# Interactive Proofs and the Hardness of Approximating Cliques

Uriel Feige [*] Shafi Goldwasser [†] Laszlo Lovasz [‡]
Shmuel Safra [§] Mario Szegedy [¶]

**Abstract**

The contribution of this paper is two-fold. First, a connection is shown between approximating the size of the largest clique in a graph and multi-prover interactive proofs. Second, an efficient multi-prover interactive proof for NP languages is constructed, where the verifier uses very few random bits and communication bits. Last, the connection between cliques and efficient multi-prover interactive proofs, is shown to yield hardness results on the complexity of approximating the size of the largest clique in a graph.

Of independent interest is our proof of correctness for the multilinearity test of functions.

## 1 Introduction

A *clique* in a graph is a subset of the vertices, any two of which are connected by an edge. Computing the size of the largest clique in a graph $G$, denoted $\omega(G)$, is one of the first problems shown **NP**-complete in Karp's well known paper on **NP**-completeness [26]. In this paper, we consider the problem of approximating $\omega(G)$ within a given factor. We say that function $f(x)$ *approximates* (from below) $g(x)$ within a factor $h(x)$ iff $1 \leq \frac{g(x)}{f(x)} \leq h(x)$.

[*]Department of Applied Math and Computer Science, the Weizmann Institute, Rehovot 76100, Israel. Part of this work was done while the author was visiting Princeton University.

[†]Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139. Part of this work was done while the author was visiting Princeton University.

[‡]Department of Computer Science, Yale University, New Haven, CT 06517. Part of this work was done while the author was visiting Princeton University.

[§]Department of Computer Science, Tel Aviv University, Tel Aviv, Israel. Part of this work was done while the author was visiting Princeton University.

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

The best upper bound known, is that $\omega(G)$ can be approximated within a factor of $\frac{n}{\log^2 n}$ [10] in polynomial time. It is natural to ask by how much this upper bound can be improved, and whether there is some factor within which approximating $\omega(G)$ is hard.

Proving that problem $L$ is **NP**-complete is usually taken as evidence as to the hardness of $L$. Since the best known decision procedure for **NP** runs in exponential time, showing that if $L \in \mathbf{P}$ then $\mathbf{NP} \subseteq DTIME(T(n))$ for some subexponential function $T$, may also be regarded as evidence that $L$ is hard. The smaller $T$ is, the stronger the evidence.

We prove two results of this type which in a sense trade the quality of approximation versus the complexity of $T$. First, if there exists a polynomial time algorithm which approximates $\omega(G)$ within any constant factor, then $\mathbf{NP} \subseteq DTIME(n^{O(\log\log n)})$. Secondly, if for some $\epsilon > 0$ there exist a $\tilde{\mathbf{P}}$ (*quasi*-polynomial time, $= \cup_{k>0} DTIME(n^{\log^k n})$ algorithm that approximates $\omega(G)$ within a factor $2^{\log^{1-\epsilon} n}$, then $\mathbf{NP} \subseteq \tilde{\mathbf{P}}$. Note that if $\mathbf{NP} \subseteq \tilde{\mathbf{P}}$ then NEXPTIME = EXPTIME.

The first result starts with a stronger assumption – the existence of a constant factor approximation, and gets as close to $\mathbf{P} = \mathbf{NP}$ as we were able to prove. The second result gives evidence that even within a large factor no approximation procedure for clique exists, discouraging attempts to extract even such little information about $\omega(G)$ in reasonable time.

The most interesting part of our work are the techniques used. We present a new connection between the complexity of approximation problems and probabilistic proofs. In particular, we show a connection between the complexity of approximating the size of the maximum clique in a graph and multi-prover interactive proofs.

An overview of our method is as follows. We show how to reduce an instance of a multi-prover interactive protocol for language $L$ on input $x$, into a graph $G_x$, whose size is exponential in the number of random bits $r$ used by the verifier in the multi-prover interactive orotocol, and in the number of answer bits $c$ the verifier received from the prover. The size of the maximum clique in $G_x$ corresponds exactly to the maximum acceptance probability of the input $x$ by the verifier. Suppose now that one can construct a multi-prover interactive proof with a gap of factor $g$ between the acceptance probability of the input $x$ in case $x \in L$ and $x \notin L$. It follows that if one can approximate the size of maximum clique in graph $G_x$ in time polynomial in size of the graph, with factor better than $g$, then one can decide membership in language $L$ in time polynomial in size of the graph. Finally, on the assumption that deciding membership in $L$ is a hard problem, we may conclude that approximating the size of maximum cliques within $g$ is a hard problem.

This approach starts to yield interesting results when the size of the graph $G_x$ is quasi-polynomially related to the size of $x$. To this end, our first effort is to show multi-prover interactive proofs for NP in which the verifier is efficient in his usage of randomness and in the number of answer bits received. More generally, we show given $L \in NTIME(T(n))$, a multi prover interactive proof for $L$ which is efficient (as related in $T(n)$) in randomness and answer bits. Let us elaborate.

## 1.1 Techniques from Interactive Proofs

We give a new characterization of **NP** in the domain of interactive proofs. We show that any language $L \in \mathbf{NP}$ is accepted by a multi-prover protocol (using the probabilistic oracle machine formulation of [19]) in which the number of random bits used by the verifier and answer bits sent by the oracle is small — $O(\log n \cdot \log\log n)$.

To do so, we consider the theorem of Babai, Fortnow and Lund ([5]), showing that NEXPTIME has multi-prover interactive proofs (This theorem implies that approximating the acceptance probability of multi-prover interactive proof systems on a given input is NEXPTIME-hard). First, we notice that this theorem can be scaled down to any non-deterministic time class $NTIME(T(n)) \subseteq NEXPTIME$, yielding a multi-prover protocol for any $L \in NTIME(T(n))$ with poly-logarithmic (in $T(n)$) number of random bits and communication bits. We then improve this bound by giving a new protocol for $L$ in which the number of answer bits and random bits is $O(\log T(n) \cdot \log\log T(n))$, while the running time of the verifier is $DTIME(T(n)^{O(1)})$. (e.g. for $L \in \mathbf{NP}$ the number of answer bits and random bits is $O(\log n \cdot \log\log n)$, and the verifier runs in polynomial time.)

**Remark:** The result that NP is recognizable by a verifier who uses $O((\log n)^c)$ randomness and answer size (which is sufficient to obtain the result that $\omega(G)$ is hard to approximate within a factor of $2^{\log^{1-\epsilon} n}$ for any $\epsilon > 0$, unless $\mathbf{NP} \subseteq \tilde{\mathbf{P}}$), was obtained [16] independently from [4] who also scaled down [5]'s protocol to the NP level. In their work, they bound the total running time of the verifier by poly(logarithmic) time rather than the particular parameters of randomness and answer size. To achieve verifer running time which is sublinear they also need to assume that the input is provided to the verifier in an error corrected form. Their motivation for scaling down the [5] protocol is not related to issues of hardness of approximation, and they attempt to optimize parameters that are different from the parameters that we optimize. The result that NP is recognizable by a verifier who uses $O(\log n \log\log n)$ randomnesss and answer size, was developed concurrently with the work of [4] . Indeed, some of the techniques used in latter versions of both works are similar.

Our work shows an interesting relation between work on interactive proof systems and long standing open problems in computational complexity theory. For earlier examples, see [21], [11], [18], [14], [13].

## 1.2 The Multi Linearity Test

An important ingrediant in the [5] protocol is a *multilinearity test*, which is a procedure of sampling a multivariate function on a small fraction of its domain, and using this random sample in order to decide whether the function is linear in each of its variables (over almost all of its domain). In [5]'s multilinearity test, the number of points that are sampled is polynomial in the number of variables. In order to design efficient multi-prover interactive proofs (leading to stronger hardness of approximation results for $\omega(G)$), it is important to have multilinearity tests in which the number of points sampled is as small as possible. We design a more efficient multilinearity test that uses a sample that is only linear in the number of variables. The analysis of our multilinearity test is simpler than the analysis presented in [5]. Moreover, our analysis is tight (up to low order additive terms) in the special case that the multivariate function differs on at most half its points from a true multilinear function. Obtaining a tight analysis for the case that the multivariate function differs from any true multilinear function on more than half of its points remains a challenging open question.

## 1.3 Previous Results on Approximation

The classification of computational problems as either tractable, i.e., in **P**, or intractable, i.e., **NP**-hard, has been quite a successful enterprise for the last twenty years. However, there was a distinct lack of techniques for classifying approximation problems. For many NP-hard optimization problems, there was neither a good approximation algorithm known, nor was there any **NP**-hardness type of evidence that the problem is hard to approximate. Approximating $\omega(G)$ is a prime example of this situation. As a less extreme example, consider the *chromatic number* problem (computing the minimum number of colors required to color the nodes of a graph so there is no monochromatic edge). It was known that approximating the chromatic number within an $\frac{n(\log\log n)^2}{\log^3 n}$ factor is in polynomial time ([34], [8],[24]), while approximating the chromatic number within any factor smaller than 2 is **NP**-hard [22]. Approximating the chromatic number within any factor between 2 and $\frac{n}{\log^3 n}$ was not known to be in **P** or to be **NP**-complete. Another example is the *vertex cover* problem (the minimum subset of vertices that contains at least one of any two adjacent vertices). The minimum size vertex cover can be approximated within a factor of $2 - \Omega(\frac{\log\log n}{\log n})$ [6, 30] in polynomial time. No **NP**-hardness results was known for approximating vertex cover.

