Probabilistic Verification of Proofs

Madhu Sudan

Abstract. Recent research in the theory of computing has led to the following intriguing result. "There exists a probabilistic verifier for proofs of mathematical assertions that looks at a proof in only a constant number of bit positions and satisfies the following properties: (Completeness) For every valid theorem there exists a proof that is always accepted. (Soundness) For invalid assertions every purported proof is rejected with some positive probability that is independent of the length of the theorem or proof." This result sheds insight into the fundamental complexity class NP and shows that it is equivalent to a seemingly smaller class of languages with efficient probabilistically checkable proofs. This result is especially significant to combinatorial optimization. For many combinatorial optimization problems it demonstrates that the task of finding even nearly-optimal solutions is computationally intractable. In this article we describe some methods used to construct such verifiers.

1991 Mathematics Subject Classification: 68Q10, 68Q15.
Keywords and Phrases: Computational complexity, Algorithms, Combinatorial optimization, Logic, Probability, Approximation.

1 Introduction

The notion of efficient verification of proofs has been a central theme in the theory of computing. The computational view of this notion abstracts the semantics of the proof system into a verification procedure or *verifier*, i.e., a polynomial time computable Boolean function described by a Turing machine. A purported theorem T and proof π are then just a sequence of bits; π proves T if the verifier accepts the pair (T, π). The purported theorem T is true if such a proof π exists. The class NP [15, 32] represents the class of all theorems with "short" proofs; and allows for very simple combinatorial descriptions of theorems and proofs. As an example, we describe the problem 3-SAT.

A 3cnf formula ϕ is described by N "clauses" $C_1, \ldots, C_N$ on n Boolean variables $x_1, \ldots, x_n$. A clause consists of up to 3 literals (i.e., a variable or its negation) and the clause is satisfied by some Boolean assignment to the variables if at least one literal is assigned a true value. The formula ϕ is said to be satisfiable, if there exists an assignment to the n variables which simultaneously satisfies all clauses. 3-SAT is the language of all satisfiable 3cnf formulae.

The NP-completeness of 3-SAT may be interpreted as follows: For any system of logic, there exists a polynomial time computable function f such given an assertion T in this system of logic and an integer n, $f(T, 1^n)$ computes a 3cnf formula that is satisfiable if and only if T has a proof of length at most n. Thus, under the equivalence class of polynomial time computation, the satisfying assignment to $f(T, 1^n)$ is a proof for the theorem T, and the statement $f(T, 1^n) \in$ 3-SAT is itself the theorem. While this method of describing theorems and proofs is equivalent to any other system of logic it has some conceptual simplicity. One formal effect that captures this simplicity is that an incorrect proof has a very local error: Such an incorrect proof is an assignment that fails to satisfy at least one clause. Hence the three bits corresponding to the assignment to the variables participating in this clause point give the explicit error in the proof. In other words every incorrect proof has a witness of the error that is at most 3 bits long. This example demonstrates some of the power of the computational view of proofs.

Over the course of the last decade a number of new computational notions of proofs have been proposed and analyzed. The common theme in these definitions is a probabilistic verifier who is allowed some small probability of making an error. One of these notions, known as a probabilistically checkable proof (PCP), is motivated by the following informally stated question: "How fast can the verifier be compared to the size of the proof?" It is easy to establish that a deterministic verifier must at the very least "look" at the whole proof. This however need not be true for probabilistic ones. The notion of "looking at a bit of the proof". can be formalized by providing the verifier with oracle access to the proof, i.e., the verifier can specify the address of a location of the proof and gets back the bit written in that location and this entire process takes only as much time as required to write the address. The number of bits of the proof that are "looked" at is now the number of oracle queries. To contrast such a verifier with the traditional verifier, one also quantifies the amount of randomness used by such a verifier. Thus we define $(r(\cdot), q(\cdot))$-restricted PCP verifier to be a probabilistic verifier with random access to a proof oracle, such that on input x of length n, the verifier tosses at most $r(n)$ coins and accesses the oracle at most $q(n)$ times, where the locations accessed are a function of the random coins. A language L is said to be in class $\mathrm{PCP}(r, q)$ if there exists an $(r(\cdot), q(\cdot))$-restricted PCP verifier V satisfying the following: If $x \in L$ there exists an oracle π such that the verifier V accepts x with probability 1 on oracle access to π. If $x \notin L$, for every π, V accepts x with probability at most $1/2$. Notice that the verifier can make mistakes when $x \notin L$. (The definition of a PCP as defined above is from [6]. Many components in this definition come from earlier works: The notion of probabilistic verifiers was first proposed in [25, 10], as part of a larger definition. The notion of oracle machine verifiers was proposed in [22]. The parameters of interest. i.e., $r(\cdot)$ and $q(\cdot)$, were implicit in [19]. A closely related definition focusing on different parameters, termed transparent proofs, was also studied by [9].)

