a hard gap at location 1 and at any other location. A hard gap at location 1 implies (assuming this conjecture) a hard gap at all locations in the interval $(\beta_P, 1]$, whereas a hard gap at any other location does not imply a hard gap at location 1 (even assuming the conjecture). In fact, there exist optimization problems for which there are hard gaps at all locations in the open interval $(\beta_P, 1)$ but not at location 1. We discuss such problems in what follows, but first, let us state our conjecture.

Informal Conjecture 1 *Let P be some* natural *optimization problem in which we are trying to determine the quality of the best solution. Suppose P has a solution interval $[\beta, 1]$, for some $0 \leq \beta < 1$. Then a hard gap at any location in $(\beta, 1]$ implies hard gaps at all locations in the open interval $(\beta, 1)$.*

In what follows, we assume the validity of Conjecture 1. Conjecture 2 (below) makes a distinction between problems which possess hard gaps at all locations in the interval $(\beta_P, 1]$

and problems that have hard gaps only in the open interval $(\beta_P, 1)$. Consider the following decision problem related to an approximation problem P. Given an instance I, determine whether $OPT(I) = N(I)$. For example, the decision problem related to MAX-SAT is SAT (i.e., given a formula φ, determine whether all its clauses can be satisfied). We partition the optimization problems into two classes according to the difficulty of their related decision problems. The first class contains optimization problems for which it is NP-hard to determine whether an input I has $OPT(I) = N(I)$. This class contains problems as MAX-SAT (proven hard to approximate by Arora *et al.* [ALMSS-92]) and MAX k-COLORABILITY (for $k \geq 3$) [PY-91] and Section 3.2). The other class contains problems for which it is easy to decide whether an input I has $OPT(I) = N(I)$. This class contains problems such as MAX-CUT (which can be also viewed as MAX 2-COLORABILITY), and MAX-2SAT (both shown hard to approximate by Papadimitriou & Yannakakis [PY-91]). (While doing this partition, we do not claim that all decision problems are either NP-hard or easy, but practically all known interesting problems of this form do fall into one of these two categories, and we are interested only in these natural problems here.) Our second belief is that any natural problem of the first class has hard gaps at all locations $\alpha \in (\beta_P, 1]$ and that any natural problem of the second class has hard gaps at all locations $\alpha \in (\beta_P, 1)$. Note that we can never have a hard gap at location β_P, since we know that all instances have $OPT(I) \geq \beta_P \cdot N(I)$ and none has $OPT(I) \leq (1 - \epsilon)\beta_P \cdot N(I)$, for any $\epsilon > 0$. Conjecture 2 states that a problem P has a hard gap at location 1 if and only if the corresponding decision problem is NP-hard. The validity of the two conjectures implies our second belief.

Informal Conjecture 2 *Let P be some* natural *optimization problem in which we seek the quality of the best solution. Suppose that P has a hard gap at some location in its solution interval. Then P has a hard gap at gap-location 1 if and only if it is NP-hard to determine whether $OPT(I) = N(I)$ (i.e., the original decision problem is NP-hard).*

Conjecture 1 can be proven for many known optimization problems using a padding argument. In these cases, one may use a padding argument to "transfer" the hard gap to any location in the open interval $(\beta, 1)$. This can be demonstrated on the problem MAX-SAT (as was done in the introduction) and on MAX k-COLORABILITY in the following way. To lower the gap, add a large enough clique to the graph and do not connect it to any original vertex. To move the gap up, add a large enough bipartite graph disconnected from the original nodes of the graph. Note that the solution interval here is $[1 - \frac{1}{k}, 1]$ since any graph has a k-coloring for which $\left(1 - \frac{1}{k}\right) \cdot |E|$ edges are consistent. Generally, note that this padding method cannot be used to transfer a hard gap from a location other than 1 to a hard gap at location 1.

Let us define a class of optimization languages for which the first conjecture can be proven. This class contains many natural optimization problems such as MAX-SAT, MAX 3-DIMENSIONAL MATCHING, etc. We define the class to contain problems which possess a "padding property", which will enable us to prove the conjecture. We will have two requirements to make of problems in this class. First, we require the existence of an efficiently computable operation which joins two instances into one. This operation, which we denote by $\circ$, will have a specific way of combining the optimum values of both instances. (We state this formally in Definition 3.4 below). The second requirement we make is that the problem has a "substantial" number of instances whose optimum lies at the endpoints of the solution interval. Formally, we have the following.

Definition 3.4 *The class* PADDABLE *of optimization problems contains all optimization problems which satisfy the following two requirements:*

1. **Padding property:** *There exists a polynomial time computable padding operation, denoted by $\circ$, which operates on any two instances I_1 and I_2 of P and has the property that for all I_1, I_2, $N(I_1 \circ I_2) = N(I_1) + N(I_2)$ and $OPT(I_1 \circ I_2) = OPT(I_1) + OPT(I_2)$.*

2. **Endpoints property:** *Let $(\beta, 1]$ be the interval solution of P. There exists a constant $c > 0$ such that for any $n \in \mathsf{N}$ it is possible to find in polynomial time two instances I_1, I_2 such that $n \leq N(I_1), N(I_2) \leq n + c$, and $\frac{OPT(I_1)}{N(I_1)} = \beta$, $\frac{OPT(I_2)}{N(I_2)} = 1$.*

For this class the conjecture can be stated as a theorem.

Theorem 3.5 *Let $P \in$* PADDABLE *be some optimization problem in which we are try2ing to determine the quality of the best solution. Suppose P has a solution interval $[\beta, 1]$ for some $0 \leq \beta < 1$. Then, a hard gap at any location in $(\beta, 1]$ implies hard gaps at all locations in the open interval $(\beta, 1)$.*

SKETCH OF PROOF. Suppose that P has a hard gap at location $\alpha_0 \in (\beta, 1]$ and consider any location $\alpha_1 \in (\beta, 1)$. Suppose w.l.o.g. that $\alpha_1 < \alpha_0$. We show that if, for any $\epsilon' > 0$, there exists a polynomial time algorithm that distinguishes between the case $OPT(I)/N(I) \leq \alpha_1 - \epsilon'$ and the case $OPT(I)/N(I) \geq \alpha_1$, then there exists a polynomial time algorithm that distinguishes (for any instance I) between the case $OPT(I)/N(I) \leq \alpha_0 - \epsilon$ and the case $OPT(I)/N(I) \geq \alpha_0$. Thus, the NP-hardness of the gap at location α_1 is implied by the NP-hardness of the gap at location α_0.