Papadimitriou and Yannakakis [32] initiated a classification of **NP** optimization problems based on their logical characterization. They define the class MAX **NP**, and use its logical characterization to infer that all problems in this class can be approximated. They define the class MAX SNP and show approximation problems which are complete for this class under a reduction that preserves constant approximation. Some examples of complete problems in this class are independent set in bounded degree graphs and satisfying the maximum number of clauses in a Boolean(CNF) formula. Panconesi and Ranjan [31] extend [32]'s approach and define the class MAX $\Pi_1$. The complete problems for MAX $\Pi_1$ cannot be approximated unless $\mathbf{P} = \mathbf{NP}$. Consequently, [31] define subclasses of MAX $\Pi_1$ for which the approximability of the complete problems was open. Approximating $\omega(G)$ (or MAX CLIQUE, in [31]'s terminology) is in RMAX(2), which is the lowest class that [31] define. Our results imply that if $\tilde{\mathbf{P}}$ approximation algorithms exist for any of the approximation classes that [31] define, or for any of the RMAX(2)-hard approximation problems, then $\mathbf{NP} \subset \tilde{\mathbf{P}}$.

Berman and Schnitger[9] show that approximating $\omega(G)$ within factor $n^\epsilon$ is hard for MAX SNP under randomized reductions. Namely, if clique has $n^\epsilon$ approximation for arbitrarily small $\epsilon$, then all problems in this class have polynomial time constant approximation schemes within factors arbitrarily close to 1. Alon and Boppana [1] show that there is no polynomial time *monotone* circuit that approximates the size of the maximum clique in an $n$-node graph up to a factor of $\frac{n}{\log^{O(1)} n}$.

## 1.4 Subsequent Work on Approximation

When our work first appeared, we raised two major open problems [17].

1. Can the methods be extended to prove that if the clique function can be approximated to within factor $f$, then $\mathbf{P} = \mathbf{NP}$? Say, even for $f = 2$? Using a similar proof outline to the one we use here to answer this question in the affirmative, would entail showing an $O(\log n)$ bound on the number of coins and answer bits that the verifier receives, whereas we show an $O(\log n \log\log n)$ bound.

2. Our methods are well suited to attack the clique approximation problem. Can similar methods be used to derive hardness results for approximating other NP-hard functions?

The announcement of the results of this paper in [17], was followed by a sequence of rapid and exciting developments in which tremendous progress was made regarding the relation between interactive proofs (in their various forms) and hardness of approximation problems.

Both of the open problems we posed received affirmative answers.

The first of these questions was solved by Arora and Safra [3]. To describe their results we adopt their notation, which has become standard by now. Let $PCP(r(n), c(n))$ denote the class of languages that have *probabilistically checkable proofs* in which the verifier uses $O(r(n))$ random bits, receives $O(c(n))$ answer bits, and the error (the probability of accepting a false "proof" of an incorrect statement) is at most 1/2. Using this terminology, the results in the current paper show that $NP \subset PCP(\log n \log\log n, \log n \log\log n)$, and that if it is easy to approximate $\omega(G)$ within some constant factor then $PCP(r,a) \in DTIME(2^{r+a})$, and thus if $NP \subset PCP(\log n, \log n)$, then it is NP-hard to approximate $\omega(G)$ within any constant factor. Arora and Safra have shown that in fact, $NP \subset PCP(\log n, \sqrt{\log n})$. They concluded that $\omega(G)$ cannot be approximated within a factor of $2^{O(\sqrt{\log n})}$, unless $\mathbf{P} = \mathbf{NP}$.

Arora, Lund, Motwani, Sudan, and Szegedy [2] improved upon the work of [3], and showed that $NP \subset PCP(\log n, 1)$. This implied that it is NP-hard to approximate $\omega(G)$ within a factor of $n^{\epsilon}$, for some $\epsilon > 0$. More importantly, the quantitative improvement in the number of answer bits received by the verifier, led also to a result that MAX-3SAT cannot be approximated within a factor of $1+\epsilon$, for some $\epsilon > 0$. Since MAX-3SAT is in MAX-SNP, the theory of MAX-SNP completeness (as developed in [32] and subsequent works) automatically resulted in similar hardness of approximation results for a large number of other optimization problems. One such optimization problem is that of vertex cover, mentioned in Section 1.3.

Both our reduction from PCP protocols to clique, and the [2] reduction to MAX-3SAT, treat the PCP characterization of NP as a blackbox. Lund and Yannakakis[29] looked more carefully at the structure of the protocols that give this characterization. Using this structure, they derived sophisticated reductions that showed that it is NP-hard to approximate the chromatic number within a factor of $n^{\epsilon}$ for some $\epsilon > 0$ (compare with Section 1.3), and that *set cover* cannot be approximated within a factor of $(\log n)/4$ unless $\tilde{\mathbf{P}} = \tilde{\mathbf{NP}}$. (Approximating set cover within a factor of $\ln n$ is in $\mathbf{P}$.)

For further references, see the survey paper of Johnson [25], and the bibliographical list in [15].

### 1.5 Roadmap

The paper is organized as follows: In section 2 we introduce some notation and the model of multi-prover interactive proofs. In section 3 we show the connection between multi-prover proofs and approximating the clique problem. In section 4 we improve the efficiency of [5]'s proof system and scale it down to complexity classes lower than NEXPTIME.

## 2 Multi-Prover Protocols

The model of multi-prover interactive proofs was introduced by Ben-Or, Goldwasser, Kilian, and Wigderson in [7]. It is defined as follows.

Let $P_1, P_2$ be infinitely powerful machines and $V$ be a probabilistic polynomial-time Turing machine, all of which share the same read-only input tape. The verifier $V$ shares communication tapes with each $P_i$, but provers $P_1$ and $P_2$ have no common tapes except the input tape. ($P_1$ does not see the conversation between $V$ and $P_2$, and $P_2$ does not see the conversation between $V$ and $P_1$).

Formally, each $P_i$ is a function from the input and the conversation with the verifier it has seen so far to a new message. Similarly, $V$ is a function from the input, a random string, and the conversation with both provers it has seen so far to a new message. $V$ is a polynomial time computable function.

At the end of the conversation, $V$ outputs *accept* (or *reject*) based on the input $x$, the random string $r$, and the entire conversation it has had with both provers. We then say that multi-prover interactive proof $(V, P_1, P_2)(x, r)$ accepts (or rejects).

**Definition 1** *A language L is* accepted *by a* multi-prover interactive proof *if:*

1. $(\forall x \in L)$ $(\exists P_1, P_2)$ *s.t.*
   $\Pr_r[(V, P_1, P_2)(x, r)$ *accepts* $] = 1$
2. $(\forall x \notin L)$ $(\forall P_1, P_2)$
   $\Pr_r[(V, P_1, P_2)(x, r)$ *accepts* $] < \frac{1}{2}$.

We let **MIP** denote the class of languages accepted by some multi-prover interactive proof.

A useful alternative formulation of **MIP** was suggested by Fortnow, Rompel and Sipser ([19]) as follows.

Let $M$ be a probabilistic polynomial time Turing machine with access to a memoryless oracle $O$ ($O$ is a function from the query sent by $M$ to an answer to $M$, i.e., $O$ gives the same answer on the same query, regardless of the history of communication between $M$ and $O$). $M$ is a polynomial-time function from an input $x$, a random string $r$, and the history of communication with oracle $O$, to $M$'s next query to the oracle. We denote by $M(x, r, h)$ the query sent by $M$ on input $x$, random string $r$ and communication history $h$ with the oracle. We write that $M^O(x, r)$ *accepts* if machine $M$ communicating with oracle $O$ on input $x$ and random string $r$ accepts $x$.

We define the class of languages that can be accepted by these machines as follows:

**Definition 2** *A language L is* accepted *by a probabilistic oracle-machine M iff*

1. *For every $x \in L$, there is an oracle $O$ such that $Pr_r[M^O(x, r)$ accepts $] = 1$.*
2. *For every $x \notin L$ and for all oracles $O$, $Pr_r[M^O(x, r)$ accepts $] < \frac{1}{2}$.*

This differs from the standard interactive protocol model in that the oracle is memoryless and thus might as well be fixed ahead of time, while in an interactive proof the prover may let his future answers depend on previous questions.

Intuitively, one may think of this oracle as representing an exponential size bounded proof that the input $x$ is in the language $L$. The machine $M$ has to verify with high probability that the proof is correct, using only the capability of choosing randomly a "small" set of places to look at in the proof.

**Theorem 1 ([19])** *$L$ is accepted by a probabilistic oracle-machine iff $L$ is accepted by a multi-prover interactive protocol.* $\square$

**Important Note:** From now on we will use the probabilistic oracle-machine formulation of **MIP** in order to prove our results.

**Theorem 2 ([5])** **MIP**$=$ *NEXPTIME.*

This is a striking phenomenon; that for any language that has an exponentially long proof of membership, there exists an alternative proof of membershoip that can be verified with high probability by a random polynomial-time machine.

There are three complexity measures that we define for a probabilistic oracle-machine $M$ on input $x$, $|x| = n$: the number of random coins $M$ tosses, the number of bits sent by the oracle to $M$, and the running time of $M$ on $x$.

**Definition 3** *Given a probabilistic oracle-machine $M$, let $r_M(n)$ be the maximum (taken over all inputs $x$ of length $n$ and all oracles $O$) number of random bits $M$ uses on input $x$, and $c_M(n)$ be the maximum (taken over all inputs $x$ of length $n$ and all oracles $O$) number of answer bits sent by the oracle to $M$. We will drop the subscript $M$ when it is obvious from the context.*

**Remark on notation** *: Arora and Safra [3] introduced the notation $PCP(r(n), c(n))$ for the set of languages $L$ accepted by probabilistic oracle machine $M$ with $r_M(n) \leq O(r(n))$ and $c_M(n) \leq O(c(n))$.*

In the [5] protocol to recognize $L \in NEXPTIME$, the probabilistic oracle machine $M$ that accepts $L$ runs in polynomial time and uses a polynomial number of random and communication bits with the appropriate oracle. Thus, $M$ uses resources that are poly-logarithmic in the running time of a non-deterministic Turing machine that recognizes $L$.

