On Approximating NP-Hard Optimization Problems

Johan Håstad

Abstract. We discuss the efficient approximability of NP-hard optimization problems. Although the methods apply to several problems we concentrate on the problem of satisfying the maximal number of equations in an over-determined system of linear equations. We show that over the field of two elements it is NP-hard to approximate this problem within a factor smaller than 2. The result extends to any Abelian group with the size of group replacing the constant 2.

1991 Mathematics Subject Classification: 68Q25,68Q99
Keywords and Phrases: Approximation algorithms, NP-hard optimization problems, Linear equations

1 Introduction

The basic entity in complexity theory is a computational problem which, from a mathematical point of view, is simply a function F from finite binary strings to finite binary strings. To make some functions more intuitive these finite binary strings should sometimes be interpreted as integers, graphs, or descriptions of polynomials. An important special case is given by decision problems where the range consists of only two strings, usually taken to be 0 or 1.

The basic notion of efficiently computable is defined as computable in time polynomial in the input-length. The class of polynomial time solvable decision problems is denoted by P. Establishing that a problem cannot be solved efficiently can sometimes be done but for many naturally occurring computational problems of combinatorial nature, no such bounds are known. Many such problems fall into the class NP; problems where positive answers have proofs that can be verified efficiently. The standard problem in NP is satisfiability (denoted SAT), i.e. the problem of given a Boolean formula φ over Boolean variables, is it possible to assign truth values to the variables to make φ true? The most common version of SAT, which is also the one we use here, is to assume that φ is a CNF-formula, i.e. a conjunction of disjunctions.

It is still unknown whether NP=P, although it is widely believed that this is not the case. It is even the case that much work in complexity theory, and indeed this paper, would have to be completely reevaluated if it turns out that NP=P. There is a group of problem, called the *NP-complete* problems, introduced by Cook [8], which have the property that they belong to P iff NP=P. Thus being NP-complete is strong evidence that a problem is computationally intractable and literally thousands of natural computational problems are today known to be NP-complete (for an outdated but still large list of hundreds of natural problems see [13]). SAT is the most well known NP-complete problem.

Many combinatorial optimization problems have a corresponding decision problem which is NP-complete. As an example take the traveling salesman problem of finding the shortest tour that visits a certain set of cities. The corresponding decision problem, namely that of, given K, determine if there is a tour of length K is NP-complete and thus solving the optimization problem exactly in polynomial time would mean that NP=P. Optimization problem with this property are called *NP-hard* (not NP-complete as they do not fall into the class NP as they are not decision problems). Solving NP-hard optimization problems exactly is thus hard, but in many practical circumstances it is almost as good to get an approximation of the optimum. Different NP-hard optimization problems behave very differently with respect to efficient approximation algorithms and this set of questions has lead to many interesting results.

The goal of this paper is to derive lower bounds on how well natural optimization problems can be approximated efficiently. The type of result we are interested in is a conclusion of the form "If optimization problem X and be approximated within factor c in polynomial time, then NP=P". The techniques we discuss give results for many optimization problem but we here concentrate on solving over-determined systems of linear equations over finite Abelian groups. For this problem we are given a set of m equations in n unknowns and the task is to determine the maximal number of equations that can be simultaneously satisfied and possibly also produce an assignment that satisfies this number of equations.

An algorithm is a c-approximation algorithm if it, for every instance, finds a solution that is within a factor c of the optimal value. Thus if the best assignment satisfies o equation, such an approximation algorithm would always find an assignment that satisfies o/c equations. For linear equations over GF[2] a random assignment satisfies, on the average, half the equations. It is hence not surprising that one can efficiently find an assignment that satisfies at least half the equations. This gives a 2-approximation algorithm and the result extends to any Abelian group G to give a $size(G)$-approximation algorithm. The main result we discuss in this paper is to prove that this simple heuristic is optimal in that for any Abelian group G and any $\epsilon > 0$ it is NP-hard to $size(G) - \epsilon$-approximate the problem of linear equations over G.

The main tool for deriving such strong approximability results was introduced in the seminal paper [10]. It gives a connection to multiprover interactive proofs and let us here give an informal description of a variant of this concept. We later give some formal definitions in Section 1.1.

