Approximating Clique is Almost NP-complete

(Preliminary Version)

U. Feige*[1] S. Goldwasser†[2] L. Lovász[1,4] S. Safra‡[2] M. Szegedy[3]

1. Princeton University
2. M.I.T.
3. University of Chicago
4. Eötvös University

Abstract

We consider the computational complexity of approximating $\omega(G)$, the size of the largest clique in a graph G. We show that

1. If there is an approximation algorithm in $\mathbf{P}$ for $\omega(G)$ within some constant factor, then $\mathbf{NP} \subseteq DTIME(n^{O(\log\log n)})$.
2. If there is an approximation algorithm in $\tilde{\mathbf{P}}$ $(= \cup_{k>0} DTIME(2^{log^k n}))$ for $\omega(G)$ within a factor of $2^{\log^{1-\epsilon} n}$ (for some $\epsilon > 0$), then $\mathbf{NP} \subseteq \tilde{\mathbf{P}}$.

We conclude that if such approximation procedures exist, then EXPTIME = NEXPTIME and $\mathbf{N\tilde{P}} = \tilde{\mathbf{P}}$.

This work uses the theorem of Babai, Fortnow and Lund that NEXPTIME has multi-prover interactive proofs. For our purpose, we scale down [BFL90]'s protocol to the NP level, and improve its efficiency. Of independent interest is our simpler proof of correctness for the multilinearity test.

1 Introduction

Let $G = \langle V, E \rangle$ be a graph. A *clique* of size k is a subset of the vertices of size k that constitutes a complete subgraph. 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 ([Kar72]).

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 $1 < \frac{f(x)}{g(x)} < h(x)$. The best upper bound known, is that $\omega(G)$ can be approximated within a factor of $\frac{n}{\log^2 n}$ ([BH90]) in polynomial time.

1.1 Our Results

We turn our attention to hardness results concerning the approximation of $\omega(G)$.

Proving that 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 $L \in \mathbf{P}$ implies that $\mathbf{NP} \subseteq DTIME(T(n))$ for some subexponential function T, may also be regarded as evidence that L is hard. Clearly, 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 $\tilde{\mathbf{P}}$ $(= \cup_{k>0} DTIME(n^{\log^k n})$ algorithm which approximates $\omega(G)$ within a factor $2^{\log^{1-\epsilon} n}$, then $\mathbf{NP} \subseteq \tilde{\mathbf{P}}$. Standard padding arguments show 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 to the non existence of an approximation procedure even within a large factor, discouraging attempts

*Author supported in part by a Weizmann Fellowship

†Part of this work was done while the author visited Princeton university. Author supported in part by NSF grant No. CCR-8657527, DARPA grant No. DAAL03-86-K-0171, and BSF grant No. 89-00312

‡Author supported in part by a Weizmann Fellowship and NSF grant CCR-8912586

to extract even such little information about $\omega(G)$ in reasonable time.

1.2 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. Much less is known about the complexity of approximation problems. For 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 is known that approximating the chromatic number within an $\frac{n(\log\log n)^2}{\log^3 n}$ factor is in polynomial time ([Wig83], [BR90],[Hal90]), while approximating the chromatic number within any factor smaller than 2 is **NP**-hard. Approximating the chromatic number within any factor between 2 and $\frac{n}{\log^3 n}$ is 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 $2-\Omega(\frac{\log\log n}{\log n})$ [BE83] in polynomial time. No **NP**-hardness results are known for approximating vertex cover.

Papadimitriou and Yannakakis [PY88] 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 [PR90] extend [PY88]'s approach and define the class MAX Π_1. The complete problems for MAX Π_1 cannot be approximated unless $\mathbf{P}=\mathbf{NP}$. Consequently, [PR90] define subclasses of MAX Π_1 for which the approximability of the complete problems was open. Approximating $\omega(G)$ (or MAX CLIQUE, in [PR90]'s terminology) is in RMAX(2), which is the lowest class that [PR90] define.