In this paper we scale down and improve the efficiency of [5]'s protocol as follows:

**Theorem 3** *Any language $L \in NP$ is accepted by a probabilistic oracle-machine $M$ such that $r_M(n)+c_M(n) \leq \log(n)\cdot\log\log(n)$, and $M$'s running time is $DTIME(n^{O(1)})$. (i.e. $NP \subset PCP(\log n \log\log n, \log n \log\log n)$).*

We prove this Theorem in Section 4.

**Remarks:**

1. For a given error probability, the bound one obtains on the answer bits plus random bits used by the probabilistic oracle machine, is better than the bound one would obtain on the answer bits plus random bits used by a verifier in the corresponding multi-prover interactive proof.
2. Although we scale down the number of random bits and answer bits, we do not scale down the running time of $M$, which remains polynomial. Indeed, a significant scaling down of this measure is not possible, since $M$ needs linear time just to read the input string. Luckily, such further scaling is not necessary for our purpose.
3. The following "scaled up" version is an easy corollary of Theorem 3:

   Any language $L \in NTIME(T(n))$ (for $T(n) \geq n$) is accepted by a probabilistic oracle-machine $M$ such that $r_M(n)+c_M(n) \leq \log(T(n))\cdot\log\log(T(n))$, and $M$'s running time is $DTIME(T(n)^{O(1)})$.

## 3 Multi-Prover Protocols and Approximating Clique

**Theorem 4** *1. If approximating $\omega(G)$ within any constant factor is in* **P** *then $NP \subseteq DTIME(n^{O(\log\log n)})$.*

*2. If, for some $\epsilon > 0$, approximating $\omega(G)$ within a factor $2^{\log^{1-\epsilon} n}$ is in $\tilde{\mathbf{P}}$, then $NP \subseteq DTIME(2^{(\log n)^k})$.*

**Proof:** We first show that if approximating $\omega(G)$ within a factor of 2 is in **P** then $NP \subseteq DTIME(n^{O(\log\log n)})$. We will then show how the approximation factor can be amplified, yielding both parts of the theorem.

Let $B$ be an algorithm which approximates $\omega(G)$ to within a factor of 2 and let $T_B(|G|)$ be its running time. Let $L$ be an NP language. Let $M$ be a probabilistic oracle machine which accepts $L$, $r(n) = r_M(n)$, $c(n) = c_M(n)$, and $T_M(n)$ be $M$'s running time on inputs $x$ of length $n$. Using the machine $M$, we reduce the question of membership of $x$ in $L$ to approximating $\omega(G)$ in a graph in the following two steps procedure:

1. Construct a graph $G_x$, $|G_x| \leq 2^{r(n)+c(n)}$ such that if $x \in L$, then $\omega(G_x) = 2^{r(n)}$ and if $x \notin L$, then $\omega(G_x) < \frac{2^{r(n)}}{2}$.
2. Run the approximation procedure $B$ on $G_x$. If the answer for the approximated $\omega(G_x)$ is greater than $\frac{2^{r(n)}}{2}$ then accept $x$, else reject $x$.

We now show how to construct, given $M$ and input $x$, a graph $G_x$ that satisfies condition 1 of the theorem.

We need to introduce the notion of accepting transcripts and consistent transcripts for this purpose.

Informally, in the following definition we will let $q_i$ denote queries, $a_i$ denote oracle answers, and a transcript to be a possible complete history of $M$'s view.

**Definition 4** *A string $t = \langle r, q_1, a_1, ..., q_l, a_l \rangle$ is a* transcript *of a probabilistic oracle-machine $M$ on input $x$, if $|r| = r(n)$, $(|a_1| + ... + |a_l|) \leq c(n)$ and for every $i$, $q_i = M(x, r, \langle q_1, a_1, ..., q_{i-1}, a_{i-1} \rangle)$. A transcript is an* accepting transcript, *if $M$ on input $x$, random string $r$ and history of communication $\langle q_1, a_1, ..., q_l, a_l \rangle$ accepts $x$.*

**Definition 5** *We say that two transcripts $t = \langle r, q_1, a_1, ..., q_l, a_l \rangle$ and $\hat{t} = \langle \hat{r}, \hat{q}_1, \hat{a}_1, ..., \hat{q}_{\hat{l}}, \hat{a}_{\hat{l}} \rangle$ are* consistent *if for every $i, j$, if $q_i = \hat{q}_j$, then $a_i = \hat{a}_j$.*

We are now ready to define the graph $G_x$, whose maximum clique size reflects membership of $x$ in $L$.

**Definition 6** *Let $M$ be a probabilistic oracle machine which accepts $L$. For an input $x$, define the graph $G_x$ as follows. The nodes of $G_x$ are all accepting transcripts of $M$ on $x$. Two nodes in $G_x$ are connected by an edge iff they are consistent.*

Our notation of a transcript contains redundant information, since $M$'s queries can be computed efficiently from the input $x$, its random bits, and the oracle's answers. We can *compress* a transcript by discarding $M$'s queries, or *expand* a compressed transcript by running $M$'s algorithm. In order to construct the set of nodes of $G_x$ we enumerate all the *compressed* transcripts (this takes time $2^{r(n)+c(n)}$), and then run $M$ on each transcript to expand it and check that it is accepting. Observe that $2^{r(n)+c(n)}$ is an upper bound on the number of nodes of $G_x$.

**Lemma 5** $\max_O \Pr_r[M^O(x, r) \text{ accepts}] \cdot 2^{r(n)} = \omega(G_x)$

**Proof:** We first show that $\max_O \Pr_r[M^O(x,r)\ \textit{accepts}] \cdot 2^{r(n)} \leq \omega(G_x)$. Let $O$ be an oracle for which $\Pr_r[M^O(x,r)\ \textit{accepts}]$ is maximal, and denote this last probability by $p$. Consider the $p \cdot 2^{r(n)}$ transcripts for which $M^O(x,r)$ accepts. Since these transcripts all correspond to the same oracle $O$, they are pairwise consistent. Hence they constitute a clique of size $p \cdot 2^{r(n)}$ in $G_x$.

We now show that $\max_O \Pr_r[M^O(x,r)\ \textit{accepts}] \cdot 2^{r(n)} \geq \omega(G_x)$. Consider a clique of maximum size in $G_x$, and let $k$ denote its size. The transcripts corresponding to any two nodes in the clique are consistent. Hence, for any query of $M$ that appears in a transcript representing a node in the clique, the same response appears in all transcripts of the clique that contain the query. Thus we can define a partial function, $O'$, from $M$'s queries that occur in the clique to the oracle's responses. We extent this partial function to an oracle $O$, by assigning arbitrary responses to queries that do not appear in transcripts that correspond to nodes of the clique. Each of the $k$ nodes of the clique is consistent with $O$. A clique in $G_x$ cannot contain two transcripts with the same random string, since this would force the transcripts to be inconsistent. Finally, each of the transcripts in the graph, and hence in the clique, is accepting. Hence $M^O$ accepts for at least $k$ random strings. □

Clearly, if $x \in L$ then $\omega(G_x) = 2^{r(n)}$ and if $x \notin L$ then $\omega(G_x) < \frac{2^{r(n)}}{2}$, and thus the condition in item 1 of the theorem holds.

The entire two-step procedure runs in time $T_B(2^{r(n)+c(n)}) + 2^{O(r(n)+c(n))}T_M(n)$. By Theorem 3, our statement regarding the hardness of approximating $\omega(G)$ within a factor of 2 is proved, since the running time of the two step procedure is bounded by $n^{O(\log\log n)}$. Note that since the running time is dominated by $2^{O(r(n)+c(n))}$ the result will hold even if we allow $T_M$ to be bounded by $2^{O(r(n)+c(n))}$.

In order to amplify the factor of approximation, we construct the graph $G'_x$ that corresponds to the protocol $M'$. $M'$ simply runs $\ell$ iterations of $M$ on input $x$, and accepts $x$ if $M$ accepts $x$ on all $\ell$ iterations. The ratio between the value of $\max_O Pr_r[M'^O(x,r)]$ when $x \in L$ and when $x \notin L$ is a factor of $2^\ell$. To prove the first part of the theorem, let $\ell$ be an arbitrary constant, and observe that $c(n)$ and $r(n)$ increase only by a factor of $\ell$. To prove the second part of the theorem, let $\ell = (r(n) + c(n))^{\frac{1-\epsilon}{\epsilon}}$. Note that by this choice of $\ell$, the number of random and answer bits used by $M'$ is still poly-logarithmic in $n$. The ratio between the size of the maximum clique in $G'_x$ when $x \in L$ and when $x \notin L$ is a factor of $2^{\log^{1-\epsilon}|G'_x|}$, where $|G'_x|$ is the size of $G'_x$. □

As an immediate corollary we get:

**Corollary 6** *If, for some $\epsilon > 0$, approximating $\omega(G)$ within a factor $2^{\log^{1-\epsilon} n}$ is in $\tilde{\mathbf{P}}$, then:*

1. $\mathbf{NP} \subseteq \mathbf{N}\tilde{\mathbf{P}} = \tilde{\mathbf{P}}$
2. *NEXPTIME = EXPTIME*

## 3.1 Graph Products and Probabilistic Oracle Machines

An alternative proof of the second part of Theorem 4 is to take the graph $G_x = \langle V, E \rangle$ which corresponds to running the protocol once and define the following graph product (see [23]): $G_x^2 = \langle V', E' \rangle$ where $V' = V \times V$ and $(\langle v_1, v_2 \rangle, \langle v_1', v_2' \rangle) \in E'$ iff $\{(v_1, v_1') \in E$ *or* $v_1 = v_1'\}$ *and* $\{(v_2, v_2') \in E$ *or* $v_2 = v_2'\}$. It is easy to see that $\omega(G_x^2) = \omega(G_x)^2$, and part 2 of the theorem follows naturally from part 1. It is interesting to note that $G_x^i$ is exactly the graph $G_x'$ resulting from repeating $M$'s protocol $i$ times. As taking graph products is a well known technique for amplifying graph properties (such as clique or chromatic number), the analogy with amplifying the probability of success of a protocol may be of further value.