Suppose we have an instance I for which we would like to determine whether $OPT(I)/N(I) \leq \alpha_0 - \epsilon$ or $OPT(I)/N(I) \geq \alpha_0$. We "pad" this instance with an instance I' such that the following two relations hold:

$$\frac{OPT(I)}{N(I)} \geq \alpha_0 \implies \frac{OPT(I \circ I')}{N(I \circ I')} \geq \alpha_1$$

$$\frac{OPT(I)}{N(I)} \leq \alpha_0 - \epsilon \implies \frac{OPT(I \circ I')}{N(I \circ I')} \leq \alpha_1 - \epsilon'$$

for the constant $\epsilon' = \epsilon/2$. The possibility to find such an instance I' in polynomial time follows from the second requirement in the definition of PADDABLE. Now, if for any constant $\epsilon' > 0$ there is a polynomial time algorithm which distinguishes between the cases $OPT(I \circ I')/N(I \circ I') \geq \alpha_1$ and $OPT(I \circ I')/N(I \circ I') \leq \alpha_1 - \epsilon'$, then we get a polynomial time algorithm that distinguishes between the cases $OPT(I)/N(I) \geq \alpha_0$ and $OPT(I)/N(I) \leq \alpha_0 - \epsilon$, and we are done. ∎

The definition of the class PADDABLE is semantic. We consider it an interesting open problem to find a syntactic definition of a class of optimization problems for which the first conjecture can be proven.

3.2 A Hard Gap for k-COLORABILITY at Gap-Location 1

Consider the problem of finding a k-coloring of a given graph G such that as many edges as possible are adjacent to two vertices of different colors.

Definition 3.6 (A CONSISTENT EDGE). *Consider a graph $G(V, E)$ and a coloring of its vertices $\sigma : V \to \{1, 2, ..., k\}$. We say that an edge $e = (v_i, v_j)$ is* consistent *regarding σ if $\sigma(v_i) \neq \sigma(v_j)$.*

Definition 3.7 (THE PROBLEM MAX k-COLORABILITY).
Input: *A graph $G(V, E)$.*
Problem: *Find the maximum number of consistent edges in G, over all k-colorings of the vertices in G.*

For $k \geq 3$, it is NP-hard to tell whether a graph is k-colorable or not. We show that for any $k \geq 3$, there exists a constant $\epsilon_k > 0$ such that unless P=NP, there is no polynomial time algorithm which can determine whether an input graph $G(V, E)$ is k-colorable, or whether any k-coloring of G has at most $(1 - \epsilon_k)|E|$ consistent edges.

Theorem 3.8 *For any $k \geq 3$,* MAX k-COLORABILITY *possesses a hard gap at location 1.*

Proof: We use a reduction from MAX 3SAT-B to MAX 3-COLORABILITY. Next, we can use techniques from [PY-91] to further reduce MAX 3-COLORABILITY to MAX k-COLORABILITY for any $k > 3$. In the reduction, we use a bipartite expander, which helps us to preserve the hard gap. The use of expanders in preserving gaps was first noticed in [PY-91]. A bipartite graph on $2 \times n$ nodes is called a bipartite *expander* with degree d and expansion factor $1+\gamma$ if every subset S of at most $n/2$ nodes of one side of the graph is adjacent to at least $(1+\gamma)|S|$ nodes on the other side. Bipartite expanders on $2 \times n$ nodes can be efficiently constructed for any $n \in \mathbb{N}$ [Mar-73, GG-81, AJ-87]. Let us first show the reduction, and then show that if the instance φ of MAX 3SAT-B is satisfiable, then the output of the reduction, $G^{\varphi}(V, E)$, is 3-colorable (Lemma 3.9 below), while if any assignment to φ satisfies at most a fraction $1 - \epsilon$ of the clauses in φ, then any 3-coloring of $G(V, E)$ induces at least $\epsilon \frac{\gamma}{2B} m \geq c|E^{\varphi}|$ inconsistent edges for some constant $c > 0$ (Lemma 3.10 below).

The reduction. We are given an instance of MAX 3SAT-B, i.e., a 3-CNF formula φ with n variables and m clauses, such that each variable appears at most B times in the formula φ. We use an extension of the standard reduction from 3SAT to 3-COLORABILITY [St-73, GJS-76]. Let us shortly describe the original reduction, which uses a gadget with nine vertices and 10 edges (Figure 3.2).

We call the top vertex g_4 the gadget head, and the three bottom vertices g_1, g_2, g_3 the gadget legs. The other vertices in the gadget are called the gadget body. The useful property of this gadget (which will be denoted "$\mathcal{P}$") is that if the three legs have the same color, then any consistent 3-coloring of the gadget assigns the gadget head the same color too, while if the three legs do not have the same color, then for any color assigned to the head, we can complete the coloring of the body consistently.

The reduction outputs $2n$ vertices (the literals vertices) labeled $x_1, \overline{x}_1, x_2, \overline{x}_2, \ldots, x_n, \overline{x}_n$, two vertices named GROUND and B, and m gadgets, one for each clause. The edges of the output graph connect x_i to $\overline{x}_i$, x_i to GROUND, $\overline{x}_i$ to GROUND for $1 \leq i \leq n$,

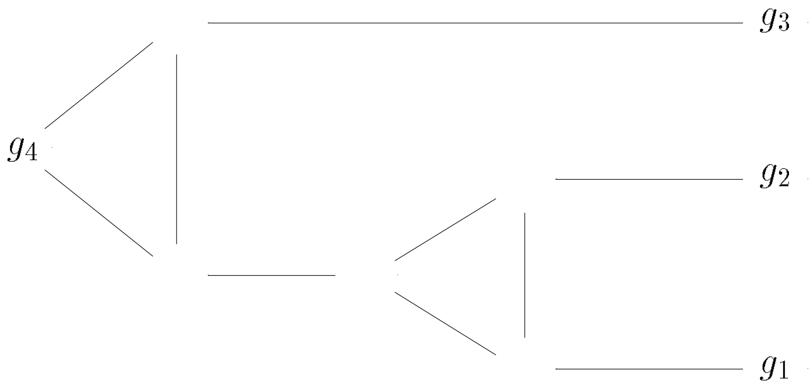

Figure 3.1: The gadget in the reduction from 3-SAT to MAX-3-COLORABILITY

all the gadget heads to B, and B to GROUND. Last, it identifies the three legs of the gadget of C_i with the vertices that correspond to the literals of C_i.