It is immediate from the definition of PCP that $\mathrm{NP} = \cup_{c>0}\mathrm{PCP}(0, n^c)$ (the verifier is not randomized, but is allowed unlimited access to the proof). The results of [8, 9, 19] showed that by allowing the verifier small amounts of randomness, the query complexity can be reduced dramatically and in particular

$\mathrm{NP} \subseteq \cup_{c>0}\mathrm{PCP}(c \log n \log\log n, c \log n \log\log n)$. Subsequently [6, 5] showed that it is possible to restrict the verifier even more significantly to just a *constant* number of queries (independent of the theorem, the proof or the system of axioms). They also reduce randomness to strictly logarithmic in input size. Specifically, they show

THEOREM 1 $\exists q < \infty$ *such that* $NP = \cup_{c>0} PCP(c \log n, q)$.

The consequences to combinatorial optimization may be described informally as follows: The notion of a PCP verifier allows one to formalize the notion of an "approximately" correct proof; and the strong results obtained above show that such a proof exists if and only if a perfectly correct proof exists. Thus the task of finding an approximately correct proof is as hard as the task of finding a perfectly correct proof. The traditional connection between proofs and optimization [15, 29, 32] now indicates that for some optimization problems (unfortunately, not necessarily natural ones) finding near-optimal solutions should be as hard as finding optimal ones. The statement can actually be formalized and made applicable even to natural optimization problems. This connection was discovered by [19] and its applicability was further extended in [5] to apply to a large number of optimization problems considered in [34].

In this article we describe some of the methods used in the construction of probabilistically checkable proofs, from a very high level. In particular, we describe some of the properties that a probabilistically checkable proof must have. We also give a hint of how such properties are effected. The primary hope is to motivate the reader to read more detailed descriptions. The concluding section includes pointers for further reading as well as to more recent work.

2 CONSTRUCTION AND VERIFICATION OF PCPs

In this section we will describe from a high-level the construction of a probabilistically checkable proof. Using the completeness of 3-SAT we will assume that we restrict our attention to theorems of the form $\phi \in$ 3-SAT, where ϕ is a 3cnf formula. It will be useful to think of ϕ as a function mapping $\{0,1\}^n$ to $\{0,1\}$, using the association that 0 represents the Boolean false and 1 represents the Boolean true. $\phi(\vec{a}) = 1$ if ϕ is satisfied by the assignment $\vec{a} \in \{0,1\}^n$. We will switch between 3 possible views of a, the proof of the theorem $\phi \in$ 3-SAT. a may be thought of as a string, as a vector (over some appropriate field containing 0 and 1), or as an oracle that on query i responds with the ith coordinate of a.