Berman and Schnitger[BS89] show that approximating $\omega(G)$ within factor n^ϵ is complete for MAX SNP. Namely, if clique has n^ϵ approximation then all problems in this class have polynomial time constant approximation schemes. Our results imply that if $\tilde{\mathbf{P}}$ approximation algorithms exist for any of the approximation classes that [PR90] define, or for any of the RMAX(2)-hard approximation problems, then $\mathbf{NP}\subset\tilde{\mathbf{P}}$. Alon and Boppana [AB87] 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 $\frac{n}{\log^{O(1)} n}$.

1.3 Techniques from Interactive Proofs

In order to prove the above, we give a new simulation of **NP** in the domain of interactive proofs. We show that any language $L\in\mathbf{NP}$ is accepted by a multi-prover protocol[1] in which the number of random and communication bits is small — $O(\log n\times\log\log n)$.

To do so, we consider the theorem of Babai, Fortnow and Lund ([BFL90]), showing that NEXPTIME has multi-prover interactive proofs (this 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 communication bits and random bits is $O(\log T(n)\times\log\log T(n))$, while the running time of the verifier is $DTIME(T(n)^{O(1)})$. (e.g. for $L\in\mathbf{NP}$ the number of communication bits and random bits is $O(\log n\times\log\log n)$, and the verifier runs in polynomial time.) Our proof for the correctness of this protocol, in particular the multilinearity test, is also simpler than that of [BFL90].

Once we have this simulation, we use it to prove our main theorem. Given any protocol and an input, we construct a graph, whose size is exponential in the number of random bits and communication bits of the protocol. The size of the largest clique in this graph is proportional to the maximum success probability of the provers to convince the verifier to accept the input as in the language L. Due to

[1] using the probabilistic oracle machine formulation of [FRS88].

the definition of multi-prover interactive proofs, we get a gap between the acceptance probability of an input by a verifier in the case the input is in the language versus the case that it is not. Accordingly there is a gap between the size of the largest clique in the corresponding graph in the case that the input is in the language versus the case that it is not. Thus we get a reduction from the problem of approximating the acceptance probability of an input by a multi-prover interactive proof to that of approximating the size of the largest clique in a graph.

Also in the past results on interactive proofs served to shed light on complexity problems [BHZ87],[FS89], [CL89], [Con91].

1.4 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 [BFL90]'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 [BGKW88]. 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 either an *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* $] < \frac{1}{4}$.

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 ([FRS88]) 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 $Pr_r[M^O(x, r)] = 1$.*
2. *For every $x \notin L$ and for all oracles O, $Pr_r[M^O(x, r)] < \frac{1}{4}$.*

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 (For details see [BFLS91]).

Theorem 1 ([FRS88]) *L is accepted by a probabilistic oracle-machine iff L is accepted by a multi-prover interactive protocol.* □

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

Theorem 2 ([BFL90]) MIP= *NEXPTIME.*

This is a striking phenomenon; that any language that has an exponentially long proof of containment, has a proof of containment 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 communicated between M and the oracle, 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 communication bits exchanged between M and an oracle. We will drop the subscript M when it is obvious from the context.*

In the [BFL90] 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 generalize and improve the [BFL90] result and show an upper bound on the number of communication and random bits used by a probabilistic oracle machine that accepts $L \in NTIME(T(n))$ for $n < T(n) < 2^{poly(n)}$ as follows:

Theorem 6 Any language $L \in NTIME(T(n))$ (for $n \leq T(n) \leq 2^{poly(n)}$) 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 $DTIME(T(n)^{O(1)})$.[2]

In section 4 we give a proof for the the special case of this theorem when $L \in NP$ (the proof of the general result is practically identical). In the case $L \in$ **NP** the number of communication bits and random bits used by the probabilistic polynomial time bounded oracle machine M which accepts L is bounded by $O(\log n \log\log n)$.

The naive first attempt at proving the above theorem (for sub-exponential $T(n)$) would be to design (using [BFL90] protocol) a probabilistic oracle-machine M for $L \in NTIME(T(n))$ which runs in polylogarithmic in $T(n)$ time – thus automatically yielding the desired bound on the number of random bits and communication bits M receives from the oracle. This however is not possible for the simple reason that M must at least read the input string (which requires linear time). The proof will require a more efficient version of the [BFL90] protocol.

