# Proof Verification and Hardness of Approximation Problems

*Sanjeev Arora*[*] *Carsten Lund*[†] *Rajeev Motwani*[‡] *Madhu Sudan*[§] *Mario Szegedy*[¶]

**Abstract**

*The class PCP*$(f(n), g(n))$ *consists of all languages* $L$ *for which there exists a polynomial-time probabilistic oracle machine that uses* $O(f(n))$ *random bits, queries* $O(g(n))$ *bits of its oracle and behaves as follows: If* $x \in L$ *then there exists an oracle* $y$ *such that the machine accepts for all random choices but if* $x \notin L$ *then for every oracle* $y$ *the machine rejects with high probability. Arora and Safra very recently characterized* $NP$ *as PCP*$(\log n, (\log\log n)^{O(1)})$*. We improve on their result by showing that* $NP = PCP(\log n, 1)$*. Our result has the following consequences:*

1. *MAXSNP-hard problems (e.g., metric TSP, MAX-SAT, MAX-CUT) do not have polynomial time approximation schemes unless P=NP.*
2. *For some* $\epsilon > 0$ *the size of the maximal clique in a graph cannot be approximated within a factor of* $n^\epsilon$ *unless P=NP.*

## 1 Introduction

The notion of NP-completeness [Coo71, Kar72, Lev73] has been used since the early seventies to show the hardness of finding optimum solutions for a large variety of combinatorial optimization problems. The apparent intractability of these problems motivated the search for approximate solutions to these hard optimization problems. For some problems this effort gave good approximation algorithms, but for others it seemed hard even to find near optimal solutions.

The task of proving hardness of the approximation versions of such problems met with limited success. For the traveling salesman problem without triangle inequality Sahni and Gonzalez [SG76] showed that finding a solution within any constant factor of optimal is also NP-hard. Garey and Johnson [GJ76] studied MAX-CLIQUE: the problem of finding the largest clique in a graph. They showed that if a polynomial time algorithm computes MAX-CLIQUE within a constant multiplicative factor then MAX-CLIQUE has a polynomial-time approximation scheme (PTAS) [GJ78], i.e., for any $c > 1$ there exists a polynomial-time algorithm that approximates MAX-CLIQUE within a factor of $c$. They used graph products to construct "gap increasing reductions" that mapped an instance of the clique problem into another instance in order to enlarge the relative gap of the clique sizes. Berman and Schnitger [BS92] used the ideas of increasing gaps to show that if the clique size of graphs with bounded co-degree (degree of the complement) does not have a randomized PTAS then there is an $\epsilon > 0$ such that MAX-CLIQUE cannot be approximated within a factor of $n^\epsilon$ in randomized polynomial-time.

Yet for the most part, not much could be said for a wide variety of problems until very recently. A connection between two seemingly unrelated areas within theoretical computer science, established by Feige et al. [FGLSS91], led to surprisingly strong hardness results for approximating optimization problems. Feige et al. exploited a recent characterization of multiprover interactive proof systems by Babai, Fortnow and Lund [BFL91] to obtain intractability results for approximating MAX-CLIQUE under the assumption that $NP \not\subseteq DTIME(n^{O(\log\log n)})$.

Recently Arora and Safra [AS92] improved on this by showing that it is NP-hard to approximate MAX-CLIQUE within any constant factor (and even within a factor of $2^{\log n/(\log\log n)^{O(1)}}$). Their solution builds on and further develops the techniques of Feige et al. and yields an elegant new characterization of the class NP in terms of probabilistically checkable proofs.

Relying on their work we further develop these techniques and show that there is no PTAS for a large number of combinatorial optimization problems unless P=NP. These problems, like traveling salesman problem with triangle inequality, minimal steiner tree, maximum directed cut, shortest superstring, etc. be-

[*] Computer Science Division, U. C. Berkeley, Berkeley, CA 94720. Supported by NSF PYI Grant CCR 8896202.

[†] AT&T Bell Labs, Murray Hill, NJ 07974.

[‡] Department of Computer Science, Stanford University, Stanford, CA 94305. Supported by NSF Grant CCR-9010517, and grants from Mitsubishi and OTL.

[§] Computer Science Division, U. C. Berkeley, Berkeley, CA 94720. Supported by NSF PYI Grant CCR 8896202.

[¶] AT&T Bell Labs, Murray Hill, NJ 07974.

long to the class of MAXSNP-hard problems, defined by Papadimitriou and Yannakakis [PY91] in terms of logic and reductions that preserves approximability.

Our result also improves the parameters for the MAX-CLIQUE result of Arora and Safra. We show, that there is an $\epsilon > 0$ such that approximating MAX-CLIQUE within a factor of $n^\epsilon$ in NP-hard.

In the next section we elaborate on the definition of the class MAXSNP and then move on to give background on probabilistically checkable proofs and formulate our main lemma, which is a characterization of NP that improves on the one in [AS92]. Then we relate this characterization to the non-approximability of MAXSNP. The rest of the paper will be devoted to the proof of the main lemma.

### 1.1 The class MAXSNP

Given a maximization problem, formulated as $opt_F(x) = \max_y F(x, y)$, an approximation algorithm $A$ is said to achieve worst-case performance ratio $\alpha(n)$ if for every input $x : F(x, A(x)) \geq \alpha(n)^{-1} opt_F(x)$, where $n$ is the size of $x$.

In 1988 Papadimitriou and Yannakakis [PY91] using Fagin's definition of NP [Fag74] observed that there is an approximation algorithm which has constant performance ratio for any maximization problem that is defined by a quantifier free first order formula $\varphi$ as

$$opt_\varphi(X) = \max_S : |\{z | \varphi(S, X, z)\}| , \tag{1}$$

where $z$ is a vector of constant number of first order variables.

**Definition 1.1 (Papadimitriou-Yannakakis)** *An optimization problem $opt(x)$ belongs to $MAXSNP$ if there is a polynomial-time algorithm that encodes input $x$ into $X$ in 1 such that $opt(x)$ becomes $opt_\varphi(X)$.*

They further considered the following type of "constant gap preserving" reductions:

**Definition 1.2 (Papadimitriou-Yannakakis)** *Let opt and opt′ be two optimization problems defined by functions $F$ and $F'$. We say that opt L-reduces to opt′ if there exist two polynomial time algorithms $f$ and $g$ and constants $\alpha, \beta > 0$ such that for each instance $x$ of opt:*

1. *Algorithm $f$ produces an instance $x'$ of opt′ such that $opt'(x') \leq \alpha opt(x)$.*
2. *For any $y'$ algorithm $g$ outputs a $y$ with the property:*

$$|F(x, y) - opt(x)| \leq \beta |F'(x', y') - opt(x')|.$$

A problem *opt* is MAXSNP-hard (complete) if every problem in MAXSNP L-reduces to it (and $opt \in MAXSNP$). Papadimitriou and Yannakakis showed that if any of the MAXSNP-hard problems have a PTAS then every problem in MAXSNP has a PTAS.