Recall that our goal is to describe an alternate proof for $\phi \in$ 3-SAT. More importantly we wish to describe a new probabilistic verifier V for proofs of satisfiability of 3cnf formulae. The verifier will make "few" queries to the new proof, an oracle A, and then cast a verdict. If $\phi \notin$ 3-SAT, then no oracle satisfies the verifier with probability $1/2$, while if $\phi \in$ 3-SAT, then there exists an oracle A such that V always accepts. In the latter case there exists a effective transformation T which transforms the proof $a \in \{0,1\}^n$ satisfying $\phi(a) = 1$ into the the oracle A. It is this transformation that will be our primary focus. For reasons of space, we will focus on the weaker goal of describing a verifier V that makes only $q = q(n) = (\log n)^{O(1)}$ queries and the transformation T for such a verifier.

2.1 MOTIVATION

We start by examining some properties such a transformation T necessarily exhibits. The first interesting property exhibited by $A = T(a)$ is its redundancy. Let us view A as a string, and suppose $\tilde{A}$ is a string obtained by randomly choosing a small fraction of the bits of A and changing them (from 0 to 1 and vice versa). A PCP verifier making q queries still accepts with probability the proof $\tilde{A}$ with high probability, where this probability tends to 1 as the fraction of errors in the proof tend to 0. Furthermore it is possible to determine the acceptance probability of the verifier on string $\tilde{A}$ in polynomial time. Thus even though $\tilde{A}$ is far from A, it preserves its "meaning" (i.e., continues to prove the statement $\phi \in$ 3-SAT.) The easiest conceivable way to achieve such an effect is to insist that $\tilde{A}$ preserves the original proof. i.e., $\vec{a}$ itself (despite the fact that 1% of its bits are erroneous). This leads us to the first property of the transformation $T(\cdot)$ that we will try to achieve. T is an error-correcting code, i.e., for any two strings a_1 and a_2, $T(a_1)$ and $T(a_2)$ differ in a constant fraction of the bits.

In particular this implies that T is an expansive mapping i.e., maps $\{0,1\}^n \rightarrow \{0,1\}^N$ for $N > n$ and hence there are many strings in the range that T does not map to. Given a formula $\phi \notin$ 3-SAT and a string $\tilde{A}$, the PCP verifier has to reject the offered proof with probability at least $\epsilon > 0$ after reading just q bits in such a proof. Furthermore, when the verifier rejects the proof, it must offer an explicit error in the proof in the three bits it reads. The error described may either claim (1) $\tilde{A}$ is not describing any string in the image of T; or (2) $\tilde{A}$ may be the encoding of some string $\vec{a}$; but $\phi(\vec{a}) \neq 1$ for any such string.

To use error of the form (1) above with some string A, it must be that there exist indices $i_1, \ldots, i_q \in [N]$ such that for any string $T(\vec{a})$, the projection to the coordinates $i_1, \ldots, i_q$ does not agree with the projection of A to the same coordinates. We say that an error-correcting code T is q-locally checkable if for every string A that is not in the range of T, there exist indices $i_1, \ldots, i_q \in [N]$ with this property. It will be our goal to come up with an appropriate q-locally checkable code T, for relatively small q.

Finally, T will need to have a "semantic" part: i.e., somehow T must be dependent on ϕ, in order for it to exploit the error condition in (2) above. Summarizing, in the next sections we will describe a transformation T that is an error-correcting code, with good local checkability, that will somehow reveal the truth of the statement $\phi \in$ 3-SAT.

2.2 THE TRANSFORMATION

We start with a simple transformation which leads to some error-correction properties. (Here and later we use $[n]$ to denote the set $\{1, \ldots, n\}$.) The simplest method for adding some error-correcting feature to any information string is to encode it using the Reed-Solomon code. Specifically, to encode the information string $a_1, \ldots, a_n$ we pick a finite field F of order $\Omega(n)$ and an injective map $b : [n] \rightarrow F$.

We then pick a polynomial $P_a : F \to F$ of degree at most n such that $P_a(b(i)) = a_i$ for $i \in [n]$ and our first transformation, denoted T_1 is given by $T_1(a) = (P_a(z))_{z\in F}$ be the encoding of a. Notice that the encoding does not give us a string in $\{0,1\}^N$, but rather an element of $F^{|F|}$ that we view as a string over F. Using the elementary property that two distinct degree n polynomials can agree in at most n places, we find that this transformation is very redundant. Specifically $T_1(a_1)$ and $T_1(a_2)$ have a Hamming distance of at least $|F| - n$ when viewed as strings over F.