NP can be viewed as a proof system where a single prover P tries to convince a polynomial time verifier V that a statement is true. For concreteness let us assume that the statement is that a formula φ is satisfiable. In this case, P displays a satisfying assignment and V can easily check that it is a correct proof. This proof system is complete since every satisfiable φ admits a correct proof, and it is sound since V can never be made to accept an incorrect statement.

If φ contains n variables, V reads n bits in the above proof. Suppose we limit V to reading fewer bits where the most extreme case would be to let this number be constant independent of the the number of variables in φ. It is not hard to see that the latter is impossible unless we relax the requirements of the proof. The proof remains a finite binary string, but we allow the verifier to make random choices. This means that given φ we should now speak of the probability that V accepts a certain proof π. Soundness is relaxed in that when φ is not satisfiable then there is some constant $s < 1$ such that for any proof π the probability that V accepts is bounded by s. A bit surprisingly it turns out that it is convenient also to relax completeness in that we only require the verifier to accept a correct proof for a correct statement with probability $c > s$ where we might have $c < 1$. Note that both completeness and soundness probabilities are taken only over V's internal random choices and hence we can improve these parameters by making several independent verifications and taking a majority decision. Naturally this increases other parameters that we want to keep small such as the number of bits of the proof that V reads.

It is an amazing fact, proved by Arora et al [1], that any NP-statement has a proof of the above type, usually called probabilistically checkable proof or simply PCP, where V only reads a constant, independent of the size of the statement being verified, number of bits of the proof and achieves soundness $s = 1/2$ and completeness $c = 1$. Apart from being an amazing proof system this gives a connection to approximation of optimization problems as follows.

Fix a formula φ and consider the PCP by Arora et al. Since everything is fixed except the proof π, we have a well defined function $acc(\pi)$, the probability that V accepts a certain proof π. Consider $\max_\pi acc(\pi)$. If φ is satisfiable then this optimum is 1, while if φ is not satisfiable then the optimum is $\leq s$. Thus, even computing this optimum approximately would enable us to decide an NP-complete question. Now by choosing the test appropriately this optimization problem can be transformed to more standard optimization problems leading to the desired inapproximability results.

1.1 Probabilistic proof systems

As discussed in the introduction we are interested in proof systems where the verifier is a probabilistic Turing machine. The simplest variant is a probabilistically checkable proof.

Definition 1.1 *A* Probabilistically Checkable Proof (PCP) *with completeness c and soundness s for a language L is given by a verifier V such that for $x \in L$ there is a proof π such that V^π outputs 1 on input x with probability at least c, and for $x \notin L$ and all π the probability that V^π outputs 1 on input x is bounded by s.*

We are interested in efficient PCPs and hence we assume that V runs in worst case polynomial time. It is also important for us to efficiently enumerate all the random choice of V and hence we need that V only makes $O(\log |x|)$ binary random choices on input x. We maintain this property without mentioning it explicitly.

We also need what is generally called a two-prover one-round interactive proof. Such a verifier has two oracles but has the limitation that it can only ask one question to each oracle and that both questions have to be produced before either of them is answered. We do not limit the answer size of the oracles. We call the two oracles P_1 and P_2.

DEFINITION 1.2 *A probabilistic polynomial time Turing machine V is a verifier in a* two-prover one-round proof system *with completeness c and soundness s for a language L if on input x it produces two strings $q_1(x)$ and $q_2(x)$, such that for $x \in L$ there are two oracles P_1 and P_2 such that the probability that V accepts $(x, P_1(q_1(x)), P_2(q_2(x)))$ is c while for $x \notin L$, for any two oracles P_1 and P_2 the probability that V accepts $(x, P_1(q_1(x)), P_2(q_2(x)))$ is bounded by s.*

In all our two-prover interactive proofs the verifier always accepts a correct proof for a correct statement, i.e. we have $c = 1$ in the above definition.

BRIEF HISTORY. The notion of PCP was introduced by Arora and Safra [2]. It was a variation of randomized oracle machines discussed by Fortnow et al [12] and transparent proofs by Babai et al [4]. Multiprover interactive proofs were introduced by Ben-Or et al [7], and all these systems are variants of interactive proofs as introduced by Goldwasser, Micali, and Rackoff [14] and Babai [3].

