Some optimal inapproximability results

Johan Håstad
Royal Institute of Technology
Sweden
email:johanh@nada.kth.se

Abstract

We prove optimal, up to an arbitrary $\epsilon > 0$, inapproximability results for Max-Ek-Sat for $k \geq 3$ and optimizing the number of satisfied linear equations modulo a prime p. Max-Ek-Sat is the variant of CNF-Sat where each clause is of length exactly k. As a consequence of these results we get improved lower bounds for many problems studied previously. In particular, for Max-E2-Sat, Max-Cut, Max-Di-Cut and Vertex cover. For Max-E2-Sat the obtained lower bound is essentially $22/21 \approx 1.047$ while the strongest upper bound is around 1.074.

1 Introduction

We know that many natural optimization problems are NP-hard. This means that they are probably hard to solve exactly in the worst case. In practice, however, it is many times sufficient to get reasonable good solutions for all (or even most) instances. In this paper we study the existence of polynomial time approximation algorithms for some of the basic NP-complete problems. We say that an algorithm is a C-approximation algorithm if it for each instance produces an answer that is at most off by a factor C from the optimal answer. The fundamental question is for a given NP-complete problem, for what value of C can we hope for a polynomial time C-approximation algorithm. Posed in this generality this is a large research area with many positive and negative results. In this paper we concentrate on negative results, i.e. results of the form that for some $C > 1$ a certain problem cannot be approximated within C in polynomial time. These results are invariably based on plausible complexity theoretic assumptions, the weakest possible being NP$\neq$P since if NP=P, all considered problems can be solved exactly in polynomial time.

The most basic NP-complete problem is satisfiability of CNF-formulas and probably the most used variant of this is 3-SAT where each clause contains at most 3 clauses. For simplicity, let us assume that each clause contains exactly 3 variables. The optimization variant of this problem is to satisfy as many clauses as possible. It is not hard to see that a random assignment satisfies each clause with probability 7/8 and hence if there are m clauses it is not hard (even deterministically) to find an assignment that satisfies $7m/8$ clauses. Since we can never satisfy more than all the clauses this gives a 8/7-approximation algorithm. This was one of the first approximation algorithms considered [13] and one of the main results of this paper is that this is optimal to within an arbitrary positive additive constant ϵ.

A problem that in many respects is as basic as satisfiability is that of solving a system of linear equations over a field. One reason that not many papers are written on this subject in complexity theory is the simple and powerful procedure of Gaussian elimination which makes it solvable in polynomial time.

Gaussian elimination is, however, very sensitive to incorrect equations. In particular, if we are given an overdetermined system of equations it is not clear how to efficiently find the "best solution", at least if we interpret this as the assignment satisfying the maximal number of equations. It is not hard to see that this problem is NP-complete over the field of two elements. In fact, the special case of having equations of the form $x_i + x_j = 1$ is equivalent to Max-Cut. We believe that as an optimization problem this problem will play a natural and important role. As with 3-SAT there is an obvious approximation algorithm that just does as well as picking the variables at random and in this case a random assignment satisfies half the equations and thus this yields a 2-approximation algorithm. One of the main results of this paper is to prove that this is, again upto an arbitrary $\epsilon > 0$ and based on NP$\neq$P, the best possible for a polynomial time approximation algorithm.

Other results included in this paper are similar results for linear equations mod p where p is a fixed prime. By reduction we get improved constants for Max-2-Sat, Max-Cut and Max-Di-Cut and Vertex Cover. These reduction are all done from the problem with linear equations modulo 2 with 3 variables in each equation.

1.1 Short history and our contribution

The question of proving NP-hardness of some approximation problems was discussed at length already in the book by Garey and Johnson [10], but really strong results were not obtained until the connection with multiprover interactive

STOC '97 El Paso, Texas USA

proofs, introduced for a different reason by [7], was discovered in the seminal paper of Feige et al. [9] in 1990. The two proof models that we need in this paper are that of two prover interactive proofs and that of probabilistically checkable proofs (PCP). In the first model the verifier interacts with two provers who cannot communicate with each other and in the second the verifier does spot checks in a written proof. The first result proving hardness for the problems we are discussing here using these methods was obtained in the fundamental paper by Arora et al [1] were it was established that NP had PCP reading only a constant number of bits and using a logarithmic number of random coins. This result implies that there is some constant $C > 1$ such that Max-3-Sat could not be approximated within C unless NP=P. The first explicit constant was given by Bellare et al [6] and based on slightly stronger hypothesis they achieved the constant 94/93. Bellare and Sudan improved this to 66/65 and the strongest result prior to our results here are by Bellare, Goldreich and Sudan [5] obtaining the bound 27/26. This paper [5] also studied the problem of linear equations mod 2 and obtained inapproximability constant 1.13.

The improvement in the constants has many times been obtained by extracting some important property from a previous protocol, using that protocol as a black box and then adding some conceptually new construction. This is more or less what we do also in the current paper. The needed starting point for our construction is the inapproximability result for 3-SAT of [1] described above. This translates naturally into a constant error, two-prover multiprover interactive proof for satisfiability. Namely, choose a clause at random and then randomly a variable appearing in that clause. Ask one prover for the values of all variables in the clause while the other is asked for chosen variable. The verifier accepts if the answers are consistent and satisfies the clause. It is easy to see that this proof system has constant error rate for any non-satisfiable formula output by the construction of [1]. This protocol has two important properties that we need. It exchanges a constant number of bits and one answer is a projection of the other.

To modify the protocol we first do a parallel version of the above two-prover proof to bring down the acceptance probability for unsatisfiable formulas. For the basic result we only need that the error probability goes to 0 when the number of parallel repetitions increase. However, the result by Verbitsky [18] is not sufficient and that it we need that for any constant ϵ there is another constant v such that v repetitions is enough to make the acceptance probability below ϵ. In [18], v depends on the number of possible questions, and thus we we need the strong results by Raz [16], saying that the error probability decreases as c^v where the constant c only depends on the answer size. We then transfer this parallelized two-prover interactive proof to a PCP where instead of writing down answers, we ask the prover to write down the *long code* of the answers.

The long code, one of the great inventions of [5], of an input $x \in \{0,1\}^v$ is the value of every function $f : \{0,1\}^v \mapsto \{0,1\}$ on x and thus it is a string of length 2^{2^v}. It is a very wasteful encoding but since v is a constant we can afford it. The long code is universal in that it contains every other binary code as a sub code. Thus it never hurts to have this code available, but it is still surprising that it is beneficial to have such a wasteful code. We do not know how to make our construction work with a smaller code.

Said in other words our proof system is a combination of the proof system of [1] and a new inner verifier (as introduced in [5]). Thus we once more display the power of proof composition introduced in [2] although in an unusually simple form.