3 Multi-Prover Protocols and Approximating Clique

Theorem 3 (Main) *1. If approximating $\omega(G)$ within any constant factor is in* **P** *then for any $T(n) \geq n$, $NTIME(T(n)) \subseteq DTIME(T(n)^{O(\log\log T(n))})$.*

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

Proof: 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 $L \in NTIME(T(n)) \subset NEXPTIME$, and M be a probabilistic oracle machine which accepts L. Let x be an input to M

[2] It is worth noting that the bound we obtain on the communication bits plus random bits used by the probabilistic oracle machine, is better than the bound one would obtain on the communication bits plus random bits used by a verifier in the corresponding multi-prover interactive proof accepting L achieving the same error probability.

such that $|x| = n$, Using the protocol 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 $\omega(G_x) < \frac{2^{r_M(n)}}{4}$.
2. Run the approximation procedure B on G_x. If the answer for the approximated $\omega(G_x)$ is greater than $\frac{2^{r_M(n)+c_M(n)}}{2}$ 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 above.

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 $|r| = r(n)$, $|\langle q_1, a_1, ..., q_l, a_l \rangle| \leq c(n)$ 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 $a_i = \hat{a}_j$.*

The nodes of G_x are all the accepting transcripts. In order to construct the set of nodes of G_x we enumerate all transcripts (this takes exponential time in the length of the longest transcript i.e $2^{r(n)+c(n)}$), and then run M on each transcript to check that it is accepting. Two nodes (accepting transcripts) in G_x are connected iff they are consistent.

Lemma 4 $\max_O \Pr_r[M^O(x,r)accepts] \times 2^{r(n)} = \omega(G_x)$

Proof:

$\leq$ For any oracle O such that $\Pr_r[M^O(x,r)accepts] = p$, consider the $p \times 2^{r(n)}$ transcripts for which $M^O(x,r)$ accepts; these constitute a clique in G_x.

$\geq$ A clique of size k of G_x defines a partial function, O', from M's legal queries to the oracle's response (since in any clique, for any query there is at most one response). Let O be any extension of O', then M^O accepts for at least k random strings (a clique in G_x cannot contain two transcripts with the same random string).

□

Clearly, now if $x \in L$ then $\omega(G_x) = 2^{r(n)}$ and if $x \notin L$ then $\omega(G_x) < \frac{2^{r(n)}}{4}$.

The entire two-step procedure runs in time $T_B(2^{O(r(n)+c(n))}) + 2^{O(r(n)+c(n))}T_M(n)$ where T_M is the running time[3] of M.

By Theorem 6, the first part of our theorem is proved, since the running time of the two step procedure is bounded by $T(n)^{O(\log\log T(n))}$.

In order to prove the second part of our theorem, we construct the graph G'_x that corresponds to the protocol M'. M' simply runs $l = \log^{\frac{1-\epsilon}{\epsilon}} T(n)$ iterations of M on input x and accept x if M accepts x on all l iterations. Note that by our choice of l, the number of random and communication bits used by M' is still poly-logarithmic in $T(n)$. For $x \notin L$, the $\max_O Pr_r[M'^O(x,r)]$ has been decreased to $\frac{1}{2^l}$. Thus, the difference 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 5 *If, for some $\epsilon > 0$, approximating $\omega(G)$ within a factor $2^{\log^{1-\epsilon} n}$ is in $\tilde{\mathbf{P}}$, then:*

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

3.1 Graph Products and Protocols

An alternative proof of the second part of the theorem is to take the graph $G_x = \langle V, E \rangle$ which corresponds to running the protocol once and define the

[3]Note that since this 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 $2^{O(r(n)+c(n))}$.

following graph product (see [GJ79]) $G_x^2 = \langle V', E' \rangle$ where $V' = V \times V$ and $(\langle v_1, v_2 \rangle, \langle v_1', v_2' \rangle) \in E'$ iff $(v_1, v_1') \in E$ *and* $(v_2, v_2') \in E$. 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 the protocol M i times. As taking graph product 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.