Our extension proceeds as follows. We duplicate the vertex B m times to get $B_1, B_2, \ldots, B_m$, which are all connected to GROUND, and we connect them to the m gadget heads using a bipartite expander. Namely, one side of the expander (which we call *the upper side*) contains $B_1, B_2, \ldots, B_m$, and the other (*the lower side*) contains the gadget heads. The resulting graph is illustrated in Figure 3.2.

Note that, for simplicity, we have drawn the vertex GROUND twice in the figure. Note also that the number of edges in the output graph is $O(m)$.

Lemma 3.9 *If φ is satisfiable, then G^φ is 3-colorable.*

Proof: We use the colors T,F,G. Color the vertex GROUND with the color G. Color each literal-vertex with T if the corresponding literal is assigned TRUE by the satisfying assignment τ of φ, or with F otherwise. Color $B_1, B_2, \ldots, B_m$ with F and the gadget heads with T. It remains to color the body vertices of the clauses-gadgets. Recall that each gadget head is colored T and that since the truth assignment τ satisfies φ, then at least one gadget leg must be colored T (This is the literal-vertex that corresponds to the literal that is assigned TRUE by τ). By property $\mathcal{P}$ of the gadget, we get that it is possible to complete the coloring of all the gadget-bodies consistently. ■

Lemma 3.10 *If there exists a 3-coloring of G^φ which induces δm inconsistent edges, then there is an assignment to φ that satisfies at least $\left(1 - \frac{2B\delta}{\gamma}\right) m$ clauses in φ, where $1+\gamma$ is the*

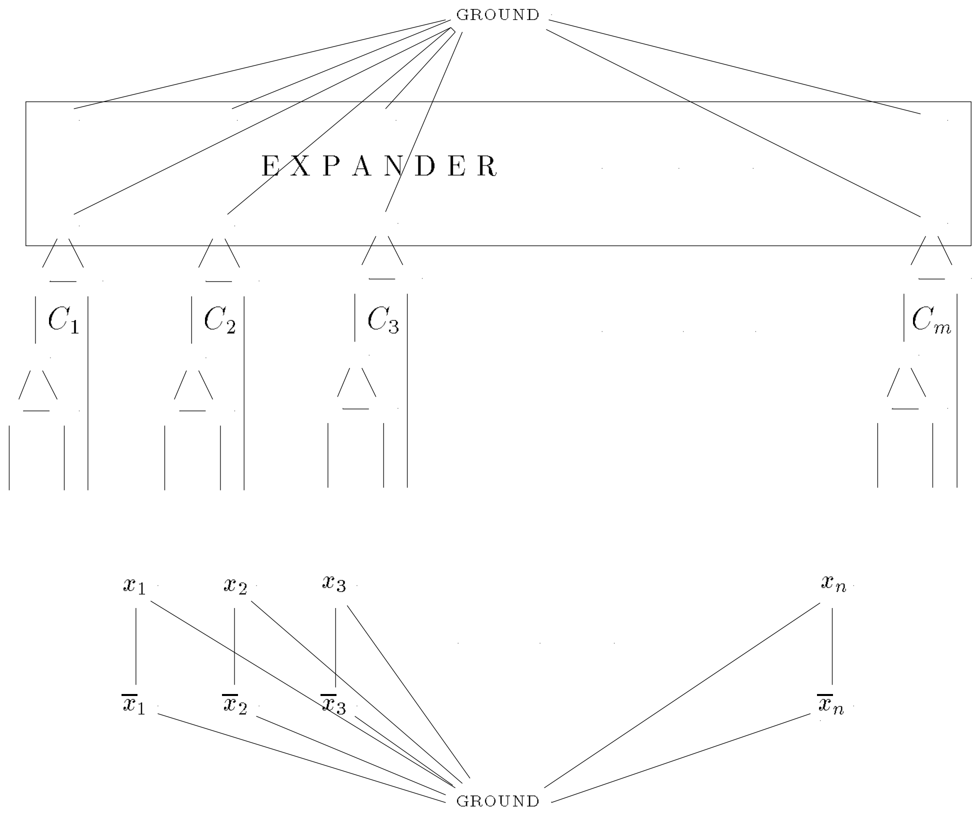

Figure 3.2: The reduction from 3-SAT to MAX-3-COLORABILITY

expansion rate of the polynomial time constructible expander that is used in the construction of G^φ.

Proof: The lemma is trivially valid for $\delta \geq 1/4$, since for any formula there exists an assignment that satisfies at least half of its clauses. Therefore, we restrict ourselves to $\delta < 1/4$. Given a 3-coloring of G^φ, we first select names for the 3 colors and define an assignment to the variables of φ. Denote the color of the vertex GROUND by G. The majority color in the heads of all gadgets is denoted T (if there are two candidate colors appearing the same number of times, select one of them arbitrarily). Note that the color T is different from G since all the m gadget-heads are connected to the GROUND vertex and we have less than $m/4$ inconsistent edges. The third color is denoted F. Now, to fix the assignment for the variable x_i, $1 \leq i \leq n$, we consider the vertex labeled x_i. If it is colored T, we assign the value TRUE to x_i. Otherwise, the assignment for x_i is FALSE. We claim that this assignment satisfies at least $\left(1 - \frac{2B\delta}{\gamma}\right) m$ clauses.

By the property of the gadget, there is no consistent 3-coloring of the vertices of the gadget such that the head of the gadget is colored T and its three bottom vertices (the three literal vertices) are colored F. Therefore, if the gadget of the clause C_j, $1 \leq j \leq m$, has all its edges consistent and its head colored T, then one of its literals must be colored T or G. In other words, if a clause $C_j = (l_{j_1} \vee l_{j_2} \vee l_{j_3})$, $1 \leq j \leq m$, is not satisfied by our assignment, i.e., all its literals are assigned FALSE by our assignment, then one of the following conditions must be met:

1. One of the edges in the gadget of C_j is inconsistent.

2. The gadget head is not colored T.

3. One of the literals in the clause is assigned FALSE by our assignment, and its vertex is not colored F.

We show that these three conditions cannot be met "too many times" in the graph G^φ by showing that fulfillment of these conditions implies inconsistent edges. Denote by K_1, K_2 and K_3 the number of times that conditions 1, 2, and 3 are met in the graph G^φ. Clearly, the number of clauses that are not satisfied by our assignment is at most $K_1 + K_2 + K_3$. We claim that

$$K_1 + \frac{\gamma}{2} K_2 + \frac{1}{B} K_3 \ < \ \delta m. \tag{3.1}$$

Assuming this (the proof follows) we get that

$$K_1 + K_2 + K_3 \;\leq\; \frac{2B}{\gamma}\delta m$$

and therefore, the number of clauses that are not satisfied by our assignment is at most $\frac{2B\delta}{\gamma}m$, as needed.