Our main results are obtained by simply reading 3 bits, each random but correlated, of these codes and then decide whether to accept or reject based on the read bits. For the result of linear equations we have a proof system that accepts or rejects depending on the exclusive-or of these bits and this proof system has completeness $1-\epsilon$ and soundness $1/2+\epsilon$ for an arbitrary ϵ. Note that we have a PCP that is very closely connected to the optimization problem we are studying. This is of course no accident and we believe, as advocated by [5], that to get strong inapproximability results one needs to design a PCP with the intended application in mind.

The result about 3-SAT follows in the approximation sense from the result on linear equations. However, there is an imperfection in that it only proves that for any ϵ it is NP-hard to distinguish those formulas where you can satisfy $1-\epsilon$ fraction of the clauses and those where you can satisfy a fraction $7/8+\epsilon$. As discussed in [15] we want the correct gap location which in this case means that it is NP-hard to distinguish satisfiable formulas from those where you can only satisfy a fraction $7/8+\epsilon$. We establish this by a separate, more complicated proof.

An outline of the papers is as follows. In Section 2 we give the result for linear equations. It is the simplest result to prove and it gives most of our corollaries. In Section 3 we give the results on Max-k-Sat. We obtain some results for other problems in Section 4 and we compare with known upper bounds to study the gaps in Section 5. We finally briefly discuss how to make our arbitrary constants be functions of the inputs length in Section 6 and end by some concluding remarks.

2 Linear equations

We start with a 3-SAT formula φ as we know can be produced by [1] and [14]. Either φ is satisfiable or we can satisfy at most a fraction $h < 1$ of the clauses and it is NP-hard to distinguish the two cases. Each variable appears exactly 6 times. We want to produce a reduction to linear equations mod 2. The equations we produce contain 3 variables in each equation.

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

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

We have by a straightforward analysis.

Lemma 2.1 *If any assignment satisfies at most a fraction h of the clauses of φ, then V accepts with probability at most $(2+h)/3$.*

We now concentrate on the game consisting of v copies of this basic game. I.e. the verifier picks v random clauses and then one random variable from each chosen clause. The clauses are sent to P_1 while the picked variables are sent to

P_2. The verifier can again be made to always accept when φ is satisfiable while the acceptance probability in the other case is bounded by $acc(v)$ which, as proved by Raz [16] is bounded by c^v for some $c < 1$.

To fix notation, V is the set of variables chosen and W the set of variables in the clauses. Thus V is of size v and W is of size $3v$. Since v is a constant, with probability $1 - O(1/n)$ the chosen clauses are disjoint and the variables in V different. For notational simplicity we assume that this is always the case.

The PCP contains for each V and W the long code of a fixed satisfying assignment to the given formula restricted to that subset. We denote the supposed long codes for generic sets V and W by the functions A and B respectively. Of course A and B depends on the identity of V and W but we leave this dependence implicit. We should check that this long codes have the appropriate properties but let us take a detour to gather intuition and consider the simpler problem of testing only whether A is a correct long code.

We want to to read three bits $A(f_0), A(f_1)$ and $A(f_2)$ and accept if these three bits satisfy some property. If we are aiming at proving a result for linear equations this property should be that the exclusive-or of the 3 bits is 0. If we want perfect completeness and know nothing about the input for which A is the long code, the only way we can assure this is by letting the three bits $f_0(x), f_1(x)$ and $f_2(x)$ have the correct exclusive-or for any input x. The natural way to achieve this is to pick f_0 and f_1 randomly then determine f_2 pointwise by setting $f_2(x) = xor(f_0(x), f_1(x))$. A moments reflection shows that this is not the test we want since it is a linear test and in fact any A which is an exclusive-or of an arbitrary number of long codes will pass the test. The analysis of this test as given in [4] (for the case when A is a normal function with n bit strings as arguments) is, however, illuminating and hence let us repeat it in the current situation, which only differs from the original case in notation.

For notational convenience let all functions (both ordinary functions like f and long codes like A) take values in ± 1. In this notation exclusive-or becomes multiplication and to make this correspondence the usual we let -1 correspond to "true" while 1 corresponds to "false". The Fourier coefficients of A are defined as

$$\hat{A}_\alpha = 2^{-2^v} \sum_f A(f) \prod_{x \in \alpha} f(x)$$

where $\alpha \subseteq \{0,1\}^v$. The transformation can be inverted as

$$A(f) = \sum_{\alpha \subseteq \{0,1\}^v} \hat{A}_\alpha \prod_{x \in \alpha} f(x). \quad (1)$$

The Fourier coefficients are thus rational numbers and by Plancerel's identity

$$\sum_\alpha \hat{A}_\alpha^2 = 2^{-2^v} \sum_g A^2(g) = 1$$

where the last equality follows from the fact that $A^2(g) = 1$ for all g.

The reader might be more familiar with the Fourier transform of ordinary functions and hence with the formulas (again in ± 1 notation)

$$\hat{f}_\alpha = 2^{-n} \sum_x f(x) \prod_{i \in \alpha} x_i$$

and

$$f(x) = \sum_{\alpha \subseteq [n]} \hat{f}_\alpha \prod_{i \in \alpha} x_i.$$

After a moments reflection one notes that the only difference is that $\{0,1\}^v$ takes the place of $[n]$. Normal n bit strings can be thought of as mappings from $[n]$ to ± 1 while our basic unit now is mappings from $\{0,1\}^v$ to ± 1.

To evaluate the test described above note that $A(f_0)A(f_1)A(f_2)$ is one when the test accepts and negative one when it fails and thus we want to evaluate $E(A(f_0)A(f_1)A(f_2))$ which is the *advantage* of the test, i.e. the probability of accept minus the probability of reject. Using (1) we need to estimate

$$\sum_{\alpha_0,\alpha_1,\alpha_2} \hat{A}_{\alpha_0}\hat{A}_{\alpha_1}\hat{A}_{\alpha_2} E\Big(\prod_{x \in \alpha_0} f_0(x) \prod_{x \in \alpha_1} f_1(x) \prod_{x \in \alpha_2} f_2(x)\Big). \quad (2)$$

The key is now that if not $\alpha_1 = \alpha_2 = \alpha_3$ then then inner expected value is 0. This follows since

$$\prod_{x \in \alpha_0} f_0(x) \prod_{x \in \alpha_1} f_1(x) \prod_{x \in \alpha_2} f_2(x) = \prod_{x \in \alpha_0 \Delta \alpha_2} f_0(x) \prod_{x \in \alpha_1 \Delta \alpha_2} f_1(x)$$

where Δ is symmetric difference operator. Thus the expression (2) simplifies to $\sum_\alpha \hat{A}_\alpha^3$ and this is bounded by $\max \hat{A}_\alpha \sum_\alpha \hat{A}_\alpha^2 = \max \hat{A}_\alpha$ and thus a function A that passes this test with a high probability is close to a linear combination of long codes.