## 3.2 A Comparison Between Graphs and Oracles

We represented possible executions of probabilistic oracle machines as graphs, where oracle like behavior translates into rules of how to place edges in the graph. The rule we used was that two nodes are connected if identical questions have identical answers. However, more generally, we would like two nodes to be connected by an edge if and only if the joint answers represented by these nodes are consistent with the assumption that the input $x$ should be accepted. This more general rule may reduce the size of the maximum clique in the case that $x \notin L$ to a value smaller than the one implied by the error probability of the corresponding proof system. It would be interesting to see if this observation can be used to obtain better bounds on clique approximation, or in other future work.

# 4 Efficient Multi-Prover Protocols for NP

In this section we consider a scaled down analogue of the theorem of Babai, Fortnow and Lund [5] showing that $NEXPTIME = \mathbf{MIP}$. For any NP language $L$, we construct a probabilistic oracle machine $M$ which accepts $L$. Our protocol has the same general structure as that of [5] but some features are modified so as to improve its efficiency. These include the use of arithmetic over finite fields instead of over the integers, [1] the use of pseudo-random sampling, and a tighter analysis of the multilinearity test.

We now proceed to describe the ingredients of our protocol.

## 4.1 Arithmetization

We follow ideas developed in [28], [33], [5].

[1] A similar change is also suggested in [5] and [4].

If suffices to show a procedure for deciding the **NP**-complete language 3-SAT. Let $f$ be a given 3CNF formula of length $n$. Let $m$ be the smallest integer satisfying $2^m \geq n$. Let $c$ denote a clause number and $\vec{v}$ denote a variable name. (To prepare for the later multilinear representation we represent the variable names as strings in $\{0,1\}^m$).

For ($j = 1, 2, 3$) and for clause $c$, let $\chi_{j,c}\colon \{0,1\}^m \to \{0,1\}$ be an indicator function satisfying:

$$\chi_{j,c}(\vec{v}) = \begin{cases} 1 & \text{if } \vec{v} \text{ is the } j\text{th variable of } c \\ 0 & \text{otherwise} \end{cases}$$

As an example, assume that $m = 3$, and that the first variable of the fourth clause is $x_5$ (represented in binary as 101). Then $\chi_{1,4}(\vec{v})$ is the function $v_1(1 - v_2)v_3$, which evaluates to 1 if $v_1 = 1$, $v_2 = 0$, $v_3 = 1$, and evaluates to 0 otherwise. Note that $\chi_{1,4}$ is linear in each of its variables, and this is the general rule for each of the $\chi_{j,c}$.

For $j$ and $c$ as before, let $s_{j,c}$ be a shorthand notation satisfying:

$$s_{j,c} = \begin{cases} 1 & \text{if the } j\text{th variable of } c \text{ is positive} \\ 0 & \text{if complemented} \end{cases}$$

A truth assignment $A$ is a function from variable names to Boolean values

$$A\colon \{0,1\}^m \to \{0,1\}\,.$$

The following expression over any field $\mathcal{F}$ ($CC$ standing for clause check)

$$CC(A,c) = \sum_{\vec{v}_1,\vec{v}_2,\vec{v}_3 \in \{0,1\}^m} \prod_{j=1}^{3} \chi_{j,c}(\vec{v}_j) \cdot (s_{j,c} - A(\vec{v}_j))$$

evaluates to 0 iff $A$ satisfies clause $c$ of $f$.

In order to verify that $A$ is a satisfying assignment, one needs to check that for all $c$, $CC(A,c) = 0$. An oracle machine can do this by requesting the complete description of $A$ from the oracle. However, this uses a polynomial number of answer bits. We develop a different procedure which requires only $O(\log n \log\log n)$ answer bits.

Let $\vec{v} = v_1, .., v_m$. We extend the arithmetization in such a way that $v_i$ will take arbitrary values from $\mathcal{F}$. This requires to extend the domain of the functions $A$ and $\chi_{j,c}$ from $\{0,1\}^m$ to $\mathcal{F}^m$.

**Definition 7** *Let $\mathcal{F}$ be any field. Given a function $f\colon \{0,1\}^m \to \mathcal{F}$, the* multilinear extension *of $f$ over $\mathcal{F}$, denoted $\hat{f}\colon \mathcal{F}^m \to \mathcal{F}$, is such that:*

*1. $\forall \vec{y} = y_1, ..., y_m$, where $y_i \in \{0,1\}$: $\hat{f}(\vec{y}) = f(\vec{y})$*

*2. $\hat{f}$ can be expressed as a multinomial of degree 1 in all variables $y_1, ..., y_m$.*

The procedure of constructing the multilinear extension of a multi-variate Boolean function $f$ is well known, and we present it for completeness. Represent $f$ in Disjunctive Normal Form. Arithmetize by replacing any *or* by addition, replacing any *and* by multiplication, and replacing any occurance of a negated variable $y_i$ by the expression $(1 - y_i)$. To obtain $\hat{f}$, simply let all variables range over $\mathcal{F}$.

$M$ can easily compute multilinear arithmetic expressions for each $\chi_{j,c}$, and can easily determine each $s_{j,c}$. The oracle can hold the multilinear representation of $A$. Each $CC(A, c)$ is a multinomial of degree 6 in each of the variables $v_i$. Since $M$ does not know the value of $A$ on each $\vec{v} \in \{0,1\}^m$, $M$ cannot check that the value of each $CC(A, c)$ is 0. However, we can use [5]'s procedure for this purpose. It has two major components:

1. $M$ verifies that the extended $A$ that the oracle holds is (almost) multilinear, and thus the extended $CC$ is a multinomial of constant degree.
2. Assuming that $CC$ is a multinomial of constant degree, $M$ can now use an [28] type protocol to verify, with high probability, that for all $c$,, $CC(A, c) = 0$. In this part, the oracle provides $M$ with the value of the multilinear extension of $A$ on a small number of arguments which $M$ chooses.

We show how to do part 1 in subsection 4.3, and part 2 in subsection 4.2

## 4.2 Verifying Simultaneously that All Clauses Evaluate to Zero

We now show how to verify with high probability that for all $c$, $CC(A, c) = 0$. Clearly, this would imply that $A$ satisfies $f$. All arithmetic in this and the next section is done over the finite field $\mathcal{F}$, where $\mathcal{F}$ is some "large enough" finite field ($|\mathcal{F}| > 100m$ suffices).

We first show how to verify that the arithmetic expression of a single clause evaluates to 0. We then show how all clauses can be tested simultaneously.

### 4.2.1 The Sum-Check Protocol

Given a multinomial $\varphi$ of constant degree $d$ (in each of its variables) and a constant $c$, we want to check that

$$\sum_{\vec{y} \in \{0,1\}^m} \varphi(\vec{y}) = c \qquad (*)$$

This can be done inefficiently by evaluating $\varphi$ on the $2^m$ different points $\vec{y} \in \{0,1\}^m$. However, [28] developed a procedure for checking $(*)$ which requires the evaluation of $\varphi$ only on a single random point $\vec{y}$ which belongs to the extended field $\mathcal{F}^m$. [28] (and [33]) used this protocol because the number of terms in the summation was too large for the performing a direct computation. In contrast, we use the protocol because $M$ does not have an explicit expression for $\varphi$ ($M$ does not know $A$), and the amount of communication required to obtain such an expression is prohibitingly large. In [5], this protocol was used for both the above reasons.

We describe an adaptation of the [28] protocol to our purpose. Since $\varphi$ is a multinomial of degree $d$, the function

$$g(y_1) = \sum_{y_2,\ldots,y_m \in \{0,1\}} \varphi(\vec{y})$$

is a polynomial of degree $d$.

To prove $(*)$, the oracle sends $M$ a polynomial $g'$ of degree $d$, and claims that $g' = g$. If $(*)$ is false, then there are two possibilities:

1. $g'(0) + g'(1) \neq c$ (which $M$ can determine easily),

2. $g' \neq g$, and since both $g$ and $g'$ are polynomials of degree $d$, they agree on at most $d$ values. To check this condition, $M$ picks at random a $k \in \mathcal{F}$ and checks whether $g'(k) = \sum_{y_2,\ldots,y_m \in \{0,1\}, y_1 = k} \varphi(\vec{y})$. Note, that if $g' \neq g$ then with probability at least $1 - \frac{d}{|F|}$ (taken over the choices of $k$), $g'(k) \neq g(k) = \sum_{y_2,\ldots,y_m \in \{0,1\}, y_1 = k} \varphi(\vec{y})$. Note also, that this condition is of the same form as (*), except for only $m-1$ variables and will be checked recursively.

$M$ continues this procedure for $m-1$ more rounds. In round $i$, $M$ picks a random value for $y_{i-1}$, sends to the oracle the values of all variables instantiated so far (which are $y_1 \ldots y_{i-1}$), and requests a $d$ degree polynomial in the variable $y_i$ to test the polynomial that results from the previous instantiations. When all variables are finally instantiated, $M$ checks that for $\vec{y}$ that was formed, $\phi(\vec{y})$ indeed achieves the value claimed. For $|F| > \frac{md}{\epsilon}$, the probability of falsely accepting is bounded by the – the sum over $m$ rounds of possible error probability $d/|F|$ in every round, which is at most $\epsilon$. Note that, the number of random bits used by $M$ and answer bits received by $M$ is $O(m \log |\mathcal{F}|) = O(m \log m)$ for constant $d$ and $\epsilon$. An important observation is that the test requires that $M$ checks the value of $\varphi$ only at one single point when $\vec{y}$ is fully instantiated.