4 Scaled Down BFL Protocol

In this section we consider a scaled down analogue of the theorem of [BFL90] showing that $NEXPTIME = \mathbf{MIP}$. Let $n \leq T(n) \leq 2^{poly(n)}$. For any $L \in NTIME(T(n))$, we give a new probabilistic oracle machine M which accepts L. Our protocol has the same general structure as that of [BFL90] but uses arithmetic over finite field instead of over the integers [4] and some other features are also modified. The new protocol provides a better bound on the number of random and communication bits between machine M and oracle O, than previously achieved by ([BFL90]). The multilinearity test presented is more economical than that of [BFL90], and has a simpler proof of correctness.

Theorem 6 *Any language $L \in NTIME(T(n))$ (for $n \leq T(n) \leq 2^{poly(n)}$) is accepted by a probabilistic oracle-machine M such that $r_M(n) + c_M(n) \leq O(\log T(n) \cdot \log\log T(n))$, and M runs in time $DTIME(T(n)^{O(1)})$.*

For $L \in \mathbf{NP}$ the bound on the number of communication bits and random bits obtained is $O(\log n \log\log n)$, and the verifier will run in polynomial time. For simplicity of notation, we give the proof for the special case that $L \in \mathbf{NP}$. The proof for the general case has the exact same outline.

Proof:(Special Case). Let $L \in \mathbf{NP}$. Let w such that $|w| = n$ be the input to M. By Cook's theorem, we can reduce the problem of membership of w in L to satisfiability of 3CNF formula f with n^c variables and n^c clauses. Set the integer m such that $2^m \geq n^c$.

Let $f(x_1, ..., x_{2^m})$ be a given 3CNF formula with 2^m variables. Let $\vec{c}$ denote a clause number and $\vec{v}$ denote a variable number. (To prepare for the later multilinear representation we represent both the clauses and the variables as strings in $\{0,1\}^m$).

For $(j = 1, 2, 3)$ let $\chi_j : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$. such that

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

and $s_j : \{0,1\}^m \to \{0,1\}$ such that

$$s_j(\vec{c}) = \begin{cases} 1 & \text{if the } j\text{th variable of } \vec{c} \text{ is positive} \\ 0 & \text{if complemented} \end{cases}$$

A truth assignment A is a function

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

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

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

evaluates to 0 iff A satisfies clause $\vec{c}$ of f.

In order to verify that A is a satisfying assignment, one needs to check that $\forall \vec{c}: CC(A, \vec{c}) = 0$.

Let $\vec{c} = c_1, ..., c_m$ and $\vec{v} = v_1, .., v_m$. We extend the arithmetization in such a way that c_i and v_i will take arbitrary values from $\mathcal{F}$. This requires to extend the domain of the functions A, χ_i, and s_i $(i = 1, 2, 3)$, from $\{0,1\}^m$ to $\mathcal{F}^m$.

Definition 6 *Let $\mathcal{F}$ be any field. Given a function $f : \{0,1\}^m \to \mathcal{F}$, the* multilinear extension *of f over $\mathcal{F}$, denoted $\hat{f} : \mathcal{F}^m \to \mathcal{F}$, 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 $y_1, ..., y_m$.*

We do not prove here the following simple fact.

Fact 7 *: Let $\mathcal{F}$ be an arbitrary field. Any function $f : \{0,1\}^m \to \mathcal{F}$ has a unique multilinear extension over $\mathcal{F}$. The value of f over any point of the extended domain can be computed in time $2^{O(m)}$ given f on $\{0,1\}^m$.*

[4] A similar change is also suggested in [BFL90].

Since M knows the values of χ_i, s_i $(i = 1, 2, 3)$ over the restricted domain, it can compute the value of their multilinear extension over any point of the extended domain in polynomial time. The situation with A is different. First, unless $\mathbf{P} = \mathbf{NP}$, M can not compute A over $\{0,1\}^m$ (it may not even exist) . Moreover, M can not receive a full description of A from the oracle, as it can receive at most $O(m \log m)$ bits of information from the oracle. These are way too few to compute the multilinear extension of A. Instead the oracle provides M with the multilinear extension of A on arguments of M's choice. M must check that the answers of the oracle indeed correspond to a multilinear extension of a boolean function A with high probability.