1.2 ESSENTIAL PREVIOUS WORK

The surprising power of interactive proofs was first established in the case of one prover [17], [20] and then for many provers [5]. After the fundamental connection with approximation was discovered [10] the parameters of the proofs improved, culminating in the result [2, 1] that it is sufficient to read a constant number of bits. Using a transformation of [18] and massaging the result slightly we arrive at the following theorem.

THEOREM 1.3 *[1] Let L be a language in NP and x be a string. There is a universal constant $c < 1$ such that, we can in time polynomial in size(x) produce a CNF formula $\varphi_{x,L}$ with exactly 3 literals in each clause such that if $x \in L$ then $\varphi_{x,L}$ is satisfiable while if $x \notin L$, any assignment satisfies at most a fraction c of the clauses of $\varphi_{x,L}$. Furthermore, we can assume that each variable appears exactly 12 times.*

Let us now turn to two-prover interactive proofs. Given a one-round protocol with soundness s and completeness 1 we can repeat it k times in sequence improving the soundness to s^k. This creates a many round protocols, whereas we need our protocols to remain one-round. This can be done by parallel repetition in that V repeats his random choices to choose k pairs of questions $(q_1^{(i)}, q_2^{(i)})_{i=1}^k$ and sends $(q_1^{(i)})_{i=1}^k$ to P_1 and $(q_2^{(i)})_{i=1}^k$ to P_2 all at once. V then receives k answers from each prover and accepts if it would have accepted in all k protocols given each individual answer. The soundness of such a protocol can be greater than s^k, but Raz [19] proved that when the answer size is small the soundness is exponentially decreasing with k.

THEOREM 1.4 *[19] For all integers d and $s < 1$, there exists $c_{d,s} < 1$ such that given a two-prover one-round proof system with soundness s and answer sizes bounded by d, then for all integers k the soundness of k protocols run in parallel is bounded above by $c_{d,s}^k$.*

1.3 Definitions for approximation algorithms

Definition 1.5 *Let O be a maximization problem. For an instance x of O let $OPT(x)$ be the optimal value. An efficient C-approximation algorithm is an algorithm that on any input x outputs a number V such that $OPT(x)/C \leq V \leq OPT(x)$ and runs in worst case polynomial time.*

2 First steps towards a good proof system

We want to construct a proof system for an arbitrary language in NP and let us start by an overview.

We start by a simple two-prover one-round protocol which is obtained more or less immediately from Theorem 1.3. We improve the soundness of this protocol by running several copies of it in parallel and using Theorem 1.4. It is possible to transform this improved two-prover protocol to a PCP simply by writing down the prover answers. The answers are, however, long and since we want to keep the number of read bits very small we write the answers in a more useful form by asking the prover to supply the value of all Boolean functions of these answers. This is the long code of the answers as defined in [6]. We now proceed to give the details.

Suppose $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, where C_j contains the variables x_{a_j}, x_{b_j} and x_{c_j}. Consider the following one-round two-prover interactive proof.

Simple two-prover protocol

1. V chooses $j \in [m]$ and $k \in \{a_j, b_j, c_j\}$ both uniformly at random. V sends j to P_1 and k to P_2.
2. V receives values for x_{a_j}, x_{b_j} and x_{c_j} from P_1 and for x_k from P_2. V accepts if the two values for x_k agree and C_j is satisfied.

We have the following proposition which can be proved by a straightforward argument which we omit.

Proposition 2.1 *If any assignment satisfies at most a fraction e of the clauses of φ, then V accepts in the simple two prover protocol with probability at most $(2+e)/3$.*

We now concentrate a protocol we called the *u-parallel 2-prover game* and which consists of u copies of this basic game. That is, the verifier picks u clauses $(C_{j_k})_{k=1}^{u}$ and then uniformly at random for each k he picks a random variable x_{i_k} contained in C_{j_k}. The variables of $(C_{j_k})_{k=1}^{u}$ are sent to P_1 while $(x_{i_k})_{k=1}^{u}$ are sent to P_2. The two provers respond with assignments on the queried variables and the verifier checks that the values are consistent and that the chosen clauses are satisfied. We get completeness 1 and the soundness is in the case of $\varphi_{x,L}$ of Theorem 1.3 is, by Theorem 1.4 and Proposition 2.1, bounded by c_1^u for some constant $c_1 < 1$. To fix notation, Let $U = \{x_{i_1}, x_{i_2} \ldots x_{i_u}\}$ be the set of variables sent to P_2, and W the set of variables sent to P_1.