It remains to prove Equation (3.1), i.e., to show that there are at least $K_1 + \frac{\gamma}{2}K_2 + \frac{1}{B}K_3$ inconsistent edges in G^{φ}. We partition the edges in the graph into three disjoint subsets, and give a lower bound on the number of inconsistent edges in each subset (given K_1, K_2, and K_3). Since the subsets are disjoint, we can sum these lower bounds into a single lower bound on the number of inconsistent edges in the graph. The first subset consists of the edges inside the gadgets. By the definition of K_1, we have at least K_1 inconsistent edges in this subset. Next, we consider all the edges that connect the literal-vertices to each other and to the vertex GROUND. Recall that each literal appears in at most B clauses and therefore, if we have K_3 clauses connected to literals having the property of condition 3, then there are at least $\frac{1}{B}K_3$ literals that are assigned FALSE by our assignment and whose vertices are not colored F. We would like to show that for each such literal, there is a unique inconsistent edge. If the vertex l_i or the vertex $\overline{l_i}$ is colored G, then there exists an inconsistent edge between that vertex and the vertex GROUND and we are done for that literal. So, assume this is not the case. Since the vertex l_i is not colored F and not colored G, then it is colored T, and since we assigned FALSE to the literal l_i, the vertex $\overline{l_i}$ must be colored T also and we get an inconsistent edge $(l_i, \overline{l_i})$. Note that for each vertex that satisfies the above condition we have a different inconsistent edge. Therefore, we have at least $\frac{1}{B}K_3$ inconsistent edges in the second subset.

The remaining edges consist of the edges of the expander and the edges that connect the expander to the vertex GROUND. We claim that if K_2 vertices of the lower side of the expander are not colored T, then at least $\frac{\gamma}{2}K_2$ edges in this set are inconsistent. Denote by l_1, l_2, and l_3 the number of vertices in the lower side of the expander that are colored T, F, and G correspondingly. Recall that $K_2 = l_2 + l_3$ is the number of expander lower vertices (which are gadget-head vertices) that are not colored T. We show that the number of inconsistent edges in this last set of edges is at least

$$\max(l_3, \gamma l_2) \;\geq\; \frac{\gamma}{2}K_3.$$

Clearly, there are at least l_3 inconsistent edges in this set because there are l_3 different edges that connect the vertex GROUND, which is colored G, to gadget-head vertices with the same color. On the other hand, consider the l_2 expander lower vertices that are colored F. By definition of the color names, l_2 is smaller than half the number of the vertices in the lower side of the expander. Using the expansion property of the expander, these nodes have a set of $(1+\gamma)l_2$ neighbors in the upper side of the expander. Denote this set of neighbors by S. We cannot use the expansion property again on the set S since we are not sure that it is small enough. However, we know that S is adjacent to at least $|S|$ vertices on the lower side. Furthermore, we can associate with each vertex in S a unique neighbor on the lower side. (This holds for all known expander constructions. Yet, this property can be simply achieved (without foiling the expander) by adding m edges that connect each vertex i on one side to vertex i on the other side.) Now, only l_2 of the vertices in S can have their associated neighbors in the lower side colored F. The other γl_2 members of S have their γl_2 twin vertices on the lower side colored G or T. Putting it all together, we have at least γl_2 vertices in S which, on one hand, are connected to a vertex colored F (by the definition of the set S), and on the other hand, are connected to unique lower vertices that are not colored F. Consider such a vertex in S. If it is colored F, then we have a unique inconsistent edge between this vertex and an F-colored lower vertex. If it is colored G, then we get a unique inconsistent edge between this vertex and the vertex GROUND. The last possibility is that this vertex is colored T. If its twin vertex is also colored T, then we have an inconsistent edge between them. Otherwise, the unique neighbor is colored G (since we know that it is not F), and we also end up with a unique inconsistent edge between the lower neighbor and the vertex GROUND. This completes the proof of Lemma 3.10, and of Theorem 3.8. ■

3.3 The Hardness of MAX NAE 3SAT and MAX 3DM

Two more problems were shown to have a hard gap at locations less than 1 although it is NP-hard to decide if an instance has its best solution at location 1. These are MAX NOT-ALL-EQUAL 3SAT [PY-91] and MAX 3-DIMENSIONAL MATCHING [Kan-91]. In this section, we show that these problems have a hard gap at the gap location 1. Let us begin with defining the problems.

Definition 3.11 (THE PROBLEM MAX NOT-ALL-EQUAL 3SAT).
Input: *A 3-CNF formula φ.*
Problem: *Find the maximum number of clauses that contain at least one true literal and at least one false literal, over all truth assignments to the variables of φ.*

Definition 3.12 (THE PROBLEM MAX 3-DIMENSIONAL MATCHING-B).
Input: *A set $M \subseteq W \times Y \times Z$, where W, Y, and Z are disjoint finite sets and each element in $W \cup Y \cup Z$ appears in the triplets of M at most B times.*
Problem: *Find the maximum number of elements in $W \cup Y \cup Z$ which appear exactly once in the triplets of M', over all $M' \subseteq M$.*

Theorem 3.13 *The problem* MAX NOT-ALL-EQUAL 3SAT *possesses a hard gap at location 1.*

Proof: We first reduce 3SAT to NOT-ALL-EQUAL 4SAT, and then reduce NOT-ALL-EQUAL 4SAT to NOT-ALL-EQUAL 3SAT. In the first reduction, we simply add a new variable to all clauses. Note that for the NOT-ALL-EQUAL problem, the number of "satisfied" clauses does not change if we use an assignment σ or its complement $\overline{\sigma}$ (the complement of an assignment is an assignment that gives the opposite value to each of the variables in the formula). Therefore, we can fix the new variable to always be assigned FALSE without changing the solution to this optimization problem, and the maximum number of clauses that can be satisfied in the input formula (to the reduction) is exactly the number of clauses that can have both a false and a TRUE literal simultaneously in the output formula. In the second reduction, we treat each clause $C_j = (l_{j_1} \vee l_{j_2} \vee l_{j_3} \vee l_{j_4})$ that contains four literals by adding a new variable y_j and replacing C_j with the two clauses: $(l_{j_1} \vee l_{j_2} \vee y_j) \wedge (\overline{y_j} \vee l_{j_3} \vee l_{j_4})$. It is easy to verify that there exists an assignment to the variables of the original formula, in which at least one of the literals $l_{j_1}, l_{j_2}, l_{j_3}, l_{j_4}$ is TRUE and at least one is false iff there exists an assignment to the variables of the new formula such that both sets of literals $\{l_{j_1}, l_{j_2}, y_j\}$ and $\{\overline{y_j}, l_{j_3}, l_{j_4}\}$ contain at least one TRUE literal and at least one false literal. Hence, the hard gap at location 1 is preserved. Note that the width of the new hard gap is at least half the width of the original gap. This is because the number of clauses is at most twice the number of clauses of the original formula, and the number of clauses that can not be satisfied does not decrease.