The protocol M works in two stages:

1. M verifies that the extended A is (almost) multilinear (multinomial of degree 1 in all variables), 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 [LFKN90] type protocol to verify, with high probability, that $\forall \vec{c}: CC(A, \vec{c}) = 0$.

We show how to do part 1 in subsection 4.2, and part 2 in subsection 4.1

4.1 How to Verify that a Function Disappears over a Large Set

We now show how to verify with high probability that $\forall \vec{c}: CC(A, \vec{c}) = 0$. Clearly, this would imply that A satisfies f. All arithmetic in this and the next section is done over the finite field $\mathcal{F}$. In order to meet our bounds we fix the size of $\mathcal{F}$ to $O(m^2)$.

Lemma 8 *Let $\mathcal{F}$ be an arbitrary finite field. There exists a family $DTIME(2^{O(m)})$ computable functions $R = \{R_i | i \leq 2^{O(m)}, R_i: \{0,1\}^m \to \mathcal{F}\}$ such that for any non-identically zero function $h: \{0,1\}^m \to \mathcal{F}$,*

$$\sum_{\vec{y} \in \{0,1\}^m} h(\vec{y}) \cdot R_i(\vec{y}) \neq 0$$

for more than 99% of i.

Proof: (Sketch.) For our purpose the following linear error correcting code over $\mathcal{F}$ will suffice: Let s be such that $|\mathcal{F}|^s \geq 2^{3m} \geq |\mathcal{F}|^{s-1}$. Let K be an s dimensional extension of $\mathcal{F}$.

Let us represent an element $k \in K$ as $k = \sum_{i=1}^{s} [k]_i b_i$, where $b_1, \ldots, b_s \in K$ is some basis of K over $\mathcal{F}$ and $[k]_i \in \mathcal{F}$ for $1 \leq i \leq s$.

For $k \in K$, $1 \leq i \leq s$ define

$$R_{k,i}(\vec{y}) = [k^{\vec{y}}]_i$$

(recall that $\vec{y}$ is a natural number). It is easy to show that the family $\{R_{k,i} \mid \ k \in K; \ 1 \leq i \leq s\}$ has the desired property. □

To show that for all $\vec{c}: CC(A, \vec{c}) = 0$, it suffices to show that the following multinomial of constant degree:

$$E(A) = \sum_{\vec{c} \in \{0,1\}^m} CC(A, \vec{c}) \cdot R_i(\vec{c})$$

evaluates to 0 with high probability over the choices of i.

The following sum-check protocol can be used to verify (with high probability) that $E(A)$ indeed evaluates to 0.

The Sum-Check Protocol:

Given a multinomial φ of constant degree d and a constant c, there is a protocol to check that

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

This protocol first appeared in [LFKN90] and used in [Sha90] and [BFL90], we recall it for completeness here. Since φ is a multinomial of degree d, the function

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

is a polynomial of degree d.

To prove $(*)$, the oracle sends M a polynomial of degree d, g', and claims that $g' = g$. If $(*)$ is false, then there are 2 possibilities:

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

2. $g' \neq g$, and since both g and g' are polynomials of degree d, they agree on at most d values.

M chooses randomly a $k \in \mathcal{F}$ and verifies that

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

This is false, with high probability, if $(*)$ is false.

M repeats this procedure for variables $y_2, y_3, \ldots$ until all the variables are instantiated; at this point M can check the assertion by himself. For $|F| > \frac{md}{\epsilon}$, the probability of falsely accepting is at most ϵ [LFKN90].

The number of random and communication bits that M uses in this part of the protocol is $O(m \log m)$.

Notice that the test requires the evaluation of φ only at one single point. Thus its reliability does not change too much even if a small constant fraction of the values of φ is incorrect.

4.2 Multilinearity Test

As part of the proof system for NEXPTIME, [BFL90] propose a procedure for testing that a function is multilinear. For a function $f: \mathcal{F}^m \to \mathcal{F}$ 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). However, in order to prove Theorem 6, we need a better bound on the performance of the test. We present a test which is very similar to the one in [BFL90], but requires only $O(m \max\{\log m, \log|\mathcal{F}|\})$ random and communication bits.

