Interactive and Probabilistic Proof-Checking

Luca Trevisan[*]

January 18, 2000

Abstract

The notion of efficient proof-checking has always been central to complexity theory, and it gave rise to the definition of the class NP. In the last 15 years there has been a number of exciting, unexpected and deep developments in complexity theory that exploited the notion of *randomized* and *interactive* proof-checking. Results developed along this line of research have diverse and powerful applications in complexity theory, cryptography, and the theory of approximation algorithms for combinatorial optimization problems.

In this paper we survey the main lines of developments in interactive and probabilistic proof-checking, with an emphasis on open questions.

1 Introduction

The notion of proof-checking is central to complexity theory. The fundamental class NP can be defined as the class of decision problems such that every YES-instance has a short and easy-to-verify proof, where *short* is formalized as *of length bounded by a polynomial in the length of the instance* and *easy* is formalized as *doable in polynomial time.* The principal justification for the conjecture that P $\neq$ NP is the intuitive observation (supported by mathematical practice so far) that proofs are easier to verify than to find, and that conjectures are easier to state than to prove. The theory of NP-completeness allows a unified treatment of the difficulty of thousands of important computational problems, and is applicable to a wide variety of contexts.

Randomized computations also have a central role in the theoretical and applied computer science (three recent books [MR95, Lub96, Gol99] survey some applications to data structures, algorithm design, cryptography and complexity theory). There is a general acceptance of the fact that randomized polynomial-time computation with bounded error-probability are a more appropriate definition of feasible computation than P.

It seems only natural to ask what the notion of proof-checking becomes once we relax our notion of efficient verification to include randomness.

The notion of interaction, another familiar one in computer science, makes also sense in the context of proof-checking. Indeed, when a mathematician explains a proof to his peers, he is typically asked questions and the discussion can branch in direction that were not planned at the beginning. Typically such discussions do not exploit all the details of a proof, but can give high confidence in its validity.

These extension of the notion of a proof might just be regarded as idle exercises (we may imagine fault-tolerant proof-systems, distributed proof-systems etc., by this token), and they would be so if their purpose was just to play with the definitions. Indeed, a sequence of unexpected and very

[*]`luca@cs.columbia.edu`. Columbia University, Department of Computer Science. Work done at DIMACS.

successful developments over the last fifteen years has shown that these definitions give rise to some classes containing important problems previously lacking a right characterization, and, more stunningly, some other such classes coincide with standard complexity classes, defined without reference to randomness and interaction. A beautiful theory has emerged to prove such results, and a number of important applications have arisen, most notably to cryptography and to the theory of approximation algorithms.

Overview of the Rest of the Paper. We divide our exposition in three sections, devoted, respectively, to Interactive Proofs, Zero Knowledge and Probabilistically Checkable Proofs. In each section we will describe the model, state the most important result (or results) and discuss the main technical tools and conceptual steps needed to prove the result. We will also state some open questions: we tried to choose those that promise to branch research into new directions, rather than consolidating what we know. In order to keep this survey short and non-technical, definitions are not fully formalized and the descriptions of the techniques are only very rough sketches. Other, more technical, surveys are referenced below.

Related Work. Oded Goldreich has written several surveys on interactive and probabilistic proof-checking, which are standard references. In particular, Chapter 2 in [Gol99] covers the same topics of this paper. Regarding treatment of specific topics, the classical results on interactive proofs have already appeared in textbooks, such as Sipser's [Sip97]. An extensive treatment of Zero Knowledge is contained in [Gol95]. A survey presenting recent tight results on PCP and non-approximability at an advanced technical level but with a presentation appropriate for graduate students (the equivalent of what [Gol95] does for zero-knowledge) is still to be written. Bellare [Bel96] surveys recent developments in the field of PCP and hardness of approximation, and the role played by the optimization of problem-specific parameters. Bellare's survey appeared two years ago and does not cover some important recent developments. Arora [Aro98] presents a broader overview on PCP and non-approximability, with less emphasis on optimal results. A technical survey by Arora and Lund [AL96] describes in detail how to use results about PCP in order to prove non-approximability results.

2 Interactive Proofs

2.1 The Model, and Its Variants

The formal definition will be more clear if we first provide a concrete example. Suppose we are given two graphs G_1 and G_2, and someone (who will be called the *prover* in the following) is insisting that they are *not* isomorphic, and she is willing to *prove* it to us. There is no known way of presenting concise proofs that two graphs are not isomorphic, but here goes an *interactive* proof. We pick at random one of the two graphs, and we apply a random automorphism (i.e. randomly permute the names of the vertices, and change edges accordingly). Then we present the new graph to the prover, and ask her whether the graph is coming from G_1 or G_2. If the graphs are indeed not isomorphic, then the prover, assuming she has unbounded computational power, can always answer correctly our questions, but if G_1 and G_2 are isomorphic, then she is seeing each time an isomorphic copy of them, and she is unable to guess which graph we started with, and she will answer correctly with probability at most $1/2$ (indeed, exactly $1/2$, as it turns out). By repeating the protocol k times, the error probability can be reduced from $1/2$ to 2^{-k}. The basic idea is even more apparent in the following familiar setting: a friend claims that she is able to tell apart Coke from Pepsi. Then we set up the following experiment: we fill two equal glasses one with Coke and one with Pepsi, we choose a random one, and we ask the friend to drink and guess which soft drink she is drinking. If the friend is really able to distinguish them, she will answer correctly each time, otherwise she will be only able to randomly guess, and so she will be mistaken half of the times.