Theorem 3.14 *For any $B \geq 3$,* MAX 3-DIMENSIONAL MATCHING-B *possesses a hard gap at location 1.*

Proof: A slight modification of the original reduction from SAT to 3-DIMENSIONAL MATCHING [Kar-72] works as a gap preserving reduction from MAX-SAT-B to MAX 3-DIMENSIONAL

MATCHING-3. We shortly describe the reduction, following the presentation of Garey & Johnson (1979). [GJ-79]. Given a formula φ with n variables and m clauses in which each variable appears at most B times, we construct three disjoint sets $W^\varphi, Y^\varphi, Z^\varphi$ and a set of triplets $M^\varphi \subseteq W^\varphi \times Y^\varphi \times Z^\varphi$. The triplets in M^φ consist of n truth-setting components (one for each variable), m satisfaction-testing components (one for each clause), and a "garbage collection" mechanism.

Let x_i be a variable that appears d_i times in the formula. The truth-setting component of x_i involves "internal" elements $a_i[k] \in W^\varphi$, $b_i[k] \in Y^\varphi$ and "external" elements $x_i[k], \overline{x}_i[k] \in Z^\varphi$, for $1 \leq k \leq d_i$. We call the $a_i[k]$'s and the $b_i[k]$'s internal because they appear only inside their truth-setting component. The external $x_i[k]$'s and $\overline{x}_i[k]$'s appear in their truth-setting components as well as in other components (which we describe later). The triplets making up the truth-setting component can be divided into two sets:

$$T_i^t = \{(\overline{x}_i[k], a_i[k], b_i[k]) \,:\, 1 \leq k \leq d_i\},$$

$$T_i^f = \{(x_i[k], a_i[k+1], b_i[k]) \,:\, 1 \leq k < d_i\} \cup \{(x_i[d_i], a_i[1], b_i[d_i])\}.$$

The property of this component is that any matching that covers all internal elements $a_i[k], b_i[k]$, $1 \leq k \leq d_i$, exactly once contains either exactly all triplets in T_i^t or exactly all triplets in T_i^f. Thus, we get that either all elements $x_i[k]$ are covered and all elements $\overline{x}_i[k]$ are not (we associate this with assigning FALSE to x_i), or all elements $\overline{x}_i[k]$ are covered and all elements $x_i[k]$ are not (this is associated with assigning TRUE to x_i). Note that the number of elements produced so far is $O(n)$ since each variable appears at most B times in φ. (We remark that the original reduction produced $O(nm)$ elements, having m external elements for each variable).

The satisfaction-testing component that stands for the clause C_j $(1 \leq j \leq m)$ contains two internal elements $s_1[j] \in W^\varphi$ and $s_2[j] \in Y^\varphi$, and at most three external elements from the truth-setting components that correspond to the literals in C_j. If C_j contains the k-th appearance of the variable x_i then we add the triplet $(x_i[k], s_1[j], s_2[j])$ if x_i appears positively in C_j or the triplet $(\overline{x}_i[k], s_1[j], s_2[j])$ if x_i appears negated in C_j. These components add $2m$ elements to the output. So far, each internal element appears in at most three triplets of M^φ and each external element appears at most twice.

Note that a matching that covers all internal elements exactly once and which does not use an external element more than once corresponds to a truth assignment that satisfies φ. Given an assignment τ that satisfies φ, we select the triplets of T_i^t to be in the matching if x_i is assigned TRUE by τ or the triplets of T_i^f otherwise. This leaves all elements $x_i[k]$ ($1 \leq k \leq d_i$) uncovered if x_i is assigned TRUE by τ or all elements $\overline{x}_i[k]$ uncovered otherwise. For each clause C_j, we select a literal that is assigned TRUE (such a literal must exist since τ satisfies φ). The element that corresponds to the appearance of this literal in C_j is not covered, since its literal is assigned TRUE. Thus, we may choose the triplet that contains this element to cover $s_1[j]$ and $s_2[j]$.

In order to cover the remaining uncovered external elements, we use a garbage collection mechanism. The original mechanism is too big (it contains $O(nm)$ elements) and does not meet the demand that each element appears in at most three triplets. We present an appropriate garbage collecting mechanism later.

Consider the other direction, in which we are given a good matching and we would like to build a satisfying assignment to φ. In order to show a hard gap in MAX 3-DIMENSIONAL MATCHING-B, we shall show that if there are "only few" violations in the given matching, then there is a truth assignment that satisfies "almost all" clauses in φ. More formally, if there exists a matching M' for which the number of internal elements which appear more than once or none at all plus the number of external elements that appear more then once is at most δm, then there exists an assignment that satisfies more than $(1-\delta B)m$ clauses.

To fix an assignment for x_i, consider the truth-setting component of x_i. We assign TRUE to x_i if the M' contains a triplet in T_i^t. Otherwise, x_i is assigned FALSE. Note that if this truth assignment does not satisfy a clause C_j, then one of the following conditions must be met.

1. The internal elements $s_1[j]$ and $s_2[j]$ do not appear in the matching M'.
2. The elements $s_1[j]$ and $s_2[j]$ appear in a triplet that contains a literal which was assigned FALSE by our assignment.

Suppose condition (1) is met K_1 times and condition (2) is met K_2 times in M'. The number of clauses in φ that are not satisfied by our assignment is at most $K_1 + K_2$. To conclude, we show that the number of violations in the matching M' is at least $2K_1 + \frac{1}{B}K_2$. Clearly, each time condition (1) is met, we have two unique internal elements that are not covered by M'. If condition (2) is met K_2 times in M', then there are at least $\frac{1}{B}K_2$ literals that are assigned FALSE and that have an associated external element covered by a satisfaction-testing component. We claim that the truth-setting component of this literal has either an internal element that does not appear uniquely in the matching M', or an external element that appears more than once. Suppose all internal elements appear exactly once. By the property of the truth-setting component, we know that all external elements associated with the literal that was assigned FALSE by our assignment are covered by the truth-setting component. Thus, the external element appears in M' at least twice.