MAX-3SAT is a typical MAXSNP-complete problem: Given a conjunction $\varphi$ of clauses, each a disjunction of 3 variables or their negation, maximize the the number of satisfied clauses over all possible assignments to the variables of $\varphi$.

The following problems are also MAXSNP-complete or hard: MAX-SAT; MAX-2SAT; INDEPENDENT SET-B; NODE COVER-B; MAX-CUT; MAX-DIRECTED CUT; METRIC TSP; STEINER TREE; SHORTEST SUPERSTRING; MULTIWAY CUTS; THREE DIMENSIONAL MATCHING. The exact definitions can be found in [PY91, PY92, BP89, BJLTY91, DJPSY92, Kan91].

### 1.2 Probabilistically checkable proofs

The notion of probabilistically checkable proofs ($PCP$) was introduced by Arora and Safra [AS92], as a slight variation of the notions of *randomized oracle machines* due to Fortnow, Rompel and Sipser [FRS88] and *transparent proofs* due to Babai, Fortnow, Levin and Szegedy [BFLS91]. All these models are in turn variations of interactive proof systems [Bab85, GMR89] and multiprover interactive proof systems [BGKW88].

**Definition 1.3 (Arora-Safra [AS92])** *A language $L$ is in PCP($f(n), g(n)$) if there is polynomial-time randomized oracle machine $M^y(r, x)$ which works as follows:*

1. *It takes input $x$ and a (random) string $r$ of length $O(f(n))$, where $n = |x|$.*
2. *Generates a query set $Q(r, x) = \{q_1, \ldots, q_m\}$ of size $m = O(g(n))$.*
3. *Reads the bits $y_{q_1}, \ldots, y_{q_m}$.*
4. *Makes a polynomial-time computation using the data $r$, $x$ and $y_{q_1}, \ldots, y_{q_m}$ and outputs $M^y(r, x) \in \{0, 1\}$.*

*Moreover the following acceptance conditions hold for some $\delta > 0$ and for all $x$:*

1. *If $x \in L$ then there exists a $y$ such that for every $r$ we have $M^y(r, x) = 1$.*
2. *If $x \notin L$ then for every $y$ we have $Prob_r(M^y(r, x)) = 0) \geq \delta$.*

Observe that the definition does not say anything about the length of $y$, but it is easy to see that we can assume without loss of generality that $|y| = 2^{O(f(n))}g(n)$. This definition is a generalization of $NP$ since it is clear that $NP = PCP(0, n^{O(1)})$.

**Theorem 1 (Arora-Safra [AS92])** $NP = PCP(\log n, (\log\log n)^{O(1)})$.

We improve on the second parameter:

**Theorem 2** $NP = PCP(\log n, 1)$.

Much of the paper will be devoted to the proof of this theorem. The connection to clique approximations, of Feige et al., shows that if $NP = PCP(\log n, g(n))$, then the clique size cannot be approximated within a factor of $2^{\Omega(\log n/g(n))}$ unless P=NP. Thus Theorem 2 has the corollary:

**Corollary 3** *There is an $\epsilon > 0$ such that to approximate MAX-CLIQUE within a factor of $n^\epsilon$ is NP-hard.*

### 1.3 Related Areas

The results in this paper borrow significantly from results in the area of self-testing/self-correcting of programs (see [BLR90], [Rub90]). The areas of self-testing/correcting are closely connected to the areas of error-detection/correction in coding theory. In particular, we observe that results from the former area can be interpreted as yielding very efficient randomized error-detecting and error-correcting schemes for some well known codes. In Section 4 we use the "linearity tester" of Blum, Luby and Rubinfeld [BLR90] as an efficient error-detection scheme for the Hadamard Codes and this plays a crucial role in our proof. Later in Section 7 we use the "low-degree test" of Rubinfeld and Sudan [RS92], with its improved efficiency due to a technical lemma from [AS92], as an efficient mechanism to test Reed Solomon Codes.

Other ingredients in our proof borrow from work done in "parallelizing" the MIP=NEXPTIME protocol [LS91],[FL92]. The result described in Section 7 uses ideas from their work.

## 2 PCP and MAXSNP

The methods of [FGLSS91] and [AS92] have been applied so far only to the clique approximation problem. Here we show that the whole class of optimization problems can be handled in the same vein. The results of this section are due to independent observations of Mario Szegedy and Madhu Sudan and appear in [AMSS92].

Proof checking itself is a maximization problem from the point of view of the prover: the task is to maximize the chance of acceptance of the verifier. More formally, let $M^y(x, r)$ be a probabilistic oracle machine and let $R$ be the set of all possible values of $r$ for a given parameter $n$. The prover's intention is to solve the following optimization problem:

$$\mathbf{opt}(x) = max_y |\{r \in R|\ M^y(x, r) = 1\}|. \qquad (2)$$

In this paper we prefer to use the following reformulation of this problem:

$$opt(x) = \max_y Prob_r M^y(x, r). \qquad (3)$$

For fixed values of $r$ and $x$ the value of $M^y(x, r)$ can be computed by a circuit, which takes as many input bits from $y$ as the size of the query set. Let us denote this circuit by $C_{x,r}$. Equation 3 can now be rewritten as

$$opt(x) = \max_y Prob_r(C_{x,r}(y) = 1) \qquad (4)$$

If $M$ is a $PCP(f(n), g(n))$ machine for a language $L$ then there exists a $\delta$ such that $opt(x)$ is 1 if $x \in L$ and $opt(x) \le 1 - \delta$ if $x \notin L$.

Conversely, if $C_{x,r}(y)$ is a family of circuits such that

1. $|r| = O(f(n))$;
2. given $x$ and $r$, the circuit $C_{x,r}$ can be built in polynomial size;
3. $C_{x,r}(y)$ feeds from at most $g(n)$ bits of $y$;
4. for every $x$ the maximization problem corresponding to Equation 4 either has solution 1 or it has solution less than $1 - \delta$ for some fixed $\delta$,

then a probabilistic oracle machine can be built that recognizes the language $x:\ opt(x) = 1$.