Formally, we have a computational model made by two communicating Turing machines, one, called the *verifier* and denoted by V is randomized and polynomial-time bounded, the other, called the *prover* and denoted by P is unbounded.[1] The two machines share a common input x and interact by sending messages in rounds. For a verifier V and a prover P, we denote by V^P the computation of V while it interacts with P. Then we say that a language L is in $IP(k)$ if there exists a verifier V and a prover P such that

- if $x \in L$ then V^P accepts x with probability 1.
- if $x \notin L$ then for every P', $V^{P'}$ accepts x with probability $\leq 1/2$.
- For every prover P', the computation $V^{P'}$ proceeds in at most k rounds of communication.

The first condition is a *completeness* condition, saying that every "correct theorem" of the form $x \in L$ has a valid proof that is accepted with probability 1. The second condition is an approximate *soundness* condition, that says that incorrect theorems may have purported proofs that fool the verifier sometimes, but only with bounded probability.

The model was developed independently by Goldwasser Mical and Rackoff and by Babai. The two paper [GMR89] and [Bab85] describe slightly different models: in particular, in [Bab85] the prover knows the random choices of the verifier. Equivalently, the verifier is restricted to send, each time, its random choices. Goldwasser and Sipser [GS86] show that the two models are the same. Their techniques, based on universal hashing, have found several other applications in the field.

We will denote by IP the class IP$(n^{O(1)})$. The class IP(1), also called MA, is the simplest case of interactive proof, and in fact it involves no interaction at all. The only message being sent is sent, without loss of generality, from the prover. It can be seen as a proof that is sent over, and then verified off-line. The only difference with NP is the randomness in the verification procedure. If sufficiently good pseudorandom generators exist, then the verification can be simulated in deterministic polynomial time. In particular, it is implicit in [IW97] that under a believable complexity assumption we have MA = NP. We remark that there is no interesting problem known to be in MA but not in NP. By [Bab85, GS86] it follows that for every constant k it holds IP(k) = IP(2) = AM, where AM is defined as the class of languages having interactive proofs where the verifier sends to the prover a random string, the prover answers, and then the verifier decides whether to accept or to reject, deterministically. As will be stated later, AM contains interesting problems not known to be contained in NP. It is also conceivable that NP = AM [KvM99].

2.2 Main Results

By definition NP $\subseteq$ IP(k) for every k. The papers introducing the notion of interactive proofs had some examples of problems having interactive proofs and not known to be in NP: quadratic non-residuosity [GMR89] and some group-theoretic problems [Bab85]. Goldreich et al. [GMW91] devised the graph non-isomorphism protocol described above. Neither of these problems is known

[1]One may wonder whether it is restrictive to assume that the prover is a Turing machine as opposed to an abstract device that can even solve undecidable problems; it turns out that every prover can be replaced by a prover in PSPACE [Fel86].

(nor believed to be) *co*-NP-hard, and for some time it was believed that IP was just slightly extending the class NP. In particular it appeared to be unlikely that interactive proofs would capture *co*-NP. The intuitive reason is that a proof of a *co*-NP complete statement has to provide evidence of exponentially many special cases, and it seemed unlikely that randomness and interaction would make this doable in polynomial time. The following result gave partial evidence in this direction.

Theorem 1 ([BHZ87]) *If* $co\text{-NP} \subseteq \text{IP}(O(1))$ *then the polynomial hierarchy collapses.*

So, under a standard complexity-theoretic assumption, *co*-NP statements do not admit interactive proofs that use a constant number of rounds. This was a very important result since, when it is combined with the IP(2) protocol for graph non-isomorphism of [GMW91] we have that graph non-isomorphism is not coNP-hard, i.e. graph isomorphism is not NP-hard (unless the polynomial hierarchy collapses). This approach has been used recently by Goldreich and Goldwasser [GG98] to show that finding *approximate solutions* for certain lattice minimization problems cannot be NP-hard (unless the polynomial hierarchy collapses), even though polynomial-time approximation algorithms are not known.

It was conjectured that Theorem 1 could be extended to IP, i.e. that one could show evidence that $co\text{-NP} \not\subseteq \text{IP}$. It was also known that there exists an oracle relative to which IP does not contain *co*-NP[FS88].

In a major breakthrough, Lund et al. [LFKN92] showed that *co*-NP is indeed contained in IP, i.e. one can prove tautologies using randomness and interaction in polynomial time. The result of [LFKN92] was even more general: they showed that the entire polynomial hierarchy is contained in IP. Shortly after, Shamir [Sha92] proved that PSPACE is contained in IP. Since the opposite containment was also known [Pap85, Fel86], the following new characterization of PSPACE had ben derived.

Theorem 2 ([LFKN92, Sha92]) IP=PSPACE.

Unlike other results mentioned in this survey, Theorem 2 has no known application outside the theory of complexity classes. But within this area its impact has been dramatic. For starters, Theorem 2 has been the first result about natural complexity classes to be proved in contrast to a relativization result. This has somewhat reduced the influence of relativization results in complexity theory. Theorem 2 also disproves the "random oracle conjecture" [CCG$^+$94]. The techniques introduced by Lund et al. [LFKN92] demonstrated the power of representing boolean formulae and assignments using polynomials, a methodology that has been refined and applied in a variety of randomized proof-checking models. Finally, the results of Lund et al. [LFKN92] demonstrated the unsuspected power of interactive proofs, and started the series of developments that will be reviewed in the rest of this paper.

2.3 Low-Degree Polynomials

In this section we will sketch the proof that $co\text{-NP} \subseteq \text{IP}$.

The proof goes as follow. Consider the following problem: given an instance φ of Max 3SAT and a number k, decide whether the instance has exactly k satisfying assignments. This problem is clearly *co*-NP-hard (unsatisfiability is the special case where $k = 0$).