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 $ML(\mathcal{F}^m)$ be the set of multilinear functions defined on $\mathcal{F}^m$ (we will sometimes use shorthand ML). 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 points of $\{x_1, \ldots, x_{|\mathcal{F}|}\} \subseteq \mathcal{F}^m$ an *aligned line* in direction x_i if they differ only in the i^{th} coordinate. The set of three distinct elements a, b and c of an aligned line is called *triple* or *aligned triple*.

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

Definition 7 *Let $f : \mathcal{F}^m \to cF$ be an arbitrary function. We say that a triple $\{a, b, c\}$ (in direction x_i) is f-linear iff the function f restricted to the three element set $\{a, b, c\}$ is linear (in x_i).*

Our test exploits the fact that if $\Delta_{ML}(f) \geq 0.1$ then at least $O(1/m)$ fraction of the triples are not f-linear (Lemma 12). Thus it is enough to check $O(n)$ randomly chosen triples for f-linearity.

MULTILINEARITY TEST (random sampling version)

1. Choose randomly 300m triples and ask the oracle for the value of f on them.
2. Accept if all of them are f-linear, else reject.

The following theorem guarantees the correctness of the test.

Let us define $T = m \binom{|\mathcal{F}|}{3} |\mathcal{F}|^{m-1}$; the number of all aligned triples. For $f : \mathcal{F}^m \to \mathcal{F}$ define

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

Theorem 9 *Let $f : \mathcal{F}^m \to \mathcal{F}$ be an arbitrary function, $\Delta_{ML}(f) \geq \frac{12m}{|\mathcal{F}|-2}$. Then $\tau(f) \geq \frac{\Delta_{ML}(f)}{30m}$.*

For the proof we split the theorem according to the value of $\Delta_{ML}(f)$:

Lemma 10 *Let $f : \mathcal{F}^m \to \mathcal{F}$ be an arbitrary function, $0.5 \geq \Delta_{ML}(f) \geq \frac{12m}{|\mathcal{F}|-2}$. Then $\tau(f) \geq \frac{\Delta_{ML}(f)}{2m}$.*

Proof: Let L be a multilinear function such that $\Delta(f, L) = \Delta_{ML}(f)$. Let G be the indicator function of $L - f$ (i.e. G is 0 where f agrees with L and 1 otherwise). We say that an aligned triple $\{a, b, c\}$ is *two colored* if $|\{G(a), G(b), G(c)\}| = 2$, otherwise $\{a, b, c\}$ is *one colored*. Note that in any f-linear two colored aligned triple, only one of the elements is colored by 0. Define:

$$\delta(f) = \frac{|\{\text{two-colored triple}\}|}{T}.$$

We show that

$$\delta(f) \leq \tau(f) + \frac{3}{|\mathcal{F}| - 2}. \qquad (1)$$

This follows from 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 $G(c) = 0$. Then for any $c' \neq c$ which satisfies $G(c') = 0$, the triple $\{a, b, c'\}$ cannot be f-linear. Since there are only $\binom{|\mathcal{F}|}{2}$ possible choices of a and b along any aligned line, Equation 1 follows.

Estimating $\delta(f)$:

Consider the graph G in which the nodes are the points of $\mathcal{F}^n$ and two nodes are connected by an edge iff the two corresponding points lie on the same aligned line. Define $S = \{x \in \mathcal{F}^n \mid G(x) = 1\}$. We show that $\delta(f)$ is proportional to $|E(S, \overline{S})|$, the number of edges between S and $\overline{S}$.

$$\frac{|E(S, \overline{S})|}{|E(G)|} = 2\delta(f)/3, \tag{2}$$

Indeed: Any triple contains three edges. Any edge is contained in $|\mathcal{F}| - 2$ triples. Any two colored triple contains exactly two edges between S and $\overline{S}$. Any edge between S and $\overline{S}$ is contained in exactly $|\mathcal{F}| - 2$ two-colored triples. Equation 2 follows.