**Definition 2.1** *We say that an optimization problem opt is well behaved if there is a $\delta$ such that for every $x$ either $opt(x) = 1$ or $opt(x) \le 1 - \delta$.*

Our observation above can be stated so that there is a one to one correspondence between well behaved optimization problems, where the underlying family of circuits (parameterized by $x$ and $r$; $|r| = f(|x|)$) is polynomially constructible and the problems in the class $PCP(f(n), g(n))$, where $g(n)$ is determined by the upper bound imposed upon the input size of circuits $C_{x,r}$ where $|x| = n$.

Let $M^y(x, r)$ be a machine PCP machine with constant size query sets that recognizes a language $L$. Then the size of the members of the the corresponding family of circuits $\{C_{x,r}|x,r\}$ is bounded, since their input size is bounded. To maximize the number of satisfiable constant size circuits each feeding from the same input is in MAX SNP. This can be seen or by reducing the problem to MAX-3SAT or directly (due to Yannakakis):

The input structure consists of a $(k+1)$-ary relation $A$ and $2^k$ unary relations $B_v$ indexed by the bit-vectors $v$ of length $k$. A tuple $(r, i_1, ..., i_k)$ is in $A$ if and only if $C_{x,r}$ inputs bits $i_1, ..., i_k$ on this choice (thus, $A$ contains one tuple for each choice). A relation $B_v$ contains a random choice $r$ if and only if the bit vector $v$ satisfies $C_{x,r}$.

The problem is to maximize over all unary relations (sets) $S$ (these are the sets of 1 bits in $y$) the number of tuples $(r, y_1, ..., y_k)$ that make the following formula true:

$$A(r, y_1, ..., y_k) \wedge \{[B_1 1..1(r) \wedge S(y_1) \ldots \wedge S(y_k)] \vee \ldots$$

$$\vee [B_0..0(r) \wedge \neg S(y_1) ... \wedge \neg S(y_k)]\}$$

When $|r| = f(n)$, the size of the instance $\{C_{x,r}|r\}$ is $2^{|r|} = 2^{O(f(n)}$. Taking the advantage of the gap $\delta$ between the optimal value $opt(x)$ for those instances $x$ that are in $L$ versus those that are not in $L$, an algorithm could solve the membership problem for $L$, which approximates the $opt(x)$ close enough to 1. Thus we conclude:

**Theorem 4** *If MAX SNP complete problems have PTAS, then $PCP(f(n), 1)$ is in $DTIME(2^{O(f(n))})$.*

## 3 Outline of the Proof of the Main Theorem

Feige et al [FGLSS91] elaborated on the techniques of Babai, Fortnow and Lund [BFL91] to show $NP \subset PCP(\log n \operatorname{loglog} n, \log n \operatorname{loglog} n)$. The $\log n \operatorname{loglog} n$ barrier stood in the way of a full characterization of $NP$ by $PCP$. Arora and Safra [AS92] introduced a new technique – recursive proof checking – that enabled them to characterize $NP$ as $PCP(\log n, \log\log n)$ [AS92]. In this section we translate this technique into our optimization problem scenario. Among the technicalities we build on data encoding stands out, and it is also a major motif in [BFL91, BFLS91, AS92].

Our proof of Theorem 2 is arrived at by the following basic steps. We observe that the recursion idea of [AS92] can be generalized to apply to any arbitrary proof system provided it satisfies certain restrictions. We shall consider a restricted type of the optimization problems discussed in Section 2.

**Definition 3.1** *A well behaved optimization problem $opt(x) = \max_y Prob_{r \in_U \{0,1\}^{f(n)}}(C_{x,r}(y) = 1)$ is restricted with parameters $(f(n), g(n))$ if*

1. *We can partition $y$ into a disjoint union of segments $y_i$, where each segment has size $O(g(n))$ and that each circuit $C_{x,r}$ depends only on a constant number of such segments.*
2. *Each circuit $C_{x,r}$ has size $(g(n))^{O(1)}$.*

*The class of languages recognized by such optimization problems we call $OPT(f(n), g(n))$.*

We remark that $PCP(f(n), g(n)) \supseteq OPT(f(n), g(n))$ is straightforward, but the other direction is true but not so straightforward.

In Sections 4 and 7, we work towards the development of such restricted proof systems. The results of these sections show that $NP \subset OPT(\text{poly}(n), 1)$ (Theorem 5) and $NP \subset OPT(\log n, \text{polylog}\, n)$ (Theorem 8). Theorems 9 and 10 show that the recursion idea applies to these proof systems, and in particular shows the following:

1. $OPT(f(n), g(n)) \subset OPT(f(n) + O(\log g(n)), (\log g(n))^{O(1)})$ and
2. $OPT(f(n), g(n)) \subset OPT(f(n) + (g(n))^{O(1)}, 1)$.

This allows us to conclude that $NP \subset OPT(\log n, \text{polyloglog}\, n)$ (by composing two $OPT(\log n, \text{polylog}\, n)$ proof systems) and then by composing this system with the $OPT(\text{poly}(n), 1)$ proof system we obtain $OPT(\log n, 1)$ proof system for $NP$.

## 4 A PCP machine for NP with constant number of queries

In this section we prove:

**Theorem 5** $NP \subseteq PCP(poly(n), 1)$.

When contrasted with $PCP(\log n, 1) = NP$ the lemma may appear weak, nevertheless a slight variation of the construction is an essential ingredient of our proof of Theorem 2.

We construct for 3SAT a $M^y(r, x)$ as in Definition 1.3. For every $x$ let us turn the 3-CNF formula represented by $x$ into an arithmetic circuit $C_x$ over $GF(2)$ with $l = O(|x|)$ gates. Look at $x$ as a Boolean formula and use DeMorgan's laws to eliminate the OR gates and add gates such that the fanin for every gate is at most 2. Thereafter replace every AND gate by a multiplication gate and every negation gate with a gate that computes the function $x \mapsto 1 - x$.

If $C_x$ is satisfiable, the successful setting of the input as well as the values of the gates for this input setting will be encoded by the prover into an exponential size table. The table will also facilitate the lookup of any quadratic expression of these values. If the table errs in more than 1% of the cases, the checking will reveal it with fixed positive probability. Lookups will be error corrected by random reduction. Looking at an appropriate random quadratic expression will reveal if the circuit makes a computational error. The checks are performed simultaneously. Only constantly many bits of the table will be checked.