Fix some big prime M (exponentially big in the number of clauses and variables in φ). It is possible to associate to φ a multivariate polynomial p over Z_M such that the number of satisfying assignments of φ is exactly

$$\sum_{a_1,\ldots,a_n \in \{0,1\}} p(a_1, \ldots . a_n)$$

and the polynomial has degree at most 3 (sums are taken mod M). The polynomial p has a compact implicit representation, and the verifier can evaluate it in points of its choice.

The protocol works as follows:

- The prover sends a univariate polynomial q_1 that is claimed to be equal to $p_1(x) := \sum_{a_2,\ldots,a_n} p(x, a_1, \ldots, a_n)$. Since p_1 is a univariate polynomial of low degree, it has a compact representation, and can indeed be sent efficiently to the verifier.
- The verifier checks that $q_1(0) + q_1(1) = k$. If this test is passed, then it just suffices to verify that $q_1 \equiv p_1$.

  The verifier picks a random $v_1 \in Z_M$ and then sends it to the prover.
- The prover sends back a univariate polynomial q_2 that is claimed to be equal to $p_2(x) := \sum_{a_3,\ldots,a_n} p(v_1, x, a_3, \ldots, a_n)$.
- The verifier checks that $q_2(0) + q_2(1) = q_1(v_1)$. Then it picks a random $v_2 \in Z_M$ and sends it to the prover.
- The prover sends back a univariate polynomial q_3 that is claimed to be equal to $p_3(x) := \sum_{a_4,\ldots,a_n} p(v_1, v_2, x, a_4, \ldots, a_n)$.
- ...
- The prover sends back a univariate polynomial q_n that is claimed to be equal to $p_n(x) := p(v_1, \ldots, v_{n-1}, x)$.
- The verifier picks a random $v_n \in Z_M$ and checks that $p(v_1, \ldots, v_n) = q_n(v_n)$.

If the statement being proved is true, and the prover behaves according to the protocol, then the verifier accepts with probability 1.

Let us see what happens when the statement is not true. If the prover sends polynomials q_i such that $p_i \equiv q_i$ for all i, then the verifier will reject at the very first step. If the prover cheats by sending different polynomials, then with high probability some inconsistency will also arise. This is proved using to the fact that two different degree-3 polynomial over Z_M can only agree on a fraction $3/M$ of their inputs (which is a negligible fraction). That is, one can prove that if $q_1 \neq p_1$, then with high probability the prover is forced to send a $q_2 \neq p_2$ (as with high probability one has $p_2(0) + p_2(1) = p_1(v_1) \neq q_1(v_1)$), and likewise a $q_3 \neq p_3$. By induction one can show that with high probability the prover, to avoid outright rejection from the verifier, has to keep sending $q_i \neq p_i$ at any step. At the end, we will have $q_n \neq p_n$, and so with high probability the final test $p(v_1, \ldots, v_n) = p_n(v_n) = q_n(v_n)$ will reject with high probability.

2.4 Open Question

After the characterization of IP in terms of PSPACE, there has not been much research about interactive proofs (as opposed to zero-knowledge proofs and probabilistically checkable proofs described in the next sections). Due to this state of affairs, some important questions (such as the one below) are still not answered.

Open Question 1 *Is there an interactive proof for co-*NP *where the prover can be implemented in polynomial time having oracle access to an NP oracle, or would such a protocol imply the collapse of the polynomial hierarchy?*

Note that the graph non-isomorphism protocol can be implemented using an oracle for graph isomorphism.[2] Likewise, Shamir's protocol [Sha92] for PSPACE-complete problems can be implemented with a PSPACE prover. It is a standard result that witnesses of NP-statements can be computed in polynomial time with oracle access to an NP-complete problems. Proving a statement is typically not much harder that deciding whether the statement is true or not; it is regrettable that we do not know whether this is true in the fundamental case of *co*-NP statements (e.g. propositional tautologies).

There might be a connection between Question 1 and the question of whether NP-complete problems can have adaptive random self-reductions. Even establishing formally a relation between these two problems would be an interesting result.

3 Zero Knowledge

3.1 The Model and Its Variants

Let us revisit the graph non-isomorphism protocol as described in Section 2.1. Every time the verifier V asks a question to the prover P, V knows which answer would be correct in the case where the graphs are isomorphic[3]. Therefore the verifier is not convinced of the non-isomorphism of the graph because it received some hints as of why the graphs are not isomorphic, but just because it has evidence that had the graphs been isomorphic, the prover would have not been able to guess all the answers all the times.

Since the verifier always knows the answers that it expects from the prover, it is fair to say that after a valid proof-interaction is completed for a valid statement, the verifier *has learnt nothing but the validity of the statement.*

The notion of zero-knowledge proof captures the above intuition (the proof-system for graph non-isomorphism is a special case of a zero-knowledge proof-system). In general, a zero knowledge proof-system is such that the exchange of messages between the prover and the verifier is not giving any information to the verifier because such exchange can be "replicated" off-line by a polynomial-time randomized algorithm (called the *simulator*).

More formally, a language L has a *perfect zero-knowledge proof-system* (denoted $L \in$ PZK) if there exists an interactive protocol (P, V) for L and a randomized algorithm S such that on input $x \in L$ the outputs of S is a random variable which has the same distribution as the sequence of messages exchanged by P and V on input x (the sample space of the latter distribution is given by the random choices of V).[4]

Since what V is receiving from the prover is information that could have been computed from x efficiently, a PZK protocol satisfies the intuitive requirement that P does not give to V any

[2]On the other hand, it is not known hot to implement the protocol of [GMR89] for quadratic non-residuosity using an oracle for quadratic residuosity.

[3]In contrast, in the protocol for *co*-NP problems the verifier would not be able to compute by itself the answers that it expects from the prover.