The above transformation has the right error-correcting property, but lacks local checkability. To get this additional property, we use the idea of encoding using multivariate polynomials. Specifically, we pick an integer m, and field F (whose size will be determined shortly), a set $H \subset F$, such that $|H|^m \geq n$ and an injective function $b : [n] \to H^m$. To encode a string $\vec{a}$, we pick an m-variate polynomial $P_a : F^m \to F$ of degree $|H|$ in each variable such that $P_a(b(i)) = a_i$ for every $i \in [n]$. (It is easy to prove that such a polynomial P_a always exists.) The total degree of such a polynomial is at most $m|H|$. The encoding of a is then simply the string $T_2(a) = (P_a(z_1,\ldots,z_m))_{z_1,\ldots,z_m\in F}$. Thus $T_2 : \{0,1\}^n \to F^{|F|^m}$ and satisfies the following distance property. For any pair of strings a_1 and a_2, $T_2(a_1)$ and $T_2(a_2)$ agree in at most $m|H|/|F|$ fraction of all indices, when viewed as strings over F. This property follows from a well-known extension of the distance property of polynomials to the multivariate case, which states that a (multivariate) polynomial of total degree d can be zero on at most $d/|F|$ fraction of the domain.

The advantage in using the multivariate polynomials is that they exhibit significantly better local checkability properties. In particular, for every function $Q : F^m \to F$ that is not a polynomial of degree d, there exist $d + 2$ points that "prove" this property. We are now ready to describe some useful choices of m, $|H|$ and $|F|$. (Incidentally, the choice of the function $b : [n] \to H^m$ does not affect the performance of the transformation T_2 in any way.) To get a good locally checkable code one would like to minimize the degree which is at most $m|H|$. However, the choice has to satisfy $|H|^m \geq n$. To ensure that $T_2(a)$ is not too long compared to a, one needs to ensure that $|F|^m$ is only polynomially larger $|H|^m$, which implies $|F|$ should be a polynomial in $|H|$. Furthermore, to get a constant distance, $|F|$ better be larger than the total degree (by at least a constant factor). One such choice of parameters is (we omit floors and ceilings in the following choices): $m = \frac{\log n}{\log\log n}$, $|H| = \log n$, and $|F| = (\log n)^2$. This creates a transformation T_2 which maps n bits to n^2 elements from a field of size $\log^2 n$, with degree and hence q-local checkability for $q \leq \log^2 n$.

We now bring in the semantic element to the error correcting code. This will take some development, so we first outline the plan for this stage. In the final construction $T(a) = T_\phi(a)$ will be a sequence of polynomials $f_0 : F^m \to F$ and $f_1, \ldots, f_k : F^{m'} \to F$, described by their value at every input in $F^{m'}$. $(k, m'$ will be specified later.) The value of the polynomial f_i at some point $u \in F^{m'}$ will be determined by a simple formula — or "construction rule" — applied to the value of the polynomial f_{i-1} at some l places $\psi_{i,1}(u), \ldots, \psi_{i,l}(u)$. (Again, l will be determined shortly.) The polynomial f_0 will be $T_2(a)$. The rules will be constructed so that f_k is identically zero if and only if a satisfies ϕ.