The following proposition bounds $|E(S, \overline{S})|$.

Proposition 11 $|E(S, \overline{S})| \geq \frac{|S||\overline{S}|}{2|\mathcal{F}|^{m-1}}$.

Proof: For arbitrary nodes $a, b \in G$, let *path(a,b)* denote the edges connecting adjacent nodes on the sequence $\{c_j\}$, for $0 \leq j \leq m$, where node c_j corresponds to the point which agrees with the point corresponding to a on the first j coordinates, and with the point corresponding to node b on the last $m - j$ coordinates. Clearly, if $a \in S$ and $b \in \overline{S}$, at least one edge in *path(a,b)* belongs to $E(S, \overline{S})$. On the other hand, each edge in G belongs to at most $2|\mathcal{F}|^{m-1}$ paths. The bound follows. □

Notice that $|S|/|V(G)| = \Delta_{ML}(f))$ and by assumption $|\overline{S}|/|V(G)| \geq 0.5$. From this, from equations 1 and 2 and from the estimate in proposition 11 we can write:

$$\tau(f) \geq \frac{3\Delta_{ML}(f)|\mathcal{F}|^{2m}}{8|\mathcal{F}|^{m-1}n\binom{|\mathcal{F}|}{2}|\mathcal{F}|^{m-1}} - \frac{3}{|\mathcal{F}| - 2}.$$

By our constrains the l.h.s. $\geq \frac{\Delta_{ML}(f)}{2m}$.

□

Lemma 12 *If $f : \mathcal{F}^m \to \mathcal{F}$ be an arbitrary function, $\Delta_{ML}(f) \geq 0.1$ then $\tau(f) \geq (1 - 1/|\mathcal{F}|)^n \frac{1}{30m}$.*

Proof:

The proof is by induction on m. The proof of the base case of the induction ($m = 1$) is simple, and is omitted. For the induction step, we prove the statement for m by fixing the first coordinate, and using the induction hypothesis for $m - 1$,

When fixing the first coordinate x_1 to $a \in \mathcal{F}$ we get a subspace S_a. Let T_a be the set of all aligned triples in direction x_1 that go through S_a and $T'_a \subseteq T_a$ is the set of those, that are not f-linear. If $|T'_a|/|T_a| \geq 0.1$, then we say that $a \in \mathcal{F}$ is credited.

Let us denote by f_a the restriction of f to S_a. We say that $a \in \mathcal{F}$ is good if $\Delta_{ML}(f_a) \geq 0.1$.

Lemma 13 *If there are at least two elements a and b of $\mathcal{F}$ that are neither good nor credited then $\Delta_{ML}(f) \leq 0.4$.*

Proof: Let L_a, L_b be a multilinear functions on S_a and S_b respectively such that $\Delta(L_a, f_a) \leq 0.1$, $\Delta(L_b, f_b) \leq 0.1$. Let L be the unique multilinear extention of L_a and L_b on $\mathcal{F}^m$. We prove that $\Delta(f, L) \leq 0.4$. Random points $r \in S_a \cup S_b$ agree with L with probability at least 9/10. Thus we have to consider only points not in $S_a \cup S_b$. Choose randomly an aligned line in the direction of x_1 and two distinct points on it r and r' such that they are neither on S_a nor on S_b. Let $x = x(r, r')$ and $y = y(r, r')$ denote the points on this line which do belong to S_a and S_b. Now

$$Prob(\{r, r', x\}, \{r, r', y\} \text{ are } f\text{-linear};$$

$$L_a(x) = f_a(x);\ \ L_b(y) = f_b(y)) \geq 0.6.$$

When the above event holds then $f(r) = L(r)$ (and f(r')=L(r')). Thus

$$Prob(f(r) = L(r) | r \notin S_a \cup S_b) \geq 0.6. \tag{3}$$

□

By Lemma 10 and the above lemma we are done if there are at least two elements of $\mathcal{F}$ that are neither good nor credited. Otherwise for each credited $a \in \mathcal{F}$ we can count $T/(30m|\mathcal{F}|)$ non f-linear triples on account of T'_a (we lose a factor

of 3 because a triple might count in three different $T'_a s$). Also for each good $a \in \mathcal{F}$ we can count $\frac{(m-1)T}{n\mathcal{F}}(1-\frac{1}{\mathcal{F}})^{m-1}\frac{1}{30(m-1)}$ non-f-linear triples by induction. The later expression is the smaller, and since at least $|\mathcal{F}|-1$ of the elements of $\mathcal{F}$ are either good or credited, we get the lower bound on the non-f-linear triples. □