In our case we want to accept only in the case of correct long codes. We cannot do this exactly but at least we can make sure that A's that pass the test are close to small exclusive-ors of long codes. The key to doing this is to introduce an error function μ by setting $\mu(x) = 1$ with probability $1 - \epsilon$ and $\mu(x) = -1$ otherwise independently for each x. We now set $f_2(x) = f_0(x)f_1(x)\mu(x)$ and use the same test. A correct proof is now accepted with probability $1 - \epsilon$ and we need to estimate the advantage of the test in general. Using the same approach as before we need to consider

$$E\Big(\prod_{x \in \alpha_0 \Delta \alpha_2} f_0(x) \prod_{x \in \alpha_1 \Delta \alpha_2} f_1(x) \prod_{x \in \alpha_2} \mu(x)\Big).$$

We see again that this is 0 unless $\alpha_1 = \alpha_2 = \alpha_3 = \alpha$ and in this case the last factor gives the result $(1-2\epsilon)^{|\alpha|}$ and thus

$$E(A(f_0)A(f_1)A(f_2)) = \sum_\alpha \hat{A}_\alpha^3 (1-2\epsilon)^{|\alpha|} \leq \max_\alpha \hat{A}_\alpha (1-2\epsilon)^{|\alpha|}$$

and thus to pass the test with high probability we need $\hat{A}_\alpha$ to be large for some α of small size. Thus even if an A that pass the test with high probability need not be a long code at least it needs to be close to a small exclusive-or of long codes and this will be sufficient for our purposes.

An observant reader might have noted that if A is the constant function 1 then it always passes the test. For this function $\hat{A}_\emptyset = 1$ while all other Fourier coefficients are 0. We eliminate this anomaly by requiring that $A(f) = -A(-f)$ for all f. This is achieved by, for each pair $(f, -f)$, having only one entry in the table. When either the value of f or $-f$ is wanted this value is accessed. There is some convention (fixed but arbitrary) which decides for which of the two values the answer should be negated. This is just a way of reformulating "folding over 1", an elegant and useful

construction introduced in [5]. It is not hard to see that this convention forces $\hat{A}_\alpha = 0$ for all α of even size and in particular for $\alpha = \emptyset$.

In our test we have two long codes A (on V) and B (on W). It turns out that it is sufficient to follow the above outline and let f_0 remain a function on V while f_1 and f_2 are functions on W. To emphasize this change we rename the three functions f, g_1 and g_2.

One final word about the long code B on W. It is supposed to be the long code of an input which satisfies the clauses used to construct W. Let us write this condition as $h(y) = -1$ (remember that -1 corresponds to "true"). Thus the functions g and $g \wedge h$ should give the same value, and thus we might as well read the latter value. In other words, we use the convention $B(g) = B'(g \wedge h)$ where B' is the actual table given. One way of viewing this is that inputs to B are not functions with domain $\{0,1\}^w$ but rather $\{0,1\}^w \cap \{y \mid h(y) = -1\}$. This is similar in spirit but different from "folding over h" as used in [5]. Naturally we make sure, in the same way as described above, that $B(g) = -B(-g)$ for any g.

After these preliminaries let us return to the real test.

1. Pick a random V and W as described above.
2. Pick independently three functions f (supported on V) and g_1 and μ (supported on W). The functions f and g_1 are picked with the uniform distribution while $\mu(y)$ is set to 1 with probability $1-\epsilon$ and to -1 with probability ϵ independently for different y. Define g_2 by setting $g_2(y) = f(y)g_1(y)\mu(y)$ for each y.
3. Accept iff $A(f)B(g_1)B(g_2) = 1$.

Note that since V is a subset of W, $f(y)$ can be defined by simply first projecting y onto V. Also, remember that all functions takes values in $\{-1,1\}$.

Note that the verifier thus accepts a correct proof with probability $1-\epsilon$. This follows since if y_0 is the restriction of the satisfying assignment to W and x_0 is the restriction to V then the three read bits in the correct proof are $f(x_0)$, $g_1(y_0)$ and $g_2(y_0)$ and, since x_0 is the projection of y_0, $f(x_0)g_1(y_0)g_2(y_0) = \mu(y_0)$ and hence it is 1 with probability $1-\epsilon$. The main claim is now:

Lemma 2.2 *For any $\epsilon > 0$, $\delta > 0$ and $v \geq C_{\epsilon,\delta}$. If the probability that the test accepts is at least $(1+\delta)/2$, then the original formula φ is satisfiable.*

Proof: The general outline of the proof is to find good strategies for the provers given a successful proof. We can clearly assume that $\delta < 1/4$.

Fix V and W and let us analyze the probability of success. Following the intuition described above we have a supposed long code A on V and we want to use the Fourier expansion

$$A(f) = \sum_{\alpha \subseteq \{0,1\}^v} \hat{A}_\alpha \prod_{x \in \alpha} f(x).$$

Similarly we have a long code B on W with the Fourier expansion

$$B(g) = \sum_{\beta \subseteq \{0,1\}^w} \hat{B}_\beta \prod_{y \in \beta} g(y).$$

By the above assumptions on how to access the long code we can assume that all coefficients of even size are 0 in both A and B. Furthermore for any β such that $\hat{B}_\beta \neq 0$ we can assume that for each $y \in \beta$, $h(y) = -1$. This follows since

$$\hat{B}_\beta = 2^{-2^w} \sum_g B'(h \wedge g) \prod_{y \in \beta} g(y)$$

and if $y_0 \in \beta$ where $h(y_0) = 1$ then the pair (g, g') where $g'(y) = g(y)$ for $y \neq y_0$ and $g'(y_0) = -g(y_0)$ cancel each other in the above sum. With the interpretation that the inputs to B are functions with domain $\{0,1\}^w \cap \{y \mid h(y) = -1\}$ we see that the condition is simply that β should be a subset of the domain.

By assumption

$$E(A(f)B(g_1)B(g_2)) \tag{3}$$

is δ and we need to investigate what implications this has for the Fourier coefficients. (3) equals

$$\sum_{\alpha,\beta_1,\beta_2} \hat{A}_\alpha \hat{B}_{\beta_1} \hat{B}_{\beta_2} E(\prod_{x \in \alpha} f(x) \prod_{y \in \beta_1} g_1(y) \prod_{y \in \beta_2} g_2(y)) =$$

$$\sum_{\alpha,\beta_1,\beta_2} \hat{A}_\alpha \hat{B}_{\beta_1} \hat{B}_{\beta_2} E(\prod_{x \in \alpha} f(x) \prod_{y \in \beta_1} g_1(y) \prod_{y \in \beta_2} (f(y)g_1(y)\mu(y))).$$

If $\beta_1 \neq \beta_2$ then since $g_1(y)$ is random and independent of everything else the inner expected value is 0. If $\beta_1 = \beta_2 = \beta$ then to calculate the inner expected value let $\pi_2(\beta)$ be the "mod 2" projection onto subsets of V. The mod 2 projection of one element is the standard projection. When projecting a set an element can be obtained many times. It is included in the target if it is obtained an odd number of times. We also have the standard projection $\pi(\beta)$.