The above protocol can be used then to check that a single clause $CC(A,c) = 0$ (i.e., $\sum_{\vec{y} \in \{0,1\}^m} \varphi(\vec{y}) = CC(A,c) = \sum_{\vec{v}_1,\vec{v}_2,\vec{v}_3 \in \{0,1\}^{3m}} \prod_{j=1}^{3} \chi_{j,c}(\vec{v}_j) \cdot (s_{j,c} - A(\vec{v}_j))$ where $\vec{y} = \vec{v}_1\vec{v}_2\vec{v}_3$). Observe that the last step of the sum-check protocol requires $M$ to compute $\prod_{j=1}^{3} \chi_{j,c}(\vec{v}_j) \cdot (s_{j,c} - A(\vec{v}_j))$, where $\vec{v}_1$, $\vec{v}_2$, and $\vec{v}_3$ have been are instantiated, but $M$ cannot compute this on its own as $M$ does not have a description of the function $A$. Instead, $M$ requests the values of $A(\vec{v}_1)$, $A(\vec{v}_2)$ and $A(\vec{v}_3)$ from the oracle and uses values received for the computation of $\prod_{j=1}^{3} \chi_{j,c}(\vec{v}_j) \cdot (s_{j,c} - A(\vec{v}_j))$. We remark that the reliability of the sum-check test does not change too much even if a small constant fraction of the values of $A$ is incorrect.

### 4.2.2 Constructing a Single Expression From Many Clauses

The sum-check protocol can be used to test the value of a single expression. However, in our case, $M$ wants to test the value of a polynomial number of expressions. Checking each one of them separately requires too much communication. Thus we describe a way of checking them simultaneously. This is done by constructing one single expression $E(A)$ whose value simultaneously reflects the values of all the $2^m$ expressions $CC(A,c)$. This single expression is still of degree 6 in each of its variables and so the sum-check protocol can be used to check its value. (i.e, $\sum_{\vec{y}\in\{0,1\}^m} = E(A)$).

A naive attempt is to construct $E(A)$ as $\sum_c CC(A,c)$. If for each clause $c$, $CC(A,c)$ evaluates to 0, we know that also $E(A)$ would evaluate to 0. Unfortunately, if there exists some $c$ such that $CC(A,c) \neq 0$ over $\mathcal{F}$, it still may be true that $E(A) = 0$ over $\mathcal{F}$.

In a more sophisticated approach $M$ chooses independently and uniformly weights $w_c \in \mathcal{F}$ for each clause $c$, sends all $w_c$ values to the oracle, and requests the oracle to convince it that $E(A) = \sum_c CC(A,c) \cdot w_c = 0$ over $\mathcal{F}$. If $\forall c$, $CC(A,c) = 0$, then also $E(A) = 0$. If $\exists c$ such that $CC(A,C) \neq 0$, then $prob_{w_c}(E(A) = 0) = 1/|\mathcal{F}|$. This randomized reduction would be good enough for our purpose, if not for one problem: $M$ uses polynomially many random bits in generating the independent $w_c$, defeating our attempt to keep their number small ($O(\log n \log\log n)$).

We now describe a solution that assigns the weights pseudo-randomly, and uses only $m \log |\mathcal{F}|$ truly random bits.

**Lemma 7** *For $0 \le i \le m-1$, let $c_i$ denote the $i^{th}$ bit of the binary representation of $c$, where $c$ is the index of a clause. Let $R = (r_0, r_1, ..., r_{m-1})$, where each $r_i$ is an element chosen independently at random from $\mathcal{F}$. Consider the family of expressions:*

$$E_R(A) = \sum_{c\in\{0,1\}^m} CC(A,c) \cdot \prod_{\{i|c_i=1\}} r_i$$

*over all possible choices of $R$.*

1. *If $\forall c$:$CC(A,c) = 0$, then $\forall R : E_R(A) = 0$.*
2. *If $\exists c$:$CC(A,c) \neq 0$, then with probability at least $1 - m/|\mathcal{F}|$ (over the choices of $R$): $E_R(A) \neq 0$.*

**Proof:** View the $2^m$ expressions $CC(A,c)$, $c \in \{0,1\}^m$ as the coefficients of an $m$-variable multilinear function over $\mathcal{F}$ where the variables are $r_0, ..., r_{m-1}$. Then, the expression $E_R(A)$ corresponds to evaluating this multilinear function on a point $R \in \mathcal{F}^m$. It is well known that unless all coefficients are 0, a multilinear function is nonzero on at least a $(1 - 1/|\mathcal{F}|)^m$-fraction of its points (easy proof by induction on $m$). The proof of the lemma follows.

$\square$

**Remark:** Alternative constructions are described in a former conference proceeding version of our paper, as well as in [5] and [4]. The construction above can be modified so that it uses only $O(m)$ random bits, by choosing $k = \Theta(\log m)$, and viewing the $2^m$ expressions $CC(A,c)$ as the coefficients of an $m/k$-variable function of degree $2^k$ over $\mathcal{F}$. Observe that in effect we use a linear error detecting code of large Hamming distance over the alphabet $\mathcal{F}$. The message (sequence of values of the individual expressions) of length $2^m$ is encoded by a code word of length $2^{|R|}$, and the verifier checks whether a random character from this code word is zero. Other codes can be used instead of our construction.

Let us summarize the procedure for verifying simultaneously that for all clauses $c$, $CC(A,c) = 0$. Machine $M$ chooses at random $R \in \mathcal{F}^m$, and applies the check-sum procedure to expression $E_R(A)$. The check-sum procedure can be applied as expression $E_R(A)$ (for fixed $R$ and summing over $c$) is a multinomial of degree 6 in each variable with $3m$ variables $\vec{v_1}\vec{v_2}\vec{v_3}$ in $\mathcal{F}$, and at the final stage of the checksum procedure applied to $E_R(A)$, machine $M$ queries the oracle for the value of the extended $A$ at the three independently distributed locations chosen during the checksum protocol.

## 4.3 The Multilinearity Test

As part of the proof system for NEXPTIME, [5] propose a procedure for testing that a function is multilinear. For a function $f \colon \mathcal{F}^m \to \mathcal{F}$ this test succeeds always when $f$ is multilinear and fails with high probability if $f$ is not close to a multilinear function (as defined below). Unfortunately, the test of [5] is not efficient enough for our purpose.

In order to prove Theorem 3 we devise a new multi-linearity test which requires only $O(m \log |\mathcal{F}|)$ random and answer bits. We do this by first simplifying the analysis of the test in [5] and obtaining tighter bounds, and second using pseudo-random (rather than random) sampling for choosing the points of $f$ to be queried in the test. The structure of the resulting test is quite similar to the multi-linearity test of [5].

**Remark:** The test presented below is similar to the test that appeared in a preliminary version of this paper [17], but the analysis of this test is improved. In another (unpublished) version of this paper, we proposed a test with a somewhat different structure, with the same complexity $O(m \log |\mathcal{F}|)$. Generalizations of this other test and simplification of its analysis are presented in [20].

### 4.3.1 Developing The Mathematical Background

First we introduce some notation; the *distance* between two functions $f_1, f_2 : \mathcal{F}^m \to \mathcal{F}$,

$$\Delta(f_1, f_2) = \frac{|\{\vec{y} | f_1(\vec{y}) \neq f_2(\vec{y})\}|}{|\{\text{all } \vec{y}\}|}$$

i.e., the fraction of $\mathcal{F}^m$ on which $f_1$ and $f_2$ disagree. Let $ML$ be the set of multilinear functions defined on $\mathcal{F}^m$. We define the *minimal distance* between a function $f : \mathcal{F}^m \to \mathcal{F}$ and a multilinear function

$$\Delta_{ML}(f) = \min_{f' \in ML} \Delta(f, f').$$

We call a set of $|\mathcal{F}|$ points $\left\{y_1, \ldots, y_{|\mathcal{F}|}\right\} \subseteq \mathcal{F}^m$ an *aligned line* in direction $i$ if they differ only in the $i^{th}$ coordinate. A set of three distinct points $a$, $b$ and $c$ of an aligned line is called a *triple* in direction $i$.

It is easy to see that a function is multilinear if and only if it is linear over all possible triples.

**Definition 8** *Let $f : \mathcal{F}^m \to \mathcal{F}$ be an arbitrary function. We say that a triple $\{a, b, c\}$ (in direction $i$) is $f$-linear iff there exists a function $g$ which is linear in the $i$th variable and which satisfies $g(a) = f(a)$, $g(b) = f(b)$ and $g(c) = f(c)$.*

Let $T = m \begin{pmatrix} |\mathcal{F}| \\ 3 \end{pmatrix} |\mathcal{F}|^{m-1}$ denote the number of all triples. For $f : \mathcal{F}^m \to \mathcal{F}$ define

$$\tau(f) = \frac{|\,\{\text{non } f\text{-linear triples}\}\,|}{T}.$$

Our test checks random triples for $f$-linearity. The following theorem establishes the necessary connection between $f$-linearity of triples and multilinearity of $f$. It shows that if a function $f$ is "far enough" from multilinear (that is, any multilinear function disagrees with $f$ on a constant fraction of the points in $\mathcal{F}^m$), then a substantial fraction ($\Omega(1/m)$) of the triples are not $f$-linear. We remark that there is interplay between various constants (specifying $|\mathcal{F}|$, $\Delta_{ML}(f)$, and $\tau(f)$) in the statement of the theorem and in its proof, and our choice of particular values is somewhat arbitrary.