4.2.1 Multilinearity Test: Efficient Version

The protocol in the previous section uses $O(m \log T) = O(m^2 \log m)$ random bits. To improve this we use pseudorandom sampling.

The set of all triples will be called $\mathcal{T}$. Consider a family of subets $\{H_i \subseteq \mathcal{F} \mid 1 \leq i \leq T^2\}$ of $\mathcal{F}$ such that

1. For $1 \leq i \leq T^2$ the set H_i has size $300m$.
2. For every set $H \in \mathcal{T}$, $|H| \geq T/m$ the probability that a randomly cosen H_i intersects H is at least 0.9.

The modified protocol is:

MULTILINEARITY TEST (pseudorandom sampling version)

1. Choose a randomly $1 \leq i \leq T^2$ and ask the oracle for the value of f on all of the triples that belong to H_i.
2. Accept if all of the triples are f-linear, else reject.

The number of random and communication bits this protocol uses is $O(m \log m)$. To construct the pseudorandom system of sets that we need, we can use pairwise independent sampling [CG89] (details omitted).

□

5 Open Problems

The most intriguing problem that remains is: Suppose that $\omega(G)$ can be approximated within a factor of 2. Does this imply that $P = NP$?

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

Acknowledgment

We thank Mike Sipser, Avrim Blum and László Babai for many discussions on this paper. We thank Sasha Shen for clarifying observations on multilinearity tests.

References

[AB87] N. Alon and R. Boppana. The monotone circuit complexity of boolean functions. In *Combinatorica 7*, pages 1–22, 1987.

[BE83] 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.

[BFL90] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. In *Proc. 31th IEEE Symp. on Foundations of Computer Science*, pages 16–25, 1990.

[BFLS91] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in poly-logarithmic time. In *Proc. 23rd ACM Symp. on Theory of Computing*, 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.

[BH90] 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.

[BHZ87] R. B. Boppana, J. Hastad, and S. Zachos. Does co-np have short interactive proofs. In *Inform. Process. Lett., 25*, pages 127–132, 1987.

[BR90] B. Berger and J. Rompel. A better performance guarantee for approximate graph coloring. *Algorithmica*, 5(4):459–466, 1990.

[BS89] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. In *6th STACS. Lecture Notes in Computer Science 349*, pages 256–267, 1989.

[CG89] B. Chor and O. Goldreich. On the power of two-point based sampling. *Journal of Complexity*, 5:96–106, 1989.

[CL89] 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.

[Con91] A. Condon. The complexity of the max word problem. In *8th STACS*, 1991.

[FRS88] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. In *Proc. 3rd STRUCTURES*, pages 156–161, 1988.

[FS89] U. Feige and A. Shamir. Multi-oracle interactive protocols with space bounded verifiers. In *Proc. 4th STRUCTURES*, pages 158–164, 1989.

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

[Hal90] M. M. Halldórsson. A still better performance guarantee for approximate graph coloring. Technical Report 90-44, 1990.

[Kar72] 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.

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *STOC90*, pages 2–10, 1990.

[LV89] N. Linial and U. Vazirani. Graph products and chromatic numbers. In *Proc. 30th IEEE Symp. on Foundations of Computer Science*, 1989.

[PR90] A. Panconesi and D. Ranjan. Quantifiers and approximation. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 446–456, 1990.

[PY88] C. Papadimitriou and M Yannakakis. Optimization, approximation and complexity classes. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 229–234, 1988.

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

[Wig83] A. Wigderson. Improving the performance guarantee for approximate graph coloring. *Journal of the ACM*, 30(4):729–735, 1983.