To estimate the inner expected value observe first that the only dependences that exist are between values that have the same projection on V. Thus we need to estimate the expected value of expressions of the form

$$\prod_y f(y)\mu(y) \quad \text{and} \quad f(x) \prod_y f(y)\mu(y)$$

where all y have the same projection, x, onto V and then the general case follows by multiplication. By definition $f(y) = f(x)$ for all considered y. Let s_x be the number of y in the above product. Now $E(\prod_y \mu(y)) = (1-2\epsilon)^{s_x}$ while $E(f(x)^t)$ is 1 if t is even and 0 otherwise. The two quantities are independent and thus the second fact implies that the expected value is 0 unless for each occurring x, s_x is even iff $x \notin \alpha$. This is just the same as $\pi_2(\beta) = \alpha$ and thus the overall sum equals

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

We want to prove that if the expected value of this (over random choices of V and W) is at least δ then we have a good strategy of the provers.

We first identify parts of the sum that are always small. Since each β appears for exactly one α and, by Parseval's identity, $\sum \hat{B}_\beta^2 = 1$ and $|\hat{A}_\alpha| \leq 1$, the sum over all β with $|\beta| \geq \lceil \epsilon^{-1} \log \delta \rceil \stackrel{\text{def}}{=} k$ is bounded by $\delta/4$. Since the size of a projection is always smaller than the original set, we can discard also all α of size at least $k+1$.

We can also discard all terms for which $|\hat{A}_\alpha| \leq \delta/4$, since this part of the sum is bounded by $\delta/4 \sum_\beta \hat{B}_\beta^2 \leq \delta/4$.

To sum up, if we discard the terms considered ($|\beta| > k$, or $|\hat{A}_\alpha| \leq \delta/4$) in (4) then the expected value of the remaining sum is at least $\delta/2$. Now we define good randomized strategies for the provers. These can, at least in principle, be converted to optimal deterministic strategies that do at least as well.

The strategy of P_2 is first to pick a random α with $|\alpha| \leq k$ and $|\hat{A}_\alpha| \geq \delta/4$. The probability of picking α is defined to be proportional to $|\hat{A}_\alpha|$. It is easy to see that it is at least $\delta/4|\hat{A}_\alpha|$ since summing over all such α we have

$$\sum |\hat{A}_\alpha| \leq \frac{4}{\delta} \sum_\alpha \hat{A}_\alpha^2 = \frac{4}{\delta}.$$

P_2 sends a random $x \in \alpha$.

Let us point of two minor details. First note that the set of possible α might be empty for some choices of V since we only have that the expected value (over the choice of V and W) of the sum (4) is large. For V which this is the case P_2 can do anything but from an analysis point of view, we assume that P_2 gives up. Secondly note that all possible α are non-empty. This follows since $\hat{A}_\emptyset = 0$.

The strategy of P_1 is to pick a random β with $|\beta| \leq k$ where the probability of picking a specific β is proportional to $\hat{B}_\beta^2$ (and thus at least this number since $\sum_\beta \hat{B}_\beta^2 = 1$). P_1 then picks a random $y \in \beta$. The same minor details pointed out above for the strategy P_2 applies to the strategy of P_1.

Let us evaluate the success-rate of this strategy. Clearly it is at least k^{-2} times the probability that for the picked α and β we have $\alpha \cap \pi(\beta) \neq \emptyset$. The latter probability is at least

$$\delta/4 \sum_\alpha \sum_{\beta|\pi_2(\beta)=\alpha, |\beta|\leq k, |\hat{A}_\alpha| \geq \delta/4} \hat{A}_\alpha B_\beta^2.$$

Since this sum is larger than (4) over the same range, the expected value is at least $\delta/2$ and thus the probability that the verifier accepts with the given strategy is at least $k^{-2}\delta^2/8$. For sufficiently large v this means that the acceptance probability is greater than $acc(v)$ and we have proved Lemma 2.2.

Let E3-Lin-p be the problem of given a number of linear equations mod p with exact 3 variables in each equation, maximize the number of satisfied equations.

Theorem 2.3 *For any $\epsilon > 0$ it is NP-hard to approximate E3-Lin-2 within a factor $2 - \epsilon$.*

Proof: In the above PCP think of each bit in the proof as a variable. Succeeding at an instance of the test is equivalent to satisfying a linear equation with 3 variables. The set of all equations given by all possible random strings gives a set of equations with weights where the weights are given by the probability that a certain triplet is chosen and thus the total weight is one. If φ is satisfiable we can satisfy equations with a total weight which is at least $1 - \epsilon$ and otherwise the total weight of equations that can be satisfied is only at most $(1 + \delta)/2$. We need only eliminate the weights to prove the theorem. Suppose we have m equations. For an equation of weight w we simply write $\lceil wm^2 \rceil$ copies, each of weight one. It is easy to check that the approximation properties are preserved upto a small factor. Since v is constant the reduction runs in polynomial time and finally since ϵ and δ in Lemma 2.2 are arbitrary numbers larger than 0, the theorem follows. ∎

Note that there is a meta reason that we have to introduce the error function μ and make our test have non perfect completeness. If we had perfect completeness then the equations produced in the proof of Theorem 2.3 could all be satisfied simultaneously. However, to decide if a set of linear equations have a common solution is polynomial time by Gaussian elimination.

Next we extend the theorem to general prime moduli.

Theorem 2.4 *For any $\epsilon > 0$ it is NP-hard to approximate E3-Lin-p within a factor $p - \epsilon$.*

Proof: We construct a proof-system similar to the one considered in the proof of Theorem 2.3, i.e. we start with the parallelized two-prover protocol and chose V and W in the same way as before. Instead of letting f be a function from $\{0,1\}^v$ to $\{-1,1\}$ we let the range be the p'th roots of unity and let ζ be such a root. Whenever convenient we identify i mod p and ζ^i. We change the range for g_1 and g_2 similarly. We also extend the long code to give the value of all such functions on the given input. The range is then naturally also p'th roots of unity.

The test chooses f and g_1 uniformly and independently. For the error function, $\mu(y)$ is again 1 with probability $1 - \epsilon$ and otherwise chosen uniformly among all other values again independently for each y. Finally, $g_2(y) = (f(y)g_1(y)\mu_2(y))^{-1}$ for every y. The test accepts if $A(f)B(g_1)B(g_2) = 1$ and rejects otherwise. Note that the verifier again accepts a correct proof with probability $1 - \epsilon$ and we need to analyze what happens with incorrect proofs.