To get such a sequence, we start by "arithmetizing" the notion of a 3-SAT formula and the notion of satisfiability of a clause. Recall that a variable is specified by an index in $[n]$ and thus a literal can be specified as an element of $[n]\times\{0,1\}$. A clause is a triple of literals and thus an element of $[n]\times[n]\times[n]\times\{0,1\}^3$. A 3cnf formula ϕ can thus be described by a function ϕ' mapping $[n]\times[n]\times[n]\times\{0,1\}^3$ to $\{0,1\}$. $\phi'(i,j,k,b_1,b_2,b_3)=1$ iff the clause with literals (i,b_1), (j,b_2) and (k,b_3) occurs in ϕ. Using the function $b:[n]\to H^m$, we can identify a clause with an element of H^{3m+3}. We say that a polynomial $\hat\phi : F^{3m+3}\to F$ of degree less than $|H|$ in each variable is the arithmetization of ϕ if $\phi'(i,j,k,b_1,b_2,b_3)=\hat\phi(b(i),b(j),b(k),b_1,b_2,b_3)$ for any $i,j,k\in[n]$ and $b_1,b_2,b_3\in\{0,1\}$. We will fix $\hat\phi(u)=0$ for all other $u\in H^{3m+3}$. As claimed earlier, it can be shown that such a polynomial $\hat\phi$ does exist and that it can be computed in polynomial time from ϕ.

We now move on to the task of arithmetizing the notion of satisfiability. Given a clause C on literals (i,b_1) (j,b_2) and (k,b_3), and an assignment $a_1,\ldots,a_n$ that has been transformed by the transformation T_2 into the polynomial $f_0:F^m\to F$, notice that the formula $(f_0(b(i))-b_1)\cdot(f_0(b(j))-b_2)\cdot(f_0(b(k))-b_3)$ is 0 if and only if the clause C is satisfied. This leads us to the definition of the polynomial $f_1:F^{3m+3}\to F$ to be

$$f_1(u,v,w,b_1,b_2,b_3)=\hat\phi(u,v,w,b_1,b_2,b_3)(f_0(u)-b_1)(f_0(v)-b_2)(f_0(w)-b_3) \quad (1)$$

where $u,v,w\in F^m$ and $b_1,b_2,b_3\in F$. By construction it is clear that f_1 is a polynomial on $m'=3m+3$ variables having degree at most $2|H|$ in each variable; and furthermore is identically zero on the domain $H^{m'}$ if and only if ϕ is satisfied by the assignment a.

This is close in spirit to what we desire. In what follows, we will develop a sequence of polynomials which will move the condition on f_1 being zero on the domain H^m to the condition that $f_{m'+1}$ being zero on $F^{m'}$. This will be achieved inductively: specifically we will define $f_i:F^{m'}\to F$ to be such that f_{i+1} is zero on the domain $F^i\times H^{m-i}$ if and only if f_i is zero on the domain $F^{i-1}\times H^{m-i+1}$. This is achieved by the following rule.

$$f_{i+1}(r_1,\ldots,r_i;z_{i+1},\ldots,z_{m'})=\sum_{j=1}^{|H|} r_i^j\cdot f_i(r_1,\ldots,r_{i-1};\zeta_j;z_{i+1},\ldots,z_{m'}), \quad (2)$$

where $\zeta_1,\ldots,\zeta_{|H|}$ is any enumeration of the elements of H. It is easy to argue that the polynomials f_{i+1} satisfies the desired property by an inductive argument on i. This concludes the transformation T, that we summarize as follows: given ϕ,a, we pick a field F, a subset H, an integer m, an injective map $b:[n]\to H^m$ and an enumeration $\zeta_1,\ldots,\zeta_{|H|}$ of the elements of H. We then let $f_0=T_2(a)$, f_1 be as defined by (1), $f_2,\ldots,f_{m'+1}$ be as defined by (2) and let $T(a)=(f_0,\ldots,f_{m'+1})$.

2.3 The verification

We now describe the verifier for the transformation T. The verifier will be given oracles for functions $f_0,\ldots,f_{m'+1}$ and needs to verify that (a) The oracles f_i, $i\in\{0,m'+1\}$ describe polynomials of the correct degree. (b) For every $i\in[m'+1]$, the polynomial f_i is constructed from p_{i-1} using (1) or (2) (as appropriate). (c) The polynomial $f_{m'+1}$ is identically zero. Assuming that the functions $f_0, \ldots, f_{m'+1}$ are indeed polynomials of the correct degree, (b) and (c) can be verified very easily, probabilistically. To verify (c) the verifier queries the oracle $f_{m'+1}$ at a randomly chosen input $u \in F^{m'}$. By the distance property of polynomials, if $f_{m'+1}$ is not identically zero then $f_{m'+1}(u) \neq 0$ with high probability. To check (b) the verifier checks that the appropriate rule (1) or (2) holds for the oracles f_{i+1} and f_i for randomly chosen u. If the polynomial f_{i+1} is not identical to the polynomial obtained by applying the rule to f_i, then this difference will be witnessed by the point u with high probability. (Once again the distance property of polynomial is being used here.)