**Theorem 8** *Let $|\mathcal{F}| \geq 20m$, and let $f : \mathcal{F}^m \to \mathcal{F}$ be an arbitrary function, such that $\Delta_{ML}(f) \geq \frac{1}{10}$. Then $\tau(f) > \frac{1}{10m}$.*

**Proof:** We first give a high level overview of our proof. It is composed of two main parts.

The first part (Lemma 9 and Corollary 10 deals with the case that there exists a multilinear function $L$ such that $\Delta_{ML} \leq \frac{9}{10}$, or actually more generally an $L$ that agrees with $f$ on a constant fraction of $\mathcal{F}^m$. For this case we show that $\Omega(1/m)$ of the triples are such that: at least one of their points falls in the region where $L$ and $f$ agree, and at least one of their points falls in the region where $L$ and $f$ disagree (so called *two colored* points). We then show that the vast majority of two colored triples are not $f$-linear. We remark that for the special case that $\Delta_{ML}(f) \leq 1/2$, the bounds that we obtain are best possible (upto low order terms).

The second part of the proof (Lemma 11 ). deals with the case that $\Delta_{ML}(f) > 9/10$. We need to show that $\Omega(1/m)$ of the triples are not $f$-linear, a simple thing to show if $f$ were random. But can it be the case that $f$ is composed of many segments of different multilinear functions, none of which covers a constant fraction of $f$, such that most aligned lines agree with at least one of these multilinear functions? Lemma 12 shows that this is not possible. This is shown by induction on the dimension as follows. Let's look at direction $x_1$. If most aligned lines in the $x_1$ direction are (approximately) $f$-linear, and most hyperplanes perpendicular to the $x_1$ direction are (approximately) $f$-multilinear, then we can find a single multilinear function that agrees with $f$ on a constant fraction of $\mathcal{F}^m$. Hence if $f$ is "far" from multilinear, then either many aligned lines in direction $x_1$ are "far" from linear, in which case a random triple in direction $x_1$ would not be linear, or many hyperplanes perpendicular to direction $x_1$ are "far" from multilinear, in which case we can proceed by induction on these hyperplanes (having only $m-1$ dimensions).

The rest of this subsection is devoted to a a detailed proof of Theorem 8. (Readers that are satisfied with the high level overview of the proof may go directly to Section 4.3.2.)

**Lemma 9** *Let $f : \mathcal{F}^m \rightarrow \mathcal{F}$ be an arbitrary function. Then*

$$\tau(f) \geq \frac{3(1-\Delta_{ML}(f))\Delta_{ML}(f)}{m} - \frac{3}{|\mathcal{F}|}$$

.

**Proof:** Let $L$ be a multilinear function such that $\Delta(f, L) = \Delta_{ML}(f)$. Let $G$ be the indicator function of $f - L$ (i.e. $G$ is 0 where $f$ agrees with $L$ and 1 otherwise). We say that a triple $\{a, b, c\}$ is *one colored* if $G(a) = G(b) = G(c)$ and *two colored* otherwise. In order to prove Lemma 9, we show that the number of two colored triples is "large" (at least a fraction of $3(1-\Delta_{ML}(f))\Delta_{ML}(f)/m$ of all triples), whereas only a "small" number of two colored triples are $f$-linear (at most a fraction of $3/|F|$ of all triples).

*Lower bounding the number of two colored triples:*

The event $\mathcal{E}$ that a triple is two colored is the disjoint union of the following three mutually exclusive events: $\mathcal{E}_1$: $G(a) = 0$ and $G(b) = 1$; $\mathcal{E}_2$: $G(b) = 0$ and $G(c) = 1$; $\mathcal{E}_3$: $G(c) = 0$ and $G(a) = 1$. By symmetry, $Prob(\mathcal{E}) = 3Prob(\mathcal{E}_1)$. Thus, it suffices to bound the value of $Prob(\mathcal{E}_1)$.

The points $a$ and $b$ are each chosen at random with uniform probability from the set $\mathcal{F}^m$. Had they been chosen independently, then $Prob(\mathcal{E}_1)$ would have been exactly $(1 - \Delta_{ML}(f))\Delta_{ML}(f)$. However, $a$ and $b$ are not independent, as they are chosen to agree on all their coordinates but one. To quantify the effect of this dependency, we present a two-stage processes for choosing $a$ and $b$, which is equivalent to the actual process used.

1. Select two points $p, q$ independently at random from $\mathcal{F}^m$.
2. Select an index $j$ at random, between 1 and $m$. Let $a$ and $b$ both agree with $p$ on their first $j-1$ coordinates, both agree with $q$ on their last $m-j$ coordinates, and for the $j^{th}$ coordinate, point $a$ agrees with $p$, whereas point $b$ agrees with $q$.

Clearly, if $G(p) = 0$ and $G(q) = 1$, then there exists a choice of $j$ such that $\mathcal{E}_1$ holds. It follows that $prob(\mathcal{E}_1) \geq \frac{(1-\Delta_{ML}(f))\Delta_{ML}(f)}{m}$.

*Upper bounding the number of $f$-linear two colored triples:*

Two colored triples can be of two types: (1) exactly one of the points in the triple is colored 1 ; (2) exactly one of the points in the triple is colored 0.

Type (1) means that on two points of the triple, $f$ and $L$ agree, and on one point they disagree. As $L$ is a multilinear function and thus in particular linear in the $i$th direction, and two *different* linear functions agree on at most one point of a triple (otherwise they will be the same function), it follows that type (1) triple cannot be $f$-linear.

Thus, it suffices to upper bound the number of $f$-linear type (2) triples. In order to do this, we again use the fact than any two different linear functions agree on at most one point. Consider an arbitrary triple $\{a, b, c\}$ that is two colored and $f$-linear. W.l.o.g., assume that $G(a) = G(b) = 1$ and $G(c) = 0$, and that $g$ is the unique linear function in the $i$th direction such that $f(a) = g(a)$ and $f(b) = g(b)$ (and $f(c) = g(c)$, by our assumption that $\{a, b, c\}$ is $f$-linear). Then, for any $c' \neq c$ which satisfies $G(c') = 0$, the triple $\{a, b, c'\}$ cannot be $f$-linear. This is argued as follows: suppose to the contrary that the triple $\{a, b, c'\}$ was $f$-linear. Then $f(c') = g(c')$. It follows that $g$ agrees with $L$ on two points ($c$ and $c'$), and hence must agree everywhere, contradicting the assumption that $G(a) = 1$. By how much have we restricted the number of $f$-linear triples? There are only $\begin{pmatrix} |\mathcal{F}| - 1 \\ 2 \end{pmatrix}$ possible choices of $a$ and $b$ with $G(a) = G(b) = 1$ along any aligned line which contains a point $c$ satisfying

$G(c) = 0$, whereas there are $\begin{pmatrix} |\mathcal{F}| \\ 3 \end{pmatrix}$ triples along any aligned line. It follows that of all the triples, only a fraction of at most $3/|F|$ can be two colored and $f$-linear.

Combining the lower bound and the upper bound, the proof of Lemma 9 follows. □

By substituting the appropriate values in Lemma 9 we get the following corollary.

**Corollary 10** *Let $|\mathcal{F}| \geq 20m$, and let $f : \mathcal{F}^m \to \mathcal{F}$ be an arbitrary function, $\frac{1}{10} \leq \Delta_{ML}(f) \leq \frac{9}{10}$. Then $\tau(f) > \frac{1}{9m}$.*

We are now ready to address the case that $\Delta_{ML}(f)$ is large (i.e in particular, larger than $\frac{9}{10}$).

**Lemma 11** *Let $|F| \geq 20m$ and let $f : \mathcal{F}^m \to \mathcal{F}$ be an arbitrary function satisfying $\Delta_{ML}(f) > 9/10$. Then $\tau(f) > (1 - 1/|\mathcal{F}|)^{(m-1)} \frac{1}{9m}$.*

**Proof:** The proof is by induction on $m$. For the base case of the induction ($m = 1$), we need to prove that $\Delta_{ML}(f) > 9/10$ implies that $\tau(f) > 1/9$. Assume to the contrary that $\tau(f) \leq \frac{1}{9}$. Hence the probability that a random triple is $f$-linear is at least $8/9$. An averaging argument shows that there exist a pair of points $a, b \in \mathcal{F}$, such that at least $8/9$ of the remaining points are co-linear with $a$ and $b$. Thus $\Delta_{ML}(f) < 1/9$, contradicting the assumption that $\Delta_{ML}(f) > 9/10$.

For the induction step, we prove the statement for $m$ by fixing the first coordinate, and using the induction hypothesis on the other $m - 1$ coordinates. Actually, since the statement of Lemma 11 does not make a strong enough induction hypothesis, we will also rely on Corollary 10 to make the induction step go through.

When fixing the value of the first coordinate $x_1$ to $a \in \mathcal{F}$ we get a subspace $\mathcal{F}^m_{x_1=a}$. Let us denote by $f_a$ the restriction of $f$ to $\mathcal{F}^m_{x_1=a}$.

Let $T_a$ be the set of all triples along the $x_1$ coordinate for which one point is in $\mathcal{F}^m_{x_1=a}$ (hence the other two points are not in $\mathcal{F}^m_{x_1=a}$), and let $T'_a \subseteq T_a$ be the set of those, that are not $f$-linear. Let $\tau_a = |T'_a|/|T_a|$.

We proceed to (1) show that for at most one value of $a$ both[2] $\delta_{ML}(f_a) < \frac{1}{10}$ and $\tau_a < \frac{1}{3}$; and (2) for all other values $b$ of $x_1$, bound from below the number of triples that are not $f$-linear. We use these facts to show that the induction step goes through.