As discussed in the introduction to this section we want to replace this two-prover interactive proof by a PCP consisting of the answers of P_1 and P_2 given in a more redundant form. We use the powerful long code introduced in [6].

DEFINITION 2.2 *The* long code *of a string x of length w is of length 2^{2^w}. The coordinates of the codeword are identified with all possible functions $f : \{0,1\}^w \mapsto \{0,1\}$ and the value of coordinate f is $f(x)$.*

Before we continue, let us fix some more notation. The written part of the PCP described above is called a *Standard Written Proof of size u* or simply SWP(u). Let $\mathcal{F}_T$ denote the set of functions on a set T and let A_T be the supposed long code of the restriction of the satisfying assignment to the set T. It is convenient to have $\{-1,1\}$ as our set of two values for Boolean functions and Boolean variables and thus exclusive-or turns into multiplication. For the supposed long code A_T we assume that $A_T(f) = -A_T(-f)$. This is achieved by, for each pair $(f,-f)$, having only one value in A_T. This value is negated if the value of $A_T(-f)$ is needed. For the tables A_W, we know that it should be a long code for some assignment that satisfies $\wedge_k C_{j_k}$ and instead of storing an entry for each $g \in \mathcal{F}_W$ we only store an entry for each function of the form $g \wedge (\wedge_k C_{j_k})$. When we want the value of a function h we access the entry for $h \wedge (\wedge_k C_{j_k})$. A SWP($u$) is correct for φ if there is an assignment x that satisfies φ and thus that $A_T(f) = f(x|_T)$ for any supposed long code A_T for any set T obtained as U or W in a run of the u-parallel 2-prover game.

We need the discrete Fourier transform defined by

$$\hat{A}_{\alpha,T} = 2^{-2^{size(T)}} \sum_f A_T(f) \prod_{x\in\alpha} f(x)$$

for any $\alpha \subseteq \{-1,1\}^T$. It is inverted by

$$A_T(f) = \sum_\alpha \hat{A}_{\alpha,T} \prod_{x\in\alpha} f(x).$$

and since $|A(f)| = 1$ for any f we have, by Parseval's identity, $\sum_\alpha \hat{A}^2_{\alpha,T} = 1$. For a set $\beta \subseteq \{-1,1\}^W$ and $U \subset W$ we let $\pi_2^U(\beta) \subseteq \{-1,1\}^U$ be those elements that have an odd number of extensions in β. This a mod 2 projection and note that it might be empty even if β is not empty.

3 LINEAR EQUATIONS

We now study the optimization problem of satisfying the maximal number of equations mod 2. For natural reasons we want to design a test for SWP(u) that accepts depending only on the exclusive-or of three bits of the proof.

TEST $L_2^\epsilon(u)$

WRITTEN PROOF. A SWP(u).

Desired property. To check that it is a correct SWP(u) for a given formula $\varphi = C_1 \wedge C_2 \dots C_m$.
Verifier. Choose set U and W as in the u-parallel 2-prover game. Choose $f \in \mathcal{F}_U$ and $g_1 \in \mathcal{F}_W$ with the uniform probability. Choose a function $\mu \in \mathcal{F}_W$ by setting $\mu(y) = 1$ with probability $1-\epsilon$ and $\mu(y) = -1$ otherwise, independently for each $y \in \{-1,1\}^W$. Define g_2 by for each $y \in \{-1,1\}^W$, $g_2(y) = f(y|_U)g_1(y)\mu(y)$. Accept if $A_U(f)A_W(g_1)A_W(g_2) = 1$.

We need to analyze the soundness and completeness of this test.

Lemma 3.1 *The completeness of Test $L_2^\epsilon(u)$ is at least* $1-\epsilon$.