Thus the entire verification process reduces to the task of checking condition (a). A test for this condition is termed a "low-degree test" and has been a subject of active investigation recently. Specifically a low-degree test probabilistically queries q locations in an oracle Q and behaves as follows: If Q is a polynomial of total degree d, then the test accepts with probability 1. If the test accepts with probability $1-\delta$, then there is a polynomial $P : F^w \to F$ such that Q and P agree in all but at most ϵ fraction of the inputs, where ϵ, δ are parameters associated with the test.

To test that Q is a polynomial of degree at most d, one exploits the geometry of the space F^m as follows: For $u, v \in F^m$ and $t \in F$, let $l_{u,v}(t) = u + tv$ and let the line through u with slope v, denoted $l_{u,v}$, be the parameterized set of points $\{l_{u,v}(t) | t \in F\}$. It is immediate that for any polynomial $P : F^m \to F$ of degree at most and a line $l = l_{u,v}$, the function $P|_l : F \to F$ given by $P|_l(t) = P(l_{u,v}(t))$ is a univariate polynomial of degree at most d. Based on this observation a low-degree test was proposed in [37]: "Pick u, v uniformly and independently at random from F^m and verify that the points $\{(t, Q(l_{u,v}(t))) | t \in F\}$ are described by a univariate polynomial of degree at most d." It is clear that the tester makes $|F|$ queries to the oracle for Q and accepts all degree d polynomials. The "converse" is harder to prove and we will not attempt to hint at the proof here. A sequence of results [37, 6, 5, 36, 7] concludes showing that this test works for every $\epsilon < \delta < 1$, provided $|F|$ is polynomially larger that $d/(1-\delta)$.

We are still not done, since the low-degree test does not guarantee that the oracle Q is always equal to a low-degree polynomial, but only close to one. To patch this problem, we again resort to the error-correcting nature of polynomials; by using a probabilistic (and highly-efficient) error-correcting algorithm C for low-degree polynomials, due to [11]. C will have oracle access to some function $Q : F^w \to F$ and behave as follows on input $u \in F^w$: If Q is a degree d polynomial, it will return $Q(u)$. If Q is ϵ-close to a degree d polynomial P, it will return $P(u)$ or "error" with high probability (over its internal coin tosses). Again C uses the property of lines in F^m. "Given $\vec{u} \in F^w$ C picks at random $\vec{v} \in F^w$ and considers the function $q(t) = Q|_{l_{\vec{u},\vec{v}}}(t)$. If q is a polynomial in t of degree at most d, then outputs $q(0)$ else outputs error." A simple probabilistic argument shows that C has the desired properties for every $\epsilon < 1$, provided F is large enough.

We are now ready to specify the complete PCP verifier for verifying $\phi \in$ 3-SAT. The verifier has access to the oracles $f_0, \ldots, f_{m'+1}$. It performs a low-degree test on every oracle f_i, $i \in \{0, \ldots, m'+1\}$. If all low-degree tests pass, it then picks a random $u \in F^{m'}$ and verifies that for every i that the oracles $C^{f_{i+1}}$ satisfies the appropriate rule (1 or 2) w.r.t C^{f_i}. (Notice that we are now working with the oracles C^{f_i} rather than f_i. This is the right choice, since C^{f_i} is a polynomial — not merely close to one.) Finally it checks that $C^{f_{m'+1}}(u) = 0$. If all checks pass, then it accepts the proof, else it rejects the proof. Thus for the choices of m, $|F|$ etc. as made above, the construction yields a verifier making a total of $O(\log^c n)$ queries to all the oracles, for some absolute constant c. A formalization of the arguments above yields (modulo the analysis of the low-degree test) that the verifier accepts incorrect proofs with probability $o(1)$, as $n \to \infty$. Thus we conclude:

THEOREM 2 *NP* $\subseteq$ *PCP*$(O(\log n), \log^c n)$.