We have the two long codes A and B of the restriction of the supposed satisfying assignment to the sets V and W respectively. For natural reasons we use a mod p Fourier transform to analyze the situation this time. The indices of the coefficients α and β are this time mappings from $\{0,1\}^v$ and $\{0,1\}^w$ respectively to the integer $\{0,1,\ldots p-1\}$. The weight $|\beta|$ of β is defined to be the sum of the coordinates. We have

$$A(f) = \sum_\alpha \hat{A}_\alpha \prod_x f(x)^{\alpha(x)}.$$

and

$$B(g) = \sum_\beta \hat{B}_\beta \prod_y g(y)^{\beta(y)}.$$

By similar convention to the previous section we can assume that $A(\zeta f) = \zeta A(f)$ and $B(\zeta g) = \zeta B(g)$. This implies that $|\alpha|, |\beta| \equiv 1 \bmod p$ for all nonzero coefficients. Furthermore, we can again suppose that if $\hat{B}_\beta \neq 0$ then each element y such that $\beta(y) \neq 0$ satisfies the corresponding clauses.

To evaluate the quality of the test note that if we let σ_i $i = 1 \ldots p-1$ be the automorphisms sending ζ to ζ^i then the quantity

$$\sum_{i=1}^{p-1} \sigma_i(A(f)B(g_1)B(g_2))$$

is $p - 1$ if the test succeeds and -1 otherwise. Assume that this quantity has expected value δ. This corresponds to an acceptance probability $(1 + \delta)/p$. We want to prove that if v is sufficiently large this implies that φ is satisfiable. We need to estimate

$$E(\sigma_i(A(f)B(g_1)B(g_2))).$$

We expand this using the Fourier transform getting

$$\sigma_i(\sum_{\alpha,\beta_1,\beta_2} \hat{A}_\alpha \hat{B}_{\beta_1} \hat{B}_{\beta_2} E(\prod_x f(x)^{\alpha(x)} \prod_y g_1(y)^{\beta_1(y)} g_2(y)^{\beta_2(y)})).$$

The inner expected value is

$$E(\prod_x f(x)^{\alpha(x)} \prod_y g_1(y)^{\beta_1(y)-\beta_2(y)} (f(x)\mu(y))^{-\beta_2(y)}).$$

First note that if $\beta_1(y) \neq \beta_2(y)$ for some y, then the expected value in question is clearly zero. Now μ and f are independent and since $\sum_{i=0}^{p-1} \zeta^i = 0$ we have that

$$E(\mu(y)^t) = (1-\epsilon) + \frac{\epsilon}{p-1} \sum_{i=1}^{p-1} \zeta^i = 1 - \epsilon(1 + \frac{1}{p-1})$$

for any $t \neq 0 \ mod \ p$ Finally,

$$E(\prod_x f(x)^{\alpha(x)} \prod_y f(x)^{-\beta_2(y)})$$

is 0 if not $\prod_{y \in \pi^{-1}(x)} \beta_2(y) = \alpha(x)$ for all x in which case it is 1. This condition is just $\pi_p(\beta) = \alpha$ where π_p is the natural generalization of π_2. If we let $s(\beta)$ be the number of nonzero components of β then we get the total estimate

$$\sum_i \sigma_i \left(\sum_\alpha A_\alpha \sum_{\beta \mid \pi_p(\beta)=\alpha} \hat{B}_\beta^2 (1 - \epsilon(1 + \frac{1}{p-1}))^{s(\beta)} \right).$$

As before, since $\sum_\beta |B_\beta^2|=1$, we can discard all terms with $s(\beta) \geq 2\epsilon^{-1} \log(p-1)\delta^{-1}$ and $|A_\alpha| \leq \delta(4(p-1))^{-1}$ without affecting the sum by more than $\delta/2$. Now essentially the same strategies for the provers as used in the proof of Theorem 2.3 convince the verifier with a large probability and hence φ is satisfiable if v is sufficiently large. Finally, coding each number in the proof as variable we get a system of linear equations mod p we have proved Theorem 2.4. ∎

The main version of linear equations that is not included in the above theorems is the case when we have two variables in each equation. In the mod 2 case, this problem is a generalization of Max-Cut in that if we only allowed equations of the form $x_i + x_j = 1$ then it is exactly Max-Cut. Adding equations of the form $x_i + x_j = 0$ makes the problem more general, but it does not prevent the use of semidefinite programming (as in [11]) to get an approximation algorithm that performs as well as for Max-Cut. For lower bounds we have

Theorem 2.5 *For any $\epsilon > 0$ is is NP-hard to approximate E2-Lin-2 within a factor $12/11 - \epsilon$.*

Proof: This follows from a reduction from E3-Lin-2. This is done by making a local reduction through a gadget as defined in [17]. For a discussion on how to construct and use gadgets we refer to [17]. Here we just note that the existence of a 6-gadget implies the theorem. ∎

3 Satisfiability problems

We start with a direct consequence of Theorem 2.3.

Theorem 3.1 *For any $\epsilon > 0$ it is NP-hard to approximate E3-SAT within a factor $8/7 - \epsilon$.*

Proof: We take a direct reduction from E3-Lin-2. An equation $x+y+z = 0$ for three literals x, y and z is replaced by the clauses $(x \vee y \vee \bar{z})$, $(x \vee \bar{y} \vee z)$, $(\bar{x} \vee y \vee z)$, and $(\bar{x} \vee \bar{y} \vee \bar{z})$. An assignment that satisfies the linear equation satisfies all the clauses while an assignment that does not satisfy the linear equation satisfies 3 of the 4 equations. The theorem follows by a simple calculation. ∎

Note that in terms of the gadget language of [17] the proof is simply constructing a trivial 4-gadget.

Something that is not desirable in the above reduction is that we show that it is hard to distinguish the two cases of when we can satisfy a fraction $1 - \epsilon$ of the clauses and a fraction $7/8 + \epsilon$. We would prefer to have the first class be the set of satisfiable 3-SAT formulas. The proof of this is more complicated. As a warmup we first prove the desired approximation result maintaining perfect completeness for E4-SAT.

Theorem 3.2 *For any $\epsilon > 0$ it is NP-hard to distinguish satisfiable E4-SAT formulas from E4-SAT formulas for which only a fraction $15/16 + \epsilon$ of the clauses can be satisfied.*

Proof: The written proof is the same as form Theorem 2.3 but it is checked in a different way. We choose V and W as before and pick f and g_1 uniformly at random as functions on V and W respectively. Finally, g_2 and g_3 are determined pointwise by:

- If $g_1(y) = 1$ then $g_2(y)$ and $g_3(y)$ are chosen randomly to satisfy $g_2(y)g_3(y) = -f(x)$.
- If $g_1(y) = -1$ then $g_2(y)$ and $g_3(y)$ are chosen randomly and independently.

The test is then to check that $A(f) \vee B(g_1) \vee B(g_2) \vee B(g_3)$ (remember that -1 corresponds to true) is true. It is not hard to see that we get perfect completeness.