Introduce variables $g_1, \ldots, g_l$ corresponding to the gates of $C_x$. We assume that $g_l$ corresponds to the output gate. We define:

$$\mathcal{P}_1 = \{\sum_{i=1}^{l} c_i g_i \mid c_i \in GF(2) \text{ for } 1 \le i \le l\};$$

$$\mathcal{P}_2 = \{c_0 + \sum_{i=1}^{l} c_i g_i + \sum_{1\le i<j\le l}^{l} c_{ij} g_i g_j \mid c_i,\, c_{ij} \in GF(2)\}.$$

We identify the index set of $y$ with $\mathcal{P}_2$ i.e., $y$ will have lenght $l^2 + l$ and for any $P \in \mathcal{P}_2$ we have a bit $y_P$ in $y$.

We construct a probability distribution $\mathcal{D}$ over $\mathcal{P}_2$ that depends on $C_x$ (thus on $x$). For every $1 \le i \le l$ let $P_i$ be the polynomial that is 0 if and only if the gate $g_i$ functions properly. That is, if $g_i$ is supposed to compute the product of $g_j$ and $g_k$, then $P_i = g_i - g_j g_k$. Similarly, for gates that compute addition, $P_i = g_i - (g_j + g_k)$, and for negation gates $P_i = g_i - g_j + 1$. If $g_i$ is one of the input gates then $P_i = 0$. We also introduce a polynomial $P_{l+1}$, which is 0 if and only if the output gate is 1: $P_{l+1} = g_l - 1$.

Define

$$P_\lambda = \sum_{i=1}^{l+1} \lambda_i P_i,$$

where $\lambda_i$ are independent uniformly distributed random variables over $GF(2)$; $\lambda = (\lambda_1, \ldots, \lambda_l)$. Clearly $P_\lambda \in \mathcal{P}_2$. Distribution $\mathcal{D}$ is defined by the way a random instance is generated: we take a random seed $\lambda$ and compute $P_\lambda$.

$M^y(x, r)$ is the value of the predicate $I_1 \wedge I_2 \wedge I_3 \wedge I_4$, where

$I_1:\ y_P + y_Q = y_{P+Q}$

$I_2:\ y_Z + y_{Z+1} - 1 = 0$

$I_3:\ (y_R + y_{R+L_1})(y_S + y_{S+L_2}) = (y_T + y_{T+L_1 L_2})$

$I_4:\ y_U + y_{U+V} = 0$

$P, Q, R, S, T, U, Z \in_{\mathcal{U}} \mathcal{P}_2$; $L_1, L_2 \in_{\mathcal{U}} \mathcal{P}_1$; $V \in_{\mathcal{D}} \mathcal{P}_2$, where $\in_{\mathcal{U}}$ means "chosen according to the uniform distribution" and $\in_{\mathcal{D}}$ means "chosen according to the distribution $\mathcal{D}$." P,Q,R,S,T,U,Z,$L_1$,$L_2$ are generated in an obvious manner; $V$ is generated from a random seed $\lambda$. Altogether $6(1 + l + \binom{l}{2}) + 2l + (l+1)$ random bits are used. All the computations can be done in polynomial time. The number of query bits is 13: $q_1 = P; \ldots; q_{13} = U + V$. We are yet to prove that the acceptance conditions hold.

If $x \in L$ then there is an assignment $A$ to the variables of $x$ that $x(A) = 1$. Evaluate $C_x$ on $A$. We define $\gamma_i$ as the value of the gate $g_i$ under this evaluation. Clearly $P_i(\gamma) = 0$ for $1 \le i \le l$, and $P_{l+1}(\gamma) = 0$, where $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_l)$. The latter because $C_x(A) = x(A) = 1$.

Define $y$ by $y_P = P(\gamma)$ for $P \in \mathcal{P}_2$.

$I_4$ holds for every $r$, since

$$y_U + y_{U+V} = U(\gamma) + (U+V)(\gamma) =$$

$$V(\gamma) = \sum_{i=1}^{l+1} \lambda_i P_i(\gamma).$$

The latter sum evaluates to 0 since $\gamma$ was defined so that each $P_i(\gamma)$ is 0. To show that $I_1, I_2$ and $I_3$ hold is even more obvious.

We prove now that if $x \notin L$ then the test fails with probability at least 0.01 for every $y$. Assume the opposite, that is that $I_1 \wedge I_2 \wedge I_3 \wedge I_4$ holds with probability at least 0.99 for some $y$.

First we show that there is a linear function $\mathbf{L}$ from $\mathcal{P}_2$ to $GF(2)$ such that

$$Prob_{P \in_{\mathcal{U}} \mathcal{P}_2}(y_P = \mathbf{L}(P)) \ge 0.98. \tag{5}$$

Since $I_1$ holds with probability at least 0.99 we have that $Prob(y_P + y_Q = y_{P+Q}) \ge 0.99$.

The existence of $\mathbf{L}$ now follows directly from the following lemma, due to Blum, Luby and Rubinfeld [BLR90]. The bound we state here appears in Rubinfeld [Rub90] and Gemmel et al. [GLRSW91].

**Lemma 6** *Let $g$ be a function such that*

$$Prob_{x,y}(g(x) + g(y) \ne g(x+y)) \le \delta/2$$

*then there exists a linear function* $\mathbf{L}$ *such that* $Prob(g(x) \ne \mathbf{L}(x)) \le \delta$.

Whereas $y_P$ is not always $\mathbf{L}(P)$, Equation 5 gives that for every $P \in \mathcal{P}_2$:

$$Prob_{R \in \mathcal{P}_2}(\mathbf{L}(R) = y_{R+P} + y_P) \ge 0.96. \tag{6}$$

This fact is referred to as "random reduction." Three instances of random reduction occur in the predicate $I_3$ and one instance in predicates $I_2$ and $I_4$. Our indirect assumption and Inequality 6 allow us to write:

$$Prob_{L_1, L_2 \in_{\mathcal{U}} \mathcal{P}_1}(\mathbf{L}(L_1)\mathbf{L}(L_2) = \mathbf{L}(L_1 L_2)) \ge 0.87, \tag{7}$$

$$\mathbf{L}(1) = 1, \tag{8}$$

$$Prob_{V \in_{\mathcal{D}} \mathcal{P}_2}(\mathbf{L}(V) = 0) \geq 0.95. \tag{9}$$

We claim that $\mathbf{L}(g_i)\mathbf{L}(g_j) = \mathbf{L}(g_i g_j)$ for all $1 \leq i, j \leq l$. Consider the matrices $Y$ and $Z$ defined by $Y(i,j) = \mathbf{L}(g_i)\mathbf{L}(g_j)$ and $Z(i,j) = \mathbf{L}(g_i g_j)$. Let the vector $v = v(L_1)$ be the vector of the coefficients of the linear function $L_1$. Similarly $w = w(L_2)$ is the vector of the coefficients of $L_2$. $\mathbf{L}(L_1)\mathbf{L}(L_2) = v^T Y w$ and $\mathbf{L}(L_1 L_2) = v^T Z w$. If the linear forms $L_1$ and $L_2$ are drawn randomly and independently from $\mathcal{P}_1$ then the vectors $v$ and $w$ are drawn randomly and independently from $GF(2)^l$.