Proof: Fix a correct SWP(u) obtained from the assignment x satisfying φ We claim that V accepts unless $\mu(x|_W) = -1$ and leave the verification to the reader. ∎

Lemma 3.2 *For any $\epsilon > 0$, $\delta > 0$, suppose that the probability that the verifier of Test $L_2^\epsilon(u)$ accepts is $(1+\delta)/2$. Then there is a strategy for P_1 and P_2 in the u-parallel two prover protocol that makes the verifier of that protocol accept with probability at least $\epsilon\delta^2/2$.*

Proof: Let us first fix U and W and for notational convenience we denote the function A_U by A and the function A_W by B. We want to consider

$$E_{f,g_1,\mu}[A(f)B(g_1)B(g_2)] \tag{1}$$

since by the assumption of the lemma

$$E_{U,W,f,g_1,\mu}[A_U(f)A_W(g_1)A_W(g_2)] = \delta. \tag{2}$$

Using the Fourier expansion and moving the expected value inside (1) equals

$$\sum_{\alpha,\beta_1,\beta_2} \hat{A}_\alpha \hat{B}_{\beta_1} \hat{B}_{\beta_2} E_{f,g_1,\mu}\left[\prod_{x\in\alpha} f(x) \prod_{y\in\beta_1} g_1(y) \prod_{y\in\beta_2} (f(y|_U)g_1(y)\mu(y))\right]. \tag{3}$$

If $\beta_1 \neq \beta_2$ then since $g_1(y)$ for $y \in \beta_1 \Delta \beta_2$ is random and independent of all other variables the inner expected value in this case is 0 and thus we can disregard all terms except those with $\beta_1 = \beta_2 = \beta$. Now consider such a term and let s_x be number of $y \in \beta$ such that $y|_U = x$. Since $f(x)$ is random and independent for different x, unless for every x either $x \in \alpha$ and s_x is odd or $x \notin \alpha$ and s_x is even again the inner expected value is 0. These conditions imply that we only keep terms with $\pi_2^U(\beta) = \alpha$ and finally since $E_\mu[\prod_{y\in\beta}\mu(y)] = (1-2\epsilon)^{size(\beta)}$ we have reduced the sum (1) to

$$\sum_\alpha \sum_{\beta|\pi_2^U(\beta)=\alpha} \hat{A}_\alpha \hat{B}_\beta^2 (1-2\epsilon)^{size(\beta)}. \tag{4}$$

We want to prove that if the expected value of this (over random choices of U and W) is at least δ then we have a good strategy of the provers. We define good randomized strategies for P_1 and P_2.

The strategy of P_2 is first to pick a random α with and $\hat{A}_\alpha \geq \delta/2$. The probability of picking α is defined to be proportional to $\hat{A}_\alpha$ and hence by Parseval's identity it is at least $\delta\hat{A}_\alpha/2$. P_2 sends a random $x \in \alpha$. Note that α is nonempty since $A_U(f) = -A_U(-f)$ implies that $\hat{A}_\emptyset = 0$.

The strategy of P_1 is to pick a random β with probability $\hat{B}_\beta^2$ and then answer with a random $y \in \beta$.

Let us evaluate the success-rate of this strategy. By the property that $A_W(h)$ only depends on $h \wedge (\wedge_k C_{j_k})$ it is not hard to establish that every y sent by P_1 satisfies the corresponding clauses and thus we only need to look at the probability that the answers are consistent. This probability is at least $size(\beta)^{-1}$ times the probability that for the picked α and β we have $\alpha = \pi_2^U(\beta)$. The probability of picking a specific pair α and β is, provided $\hat{A}_\alpha > \delta/2$, at least $\hat{A}_\alpha \hat{B}_\beta^2 \delta/2$ and thus the overall success-rate for a fixed choice of U and W is at least

$$\delta/2 \sum_{\alpha | \hat{A}_\alpha | \geq \delta/2} \sum_{\beta | \pi_2^U(\beta)=\alpha} \hat{A}_\alpha \hat{B}_\beta^2 size(\beta)^{-1}. \tag{5}$$

Comparing this sum to (4) and making some calculations one can establish that expected value over U and W is at least $\delta^2\epsilon/2$ and the proof of Lemma 3.2 is complete. ∎

Armed with the very efficient PCP given by Test $L_2^\epsilon(u)$ we can now establish the main theorem of this paper.