**Lemma 12** *If along the first coordinate there are two distinct values $a, b \in \mathcal{F}$ that satisfy*

[2]i.e only with low probability, most aligned lines in direction $x_1$ are (approximately) $f$-multilinear, and most hyperplanes perpendicular to $x_1$ are (approximately) $f$-multilinear

*1.* $\Delta_{ML}(f_a) < 1/10$

*2.* $\Delta_{ML}(f_b) < 1/10$

*3.* $\tau_a < 1/3$

*4.* $\tau_b < 1/3$

*then* $\Delta_{ML}(f) < 9/10$.

**Proof:** Let $L_a$, $L_b$ be multilinear functions on $\mathcal{F}_{x_1=a}$ and $\mathcal{F}_{x_1=b}$ respectively such that $\Delta(f_a, L_a) < 1/10$, $\Delta(f_b, L_b) < 1/10$. Let

$$L(x_1, \ldots, x_m) = L_a(x_2, \ldots, x_m) + \frac{x_1 - a}{b - a}(L_b(x_2, \ldots, x_m) - L_a(x_2, \ldots, x_m))$$

be the unique multilinear extention of $L_a$ and $L_b$ on $\mathcal{F}^m$. We prove that $\Delta(f, L) < 9/10$.

Consider two random points $r = (r_1, r_2, \ldots, r_m)$ and $r' = (r'_1, r_2, \ldots, r_m)$ in $\mathcal{F}^m$, such that both $r$ and $r'$ lie along the same aligned line in the direction of $x_1$. We will show that $Pr[f(r) = L(r)] > 1/10$. Note that if $r \in \mathcal{F}_{x_1=a} \cup \mathcal{F}_{x_1=b}$, then by conditions (1) and (2) above, $r$ agrees with $L$ with probability at least 9/10. Hence it remains to consider the case that $r_1 \notin \{a, b\}$.

Let $r^a = (a, r_2, \ldots, r_m)$ and $r^b = (b, r_2, \ldots, r_m)$ denote the points (on the line joining $r$ and $r'$) which belong to $\mathcal{F}_{x_1=a}$ and $\mathcal{F}_{x_1=b}$ respectively. For simplicity, we assume that $r'_1 \notin \{a, b\}$, without significantly affecting our analysis. By condition (3) above, the probability that the triple $\{r^a, r, r'\}$ is $f$-linear is at least 2/3. By condition (4) above, the probability that the triple $\{r^b, r, r'\}$ is $f$-linear is at least 2/3. When both the above triples are $f$-linear, and when the events $f_a(r^a) = L_a(r^a)$ and $f_b(r^b) = L_b(r^b)$ also hold (each occurring with probability at least 9/10), then $f(r) = L(r)$. Hence $Pr[f(r) = L(r)] \geq 1 - 1/3 - 1/3 - 1/10 - 1/10 > 1/10$. $\square$

We are now ready to obtain a count of the number of triples that are not $f$-linear. Since Lemma 11 assumes that $\Delta_{ML}(f) > 9/10$, it follows from Lemma 12 that for at most one point $a \in \mathcal{F}$, both $\Delta_{ML}(f_a) < 1/10$ and $\tau_a < 1/3$ hold. For any other $b \in \mathcal{F}$, we distinguish between three possibilities (in what follows, recall that $T$ specifies the number of all triples, and observe that $\frac{(m-1)T}{m|\mathcal{F}|}$ is the number of triples that have a particular fixed value $b$ along their first coordinate):

Case 1: $1/10 \leq \Delta_{ML}(f_b) \leq 9/10$. There are at least $\frac{(m-1)T}{m|\mathcal{F}|}\frac{1}{9(m-1)}$ non-$f$-linear triples in $f_b$ by Corollary 10.

Case 2: $\Delta_{ML}(f_b) > 9/10$. There are at least $\frac{(m-1)T}{m|\mathcal{F}|}(1-\frac{1}{\mathcal{F}})^{m-2}\frac{1}{9(m-1)}$ non-$f$-linear triples in $f_b$ by the induction hypothesis.

Case 3: $\tau_b \geq 1/3$. There are at least $\tau_b T_b \geq \frac{1}{3}\frac{T}{m}\frac{3}{|\mathcal{F}|}$ triples that pass through $\mathcal{F}_{x_1=b}$ in direction $x_1$ and are not $f$-linear. Amortizing over all possible choices of $b$ with $\tau_b \geq 1/3$, we may lose an additional factor of 3 because a triple might count in three different $T_b's$. Hence the amortized contribution of $b$ to the number of non-$f$-linear triples is $T/3m|\mathcal{F}|$.

Dividing the bounds in each of the above three possibilities by $T$, and summing over the $|\mathcal{F}|-1$ possible values of $b$, we obtain:

$$\tau(f) \geq (|\mathcal{F}|-1)\min[\frac{(m-1)}{m|\mathcal{F}|}\frac{1}{9(m-1)}\ ;\ \frac{(m-1)}{m|\mathcal{F}|}(1-\frac{1}{\mathcal{F}})^{m-2}\frac{1}{9(m-1)}\ ;\ \frac{1}{3m|\mathcal{F}|}]$$

$$\geq (1-1/|\mathcal{F}|)^{(m-1)}\frac{1}{9m}$$

□

In order to complete the proof of Theorem 8 we simplify the bound obtained in Lemma 11, using the assumption that $|\mathcal{F}| \geq 20m$.

$$\tau(f) > (1-1/|\mathcal{F}|)^{(m-1)}\frac{1}{9m} > \frac{1}{10m}$$

□

An interesting open question arises from our analysis of the multilinearity test: We show (lemma 9) that if a function is relatively close to multilinear ($\Delta_{ML}(f) = 1/2$), then a fair fraction of the triples (about $3/4m$) are not $f$-linear. However, for functions that are further from multilinear ($\Delta_{ML}(f) > 9/10$), we could show only a smaller fraction of the triples (around $1/9m$) that are not $f$-linear (Lemma 11). Can our analysis be significantly improved in the latter case, or does this degradation reflect a true property of multivariate functions? (A more careful choice of the parameters in our analysis somewhat improves the bound on $\tau(f)$ for the case $\Delta_{ML}(f) > 9/10$. However, this improvement is insignificant with respect to the goals of this paper, and with respect to the open question stated above.)

### 4.3.2 The Test Itself

We devise a multilinearity test with the following properties:

1. If $f$ is multilinear, the test always accepts.

2. If $\Delta_{ML}(f) \geq 0.1$, the test rejects with probability at least $1/2$.

Since we may assume that $|F| \geq 20m$, Theorem 8 naturally suggests the following test:

**MULTILINEARITY TEST (random sampling version)**

1. Randomly choose $10m$ triples. Ask the oracle for the value of $f$ on each point of every triple.
2. Accept iff all triples are $f$-linear.

The above test is wasteful in the number of random bits that the verifier uses. Generating each of the $O(m)$ test triples requires $O(m \log |F|)$ random bits, whereas each of the $O(m)$ replies of the oracle requires only $\log |\mathcal{F}|$ bits. Thus instead of generating each of the sample triples independently, we want an easy to compute sampling procedure which uses only $O(m \log |F|)$ random bits, generates $O(m)$ sample triples, and "hits" a non-$f$-linear triple with probability at least $1/2$.

Problems of this type can be handled by the method of *two-point sampling* [12]. The basic idea behind two point sampling techniques is that pairwise independent sample points (or sample triples, in our case) share many of the properties of mutually independent sample points especially with respect to hitting sets of small density with constant probability, but require much less random bits to generate. One possible implementation is as follows.

Identify the set of sample points (all triples) $\mathcal{T}$ with a field $\mathcal{K}$. Consider the family

$$HASH = \{g_{a,b} | a, b \in \mathcal{K}\},$$

where $g_{a,b}$ is defined as the function $ax + b$ computed over the field $\mathcal{K}$.

Pick any set $K \subseteq \mathcal{K}$ such that $|K| = 20m$. The set system $\{K_{a,b} \mid a, b \in \mathcal{K}\}$ is defined by $K_{a,b} = \bigcup_{x \in K} g_{a,b}(x)$ for $a, b \in \mathcal{K}$. The desired properties of this system follow from the following properties of the hash functions:

1. (Uniformity) For any pair $x, y \in \mathcal{K}$ the probability for a randomly chosen hash function $g_{a,b}$ that $g_{a,b}(x) = y$ is exactly $1/|\mathcal{K}|$.
2. (Pairwise Independence) For any $x_1 \neq x_2 \in \mathcal{K}$ and for any $y_1, y_2 \in \mathcal{K}$ the probability that for a randomly chosen hash function $g_{a,b}$ the equations $g_{a,b}(x_1) = y_1$ and $g_{a,b}(x_2) = y_2$ hold is $1/|\mathcal{K}|^2$.

Consider a subset $H \subseteq \mathcal{K}$, and denote $|H|/|\mathcal{K}|$ by $p$. (In our case, $H$ corresponds to the set of triples that are not $f$-linear, and $p \geq 1/10m$, by Theorem 8.) For

$x \in \mathcal{K}$ let $V_x$ be the indicator function of the event that a random $g_{a,b}$ maps $x$ into $H$. $E(V_x) = p$, because of the uniformity property. Consider

$$V = \sum_{x \in K} V_x.$$

The probability that a random $K_{a,b}$ intersects with $H$ is exactly $Prob(V > 0)$. We have $E(V) = 20pm$ and $\sigma^2(V) = 20p(1-p)m$ (because of the pairwise independence of the variables $V_x$). From the Chebyshev inequality:

$$Prob(V \leq 0) \leq \frac{\sigma^2(V)}{(E(V))^2} = \frac{1-p}{20pm} < \frac{1}{2}.$$

For other suggested implementations of two-point sampling, the reader is referred to [12].

We are now ready to present our efficient multilinearity test.

**PAIR-WISE INDEPENDENT MULTILINEARITY TEST (pair-wise independent sampling version)**

1. Randomly choose two initialization points for the two point sampling procedure. Deterministically generate $20m$ pairwise independent sample triples. Ask the oracle for the value of $f$ on each point of every sample triple.
2. Accept iff all of the sample triples are $f$-linear.