We now use the argument of Freivald [Fre79] to show that $Y = Z$. It is easily verified that if $Y \neq Z$ then with probability at least 0.5, $v^T Y \neq v^T Z$. Moreover, for vectors $\alpha, \beta \in \mathcal{Z}_2^n$ and $w \in_R \mathcal{Z}_2^n$, if $\alpha \neq \beta$ then with probability at least 0.5, $\alpha^T w \neq \beta^T w$. It follows that for random $v$ and $w$, if $Y \neq Z$ then with probability at least 0.25, $v^T Y w \neq v^T Z w$. Thus from Inequality 7 we get that $Y = Z$. This in turn vindicates our claim.

The values $\mathbf{L}(g_i)$ for $i = 1, \ldots, l$ and $\mathbf{L}(g_i g_j) = \mathbf{L}(g_i)\mathbf{L}(g_j)$ determine the value of $\mathbf{L}(P)$ for any $P \in \mathcal{P}_2$ by linearity. Let us denote $\mathbf{L}(g_i)$ by $\gamma_i$. Then $\mathbf{L}(P)$ is expressed as $P(\gamma)$ (note that we here use that $\mathbf{L}(1) = 1$).

Now, since $x$ is not satisfiable, $\mathbf{L}(P_i) = P_i(\gamma) \neq 0$ for at least one of $1 \leq i \leq l+1$. But then

$$Prob_{V \in_{\mathcal{D}} \mathcal{P}_2}(\mathbf{L}(V) = 0) =$$

$$Prob_{V \in_{\mathcal{D}} \mathcal{P}_2}(V(\gamma) = 0) =$$

$$Prob_{\lambda \in_{\mathcal{U}} GF(2)^l}(\sum_{i=1}^{l+1} \lambda_i P_i(\gamma) = 0) = 0.5,$$

which contradicts Inequality 9. □

Later we need the following interpretation of the theorem:

**Theorem 7** *Let $C(y)$ be a circuit, where $|y| = n$. We can construct a family $\mathcal{F}_c = \{C_r(y') | r \in R\}$ of circuits each of size at most $O(1)$ where $y'$ is identified with $\mathcal{P}_2$ such that*

1. $|R| = 2^{O(2^n)}$;
2. *If $C(y) = 1$ then there is a setting of $y'$ such that*
   (a) $Prob_r(C_r(y') = 1) = 1$
   (b) $Prob_{Q \in \mathcal{P}_2}(y_Q + y_{Q+g_i} = g_i(y_i)) = 1$, *where $g(y)$ is the value of gate $g$ for input $y$.*
3. *If $Prob_r(C_r(y') = 1) \geq 1 - \delta$ then there is a $y$ such that $C(y) = 1$ and for every gate $g_i$ of $C$ $Prob_{Q \in \mathcal{P}_2}(y_Q + y_{Q+g_i} = g_i(y_i)) \geq 1 - 4\delta$.*

# 5 Reducing the number of query bits

In this section we reduce the size of the circuits that define well behaved optimalization problem. We will heavily use ideas in [BFLS91, AS92].

## 5.1 Encoding schemes

An encoding scheme $E = \{E_n\}$ is a polynomial time computable sequence of functions $E_n : \{0,1\}^n \rightarrow \{0,1\}^{p(n)}$, where $p(n)$ is a function that is polynomial in $n$. Moreover there is a $\delta > 0$ such that for every $n$ if $x$ and $x'$ are two different strings of length $n$ then the hamming distance $dist(x, x')$ between $E(x)$ and $E(x')$ is at least $\delta n$. An encoding scheme can be constructed using the Reed-Solomon codes [MS81]. For an encoding scheme we define the decoding $E^{-1}(z)$ as the $x$ that minimizes $dist(E(x), z)$.

## 5.2 Circuit verification

The theorem of proof verification in [BFLS91] we turn into circuit verification.

**Theorem 8** *Let $C(y)$ be an arbitrary circuit. Then a polynomial size family of circuits $C_r(y', y'')$ can be computed in polynomial-time from $C$ and $r$ with the properties:*

1. *Each circuit has size $(\log n)^{O(1)}$ and each of them depends only on a constant number of input bits from $y'$.*
2. *If $C_x(y) = 1$, then there exists a $y''$ such that $Prob_r(C_r(E(y), y'') = 1) = 1$.*
3. *If $Prob_r(C_r(y', y'') = 1) \geq 1 - \delta$ then $C(E^{-1}(y')) = 1$.*

The theorem can be generalized to circuits with constant number of inputs $y_1, y_2, \ldots, y_c$:

**Theorem 9** *Let $C(y_1, y_2, \ldots, y_c)$ be an arbitrary circuit. Then a polynomial size family of circuits $C_r(y'_1, \ldots, y'_c, y'')$ can be computed in polynomial-time from $C$ and $r$ with the properties:*

1. *Each circuit has size $(\log n)^{O(1)}$ and each of them depends only on a constant number of input bits from $y'_i$ for all $i = 1, 2, \ldots, c$.*
2. *If $C_x(y) = 1$, then there exists a $y''$ such that $Prob_r(C_r(E(y_1), \ldots, E(y_c), y'') = 1) = 1$.*
3. *If $Prob_r(C_r(y', y'') = 1) \geq 1 - \delta$ then $C(E^{-1}(y'_1), \ldots, E^{-1}(y_c)) = 1$.*

**Theorem 10** *In Theorem 9 we can also assume that $y''$ in segmented in such a way that the corresponding family of circuits meets the the segmentation requirement in Definition 3.1.*

The proof of this theorem is the topic of Section 7.

### 5.3 The reduction step

We show that we can take a well behaved optimization problem and use the circuit verification results from the previous section and obtain another well behaved optimization problem that recognizes the same language but where the circuits are much smaller. The basic idea is to use the circuit verification to verify that the circuits describing the optimization problem is satisfied.

**Theorem 11** $OPT(f(n), g(n)) \subset OPT(f(n) + O(\log g(n)), (\log g(n))^{O(1)})$.

**Proof:** Let $opt(x) = \max_y Prob_r(C_{x,r}(y) = 1)$ be a well behaved and restricted optimization problem that recognizes a language in $OPT(f(n), g(n))$. We construct another $opt'$ as follows. Fix $x$ and $r$. Without loss of generality for notational simplicity we assume that the circuit $C_{x,r}(y', y'')$ relies only on segments $y_1, \ldots, y_c$. Consider the family of circuits $\mathcal{F}_{x,r} = \{C_{x,r,r'}(y'_1, \ldots, y'_c, y''_r)|r')\}$ given by Theorem 10.