Let us calculate the probability that we accept in general. We have that

$$1 - \frac{1}{16}(1 + A(f))(1 + B(g_1))(1 + B(g_2))(1 + B(g_3))$$

is 1 if the test accepts and 0 otherwise. We need to estimate the expected value of this which gives the probability of success. We expand the product and estimate the expected value of each term separately. The only expected values that might be nonzero are the ones containing both $B(g_2)$ and $B(g_3)$. The reason for this is that both the collections (f, g_1, g_2) and (f, g_1, g_3) form independent random variables and the expected value of each term is 0. Thus we need consider the expected values of

1. $B(g_2)B(g_3)$.
2. $A(f)B(g_2)B(g_3)$.
3. $B(g_1)B(g_2)B(g_3)$.
4. $A(f)B(g_1)B(g_2)B(g_3)$.

We estimate each term by expanding the Fourier transform and switching the order of summation. The calculations are very similar in all the cases and let us start with the last term which contains all types of Fourier-terms to be considered. We get the expected value to be

$$\sum_{\alpha,\beta_1,\beta_2\beta_3} \hat{A}_\alpha \hat{B}_{\beta_1}\hat{B}_{\beta_2}\hat{B}_{\beta_3} E(\prod_{x\in\alpha} f(x) \prod_{y\in\beta_1} g_1(y) \prod_{y\in\beta_2} g_2(y) \prod_{y\in\beta_3} g_3(y))$$

Now, as before, any term with $\beta_2 \neq \beta_3$ has expected value 0. The parts of the product with different projections onto V are independent. Finally, if β_1 is not a subset of $\beta = \beta_2 = \beta_3$ the corresponding term also has expected value 0. Thus we need to estimate values of expressions of the form

$$\prod_{y\in\beta_1} g_1(y) \prod_{y\in\beta} g_2(y)g_3(y)$$

and

$$f(x) \prod_{y\in\beta_1} g_1(y) \prod_{y\in\beta} g_2(y)g_3(y)$$

where $\beta_1 \subseteq \beta$ and all elements of β project onto a fixed element x of V. When $g_1(y) = -1$ the expected value over the rest is 0 so the only part contributing to the expected value is when $g_1(y) = 1$ for all $y \in \beta$. This choice happens with probability $2^{-|\beta|}$ and then the first expression is equal to $(-f(x))^{|\beta|}$ while the second is equal to $f(x)(-f(x))^{|\beta|}$. This means that the expected value of the first type is $2^{-|\beta|}$ when $|\beta|$ is even and 0 otherwise. For the second case we get $-2^{-|\beta|}$ when $|\beta|$ is odd and 0 otherwise. This implies that we get the total estimate

$$\sum_\alpha |A_\alpha| \sum_{\beta,\pi_2(\beta)=\alpha} \hat{B}_\beta^2 2^{-|\beta|} | \sum_{\beta_1\subseteq\beta} \hat{B}_{\beta_1}|.$$

The inner sum is easily bounded using Cauchy-Schwartz inequality as

$$|\sum_{\beta_1\subseteq\beta} \hat{B}_{\beta_1}| \leq \left(\sum_{\beta_1\subseteq\beta} 1\right)^{1/2} \left(\sum_{\beta_1\subseteq\beta} \hat{B}_{\beta_1}^2\right)^{1/2} \leq 2^{|\beta|/2},$$

and substituting this we get the bound

$$\sum_\alpha |A_\alpha| \sum_{\beta,\pi_2(\beta)=\alpha} \hat{B}_\beta^2 2^{-|\beta|/2}.$$

which is a very familiar type of sum. Before continuing, let us consider the other expected values we need to estimate.

For $E(B(g_2)B(g_3))$ we only have the terms with $\alpha = \emptyset$ and since all β have odd size we get expected value 0 in this case. The same argument applies to $E(B(g_1)B(g_2)B(g_3))$. For $E(A(f)B(g_2)B(g_3))$ we get the same expression as for $E(A(f)B(g_1)B(g_2)B(g_3))$ without the summation over β_1 and thus in this case we get a stronger bound. Thus to have an acceptance probability greater than $15/16 + \epsilon$ we need that the expected value of

$$\sum_\alpha |A_\alpha| \sum_{\beta,\pi_2(\beta)=\alpha} \hat{B}_\beta^2 2^{-|\beta|/2}.$$

is at least $\epsilon/2$. By the standard argument we see that only α and β of size at most $4\log(1/\epsilon)$ matter and that we can disregard α with $|A_\alpha| \leq \epsilon/4$. Defining the same strategy as before for the provers we see that φ is satisfiable provided that v is sufficiently large.

Finally, interpreting each bit in the proof as a variable we get a 4SAT formula and we have proved Theorem 3.2. ∎

Remark 3.3 *Note that the proof can actually, virtually without change, prove inapproximability results for other optimization problems. Namely, for the constructed functions the quadruple $(f(x), g_1(y), g_2(y), g_3(y))$ never takes any of the values $(1,1,1,1), (1,1,-1,-1), (-1,1,1,-1), (-1,1,-1,1)$ and thus consider the optimization problem where we are given ordered quadruples of literals and for an assignment of the variables, a quadruple is satisfied if it takes none of the above stated values. We now want to satisfy as many of the quadruples as possible. This is a well defined approximation problem and we claim that the above proof shows that for any $\epsilon > 0$ it is NP-hard to distinguish formulas where you can satisfy all the quadruples and those where you can satisfy $3/4 + \epsilon$ of the quadruples.*

The argument applies to any predicate on 4 variables whose zero-set is a subset of the above set. The number $3/4 + \epsilon$ is replaced by $f/16 + \epsilon$ where f is the number of strings of length 4 satisfying the predicate. Thus we get examples with $12 \leq f \leq 15$.

Let us do the stronger for Max-E3-Sat with perfect completeness.

Theorem 3.4 *For any $\epsilon > 0$ it is NP-hard to distinguish satisfiable E3-SAT formulas from E3-SAT formulas for which only a fraction $7/8 + \epsilon$ of the clauses can be satisfied.*

Proof: While the overall structure of the proof is similar to the previous cases a number of complications arise. We first describe a test with a parameter $\epsilon < 1/2$

We choose V and W as before and then we first pick f randomly as a function on V. We then pick g_1 and g_2 by

- If $f(y) = 1$ then $g_1(y)$ and $g_2(y)$ are chosen randomly to satisfy $g_1(y)g_2(y) = -1$.
- If $f(y) = -1$ then $g_1(y)$ is picked at random usual but we also use an error function $\mu(y)$ which is 1 with probability $1 - \epsilon$ and -1 otherwise. We set $g_2(y) = \mu(y)g_1(y)$.

We call this distribution of functions D_ϵ. The test is to check that $A(f) \vee B(g_1) \vee B(g_2)$ is true. It is easy to see that we get perfect completeness.