THEOREM 3.3 *For any $\epsilon > 0$ it is NP-hard to approximate the problem of maximizing the number of satisfied equation in a system of linear equations mod 2 within a factor $2 - \epsilon$. The result applies to systems with only 3 variables in each equation.*

PROOF: *(Sketch)* Let L be an arbitrary language in NP and given an input x, create the formula $\varphi_{x,L}$ as given in Theorem 1.3. Let δ be small positive number to be determined and consider test $\mathrm{L}_2^\delta(u)$ where u is chosen sufficiently large so that the acceptance probability in the u-parallel 2-prover game is smaller than $\delta^3/2$.

For each bit b in a SWP(u) introduce a variable x_b. To accept in the test $L_2^\delta(u)$ is equivalent to the condition

$$b_{U,f} b_{W,g_1} b_{W,g_2} = c$$

where $b_{U,f}$, b_{W,g_1} and b_{W,g_2} are the bits in the proof corresponding to $A_U(f)$, $A_W(g_1)$ and $A_W(g_2)$, respectively[1]. Write down the equation

$$x_{b_{U,f}} x_{b_{W,g_1}} x_{b_{W,g_2}} = c$$

[1]One might think that the right hand size would always be 1, but because of our convention on having one entry in A_U to represent the value on two functions this might be the case since the value corresponding to $A_U(f)$ in the proof might actually give the value of $A_U(-f)$

with a weight that is equal to the probability that the verifier chooses the tuple (U, W, f, g_1, g_2). Now each proof corresponds to an assignment to the variables x_b and the total weight of all satisfied equations is exactly the probability that this proof is accepted. This implies that if $x \in L$ this maximal weight is $1 - \delta$ while if $x \notin L$, it is, in view of Lemma 3.2 and the choice of u, at most $(1+\delta)/2$. It is not difficult to check that we have a polynomial number of equations and an approximation algorithm with performance ratio smaller than $2 - \epsilon$ would enable us, for sufficiently small δ, to answer a NP-hard question.

As is standard, the weights can be eliminated by duplicating each equation a suitable number of times. This creates a slight degrade in the value of ϵ, but since ϵ is arbitrary anyway this can easily be compensated. We omit the details. ∎

Note that there is a meta reason that we have to introduce the error function μ and make our test have non perfect completeness. If we had perfect completeness then the equations produced in the proof of Theorem 3.3 could all be satisfied simultaneously. However, to decide if a set of linear equations have a common solution can be done in polynomial time by Gaussian elimination.

Finally, let us just state the extension to an arbitrary Abelian group.

Theorem 3.4 *For any $\epsilon > 0$ and any Abelian group G, given a system of linear equations over G, it is NP-hard to approximate the maximal number of simultaneously satisfiable equations within a factor $size(G) - \epsilon$. The result applies to systems with only 3 variables in each equation.*

4 Final remarks

As mentioned in the introduction the efficient multiprover interactive proofs give strong inapproximability results for many combinatorial optimization problems.

Independence number is to, given a graph G, find the largest set S of nodes such that no two nodes in S ares pairwise connected. It is established in [15] that it is, assuming that NP cannot be done in probabilistic polynomial time, for any $\epsilon > 0$, hard to approximate independence number within $n^{1-\epsilon}$ where n is the number of nodes G. A very related problem is that of chromatic number where we want to color the nodes in G with the minimal number of colors so that adjacent nodes get different colors. The result for independence number can be extended to chromatic number [11]. The problem of set cover is that given a number of subsets S_i of $[n]$ to find the minimal size sub-collection of the S_i that covers the entire set. This problem is, under standard complexity assumptions, hard to approximate within $(1 + o(1)) \ln n$ [9] and this result is tight. For inapproximability results on other problem, some optimal and some non-optimal we refer to the full versions of [6] and [16].

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof verification and intractability of approximation problems. Proc. of the 33rd Annual IEEE Symposium on Foundations of Computer Science, Pittsburgh, 1992, pp 14-23.