It remains to show how to do garbage collection with $O(m)$ elements such that no element appears in more than three triplets. Note that the existence of a hard gap is already proven, but we must show a garbage collection mechanism in order to place the hard gap at location 1. An elegant way to solve both problems of the original garbage collection mechanism is due to Garey and Johnson (private communications). Create three independent copies of the construction described, with the roles of W^{φ}, Y^{φ}, and Z^{φ} cyclically permuted between the three copies. Now, for each external element $x_i[k]$ and $\overline{x}_i[k]$ (currently included in just two triplets), add a single triplet including the three copies (one from each copy of the overall construction). This method shrinks the width of the gap by a constant factor (at location 1), and therefore is sufficient to prove the theorem.

3.4 Some More Hard Problems

In the introduction, we discussed the difference between optimization problems in which we seek the value of the largest solution and optimization problems in which we seek the quality of the best solution. In this section, we consider three optimization problems that seek the quality of the best solution, and which were considered before only in the largest solution version. We believe that these new optimization problems are interesting and so we give their definitions and investigate their hardness properties. Specifically, we treat approximation versions of VERTEX COVER, SET SPLITTING, and EDGE COLORING (CHROMATIC INDEX).

Let us start with the definitions. We say that an edge (v_i, v_j) in a graph $G(V, E)$ is *covered* by a subset $V' \subseteq V$ if $v_i \in V'$ or $v_j \in V'$.

Definition 3.15 (THE PROBLEM MAX VERTEX COVER-B).
Input: *A graph $G(V, E)$ with degree at most B and an integer K.*
Problem: *Find the maximum number of edges in G that V' covers, over all subsets $V' \subseteq V$ of cardinality K.*

Note the difference between this problem and the optimization problem MIN VERTEX COVER treated in [PY-91]. Their problem was to minimize the size of the vertex cover (which covers all edges). In MAX VERTEX COVER, we are given the size of the cover, K, in the input, and we are trying to maximize the number of edges that are covered.

Definition 3.16 (THE PROBLEM MAX SET SPLITTING).
Input: *A collection C of subsets of a finite set S.*
Problem: *Find the maximum number of subsets in C that are not entirely contained in either S_1 or S_2, over all partitions of S into two subsets S_1 and S_2.*

We say that a vertex $v \in V$ is consistent regarding an edge-coloring of a graph $G(V, E)$ if no two edges of the same color are adjacent to v.

Definition 3.17 (THE PROBLEM MAX k-EDGE COLORABILITY).
Input: *A graph $G(V, E)$ of degree k.*
Problem: *Find the maximum number of consistent vertices, over all edge-colorings of G with k colors.*

Theorem 3.18 *The following problems possess a hard gap at location 1:*

1. MAX VERTEX COVER-B
2. MAX k-EDGE COLORABILITY (CHROMATIC INDEX)
3. MAX SET SPLITTING

PROOF:

1. We use the identity reduction from MIN VERTEX COVER-B (a problem that was shown hard in [PY-91]). If there is a vertex cover of G of cardinality K that covers at least $(1-\epsilon)|E|$ edges in G, then there is a cover of all edges with at most $K + \epsilon|E|$ vertices. Note that $|E| \leq |V| \cdot B/2$, and recall that the hard gap for MIN VERTEX COVER-B was shown hard for $K = \Theta(|V|)$.

2. Use the original reduction of MAX-SAT to k-EDGE COLORABILITY [Hoy-81, LG-83]) only let the domain of the reduction be MAX 3SAT-B (shown hard in [PY-91]). Clearly, if there is a satisfying assignment to the input formula, then there is an edge-coloring of the output graph with k colors such that all the vertices are consistent. It is left to show that if there is a k-coloring of the edges with a small number of inconsistent vertices, then there is an assignment that satisfies almost all clauses in the input formula. This can be done using an accountancy similar to the one of Lemma 3.10.

3. The trivial reduction from MAX NOT-ALL-EQUAL 3SAT works. ■

Remark 3.19 *The approximation of* MAX SET SPLITTING *remains hard even if all subsets in C are of cardinality less than or equal to* 3 *(see the proof).*

We follow by defining the problem MAX-QER.

Definition 3.20 (THE PROBLEM MAX-QER: MAX QUADRATIC EQUATIONS OVER THE RATIONALS).
Input: *A set of quadratic equations over the rational field in the variables $x_1, \ldots, x_n$.*
Problem: *Find the maximum number of equations that are satisfied, over all assignments of rational numbers to $x_1, \ldots, x_n$.*

Theorem 3.21 MAX-QER *has a hard gap at location 1.*

Proof: Reduce MAX-3SAT-B to MAX-QER in the following way. For each variable x, add the equation $x^2 = x$. For each clause $C_i = (x \vee y \vee z)$ add a new variable c_i and produce two equations. First, write $c_i = xy$, and then use it to reduce the degree of the equation $(1-x)(1-y)(1-z) = 0$ to 2, by substituting each appearance of xy with c_i.

Note that the same proof is valid over the field of real numbers as well.

Bibliography

[ABV-95] W. Aiello, M. Bellare, and R. Venekatesan . Knowledge on the Average – Perfect, Statistical and Logarithmic. *Proceedings of the 27rd Annual ACM Symposium on the Theory of Computing,* ACM (1995).

[AH-87] W. Aiello and J. Håstad. Perfect Zero-Knowledge can be Recognized in Two Rounds. *Proceedings of the 28th Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1987).

[AJ-87] M. Ajtai. Recursive Construction for 3-Regular Expanders. In *Proc. 28th IEEE Symp. on Foundations of Computer Science*, pages 295-304, 1987.

[ADLRY-92] N. Alon, R.A. Duke, H. Lefmann, V. Rödl, and R. Yuster, The Algorithmic Aspects of the Regularity Lemma. In *Proc. 33th Ann. Symp. Found. Comput. Sci.*, 1992, 473–482.

[ALMSS-92] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, "Proof Verification and Intractability of Approximation Problems." 33st FOCS, 1992.

[AS-92] S. Arora and S. Safra. Probabilistic Checking of Proofs: A New Characterization of NP. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, 1992.

[Bab-85] L. Babai. Trading group theory for randomness. In *Proc. 17th ACM Symp. on Theory of Computing*, pages 421–420, 1985.

[BFL-90] L. Babai, L. Fortnow, and C. Lund, "Non-Deterministic Exponential Time Has Two-Prover Interactive Protocols," 31st FOCS, 1990, pp. 16-25.