[4]In fact this is the definition of *honest-verifier zero-knowledge.* In the more general, and useful, definition, we have the stronger requirement that for *every* verifier V' there is a simulator S' for the distribution of exchanges between V' and P.

information (at least not any information that V could have not computed by itself in polynomial time). Note that graph non-isomorphism is an example of problem in PZK.

One can define two more general (and interesting) classes, by relaxing the requirement on the simulator.

- A statistical zero knowledge protocol is defined similarly to a PZK protocol, except that the output of S is only required to be statistically close[5] to the distribution of exchanges between P and V. The class of decision problems that admit a statistical zero knowledge protocol is denoted by SZK.
- A computational zero knowledge protocol is defined similarly to a PZK protocol, except that the output of S is only required to be computationally indistinguishable[6] from the distribution of exchanges between P and V. The class of decision problems that admit a computational zero knowledge protocol is denoted by CZK.

In a statistical zero knowledge protocol, the prover is still not giving any information, in a strong information theoretic sense. In a computational zero-knowledge proof-system the prover may leak information, but such information is hard to extract, so for all practical purposes no information is being leaked.

3.2 Main Results

Computational Zero Knowledge

Computational zero-knowledge proofs are quite powerful.

Theorem 3 ([GMW91]) *Assuming one-way functions exist, every language L in* NP *has a* CZK *proof-system (V_L, P_L). Furthermore, on input $x \in L$, prover P_L can be implemented in polynomial time given access to a standard* NP*-witness for x.*

In other words, the result of Goldreich et al. [GMW91] shows that if we have an instance of a NP-problem and a witness for it, e.g. we have a CNF formula and a satisfying assignment, we can prove in zero-knowledge to a verifier that the formula is satisfiable, and we can answer in polynomial time to all the questions of the verifier. Several cryptographic protocols can be implemented using the result of Theorem 3 as a primitive.

The statement of Theorem 3 is a *conditional* result, because it assumes the existence of one-way functions. It appears that, given current techniques, a cryptographic assumption is necessary and the existence of one-way functions is the weakest possible. Indeed, Goldreich et al. [GMW91] show how to prove the theorem under the assumption that "commitment schemes" exist. It has been proved later that the commitment schemes exist if one-way functions exist.

For cryptographic applications it is essential that the prover strategy be implementable in polynomial time, given access to some polynomially short amount of additional information. Under such conditions, proof-systems can exist only for IP(1) problems.[7]

[5]Two random variables X and Y on $\{0,1\}^n$ are said to be statistically close if for every subset $S \subseteq \{0,1\}^n$ it holds $|\mathbf{Pr}[X \in S] - \mathbf{Pr}[Y \in S]| \leq neg(n)$ for some function neg that goes to zero faster than any inverse polynomial.

[6]Two random variables X and Y on $\{0,1\}^n$ are said to be computationally indistinguishable if for every subset $S \subseteq \{0,1\}^n$ computable in polynomial time it holds $|\mathbf{Pr}[X \in S] - \mathbf{Pr}[Y \in S]| \leq neg(n)$ for some function neg that goes to zero faster than any inverse polynomial.

[7]Indeed, Theorem 3 can be generalized to hold for IP(1), but this is not a very interesting extension since, as already mentioned, there is no interesting problem in IP(1) which is also not known to be in NP.

Even if there are no cryptographic consequences, it is of interest to see whether, giving up the requirement of efficient prover implementation, Theorem 3 can be further extended. It turns out that the strongest possible extension is true, namely every problem in IP (and hence every problem in PSPACE) admits computational zero knowledge proofs.

Theorem 4 ([IY87, BOGG$^+$88]) *If one-way functions exist, then* CZK= PSPACE.

Statistical Zero Knowledge

Interestingly, one of the main results on statistical zero-knowledge relates to the weakness of the notion.

Theorem 5 ([For87, AH91]) SZK $\subseteq$ AM $\cap$ *co*-AM.

Therefore, in light of Theorem 1 we have that if a problem has a statistical zero-knowledge proof-system then the problem is not NP-hard (unless the polynomial hierarchy collapses).

Indeed SZK is a very interesting complexity class, since it contains presumably hard problems, such as graph non-isomorphism, quadratic non-residuosity, and a problem equivalent to the discrete logarithm, and yet it cannot contain NP-hard problem. Furthermore, SZK has a number of closure properties. In particular it is closed under the complement [Oka96] and under truth-table reductions [SV99] and it has a natural complete problem [SV97]. Several technical choices in the definition of SZK yield the same class (e.g. whether the prover knows the coin tosses of the verifier or not, whether the existence of the simulator is guaranteed only for "honest verifiers" or not, and so on), which is therefore very robust.

Theorem 5 also applies to PZK as a special case. Some of the results about SZK are not known to have analogs in PZK. In particular it is not known whether PZK has a natural complete problem.

3.3 Open Questions

One outstanding open question about SZK is how "hard" it is as a class, i.e. under which sufficient conditions we can have BPP $\neq$ SZK. For example it is not known whether SZK contains a problem as hard as factoring.

Open Question 2 *Prove that if one-way functions exist then* BPP $\not\subseteq$ SZK.

A weak converse of this result is known to be true.

4 Probabilistically Checkable Proofs

4.1 The Model and the Main Result

In probabilistically checkable proofs, proof-verification is randomized but *not* interactive. In this model a proof is provided to a randomized verifier, who is trying to ascertain its correctness very efficiently and without reading it entirely. The main result is that every NP proof can be encoded in such a way that a randomized verifier can check it by only reading a constant number of bits and using a logarithmic number of random bits. A correct proof is accepted with probability 1, a "proof" of a wrong statement is accepted with probability $\leq 1/2$.

When the verifier uses a logarithmic amount of randomness and runs in polynomial time, Probabilistically Checkable Proofs are a model that restricts NP (as opposed to IP, that is a generalization) since the proof can be assumed to be of polynomial size (with no loss of generality) and a polynomial-time randomized verification procedure that uses a logarithmic number of random bits can be simulated deterministically in polynomial time.