NOTES The result from Theorem 2 is essentially due to [8, 9], though the randomness efficiency was not reduced to $O(\log n)$ until the work of [6]. To get the full effect of Theorem 1 a number of new ideas are required. A central theme is a paradigm to compose proof systems, developed by [6]. In addition [5] present two new PCP constructions to prove Theorem 1. The interested reader may read the original papers for further details. Additional details may be found in [1, 38].

Subsequently there has been a significant amount of work improving the constant q of Theorem 1. This quest was initiated in [13] and further pursued in [20, 14, 35, 12]. Recently, a surprisingly sharp result, essentially showing $q = 3$, has been obtained by [27] (see also [23] for a variant of this result). This work introduces novel techniques to analyze the soundness of verifiers and while the result does rely on some prior work, may be read completely independently.

The consequences to optimization problems have also improved significantly since the initial works of [19, 5]. In particular, a number of new optimization problems have been related to PCPs and sharp results obtained in [33, 3, 21, 18, 26, 39]. Detailed surveys of such connections are available in [4, 17]. The connections have also motivated some new systematic study of combinatorial optimization problems — see [16, 31, 30].

The renewed interest in the approximability of optimization problems has also resulted, surprisingly, in a new spurt in algorithmic results. Particularly striking results in this direction are [24, 2]. Some of these algorithmic results, in particular [28, 40], are needed to analyze the tightness of the new PCP constructions of [27].

REFERENCES

[1] S. ARORA. *Probabilistic Checking of Proofs and Hardness of Approximation Problems.* PhD thesis, U.C. Berkeley, 1994. Available from `http://www.cs.princeton.edu/~arora`.

[2] S. ARORA. Polynomial-time approximation schemes for Euclidean TSP and other geometric problems. *Proc.* 37th *Symposium on Foundations of Computer Science*, IEEE, 1996.

[3] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Computer and System Sciences*, 54(2):317-331, April 1997.

[4] S. Arora and C. Lund. Hardness of approximations. In *Approximation Algorithms for NP-hard problems*, D. Hochbaum, ed. PWS Publishing, 1996.

[5] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximations. To appear *J. ACM*, 45(3), 1998.

[6] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *J. ACM*, 45(1):70-122, 1998.

[7] S. Arora and M. Sudan. Improved low degree testing and its applications. *Proc.* 29th *Annual Symposium on Theory of Computing*, ACM, 1997.

[8] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3-40, 1991.

[9] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. *Proc.* 23rd *Annual Symposium on Theory of Computing*, ACM, 1991.

[10] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *J. Computer and System Sciences*, 36(2):254-276, 1988.

[11] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. Proc. *Symposium on Theoretical Aspects of Computer Science*, 1990.

[12] M. Bellare, O. Goldreich and M. Sudan. Free bits, PCPs and non-approximability — towards tight results. *SIAM J. Computing,* 27(3):804-915, 1998.

[13] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs. *Proc.* 25th *Annual Symposium on Theory of Computing*, ACM, 1993.

[14] M. Bellare and M. Sudan. Improved non-approximability results. *Proc.* 26th *Annual Symposium on Theory of Computing*, ACM, 1994.

[15] S. Cook. The complexity of theorem-proving procedures. *Proc.* 3rd *Annual Symposium on Theory of Computing*, ACM, 1971.

[16] N. Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *J. Computer and System Sciences*, 51(3):511-522, 1995.