[BBFG-91] R. Beigel, M. Bellare, J. Feigenbaum and S. Goldwasser. Languages that are Easier than their Proofs. *Proceedings of the 32nd Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1991).

[B-92] M. Bellare. Interactive Proofs and Approximations. Research Report 17969 (#78973), IBM research Division, T.J. Watson Research Center, Yorktown Heights, NY 10598. 1992.

[BFK-95] M. Bellare, U. Feige, J. Kilian. " On the Role of Shared Randomness in Two Proof Systems", In *Proceedings of the Third Israel Symposium on the Theory of Computing and Systems*, pp. 199-208, 1995.

[BGLR-93] M. Bellare, S. Goldwasser, C. Lund, and K. Russel. "Efficient Probabilistically Checkable Proofs: Applications to Approximation", *Proceedings of the 25rd Annual ACM Symposium on the Theory of Computing,* ACM (1993).

[BMO-90] M. Bellare, S. Micali and R. Ostrovsky. The (True) Complexity of Statistical Zero-Knowledge. *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing,* ACM (1990).

[BP-92a] M. Bellare and E. Petrank. Making Zero-Knowledge Provers Efficient. *Proceedings of the 24rd Annual ACM Symposium on the Theory of Computing,* ACM (1992).

[BP-92b] M. Bellare and E. Petrank. Quadratic Equations Over the Rationals. private communication, 1992.

[B+-88] M. Ben-Or, S. Goldwasser, O. Goldreich, J. Håstad, J. Kilian, S. Micali and P. Rogaway. Everything Provable is Provable in Zero-Knowledge. *Advances in Cryptology — Proceedings of CRYPTO 88,* Lecture Notes in Computer Science 403, Springer-Verlag (1989). S. Goldwasser, ed.

[BR-92] M. Bellare and P. Rogaway. The Complexity of Approximating a Nonlinear Program. Research Report 17831 (#78493) IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY 10598. 1992.

[BGKW-88] M. Ben-Or, S. Goldwasser, J. Kilian and A. Wigderson. Multi-prover interactive proofs: How to remove intractability. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 113–131, 1988.

[Ber-73] C. Berge. *Graphs and Hypergraphs.* North-Holland, Amsterdam, 1973.

[BMT-78] E.R. Berlekamp, R.J. McEliece, and H.C.A. van Tilborg. On the Inherent Intractability of Certain Coding Problems. *IEEE Trans. Information Theory*, 1978.

[BP-89] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Information Processing Letters*, vol. 32, pages 171–176, 1989.

[BJLTY-90] A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis. Linear Approximation of Shortest Superstrings. *Proc. 31st Symp. on Foundations of Comp. Sc.*, pages 554-562, 1990.

[BK-89] M. Blum and S. Kannan. Designing Programs that Check their Work. *Proceedings of the 21st Annual ACM Symposium on the Theory of Computing,* ACM (1989).

[BHZ-87] R. Boppana, J. Håstad and S. Zachos. Does *co-NP* Have Short Interactive Proofs". *Information Processing Letters*, Vol 25 (1987), No. 2, pp 127–132.

[CW-79] L. Carter and M. Wegman. Universal Classes of Hash Functions. *J. Computer and System Sciences* **18**, 143–154 (1979).

[Coo-71] S. A. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd ACM Symp. on Theory of Computing*, pages 151–158, 1971.

[DJPSY-92] E. Dahlhaus, D. S. Johnson, C. H. Papaditriou, P. D. Seymour and M. Yannakakis. The complexity of multiway cuts. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 241–251, 1992.

[DL-81] W. Fernandez de la Vega and G.S. Lueker. Bin Packing can be solved within $1+\epsilon$ in Linear Time. *Combinatorica*, vol. 1, pages 349–355, 1981.

[Fe-87] U. Feige. Interactive Proofs. M.Sc Thesis, Weizmann Institute of Science. August 1987.

[FGLSS-91] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 2–12, 1991.

[F-89] L. Fortnow. The Complexity of Perfect Zero-Knowledge. *Advances in Computing Research (ed. S. Micali)* Vol. 18 (1989).

[FRS-88] L. Fortnow, J. Rompel and M. Sipser. On the power of multi-prover interactive protocols. In *Proc. 3rd IEEE Symp. on Structure in Complexity Theory*, pages 156–161, 1988.

[GG-81] O. Gabber and Z. Galil. Explicit Construction of linear sized superconcentrators. *J. of Comp. and Sci.*, vol 22, pages 407-420, 1981.

[GJ-76] M.R. Garey and D.S. Johnson. The complexity of near-optimal graph coloring. *J. of the ACM*, vol 23, pages 43-49, 1976.

[GJ-79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[GJS-76] M.R. Garey, D.S. Johnson and L.J. Stockmeyer Some Simplified NP-Complete Graph Problems. *Theor. Comput. Sci.*, Vol 1, pages 237-267, 1976.

[GK-90] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *Proceedings of ICALP 90*.

[GMS-87] O. Goldreich, Y. Mansour and M. Sipser. Interactive Proof Systems: Provers that never Fail and Random Selection. *Proceedings of the 28th Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1987).

[GMW-86] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design", *Proc. 27th FOCS 86,* See also *Jour. of ACM.* Vol 38, No 1, July 1991, pp. 691–729.

[GMW-87] O. Goldreich, S. Micali, and A. Wigderson, "How to Play any Mental Game or a Completeness Theorems for Protocols of Honest Majority", STOC87.

[GO-89] O. Goldreich and Y. Oren. Definitions and Properties of Zero-Knowledge Proof Systems. Technical Report #570, Technion (1989).

[GOP-94] O. Goldreich, R. Ostrovsky, and E. Petrank. Computational Complexity and Knowledge Complexity. *Proceedings of the 26rd Annual ACM Symposium on the Theory of Computing,* ACM (1994).

[GP-91] O. Goldreich and E. Petrank. Quantifying Knowledge Complexity. *Proceedings of the 32nd Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1991). Technical Report 683, Computer Science Dept., Technion - Israel Institue of Technology.

[GMR-85] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proofs. *Proceedings of the 17th Annual ACM Symposium on the Theory of Computing,* ACM (1985).

[GMR-89] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proofs. *SIAM J. Comput.* **18** (1), 186-208 (February 1989).

[GS-89] S. Goldwasser, and M. Sipser, Private Coins vs. Public Coins in Interactive Proof Systems, *Advances in Computing Research (ed. S. Micali),* 1989, Vol. 5, pp. 73-90.