To estimate the probability of success we need to estimate the expected value of

$$1 - \frac{1}{8}(1 + A(f))(1 + B(g_1))(1 + B(g_2)).$$

We expand the product and estimate the expected value of each term separately. The only expected values that might be nonzero are the ones containing both $B(g_1)$ and $B(g_2)$. Thus we need consider the expected values of

1. $B(g_1)B(g_2)$.
2. $A(f)B(g_1)B(g_2)$.

Both these estimates require some new ideas. For lack of space we only give the relevant lemmas for the first case and a more complete discussion for the second case.

Lemma 3.5 *When the functions are chosen according to the distribution D_ϵ then*

$$|E(B(g_1)B(g_2))| \leq 3\epsilon^{1/2} + \sum_{\beta \mid \epsilon^{-1/2} \leq |\beta| \leq \epsilon^{-4/c}} \hat{B}_\beta^2.$$

This is true for a fixed W but it is expected value over the choice of V contained in W as well as the choice of f, g_1 and g_2

A key lemma for proving this is:

Lemma 3.6 *For any fixed $\beta \subseteq W$,*

$$E(1/|\pi(\beta)|) \leq |\beta|^{-c}.$$

The probability is taken over picking a random V contained in W.

Let us now consider

$$E(A(f)B(g_1)B(g_2)) \tag{5}$$

in more detail.

Lemma 3.7 *If $|E(A(f)B(g_1)B(g_2)| \geq \delta$ where the expected value is over choosing f, g_1 and g_2 from D_ϵ, then, provided $v \geq C_{\delta,\epsilon}$, φ is satisfiable.*

Proof: Using our standard Fourier expansion we transform (5) to

$$\sum_{\alpha,\beta_1,\beta_2} \hat{A}_\alpha \hat{B}_{\beta_1} \hat{B}_{\beta_2} E\left(\prod_{x\in\alpha} f(x) \prod_{y\in\beta_1} g_1(y) \prod_{y\in\beta_2} g_2(y)\right).$$

As before unless $\beta_1 = \beta_2 = \beta$ and $\alpha \subseteq \pi(\beta)$ the expected value is 0 and thus we can drop those terms. Now fix a β and consider

$$\sum_{\alpha\subseteq\pi(\beta)} \hat{A}_\alpha E\left(\prod_{x\in\alpha} f(x) \prod_{y\in\beta} g_1(y)g_2(y)\right).$$

The inner expected value is

$$\prod_{x\in\alpha\cap\pi(\beta)} (\tfrac{1}{2}((-1)^{s_x} - (1-2\epsilon)^{s_x})) \prod_{x\in\pi(\beta)/\alpha} (\tfrac{1}{2}((-1)^{s_x} + (1-2\epsilon)^{s_x})),$$

where $s_x = |\pi^{-1}(x) \cap \beta|$. Let us denote the value by $h(\alpha,\beta)$. Thus we have the estimate

$$\sum_{\alpha\subseteq\pi(\beta)} A_\alpha h(\alpha,\beta) \leq \left(\sum_{\alpha\subseteq\pi(\beta)} A_\alpha^2\right)^{1/2} \left(\sum_{\alpha\subseteq\pi(\beta)} h^2(\beta,\alpha)\right)^{1/2}$$

$$\leq \left(\sum_{\alpha\subseteq\pi(\beta)} h^2(\beta,\alpha)\right)^{1/2}.$$

It is not difficult to see that the last sum equals

$$\prod_{x\in\pi(\beta)} ((\tfrac{1}{2}((-1)^{s_x} - (1-2\epsilon)^{s_x}))^2 + (\tfrac{1}{2}((-1)^{s_x} + (1-2\epsilon)^{s_x}))^2)$$

which can be bounded by $(1-\epsilon)^{|\pi(\beta)|}$ since each term is of the form $a^2 + b^2$ where $|a| + |b| = 1$ and $\max(|a|,|b|) \leq 1-\epsilon$ and such terms are bounded by $(1-\epsilon)$ for $\epsilon < 1/2$. Thus if fact we have the upper estimate

$$\sum_\beta \hat{B}_\beta^2 (1-\epsilon)^{|\pi(\beta)|/2}$$

for (5). We are thus assuming that this expected value of this expression is at least δ, and we want to find a good strategy for the provers.

Consider the sum over all $|\beta|$ of size at least $l = (\epsilon\delta)^{-2/c}$. For each such term, by Lemma 3.6 the probability that $|\pi(\beta)| \leq (\epsilon\delta)^{-1}$ is at most $\epsilon\delta$. Thus the expected value of that part of the sum is bounded by $\delta/4$. Thus we know that

$$\left| E\left(\sum_{\alpha,\beta \mid |\beta|\leq l, \alpha\subseteq\pi(\beta)} \hat{A}_\alpha \hat{B}_\beta^2\right)\right| \geq 3\delta/4.$$

If we discard all α with $|\hat{A}_\alpha| \leq 2^{-l}\delta/4$ the number multiplying B_β^2 drops by at most $\delta/4$ and thus the value of the sum changes by at most this number.

We are not in the usual position for defining a good strategy for the provers completing the proof. The success rate of this strategy is at least $\Omega(l^{-2}2^{-l}\delta^2)$. We omit the details. ∎

We are now in position to prove Theorem 3.4. We describe the appropriate test. Given a $\delta > 0$ proceed as follows.

TEST 3S

1. Set $t = \lceil \delta^{-1} \rceil$, $\epsilon_1 = \delta^2$ and $\epsilon_i = \epsilon_{i-1}^{8/c}$ for $i = 2, 3, \ldots t$. Choose v such that $v \geq C_{\delta,\epsilon_i}$ for $1 \leq i \leq t$ where C_{δ,ϵ_i} is the constant of Lemma 3.7.
2. Choose a random j, $1 \leq j \leq t$ with uniform distribution. The choose f, g_1, g_2 according to the distribution D_{ϵ_j} and perform the given test.

Since δ was arbitrary, Lemma 3.8 completes the proof of Theorem 3.4 ∎

Lemma 3.8 *If the test 3S accepts with probability $(7+5\delta)/8$ then φ is satisfiable.*

Proof: The probability of accepting is

$$E\left(1 - \frac{1}{8}(1+A(f))(1+B(g_1))(1+B(g_2))\right) =$$

$$7/8 - \frac{1}{8}E(B(g_1)B(g_2)) - \frac{1}{8}E(A(f)B(g_1)B(g_2))$$

where f, g_1 and g_2 are chosen as described in the test. By Lemma 3.7 if φ is not satisfiable then the last term is bounded by $\delta/8$ in absolute value. On the other hand by Lemma 3.5 we have

$$|E(B(g_1)B(g_2))| \leq \frac{1}{t}\sum_{i=1}^{t} (3\epsilon_i^{1/2} + \sum_{\beta \mid \epsilon_i^{-1/2} \leq |\beta| \leq \epsilon_i^{-4/c}} \hat{B}_\beta^2) \leq$$

$$3\epsilon_1^{1/2} + \frac{1}{t} \leq 4\delta$$

since the intervals of summations are disjoint. The lemma follows. ∎

Remark 3.9 *Note also here that can get a similar theorem for another predicate. The triples $(f(y), g_1(y), g_2(y))$ never take the values $(1,1,1)$ or $(1,-1,-1)$. Thus we can consider a collection of triples of literals where we want to find an assignment that for as many triples as possible avoids giving these values. This problem is NP-hard to approximate within $4/3-\epsilon$ for any $\epsilon > 0$, even assuming perfect completeness.*

It is not hard to extend the result to longer clauses.