An earlier result was that NP proofs can be encoded in such a way that if *also the input* is encoded, then verification can be done in polylogarithmic time. In principle this could have practical application to the presentation of long and complicated proofs, such as the classification of simple groups, the four colors theorem, or Fermat's last theorem. However these encodings are applicable only if the proof is completely presented in a formal proof-system, and such encodings would be enormous.

We now give some formal definitions. In partial contrast with the previous section, we will call a *verifier* a polynomial time randomized oracle machine. A verifier V is $(r(\cdot), q(\cdot))$*-restricted* if on input a string x of length n, V queries the oracle at most $q(n)$ times and uses at most $r(n)$ random bits.

A language L admits a probabilistically checkable proof of completeness c, soundness s, query complexity q, randomness complexity r, written $L \in \mathrm{PCP}_{c,s}[r(\cdot), q(\cdot)]$ if there exists a $(r(\cdot), q(\cdot))$-restricted verifier V such that for every x

- If $x \in L$, then there exists a P such that $\mathbf{Pr}[V^P(x) \text{ accepts }] \geq c$
- If $x \notin L$, for all P, $\mathbf{Pr}[V^P(x) \text{ accepts }] \leq s$.

The main result about this model is stated as follows.

Theorem 6 (The PCP Theorem [AS98, ALM$^+$98]) $\mathrm{NP} = \mathrm{PCP}_{1,1/2}[O(\log n), O(1)]$.

Up to multiplicative constants, the result is optimal in the sense that one cannot make less than a constant number of queries, and it is possible to show that if $\mathrm{NP} \subseteq \mathrm{PCP}_{1,1/2}[o(\log n), o(\log n)]$ then $\mathrm{P} = \mathrm{NP}$ [AS98].[8]

The PCP Theorem is of interest especially for its application to proving the hardness of approximating several important optimization problems. For some of these optimization problems, techniques based on Probabilistically Checkable Proofs enable to prove hardness of approximation results that essentially match the performances of known approximation algorithms, and therefore completely resolve the question of the approximability of these problems.

Examples of non-approximability results implied by stronger versions of the PCP Theorem can be found in [Bel96, Aro98].

4.2 Origins of the Result and Overview of the Techniques

The model of Probabilistically Checkable Proofs (and the main results about it) has a long and very interesting story.

As mentioned before, the assumption that one-way functions exist appears to be unavoidable in order to prove Theorem 3. Ben-Or et al. [BOGKW88] considered a model of zero-knowledge where the verifier can interact with two (or, more generally, several) provers, who are all computationally unbounded *but* unable to communicate with each other once the protocol starts. The model can be visualized as detectives interrogating two suspects of a crime in different rooms and asking them similar questions to see if they contradict each other. The contribution of [BOGKW88] was

[8]This result has been sometimes misquoted as saying that $\mathrm{PCP}_{1,1/2}[o(\log n), o(\log n)] = \mathrm{P}$, which is unlikely to be true.

to show that a result analogous to Theorem 3 holds without cryptographic assumption in the multiple-provers setting. For cryptographic purposes, it is not unreasonable to consider a model with two non-communicating devices trying to convince a third party of the truth of a statement. For example a person may have two smart cards, and insert them in different slots of an ATM; the smart cards would be physically unable to communicate with each other, and would have to convince the ATM machine that they hold the secret key that gives access to the owner's account.[9] If the ATM has been hacked and is trying the steal the secret key, it will be unable to do that. Even if the multiple-provers model was primarily designed for zero-knowledge, it also defines a complexity class in the ordinary sense, which is called MIP. That is, MIP is the class of decision problems that admit interactive proofs with several non-communicating provers (indeed, it can be shown that two provers always suffice to simulate polynomially many provers). Fortnow, Rompel and Sipser [FRS88] show that the class MIP has the following equivalent characterization: it can be seen as the class of languages that admit exponentially long proofs of membership that can be checked in polynomial time by a randomized verifier (with bounded error probability). This class is clearly contained in NEXP, where NEXP is the class of decision problems that admit exponentially long proofs that can be checked in exponential time in the length of the input (but, without loss of generality, in polynomial time in the length of the proof itself).

Initially, it was conjectured that even MIP was only a small extension of NP, and that *co*-NP $\not\subseteq$ MIP. Shortly after Shamir's proof that IP = PSPACE [Sha92], Babai, Fortnow and Lund [BFL91] showed that MIP = NEXP. This is a truly impressive result, though the results of [LFKN92, Sha92] somewhat spoiled the surprise. The MIP = NEXP result says that for every language that admits exponentially long proofs, such proofs can be encoded in such a way that a polynomial-time randomized verifier can check them. The verifier will accept correct proofs with probability 1, and "proofs" of incorrect statements with probability $\leq 1/2$ (or, equivalently, with probability exponentially small in the length of the input). So the verifier becomes convinced of the validity of the proof even if it only looks at a negligible part of the proof itself.

It is natural to ask whether polynomially long proofs can be checked in polylogarithmic time. This question has to be phrased carefully, since a polylogarithmic time verifier cannot even read the instance, which makes it impossible to verify a proof for it. However if both the instance and the proof are encoded in a proper (efficiently computable) way, then Babai et al. show that polylogarithmic time verification is possible [BFLS91]. A variant of this result was also proved by Feige et al. [FGL$^+$91]: they show that NP-proofs have a quasi-polynomial length encoding (i.e. an encoding of length $n^{O(\log\log n)}$) such that a polynomial-time verifier can verify the correctness of the proof in polynomial time by using $O(\log n \log\log n)$ random bits and reading $O(\log n \log\log n)$ bits of the proof. The main result of Feige et al. [FGL$^+$91] was to show a connection between the computational power of such a model and the hardness of approximating the Max Clique problem.[10] They show how to encode the computation of the verifier into a graph in such a way that there is a correspondence between proofs and cliques, and between the probability that the proof be accepted and the size of the clique. A gap in the acceptance probability of YES instances and NO instances translates into a hardness of approximation result for Max Clique. The complexity of finding approximate solutions for the max Clique problem had been an open question for a long time, at least since Johnson's influential paper [Joh74] that explicitly mentions the question of finding approximation algorithms for this problem.