[H-94] J. Håstad. Perfect Zero-Knowledge in $\mathcal{AM} \cap$ co-$\mathcal{AM}$. Unpublished 2-page manuscript explaining the underlying ideas behind [AH-87]. 1994.

[HPS-93] J. Hastad, S. Phillips, and S. Safra, A well Characterized Approximation Problem. In *Proceedings of the 2nd Israel Symposium on Theory of Computing and Systems*, 1993, 261–265.

[Hoy-81] I. Hoyler. The NP-Competeness of Edge Coloring. *SIAM J. of Computation*, vol. 10, pages 718–720, 1981.

[ILe-90] R. Impagliazzo and L.A. Levin, No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random, *31st FOCS*, pp. 812-821, 1990.

[ILL-89] R. Impagliazzo, L. Levin and M. Luby. Pseudo-Random Generation from One-Way Functions. *Proceedings of the 21st Annual ACM Symposium on the Theory of Computing,* ACM (1989).

[ILu-90] R. Impagliazzo and M. Luby, One-Way Functions are Essential for Complexity Based Cryptography, *30th FOCS*, pp. 230–235, 1990.

[IY-87] R. Impagliazzo and M. Yung. Direct Minimum-Knowledge computations. *Advances in Cryptology — Proceedings of CRYPTO 87,* Lecture Notes in Computer Science 293, Springer-Verlag (1987).

[JVV-86] M. Jerrum, L. Valiant and V. Vazirani. Random Generation of Combinatorial Structures from a Uniform Distribution. *Theoretical Computer Science* **43**, 169-188 (1986).

[Kan-91] V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, vol. 37, pages 27-35, 1991.

[Kar-72] R.M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[KMR-92] D. Karger, R. Motwani and G. D. S. Ramkumar. On Approximating the Longest Path in a Graph. Manuscript, 1992.

[KK-82] N. Karmakar and R.M. Karp. An Efficient Approximation Scheme For The One-Dimensional Bin Packing Problem. In *Proc. 23rd IEEE Symp. on Foundations of Computer Science*, pages 312–320, 1982.

[LG-83] D. Leven and Z. Galil. NP-completness of finding the chromatic index of regular graphs. *J. of Algorithms* 4, pages 35-44, 1983.

[Lev-73] L. Levin. Universal'nyĭe perebornyĭe zadachi (Universal search problems, in Russian), In *Problemy Peredachi Informatsii*, vol. 9, pages 265–266, 1973. A corrected English translation appears in an appendix to Trakhtenbrot [Tra-84]

[LFKN-90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 2–10, 1990.

[Ost-91] R. Ostrovsky. One-Way Functions, Hard on Average Problems, and Statistical Zero-Knowledge Proofs. Structures 1991.

[OVY-91] R. Ostrovsky, R. Venkatesan and M. Yung. Fair Games Against an All-Powerful Adversary. *AMS DIMACS Series in Discrete Mathematics and Theoretical Computer Science.* Vol 13. (Jin-Yi Cai ed.) pp. 155-169.

[OW-93] R. Ostrovsky and A. Wigderson. One-Way Functions are Essential For Non-Trivial Zero-Knowledge, *Proc. 2nd Israeli Symp. on Theory of Computing and Systems*, 1993.

[Lov-73] L. Lovász. Coverings and Colorings of Hypergraphs. *Proc. 4-th Southern Conference on Combinatorics, Graph Theory, and Computing*, Utilitas Mathematica Publishing, Winnipeg 3-12, 1973.

[LY-92] C. Lund and M. Yannakakis. On the Hardness of Approximating Minimization Problems. *Proceedings of the 25rd Annual ACM Symposium on the Theory of Computing,* ACM (1993).

[MS-81] F. MacWilliams and N. Sloane. *The Theory of Error-Correcting Codes.* North-Holland, 1981.

[Mar-73] G.A. Margulis. Explicit Constructions of Concentrators. *Prob. Per. Infor.* vol. 9, pages 71-80, 1973. (English translation in *Problems of Infor. Trans.*, pages 325-332, 1975).

[Mot-92] R. Motwani. Lecture Notes on Approximation Algorithms. Technical Report, Dept. of Computer Science, Stanford University (1992).

[Or-87] Y. OREN. On The Cunning Power of Cheating Verifiers: Some Observations About Zero Knowledge Proofs. *Proceedings of the 28th Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1987).

[Os-91] R. OSTROVSKY. One-Way Functions, Hard on Average Problems, and Statistical Zero-Knowledge Proofs. Structures 1991.

[OVY-90] R. OSTROVSKY, R. VENKATESAN AND M. YUNG. On the Complexity of Asymmetric Games. Manuscript (1990).

[PY-91] C. H. Papadimitriou and M. Yannakakis. Optimization, Approximation, and Complexity Classes. *Journal of Computer and System Sciences*, vol. 43, pages 425–440, 1991.

[PY-92] C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two, *Mathematics of Operations Research*, to appear.

[P-92] E. Petrank. The Hardness of Approximation : Gap Location. *Computational Complexity*, Vol. 4, 1994. pp. 133-157.
A preliminary version of this paper appeared in the *Second IEEE Israel Symp. on Theory of Computation and Systems*, June 1993, pp. 275-284.

[SG-76] S. Sahni and T. Gonzalez. P-complete approximation problems. *JACM*, vol. 23, pages 555–565, 1976.

[Sha-90] A. Shamir. IP=PSPACE. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 11–15, 1990.

[Si-83] M. SIPSER. A Complexity Theoretic Approach to Randomness. *Proceedings of the 15th Annual ACM Symposium on the Theory of Computing,* ACM (1983).

[St-73] L.J. Stockmeyer, Planar 3-Colorability is NP-Complete In *SIGACT News*, Vol 5(3), pages 19–25, 1973.

[St-83] L. Stockmeyer. The Complexity of Approximate Counting. *Proceedings of the 15th Annual ACM Symposium on the Theory of Computing,* ACM (1983).

[TW-87] M. Tompa and H. Woll. Random Self-Reducibility and Zero-Knowledge Proofs of Possession of Information. *Proceedings of the 28th Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1987).

[Tra-84] B. A. Trakhtenbrot. A survey of Russian approaches to *Perebor* (brute-force search) algorithms. In *Annals of the History of Computing* vol. 6, pages 384–400, 1984.

[Yan-92] M. Yannakakis. On the Approximation of Maximum Satisfiability. In *Proc. 3rd Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 1–9, 1992.

[Z-92] D. Zuckerman. NP-Complete Problems Have a Version That's Hard to Approximate. Unpublished Manuscript. 1992.