**Lemma 13** *Let $|F| \geq 20m$ and let $f : \mathcal{F}^m \rightarrow \mathcal{F}$ be an arbitrary function.*

1. *If $f$ is multilinear, the pair-wise independent multilinearity test always accepts.*
2. *If $\Delta_{ML}(f) \geq 0.1$, the pair-wise independent multi-linearity test rejects with probability at least 1/2.*

The proof follows from theorem 8, and properties of pair-wise independent sampling as discussed above.

## 4.4 Putting the Pieces Together

We are now ready to prove Theorem 3.

**Proof:** Let $f$ be a 3CNF formula of length $n$. Set $m$ as the smallest integer satisfying $2^m \geq n$. Let $\mathcal{F}$ be the smallest finite field of order $|\mathcal{F}| > 100m$. (We remark that since $|\mathcal{F}|$ is relatively small - $O(\log n)$, it can be found in deterministic polynomial time.) All computations in the following protocol are made over $\mathcal{F}$.

1. Arithmetize $f$ and obtain multilinear representations as described in Subsection 4.1. In particular, let the (truthful) oracle hold the multilinear representation of a satisfying assignment $A$.

2. Perform on $A$ the multilinearity test (pair-wise independent sampling version) as described in Subsection 4.3.2. I.e query the oracle for the value of the extended $A$ on points in the triples secelcted by the multilinearity test. If all tested triples are $f$-linear, continue. Else, reject.

3. Select a random $R$ and construct the expression $E_R(A)$, as described in Subsection 4.2.2. Prefix this $R$ to any message sent in Step 4 of the protocol. (In Step 4 the oracle has to know which expression $M$ has in mind.)

4. Perform the sum-check protocol (as described in Subsection 4.2.1) on the expression $E_R(A)$. Reject if at any stage the oracle's replies violate the consistency check described as possibility 1 in Subsection 4.2.1. As decribed in the sum-check protocol, this step entails querys to the oracle for degree-6 polynomials corrsponding to various expressions.

5. When Step 4 ends, $M$ is left with three random arguments for the multilinear extension of $A$. One by one, request the values of $A$ on these arguments from the oracle. Plug these values in their respective locations in the instantiated $E_R(A)$, and compute its value. Accept iff this value agrees with the value computed from the last polynomial sent by the oracle.

It is easy to see that if $f$ is satisfiable, then $M$ accepts the proof of the truthful oracle. If $f$ is not satisfiable, then for any oracle, we distinguish between two cases. Denote by $g$ the function obtained by enumerating all the oracle's answers when $M$ requests values of the multilinear extention of $A$ (these questions correspond to Steps 2 and 5 in the protocol).

1. $\Delta_{ML}(g) \geq 1/10$. In this case, by lemma 13, Step 2 rejects with probability at least 1/2 (by Theorem 8, and the discussion in Subsection 4.3.2).

2. $\Delta_{ML}(g) < 1/10$. We show that Steps 3-5 reject with probability at least 1/2.

   Assume, to the contrary, that oracle $O$ is using function $g$, and the probability that $M^O$ accepts Steps 3-5 above is at least 1/2. Let $h$ be the multilinear function that satisfies $\Delta(g,h) < 1/10$. We construct a new oracle $O'$ that uses function $h$, in its responses in steps 2 and 5, and otherwise in step 4 answers in a manner identical to $O$. Namely, on any query of $M$ that does not request explicitly the value of the multilinear extention of $A$ at a point, the answer of $O'$ is identical to the answer of $O$, but in order to answer queries on the value of the multilinear extension of $A$ at a point, $O'$ use the multilinear function $h$, rather than the function $g$ that $O$ uses.

The oracle is queried on exactly three point of the multilinear extention of $A$ at step 5, and each of these points is chosen uniformly at random. Since $\Delta(g,h) < 1/10$, the probability that $O'$ is asked for the value of a point on which $g$ and $h$ disagree is at most $3/10$. Hence the probability that $M^{O'}$ accepts Steps 3-5 is at least $1/2 - 3/10 = 2/10$.

Now we compute an upper bound on the probability that $M^{O'}$ accepts Steps 3-5. We first condition over the random choice of $R$. For $f$ that is not satisfiable, the probability that $E_R(h)$ is identically 0 is at most $m/|\mathcal{F}| < 1/100$ (by lemma 7). In this case we may assume that $M^{O'}$ accepts. For the case that $E_R(h)$ is not identically 0, we use the fact that $h$ is multilinear, implying that $E_R(g)$ is of degree 6. By the analysis of the sum-check protocol, the probability that $M^{O'}$ accepts Steps 4 and 5 is at most $\frac{6\cdot 3m}{|\mathcal{F}|} < 18/100$. Hence the probability that $M^{O'}$ accepts Steps 3-5 is at most $1/100 + 18/100 < 2/10$ which is in contradiction to the hypothesis that $M^O$ accepts with probability at least $1/2$.

The total number of random bits that the verifier uses is (ignoring low order terms): $6m\log|\mathcal{F}|$ for Step 2, $m\log|\mathcal{F}|$ for Step 3, and $3m\log|\mathcal{F}|$ for Step 4, totaling $10m\log m + o(m\log m)$. The total number of bits that the oracle sends is: $60m\log|\mathcal{F}|$ in Step 2, $7m\log|\mathcal{F}|$ in Step 4, and $3|\mathcal{F}|$ in Step 5, totaling $67m\log m + o(m\log m)$. As $m \le \log n + 1$, this completes our proof of Theorem 3.

$\square$

## Acknowledgment

We thank László Babai, Avrim Blum, Oded Goldreich, Mike Sipser, and Avi Wigderson for many discussions on this work and its presentation. We thank Sasha Shen for clarifying observations on multilinearity tests.

## References

[1] N. Alon and R. Boppana. The monotone circuit complexity of boolean functions. In *Combinatorica 7*, pages 1–22, 1987.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 14–23, 1992.

[3] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 2–13, 1992.

[4] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proc. 23rd ACM Symp. on Theory of Computing*, 21–31, 1991.

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

[6] R. Bar-Yehuda and S. Even. A $2 - \frac{\log \log n}{2 \log n}$ performance ratio for the weighted vertex cover problem. Technical Report 260, Technion, Haifa, Jan 1983.

[7] 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.

[8] B. Berger and J. Rompel. A better performance guarantee for approximate graph coloring. *Algorithmica*, 5(4):459–466, 1990.

[9] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. In *Information and Computation* 96 (1992), 77–94.

[10] R. B. Boppana and M. M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. In *Proc. of 2nd Scand. Workshop on Algorithm Theory, Springer-Verlag Lecture Notes in Computer Science 447*, pages 13–25, July 1990.

[11] R. B. Boppana, J. Hastad, and S. Zachos. Does co-NP have short interactive proofs. In *Inform. Process. Lett., 25*, pages 127–132, 1987.

[12] B. Chor and O. Goldreich. On the power of two-point based sampling. *Journal of Complexity*, 5:96–106, 1989.

[13] A. Condon. The complexity of the max word problem. In *Proc. of 8th STACS*, 1991, 456–465.

[14] A. Condon and R. Lipton. On the complexity of space bounded interactive proofs. In *Proc. 30th IEEE Symp. on Foundations of Computer Science*, pages 462–467, 1989.

[15] U. Feige and S. Goldwasser (editors). The Weizmann Workshop on Probabilistic Proof Systems. *Technical report CS94-17*, Weizmann Institute, 1994.

[16] U. Feige, S. Goldwasser, L. Lovasz, and S. Safra , On the Complexity of Approximating the Maximum Size of a Clique. Unpublished preliminary report, circulated and dated Nov. 30, 1990.

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

[18] U. Feige and A. Shamir. Multi-oracle interactive protocols with space bounded verifiers. *JCSS*, 44:259–271, 1992.

[19] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. In *Proc. 3rd STRUCTURES*, pages 156–161, 1988.

[20] K. Friedl, Z. Hatsagi, A. Shen. Low degree tests. In *Proc. 5th ACM-SIAM Symp. on Discrete Algorithms*, 57–64, 1994.

[21] Goldreich, O., S. Micali, and A. Wigderson, "Proofs that Yield Nothing but their Validity and a Methodology for Cryptographic Protocol Design", *JACM*, Vol. 38, July 1991, pp. 691–729.

[22] M. Garey and D. Johnson. The complexity of near optimal graph coloring. *JACM*, 23:43–49, 1976.

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

[24] M. M. Halldórsson. A still better performance guarantee for approximate graph coloring. *Technical Report 90-44, Department of Computer Science, Rutgers University, NJ, 1990.*

[25] D. Johnson. The NP-completeness column: an ongoing guide. *J. of Algorithms* 13 (1992), 502–524.

[26] Richard 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.

[27] N. Linial and U. Vazirani. Graph products and chromatic numbers. In *Proc. 30th IEEE Symp. on Foundations of Computer Science*, 124–128, 1989.

[28] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *JACM*, 39 (1992), 859–868.

[29] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *JACM*, 41(5):960–981, 1994.

[30] B. Monien and E. Speckenmeyer. "Ramsey numbers and an approximation algorithm for the vertex cover problem". *Acta Informatica*, 22:115-123, 1985.

[31] A. Panconesi and D. Ranjan. Quantifiers and approximation. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 446–456, 1990.

[32] C. Papadimitriou and M Yannakakis. Optimization, approximation and complexity classes. *J. Computer and System Sci.* 43 (1991), 425–440. 1988.

[33] A. Shamir. IP=PSPACE. *JACM*, 39 (1992), 869–877.

[34] A. Wigderson. Improving the performance guarantee for approximate graph coloring. *Journal of the ACM*, 30(4):729–735, 1983.