The result of Feige et al. [FGL$^+$91] can be written as NP $\subseteq$

[9]In this example, the smart cards are the provers and the ATM is the verifier.

[10]A clique in a graph is a subset of vertices that are all pairwise adjacent. The Max Clique problem is, given a graph, to find the largest clique.

$\mathrm{PCP}_{1,1/2}[O(\log n \log\log n), O(\log n \log\log n)]$.

Arora and Safra [AS98] introduced several new ideas to improve on [FGL+91], and proved that $\mathrm{NP} = \mathrm{PCP}_{1,1/2}[O(\log n), O(\sqrt{\log n})]$. The main contribution of Arora and Safra is the idea of "composing" proof systems together. They first devise a modified version of the proof system of Babai et al. [BFLS91], that shows $\mathrm{NP} = \mathrm{PCP}_{1,1/2}[O(\log n), \mathrm{poly}\log n]$ (the improvement over the proof-system of [BFLS91] is that the amount of randomness is logarithmic, rather than polylogarithmic). Then they use the following line of reasoning: fix a language L and a $(O(\log n), \mathrm{poly}\log n)$-restricted verifier V for it. On an input x, simulate the computation of this verifier, and look at the $\mathrm{poly}\log n$ queries made by it. Now we have to check that V would accept the answer to these queries. The decision of V as of whether to accept or not is computed in time polynomial in the number of queries (in the construction of [BFLS91, AS98]), and can be seen as an instance of size $\mathrm{poly}\log n$ of a problem in $\mathrm{P} \subseteq \mathrm{NP}$. But then the proof-system can be recursively applied on each of these checking statements, and the resulting number of queries will be $\mathrm{poly}\log(\mathrm{poly}\log) = \mathrm{poly}(\log\log)$. In fact things are much more complicated than as described, and even in order to prove that $\mathrm{NP} = \mathrm{PCP}_{1,1/2}[O(\log n), O(\sqrt{\log n})]$ Arora and Safra have to overcome major technical difficulties and they have to provide a strong analysis of a "low-degree test".

The result of Arora and Safra gives NP-hardness of approximating Max Clique (as opposed to intractability assuming $\mathrm{NP} \not\subseteq \mathrm{DTIME}(n^{O(\log\log n)})$) and also gives a *characterization* of NP.

The next step was to realize that the reduction from PCP to Max Clique was not a isolated accident. Sudan and Szegedy (as credited in [ALM+98]) discovered that the computations of a $(O(\log n), O(1))$-restricted verifier can be encoded as instances of the Max 3SAT problem. Then, using the web of reductions between optimization problems initiated by Papadimitriou and Yannakakis [PY91], this also implies that the strength of $(O(\log n), O(1))$-restricted verifiers implies the hardness of approximating several important problems including the Traveling Salesman Problem and the Steiner Minimal Tree problems in metric spaces. This was a strong motivation to prove the PCP Theorem, that came only a few months after the initial circulation of the paper of Arora and Safra [AS98].

The proof of the PCP Theorem [ALM+98] is very long and involved. It uses the idea of composition of Arora and Safra [AS98] along with several other technical ingredients. One of them is a $(O(n^3), O(1))$-restricted verifier, that is used in composition with a $(O(\log n), \mathrm{poly}\log n)$-restricted verifier that is different from the one of [AS98] in that even if it reads $\mathrm{poly}\log n$ bits from the proof it only reads $O(1)$ blocks, each made of $\mathrm{poly}\log n$ bits (the verifier of [AS98] does not have this property). A further strengthening of the low-degree test is needed for this. We will call a verifier $(r(n), q(n), a(n))$-restricted if it uses $r(n)$ random bits and queries the proof $q(n)$ times, each time retrieving $a(n)$ bits.

At a very high level, the PCP Theorem is proved using the following steps:

1. Construct a $(O(\log n), O(1), \mathrm{poly}\log n)$-restricted verifier.

2. Construct a $(O(n^3), O(1), 1)$-restricted verifier.

3. Compose the verifier of Step (1) with itself, and derive a $(O(\log n), O(1), \mathrm{poly}\log\log n)$-restricted verifier.

4. Compose the verifier of Step (3) with the verifier of Step (2), and derive a $(O(\log n), O(1), 1)$-restricted verifier.

The first step is the hardest, and it is the main contribution of [ALM+98]. The second step has a very elegant, and not exceedingly hard, construction. The other steps are just simple invocation of results about composed verifier. Specifically, when a $(r_1(n), O(1), a_1(n))$-restricted verifier is composed to a $(r_2(n), O(1), a_2(n))$-restricted verifier, one obtains, roughly, a $(r_1 + r_2(a_1(n)), O(1), a_2(a_1(n)))$-restricted verifier.

Several stronger characterization of PCP in terms of PCP have appeared, which include [BGLR93, FK94, BS94, BGS98, Hås97a, Hås97b, RS97, AS97, Tre98, GLST98, ST98]. Much of this subsequent work has been motivated by the goal of providing stronger non-approximability results for optimization problem. A recent survey of Arora [Aro98] is a good starting point to find reference about such work.