The set of circuits $\bigcup_r \mathcal{F}_{x,r}$ is such that

1. each circuit can be computed from $x, r$ and $r'$ in polynomial time;
2. each circuit has size $\log^{O(1)} g(n)$;
3. the new input set $y' = \bigcup_i y'_i \cup \bigcup_r y_r$ can be segmented in the following way: each $y_r$ is segmented as required in Theorem 10 and the segment size is $\log^{O(1)} g(n)$. Let us decompose the first union (each $y'_i$) into segments of size 1. As it is required in Theorem 10, each $C_{x,r,r'}$ takes only a constant number of bits from $\bigcup_i y'_i$ and a constant number of segments from $\bigcup_r y_r$.

We have to argue now that the optimization problem

$$opt'(x) = \max_y Prob_{r,r'}(C_{x,r,r'}(y') = 1) \tag{10}$$

is well behaved and the underlying language is the same as for $opt$.

Assume $opt(x) = 1$. then if we set each $y'_i$ to $E(y_i)$ for $(1 \leq i \leq l)$, where $y_i$ $(1 \leq i \leq l)$ is a solution of $opt(x)$, then by Theorem 7 for every $r$ there exists a $y''_r$ such that under this setting $\alpha$ of $y'$ all the circuits $C_{x,r,r'}$ output 1. Thus $Prob_r(C_{x,r,r'}(\alpha) = 1) = 1$.

Assume now that $opt(x) \leq 1 - \delta$. Let us denote the set of all possible random strings $r$ by $R$. Let $y'$ be arbitrary. Let $y_i = E^{-1}(y'_i)$ $(1 \leq i \leq l)$. Let us denote by $B$ the set of those $r$'s for which $C_{r,x}(y_1, y_2, \ldots, y_c) = 0$. By our assumption $|B|/|R| \leq 1 - \delta'$. We have

$$Prob_{r,r'}(C_{x,r,r'}(y') = 1) =$$

$$\sum_{r \in R} \frac{Prob_{r'}(C_{x,r,r'}(y') = 1)}{|R|} \leq$$

$$\frac{|B|(1-\delta) + (|R| - |B|)}{|R|} \leq 1 - \delta\delta'. \ \Box$$

## 6 The final verification

**Theorem 12** *Let $E$ be an arbitrary encoding scheme with parameter $p(n)$. Let $C(y_1, y_2, \ldots, y_c)$ be an arbitrary circuit of size $n$. Then there is a family $\mathcal{F} = \{C_r | r \in R\}$ of circuits on inputs $y'_1, \ldots, y'_c, y''$, $\delta > 0$ such that:*

1. *$|R| = 2^{n^{O(1)}}$.*
2. *Each circuit has size $O(1)$ and can be constructed in polynomial time given $x$ and $r$.*
3. *If $C(y) = 1$, then there exists a $y''$ such that $Prob_r(C_r(E(y_1), \ldots, E(y_c), y'') = 1) = 1$.*
4. *If $Prob_r(C_r(y'_1, \ldots, y'_c, y'') = 1) \geq 1 - \delta$ then $C(E^{-1}(y'_1), \ldots, E^{-1}(y_c)) = 1$, where $\delta > 0$ is some constant depending only on $c$.*

**Proof** Let $C'(y_1, y_2, \ldots, y_c)$ be such a circuit that we obtain from $C$ by adding extra gates to it to compute all the bits of each $E(y_i)$ for $i = 1 \ldots c$. We denote by $g_{i,j}$ the gate that computes the $j^{th}$ bit of $E(i)$ $(E(i,j))$. Build a circuit verification scheme for $C'$ as in Theorem 7. We identify $y''$ with the bits of this verification scheme. Let each $y'_i$ have length $p(|y_i|)$. Now flip a coin.

1. If tail, then by further coin flips choose a circuit from the verification scheme for $C'$.
2. If head, then pick randomly a number between 1 and $c$, and compare a randomly chosen bit $y'_i[j]$ of $y'_i$ with the value that the scheme $y''$ gives for $E(i,j)$. To compute this bit we have to look at $y_Q$ and $y_{g_{i,j}+Q}$, where $Q$ is a random element of $\mathcal{P}_2$. The check $y_Q + y_{g_{i,j}+Q} = y'_i[j]$ can be computed by a constant size circuit $C_{i,j,Q}$.

Family $\mathcal{F}$ is a union of $\{C_{i,j,Q} | 1 \leq i \leq c; 1 \leq j \leq p(|y_i'|); Q \in \mathcal{P}_2\}$ and the circuits of the circuit verification scheme for $C'$. The multiplicity of each circuit in the family is proportional to the probability with which it occurs in the above checking procedure. We leave the easy proof to the reader that the scheme satisfies the conditions of the theorem. $\Box$

Using the recursion idea (Theorem 11) and the above circuit verification procedure we obtain:

**Theorem 13** *$OPT(f(n), g(n)) \subset OPT(f(n) + (g(n))^{O(1)}, 1)$.*

## 7 Segmentation

The major obstacle is that the proof $y$ can not be segmented in such a way that the verifier only accesses a constant number of segments. We overcome this by showing that we can transform an unsegmented proof into a segmented proof where the verifier only accesses a constant number of segments.

**Lemma 14** *For any polynomial-time computable collection (the old query set)*

$$\{Q_r \subset \{1, 2, \ldots, n\} | r \in \{0,1\}^t\}$$

*then there exist a polynomial-time computable family of circuits $D_{r,r'}$ and a $\delta > 0$ such that for every string $y$ such that $|y| = n$ there exists a string $\tilde{y}$ such that:*