[2] S. ARORA AND S. SAFRA. Probabilistic checking of proofs: a new characterization of NP. Journal of the ACM, Vol 45, 1998, pp 70-122.
[3] L. BABAI. Trading group theory for randomness. Proc. of the 17th Annual ACM Symposium on Theory of Computation, Providence, 1985, pp 420-429.
[4] L. BABAI, L. FORTNOW, L. LEVIN, AND M. SZEGEDY. Checking computations in polynomial time. Proc. of the 23rd Annual ACM Symposium on Theory of Computation, New Orleans, 1991, pp 21-31.
[5] L. BABAI, L. FORTNOW, AND C. LUND. Non-deterministic exponential time has two-prover interactive protocols. Computational Complexity, Vol 1, 1991, pp 3-40.
[6] M. BELLARE, O. GOLDREICH AND M. SUDAN. Free Bits, PCPs and Non-Approximability-Towards tight Results. Proc. of the 36th Annual IEEE Symposium on Foundations of Computer Science, 1995, Milwaukee, pp 422-431. Full version available from ECCC, Electronic Colloquium on Computational Complexity (http://www.eccc.uni-trier.de/eccc).
[7] M. BEN-OR, S. GOLDWASSER, J. KILIAN, AND A. WIGDERSON. Multiprover interactive proofs. How to remove intractability. Proc. of the 20th Annual ACM Symposium on Theory of Computation, Chicago, 1988, pp 113-131.
[8] S. A. COOK. The complexity of Theorem Proving Procedure. Proceeding 3rd ACM Symposium on Theory of Computing, 1971, pp 151-158.
[9] U. FEIGE. A threshold of $\ln n$ for approximating set cover. Proc. of the 28th Annual ACM Symposium on Theory of Computation, Philadelphia 1996, pp 314-318.
[10] U. FEIGE, S. GOLDWASSER, L. LOVÁSZ, S. SAFRA, AND M. SZEGEDY. Interactive proofs and the hardness of approximating cliques. Journal of the ACM, Vol, 43:2, pp 268-292.
[11] U. FEIGE AND J. KILIAN. Zero-Knowledge and the chromatic number. Proc. of the 11th Annual IEEE conference on Computational Complexity, Philadelphia 1996, pp 278-287.
[12] L. FORTNOW, J. ROMPEL, AND M. SIPSER. On the power of Multi-Prover Interactive Protocols. Proc. 3rd IEEE Symposium on Structure in Complexity Theory, pp 156-161, 1988.
[13] M.R. GAREY AND D.S. JOHNSSON. Computers and Intractability. W.H. Freeman and Company, 1979.
[14] S. GOLDWASSER, S. MICALI, AND C. RACKOFF. The knowledge complexity of interactive proof systems. SIAM Journal on Computing, Vol 18, pages 186-208, 1989.
[15] J. HÅSTAD. Clique is hard to approximate within $n^{1-\epsilon}$. Proc. of the 37th Annual IEEE Symposium on Foundations of Computer Science, Burlington 1996, pp 627-636. Full version available from ECCC, Electronic Colloquium on Computational Complexity (http://www.eccc.uni-trier.de/eccc).
[16] J. HÅSTAD. Some Optimal In-approximability Results. Proc. 29th Annual ACM Symposium on Theory of Computation, 1997, pp 1-10. Full version available from ECCC, Electronic Colloquium on Computational Complexity (http://www.eccc.uni-trier.de/eccc).
[17] C. LUND, L. FORTNOW, H. KARLOFF AND N. NISAN. Algebraic methods for interactive proof systems. Journal of the ACM, Vol 39, No 2, pp 859-868.
[18] C. PAPADIMITRIOU AND M. YANNAKAKIS. Optimization, approximation and complexity classes. Journal of Computer and System Sciences, Vol 43, 1991, pp 425-440.
[19] R. RAZ. A parallel repetition theorem. Proc. of the 27th Annual ACM Symposium on Theory of Computation, Las Vegas 1995, pp 447-456.
[20] A. SHAMIR. IP=PSPACE. Journal of the ACM, Vol 39, No 2, pp 869-877.

Johan Håstad
Dept of Numerical Analysis and
Computing Science
Royal Institute of Technology
S-100 44 Stockholm
Sweden