An interesting variant of the MIP=NEXP result was more recently discovered by Micali [Mic94]. Micali defines the notion of a *computationally sound* (abbreviated CS) proof system as follows. In a CS proof system, the prover computes a proof for a given statement, and sends it to the verifier; then the verifier checks the validity of the proof without further interaction. So far, the model is the same as the one used to define the class MA. The difference is in the soundness condition: in the definition of MA (and of interactive proofs in general) we demand that, for an incorrect statement, there is no prover strategy that will make the verifier accept with probability more than 1/2. In the case of CS proofs, one restricts this condition only to *"efficiently" computable* prover strategies, and also restricts to protocols where the correct prover strategy is "efficiently" computable. The definition of efficient depends on the class of languages that one is trying to capture. It is conjectured that every problem in EXP has a CS proof system where the prover runs in exponential time, and the computational soundness condition holds with respect to exponential time computable prover strategies (as usual, the verifier runs in polynomial time). While such a characterization is still an open question, Micali was able to prove it in the *random oracle model* [Mic94] (i.e. in a somewhat artificial setting where both the prover and the verifier have access to an exponentially long common random input). This result in the random oracle model is interesting as a "plausibility" result, supporting the conjecture that the same result holds without random oracles. Furthermore, the result in the random oracle model had an application to cryptography in recent work by Canetti et al. [CGH98].

4.3 Open Question

The progress on hardness of approximation has been steady and strong, but there are still several problems, especially several minimization problems, for which the gap between the approximation factors given by known algorithms and the known intractability results is huge. Determining the approximability of these problems, with stronger PCP results and/or new reductions, is the most important open question in the field.

Indeed, in this survey we prefer to put more emphasis on open questions that point towards new directions, rather than doing progress on established ones, and we believe that the following open question (which has been asked several times) is fundamental.

Open Question 3 *Find a simple proof of the PCP Theorem.*

This is a very general question, and it is of interest to find more specific challenges. It has been suggested to focus on statements which: (i) can be derived from (but are weaker than) the PCP Theorem, (ii) are already surprising enough to be interesting, (iii) are not known to have simple proofs; and to try and find simple proofs for such statements.

One such statement is the existence of *locally checkable* error correcting codes (this approach has been discussed, among other sources, by Arora [Aro94] and Spielman [Spi95]). Say that an error-correcting code $C \subseteq \{0,1\}^n$ having minimum distance δn[11] is "locally-checkable" if there exists a randomized procedure V that on input an element of C accepts with probability 1, and on input a string at distance $> \delta n/3$ from every element of C accepts with probability $\leq 1/2$ (the parameters have been chosen somewhat arbitrarily, but other choices are equivalent up to constant factors in the number of queries). We demand that V makes $O(1)$ queries and uses a logarithmic number of random bits. We would also like C to have a good rate[12]. Note that we are not insisting that C have efficient encoding/decoding algorithms, neither that V be efficiently implementable: we only demand that it reads a constant number of bits from its input and uses a logarithmic number of random bits.

The Hadamard code is locally checkable, but the rate is terrible. Multivariate polynomials can achieve inverse polynomial rate but error detection requires a polylogarithmic, rather than constant, number of queries. The only known way to achieve constant queries and inverse polynomial rate is to essentially do all the steps of the proof of the PCP theorem. It would be nice to have a simple non-constructive proof that such codes exist (even if V and C might be very hard to compute). It is not clear whether constant rate is achievable.

Open Question 4 *Show that locally checkable codes exist with constant rate, or prove that this is impossible.*

Since we are allowing V and C to be arbitrarily hard, this is a purely combinatorial question, with a strong information theoretic taste. Since the question does not involve computations, it should not be impossible to resolve it, either positively or negatively.

A positive resolution would be very interesting, but a negative result would be even more remarkable. Error correcting codes (without the local checkability conditions) are known with a linear minimum distance and constant rate, and such codes are also efficiently encodable and decodable. Showing that local checkability makes these codes impossible (even regardless of efficiency considerations) would disclose an information-theoretic trade-off which would be quite different from whatever we know right now about error-correcting codes.

Another important property of error correcting codes related to PCP constructions is local decodability. We say that a code $C \subseteq \{0,1\}^n$ of minimum distance δn is locally decodable if there exists a randomized algorithm D that on input a string y at distance $< \delta n/3$ from a (unique) element $x \in C$ of the code, and given an index $i \in \{1,\ldots,n\}$ returns a value $D(y,i)$ that with probability at least 3/4 is equal to x_i. Furthermore, D uses a logarithmic number of random bits and reads a constant number of bits from y.

Hadamard codes and codes based on multi-variate polynomials are locally decodable (for multi-variate polynomials the decoding procedure reads a polylogarithmic number of random bits). There is no construction of locally decodable codes based on the PCP Theorem. We note that explicit constructions of locally decodable codes with linear distance and constant rate might have practical application. Whether such codes exist or not is another major open question.

Acknowledgments

This paper is based on a lecture given at a BRICS workshop on Proof Theory and Complexity (PTAC'98). I thank the organizers for their invitation.

[11] δ should be regarded as a fixed small constant in the rest of this section.

[12] The rate of a code is $(\log |C|)/n$, the logarithm of the number of codewords divided by the length of the codewords. The larger is the rate, the more efficient is the code. Ideally, one would like the rate to be a constant independent of n. An *inverse polynomial rate* is a rate of the type $1/n^c$ with $0 < c < 1$.

Thanks to Madhu Sudan for several discussions on locally checkable codes, and to Oded Goldreich, Amit Sahai and Salil Vadhan for discussions on statistical zero knowledge.

References

[AH91] W. Aiello and J. Håstad. Statistical zero-knowledge languages can be recognized in two rounds. *Journal of Computer and System Sciences*, 42:327–345, 1991.