1. *We can partition $\tilde{y}$ into segments $\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_{\tilde{m}}$ all of length $l = O(k \log^2 n)$, where $k = \max_r |Q_r|$.*
2. *Each circuit has size $l^{O(1)}$ and has inputs from only a constant number of segments. The circuit $D_{r,r'}$ outputs either reject or a string of length $|Q_r|$. Furthermore $r'$ has length $O(\log n + \frac{\log k \log n}{\log \log n})$.*
3. *For all $r, r'$ we have that $D_{r,r'}(\tilde{y}) = y[Q_r]$, where $y[Q_r]$ is the substring of $y$ indexed by $Q_r$.*
4. *For all $z \in \{0,1\}^{\tilde{m}}$ if there exists an $r_0$ such that*

   $$Prob_{r'}(D_{r_0,r'}(z) = reject) \leq \delta$$

   *then there exists a $y$ such that for all $r$:*

   $$Prob_{r'}(D_{r,r'}(z) \notin \{reject, y[Q_r]\}) \leq 3\delta$$

**Proof:** The proof relies on the low degree code $E$ [BFL91, BFLS91] and a technical lemma about checking $E$ [RS92].

$E(y)$ is a function $E(y) : GF(p)^d \to GF(p)$, where $d = \lceil \log n / \log \log n \rceil$ and $p$ is a prime in $[(k \log n)^c, 2(k \log n)^c]$ for some constant $c$. Let $I = \{1, 2, \ldots \lceil \log n \rceil\} \subset GF(p)$. Since $|I^d| \geq n$ we can use $I^d$ as indexes of the bits of $y$. Thus we can look at $y$ as a function from $I^d$ to $\{0,1\}$. Given a function $y : I^d \to \{0,1\}$ there exists a unique multivariate polynomial $E(y)$ over $GF(p)$ that for all $\alpha \in I^d$ agrees with $y$ and the degree of any variable is bounded by $|I| - 1$. Note that the "total" degree of $E(y)$ is at most $d|I|$.

The new string $\tilde{y}$ contains three tables:

- A table $Y$ indexed by $GF(p)^d$, where segment $\alpha$ contains $E(y)(\alpha)$.
- A table $Y_{\text{lines}}$ that is indexed by $(GF(p)^l)^2$, where segment $(\alpha, \beta)$ contains the polynomial $E(y)(t\alpha + (1-t)\beta)$.
- A table $T$ that is indexed by $\{0,1\}^t \times GF(p)^d$. Let $Q_r = \{q_1, q_2, \ldots, q_k\}$. The segment $(r, \alpha)$ contains a polynomial $p_{r,\alpha}$ of degree at most $k(|I|-1)d$ such that $p_{r,\alpha}(0) = E(y)(\alpha), p_{r,\alpha}(1) = E(y)(q_1), \ldots, p_{r,\alpha}(k) = E(y)(q_k)$.

  The $p_{r,\alpha}$ exists, since it is possible to construct, by interpolation, a parameterized curve $C_{r,\alpha} : F_p \to F_p^l$ such that $C_{r,\alpha}(0) = r, C_{r,\alpha}(1) = q_1, \ldots, C_{r,\alpha}(k) = q_k$ and such that each coordinate function of $C_{r,\alpha}$ is a polynomial of degree $k$. Now $p_{r,\alpha}(t) = E(Y)(C_{r,\alpha}(t))$.

Let $r' = (\alpha, \beta, t, t')$ where $\alpha, \beta, \in GF(p)^d$ and $t, t' \in \{k+1, \ldots, p-1\}$. The circuit $D_{r,r'}$ checks that $Y[t\alpha + (1-t)\beta] = Y_{\text{lines}}[\alpha, \beta](t)$, $T[(r, \alpha)](t') = Y[C_{r,\alpha}(t')]$. If any of the checks fail then $D_{r,r'}$ rejects otherwise it's $i$th output for $i = 1, \ldots, k$ will be the least significant bit of $T[(r, \alpha)](i)$.

Note that given these definition it follows that

$$\forall r, r' : D_{r,r'}(\tilde{y}) = y[Q_r].$$

Thus we only need to prove the 4th claim, hence assume $z = (Y, Y_{\text{lines}}, T)$ is a string and for some $r_0$ we have that

$$Prob_{r'}(D_{r_0,r'}(z) \neq \text{reject}) \geq 1 - \delta.$$

This implies that

$$Prob_{(\alpha,\beta,t)}(Y[t\alpha + (1-t)\beta] \neq Y_{\text{lines}}[\alpha, \beta](t)) < \delta.$$

As in Section 4 we need a code checking lemma. This one is due to Rubinfeld and Sudan [RS92].

**Lemma 15 (Rubinfeld-Sudan [RS92])** *Let $g : GF(p)^d \mapsto GF(p)$ be a function and let $g_{lines}$ be a function that given $\alpha, \beta \in GF(p)^d$ determines a polynomial $p_{\alpha,\beta}$ of degree at most $k$ such that $p = O((kd)^c)$, where $c$ is some universal constant, and*

$$Prob_{\alpha,\beta,t}(g(t\alpha + (1-t)\beta) \neq g_{lines}(\alpha, \beta)(t)) \leq \delta$$

*then there exists a multivariate polynomial $f$ of degree at most $k$ such that $Prob_\alpha(g(\alpha) \neq f(\alpha)) \leq 2\delta$.*

Thus we can conclude that there exists a polynomial $g$ such that

$$Prob_\alpha(Y[\alpha] \neq g(\alpha)) < 2\delta. \qquad (11)$$

Let $y_i$ be the least significant bit of $g(i)$, where $i \in I^d$ (remember that we index the bits in a $n$ bit string by $I^d$).

Let $r$ be an arbitrary random string. If $T[(r, \alpha)] \neq g(C_{r,\alpha})$ then for at most $k(|I| - 1)d$ values of $t$ we have that $T[(r, \alpha)](t) = g(C_{r,\alpha}(t))$. Note that $C_{r,\alpha}(t)$ is uniformly distributed over $GF(p)^d$ when $\alpha, t \in_{\mathcal{U}} GF(p) \times \{k+1, \ldots, p-1\}$. Thus

$$\begin{aligned} & Prob_{r'}(D_{r,r'}(z) \notin \{\text{reject}, y[Q_r]\}) \\ = \;& Prob_{r'}(T[(r,\alpha)] \neq g(C_{r,\alpha}) \text{ and } \\ & \quad T[(r,\alpha)](t) = Y[C_{r,\alpha}(t)]) \\ \leq \;& Prob_{r'}(T[(r,\alpha)] \neq g(C_{r,\alpha}) \text{ and } \\ & \quad T[(r,\alpha)](t) = g[C_{r,\alpha}(t)]) + 2\delta \\ \leq \;& \frac{k(|I|-1)d}{p-k-1} + 2\delta < 3\delta \;\Box \end{aligned}$$

# 8 Further Work

Several questions remain open. There is a big gap between the (negligible) constants in the hardness results and the approximation ratio currently achievable for the MAXSNP problems. For example, MAX SAT can only be approximated to a ratio of 4/3 [Yan92], while MAX CUT and vertex cover can only be approximated to a ratio of 2 [GJ79, Mot92].