[17] P. Crescenzi and V. Kann, A compendium of NP optimization problems. Technical Report, Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza", SI/RR-95/02, 1995. Available from `http://www.nada.kth.se/ viggo/problemlist/compendium.html`.

[18] U. Feige. A threshold of ln n for Set Cover. *Proc.* 28th *Annual Symposium on Theory of Computing*, ACM, 1996.

[19] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43(2):268-292, 1996.

[20] U. Feige and J. Kilian. Two prover protocols — Low error at affordable rates. *Proc.* 26th *Annual Symposium on Theory of Computing*, ACM, 1994.

[21] U. Feige and J. Kilian. Zero knowledge and chromatic number. *Proc.* 11th *Annual Conference on Structure in Complexity Theory*, IEEE, 1996.

[22] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. *Theoretical Computer Science*, 134(2):545-557, 1994.

[23] V. GURUSWAMI, D. LEWIN, M. SUDAN AND L. TREVISAN. A tight characterization of NP with 3 query PCPs. *ECCC* Tech. Report TR98-034, 1998. Available from `http://www.eccc.uni-trier.de/eccc/`.

[24] M. GOEMANS AND D. WILLIAMSON. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115-1145, 1995.

[25] S. GOLDWASSER, S. MICALI, AND C. RACKOFF. The knowledge complexity of interactive proof-systems. *SIAM J. Computing*, 18(1):186-208, 1989.

[26] J. HÅSTAD. Clique is hard to approximate within $n^{1-\epsilon}$. *Proc.* 37th *Symposium on Foundations of Computer Science*, IEEE, 1996.

[27] J. HÅSTAD. Some optimal inapproximability results. *Proc.* 29th *Annual Symposium on Theory of Computing*, ACM, 1997.

[28] H. KARLOFF AND U. ZWICK. A 7/8-approximation algorithm for MAX 3SAT? *Proc.* 38th *Symposium on Foundations of Computer Science*, IEEE, 1997.

[29] R. KARP. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, Advances in Computing Research, pp. 85-103. Plenum Press, 1972.

[30] S. KHANNA, M. SUDAN AND L. TREVISAN. Constraint satisfaction: The approximability of minimization problems. *Proc.* 12th *Annual Conference on Structure in Complexity Theory*, IEEE, 1997.

[31] S. KHANNA, M. SUDAN, AND D. P. WILLIAMSON. A complete classification of the approximability of maximization problems derived from Boolean constraint satisfaction. *Proc.* 29th *Annual Symposium on Theory of Computing*, ACM, 1997.

[32] L. LEVIN. Universal'nyĭe perebornyĭe zadachi (Universal search problems : in Russian). *Problemy Peredachi Informatsii*, 9(3):265-266, 1973.

[33] C. LUND AND M. YANNAKAKIS. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960-981, September 1994.

[34] C. PAPADIMITRIOU AND M. YANNAKAKIS. Optimization, approximation and complexity classes. *J. Computer and System Sciences* 43(3):425-440, 1991.

[35] R. RAZ. A parallel repetition theorem. *SIAM J. Computing*, 27(3):763-803, 1998.

[36] R. RAZ AND S. SAFRA. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. *Proc.* 29th *Annual Symposium on Theory of Computing*, ACM, 1997.

[37] R. RUBINFELD AND M. SUDAN. Robust characterizations of polynomials with applications to program testing. *SIAM J. Computing* 25(2):252-271, 1996.

[38] M. SUDAN. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems.* ACM Distinguished Theses, Lecture Notes in Computer Science, no. 1001, Springer, 1996.

[39] L. TREVISAN. When Hamming meets Euclid: The approximability of geometric TSP and MST. *Proc.* 29th *Annual Symposium on Theory of Computing*, ACM, 1997.

[40] U. ZWICK. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. *Proc.* 9th *Annual Symposium on Discrete Algorithms*, ACM-SIAM, 1998.

Madhu Sudan
LCS, MIT, 545 Technology Square
Cambridge, MA 02139, U.S.A.
email: madhu@lcs.mit.edu