[AL96] S. Arora and C. Lund. Hardness of approximations. In *Approximation Algorithms for NP-hard Problems*. PWS Publishing, 1996.

[ALM$^+$98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in *Proc. of FOCS'92*.

[Aro94] S. Arora. *Probabilistic Checking of Proofs and Hardness of Approximation Problems*. PhD thesis, University of California at Berkeley, 1994.

[Aro98] S. Arora. The approximability of NP-hard problems. In *Proceedings of the 30th ACM Symposium on Theory of Computing*, pages 337–348, 1998.

[AS97] S. Arora and M. Sudan. Improved low degree testing and its applications. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 485–495, 1997.

[AS98] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. Preliminary version in *Proc. of FOCS'92*.

[Bab85] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on Theory of Computing*, pages 421–429, 1985. See also [BM88].

[Bel96] M. Bellare. Proof checking and approximation: Towards tight results. *Sigact News*, 27(1), 1996.

[BFL91] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. Preliminary version in *Proc. of FOCS'90*.

[BFLS91] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd ACM Symposium on Theory of Computing*, pages 21–31, 1991.

[BGLR93] M Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *Proceedings of the 25th ACM Symposium on Theory of Computing*, pages 294–304, 1993. See also the errata sheet in *Proc of STOC'94*.

[BGS98] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCP's and non-approximability – towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998. Preliminary version in *Proc. of FOCS'95*.

[BHZ87] R. Boppana, J. Håstad, and S. Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25:127–132, 1987.

[BM88] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988.

[BOGG+88] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway. Everything provable is provable in zero-knowledge. In *Crypto'88*, pages 37–56, 1988.

[BOGKW88] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of 20th Symposium on Theory of Computing*, pages 113–131, 1988.

[BS94] M. Bellare and M. Sudan. Improved non-approximability results. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 184–193, 1994.

[CCG+94] R. Chang, B. Chor, O. Goldreich, J. Hartmanis, J. Hastad, D. Ranjan, and P. Rohatgi. The random oracle hypothesis is false. *Journal of Computer and System Sciences*, 49(1):24–39, 1994.

[CGH98] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, pages 209–218, 1998.

[Fel86] P. Feldman. The optimum prover lives in PSPACE. Manuscript, 1986.

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

[FK94] U. Feige and J. Kilian. Two prover protocols - low error at affordable rates. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 172–183, 1994.

[For87] L. Fortnow. The complexity of perfect zero-knowledge. In *Proceedings of 19th Symposium on Theory of Computing*, pages 204–209, 1987.

[FRS88] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. In *Proceedings of the 3th IEEE Conference on Structure in Complexity Theory*, pages 156–161, 1988.

[FS88] L. Fortnow and M. Sipser. Are there interactive protocols for co-NP languages? *Information Processing Letters*, 28:249–251, 1988.

[GG98] O. Goldreich and S. Goldwasser. On the limits of non-approximability of lattice problems. In *Proceedings of the 30th ACM Symposium on Theory of Computing*, pages 1–9, 1998.

[GLST98] V. Guruswami, D. Lewin, M. Sudan, and L. Trevisan. A tight characterization of NP with 3 query PCPs. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, pages 8–17, 1998.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. Preliminary version in *Proc of STOC'85.*

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity. *Journal of the ACM*, 38(3), 1991.

[Gol95] O. Goldreich. Foundations of cryptography. Book in preparation, 1995.

[Gol99] O. Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness.* Springer-Verlag, 1999.

[GS86] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the 18th ACM Symposium on Theory of Computing*, pages 59–68, 1986.

[Hås97a] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. Technical Report TR97-38, Electronic Colloquium on Computational Complexity, 1997. Preliminary version in *Proc. of FOCS'96.*

[Hås97b] J. Håstad. Some optimal inapproximability results. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 1–10, 1997.

[IW97] R. Impagliazzo and A. Wigderson. $P = BPP$ unless E has sub-exponential circuits. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 220–229, 1997.

[IY87] R. Impagliazzo and M. Yung. Direct zero-knowledge computations. In *Crypto'87*, pages 40–51, 1987.

[Joh74] D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[KvM99] A. Klivans and D. van Milkebeek. Graph non-isomorphism has subexponential size proofs unless the polynomial hierarchy collapses. In *Proceedings of the 31st ACM Symposium on Theory of Computing*, pages 659–667, 1999.

[LFKN92] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, 1992. Preliminary version in *Proc of FOCS'90.*

[Lub96] M. Luby. *Pseudorandomness and Cryptographic Applications.* Princeton University Press, 1996.

[Mic94] S. Micali. Cs proofs. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, pages 436–453, 1994.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[Oka96] T. Okamoto. On relationships between statistical zero-knowledge proofs. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, pages 649–658, 1996.

[Pap85] C.H. Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31:288–301, 1985.

[PY91] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991. Preliminary version in *Proc. of STOC'88*.

[RS97] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 475–484, 1997.

[Sha92] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992. Preliminary version in *Proc of FOCS'90*.

[Sip97] M. Sipser. *Introduction to the Theory of Computation*. PWS, 1997.

[Spi95] D. Spielman. *Computationally Efficient Error-Correcting Codes and Holographic Proofs*. PhD thesis, MIT, 1995.

[ST98] M. Sudan and L. Trevisan. Probabilistically checkable proofs with low amortized query complexity. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, 1998.

[SV97] A. Sahai and S. Vadhan. A complete promise problem for statistical zero-knowledge. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 448–457, 1997.

[SV99] A. Sahai and S. Vadhan. Manipulating statistical difference. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 43:251–270, 1999.

[Tre98] L. Trevisan. Recycling queries in PCPs and in linearity tests. In *Proceedings of the 30th ACM Symposium on Theory of Computing*, 1998.