The recent results of Lund and Yannakakis [LY92] showed that the chromatic number is as hard to approximate as the clique and thus solce a long-standing open problem. They also show that the logarithmic ratio achievable for the set cover problem is essentially the best possible. Can the former result be extended to the case of 3-colorable graphs?

Another open problem is that of approximating the longest path in a graph. Based on our results, Karger, Motwani and Ramkumar and Azar [KMR92, Azar92] show that unless $P = NP$ there is no constant ratio approximation algorithm for longest paths. It is conjectured that this problem is as hard to approximate as clique or chromatic number.

# Acknowledgments

We acknowledge the contribution of Muli Safra. Furthermore we would like to thank the contributions of Yossi Azar, Manuel Blum, Tomas Feder, Joan Feigenbaum, Lance Fortnow, Magnus Halldorsson, David Karger, Moni Naor, Steven Phillips, Umesh Vazirani and Mihalis Yannakakis.

# References

[AMSS92] S. Arora, R. Motwani, S. Safra and M. Sudan. PCP and Approximation Problems. Unpublished note, 1992.

[AS92] S. Arora and S. Safra. Probabilistic Checking of Proofs: A New Characterization of NP. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, 1992.

[Azar92] Y. Azar. Personal Communication. 1992.

[Bab85] L. Babai. Trading group theory for randomness. In *Proc. 17th ACM Symp. on Theory of Computing*, pages 421–420, 1985.

[BFL91] L. Babai, L. Fortnow, and C. Lund. Non-deterministic Exponential Time has Two-Prover Interactive Protocols. *Computational Complexity*, vol. 1, pages 3–40, 1991.

[BFLS91] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking Computations in Polylogarithmic Time. In *Proc. 23rd ACM Symp. of Theory of Computing*, pages 21–31, 1991.

[BGKW88] M. Ben-Or, S. Goldwasser, J. Kilian and A. Wigderson. Multi-prover interactive proofs: How to remove intractability. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 113–131, 1988.

[BJLTY91] A. Blum, T. Jiang, M. Li, J. Tromp and M. Yannakakis. Linear Approximation of Shortest Superstrings. In *Proc. 23rd ACM Symp. on Theory of Computing* pages 328–336, 1991.

[BLR90] M. Blum, M. Luby, R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. In *Proc. 22th ACM Symp. on Theory of Computing*, pages 73–83, 1990.

[BP89] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Information Processing Letters*, vol. 32, pages 171–176, 1989.

[BS92] P. Berman and G. Schnitger. On the Complexity of Approximating the Independent Set Problem. *Information and Computation*, vol. 96, pages 77–94, 1992.

[Coo71] S. A. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd ACM Symp. on Theory of Computing*, pages 151–158, 1971.

[DJPSY92] E. Dahlhaus, D. S. Johnson, C. H. Papaditriou, P. D. Seymour and M. Yannakakis. The complexity of multiway cuts. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 241–451, 1992.

[Fag74] R. Fagin. Generalized First-Order Spectra and Polynomial-time Recognizable Sets. In Richard Karp (ed.), *Complexity of Computer Computations*, AMS, 1974.

[Fre79] R. Freivalds. Fast Probabilistic Algorithms. In *Mathematical Foundations of Computer Science.* Springer-Verlag Lecture Notes in Computer Science, No. 74, pages 57–69, 1979.

[FGLSS91] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 2–12, 1991.

[FL92] U. Feige and L. Lovasz. Two-Prover One-Round Proof Systems: Their Power and Their Problems. In *Proc. of 24th ACM Symp. on Theory of Computing*, pages 733–744, 1992.

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

[GJ76] Michael R. Garey and David S. Johnson. The complexity of near-optimal graph coloring. *J. of the ACM*, vol 23, pages 43-49, 1976.

[GJ78] Michael R. Garey and David S. Johnson. "Strong" NP-completeness results: motivation, examples and Implications. *J. of the ACM*, vol 25, pages 499-508, 1978.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[GLRSW91] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson. Self-Testing/Correcting for Polynomials and for Approximate Functions. In *Proc. 23th ACM Symp. on Theory of Computing*, pages 32–42, 1991.

[GMR89] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, vol. 18, pages 186–208, 1989.

[Kan91] V. Kann. Maximum bounded 3-dimensional matching is MAXSNP-complete. *Information Processing Letters*, vol. 37, pages 27-35, 1991.

[Kar72] R.M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[KMR92] D. Karger, R. Motwani and G. D. S. Ramkumar. On Approximating the Longest Path in a Graph. Manuscript, 1992.

[Lev73] L. Levin. Universal'nyĭe perebornyĭe zadachi (Universal search problems, in Russian), In *Problemy Peredachi Informatsii*, vol. 9, pages 265–266, 1973. A corrected English translation appears in an appendix to Trakhtenbrot [Tra84]

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 2–10, 1990.

[LS91] D. Lapidot and A. Shamir. Fully Parallelized Multi Prover Protocols for NEXP-time. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 13–18, 1991.

[LY92] C. Lund and M. Yannakakis. On the Hardness of Approximating Minimization Problems. Manuscript, 1992.

[MS81] F. MacWilliams and N. Sloane. *The Theory of Error-Correcting Codes.* North-Holland, 1981.

[Mot92] R. Motwani. Lecture Notes on Approximation Algorithms. Technical Report, Dept. of Computer Science, Stanford University (1992).

[PY91] C. H. Papadimitriou and M. Yannakakis. Optimization, Approximation, and Complexity Classes. *Journal of Computer and System Sciences*, vol. 43, pages 425–440, 1991.

[PY92] C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two, *Mathematics of Operations Research*, to appear.

[Rub90] R. Rubinfeld. A Mathematical Theory of Self-Checking, Self-Testing and Self-Correcting Programs. Ph.D. Thesis, Computer Science Department, U.C. Berkeley, 1990.

[RS92] R. Rubinfeld, and M. Sudan. Testing Polynomial Functions Efficiently and over Rational Domains. In *Proc. 3rd Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 23–32, 1992.

[Sha90] A. Shamir. IP=PSPACE. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 11–15, 1990.

[SG76] S. Sahni and T. Gonzalez. P-complete approximation problems. *JACM*, vol. 23, pages 555–565, 1976.

[Tra84] B. A. Trakhtenbrot. A survey of Russian approaches to *Perebor* (brute-force search) algorithms. In *Annals of the History of Computing* vol. 6, pages 384–400, 1984.

[Yan92] M. Yannakakis. On the Approximation of Maximum Satisfiability. In *Proc. 3rd Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 1–9, 1992.