Theorem 3.10 *For any $\epsilon > 0$ and any $k \geq 3$ it is NP-hard to distinguish satisfiable Ek-SAT formulas from Ek-SAT formulas for which only a fraction $1-2^{-k}+\epsilon$ of the clauses can be satisfied.*

Proof: Follows by induction over k. Change a clause C_i to the two clauses $C_i \vee z$ and $C_i \vee \bar{z}$ for a new variable z. If the number of clauses is N and the the optimal number of clauses that can be satisfied is O for the original formula, this creates an instance with $2N$ clauses and optimal value $N+O$. ∎

In fact, we can do a little bit better. By transforming clauses of length 3 to clauses of different sizes we get.

Theorem 3.11 *Let p_i, $i \geq 3$ be number such that $\sum_i p_i = 1$ and consider CNF-formulas where a fraction p_i of the clauses are of length i. For any $\epsilon > 0$ it is NP-hard to distinguish satisfiable formulas of this type and those for which only a fraction $\sum_i p_i(1-2^{-i})+\epsilon$ of the clauses can be satisfied.*

It is easy to see that also this result is tight except for the presence of ϵ.

We also get a result for E2-SAT, but we only know how to do this through a reduction.

Theorem 3.12 *For any $\epsilon > 0$ it is NP-hard to approximate E2-SAT within a factor $22/21-\epsilon$.*

Proof: This follows by a reduction from E3-Lin-2. Just use the 11-gadget of [5, 17]. ∎

4 Results for other problems

We have, following [5]:

Theorem 4.1 *For any $\epsilon > 0$ it is NP-hard to approximate vertex cover with $7/6-\epsilon$.*

Proof: Consider the proof system used for E3-Lin-2. Think of this as asking about f, then g_1 and then assuming that $B(g_2) = B(g_1)A(f)$ and rejecting if this is not the case. This is a proof system with 2 free bits and acceptance probability $(1-\epsilon)$ in the good case and at most $\frac{1}{2}+\delta$ in the bad case. Thus the sizes of the corresponding cliques are $n/4(1-\epsilon)$ and $\leq n/4(\frac{1}{2}+\delta)$. The size of the vertex covers obtained through the natural reduction are then roughly $3n/4$ and $7n/8$. The result follows. ∎

We get a couple of other results reducing from E3-Lin-2 and using the gadgets of [17].

Theorem 4.2 *For any $\epsilon > 0$ it is NP-hard to approximate undirected max-cut within a factor $17/16-\epsilon$.*

Proof: Use the 8-gadget for $a+b+c=0$ and the 9-gadget for $a+b+c=1$. If there are more equations of the second type simply complement all the variables. ∎

Theorem 4.3 *For any $\epsilon > 0$ it is NP-hard to approximate max-di-cut within $13/12-\epsilon$.*

Proof: There is a 6.5-gadget [17]. ∎

5 Gap between bounds

The results for linear equations and the k-SAT for $k \geq 3$ are tight (upto the existence of an arbitrary ϵ). For the other problems, to the best of our knowledge the gaps between the bounds are summarized below. The upper bounds are from [11, 8, 3].

	Appr. Upper	Appr. Lower
E2-SAT	1.0741	1.0476
Max-Cut	1.1383	1.0624
E2-LIN-2	1.1383	1.0909
Max-di-cut	1.164	1.0833
Vertex cover	2	1.1666

Thus a fair amount of work remains. Our conjecture would be that progress is closest at hand for vertex cover. It is also not clear what should be attacked first, the upper or the lower bounds. For the lower bounds, the optimality of all gadgets are established in [17] and thus something more essential has to be changed. For the upper bounds on the other hand, for some of the problems the strongest formulations of the semi-definite programs there are no known examples that show that the analysis is optimal. Thus already the existing algorithms might give better constants. It seems hard at this time to have any guess what the correct constants should be.

6 Getting nonconstant ϵ

There is nothing that prevents us from plugging in ϵ and δ that are non-constant while doing the proof of Theorem 2.3. To get a contradiction in the end we just need that $acc(v)$ is smaller than the accept probability for the verifier with the given strategies for the provers. The latter is $\Omega((\delta\epsilon \log 1/\delta)^2)$. Since by [16] we know that $acc(v) \leq c^v$ for some constant $c < 1$ we need $v \geq \Omega(\log(1/(\delta\epsilon)))$. The problem with this is that the number of variables in the final linear systems is around

$$m^v 2^{2^{3v}}$$

since the is the size of the long codes of all the possible W. Thus having nonconstant ϵ and δ produces a system with a nonpolynomial number of variables. However, if we are willing to assume a stronger hypothesis than NP$\neq$ P we can obtain something. Setting $v = c \log\log n$ we get:

Theorem 6.1 *Assume NP$\subseteq$ DTIME($2^{O(\log n \log\log n)}$). Then, there is a constant $c' > 0$ such for $\epsilon = (\log n)^{-c'}$, E3-Lin-2 cannot be approximated within $2-\epsilon$ in polynomial time.*

The proof of Theorem 3.2 has as good constants at that of Theorem 2.3 and hence we have

Theorem 6.2 *Assume NP$\subseteq$ DTIME($2^{O(\log n \log\log n)}$). Then, there is a constant $c' > 0$ such for $\epsilon = (\log n)^{-c'}$, satisfiable E4-SAT formulas cannot be distinguished from those where only a fraction $15/16+\epsilon$ of the clauses can be satisfied in polynomial time.*

The situation for Theorem 3.4 is, however, much worse since constants blow up to a much larger extent. Let us make a quick review using the notation of that proof.

The success probability of the final strategy of the provers is about $2^{-\Theta(l)}$ where $l = Poly((\epsilon\delta)^{-1})$. In the final test we might have $\epsilon = \epsilon_t$ where $t = \lceil \delta^{-1} \rceil$ and $\epsilon_t = \delta^{2(8/c)^{t-1}}$. Hence the final strategy has success rate

$$2^{-2^{2^{\theta(\delta^{-1})}}}$$

and this should be greater than $acc(v)$ to get a contradiction. Setting $v = \log\log n$ we can use $\delta^{-1} = \Theta(\log\log\log\log n)$.

Theorem 6.3 *Assume NP$\subseteq$ DTIME($2^{O(\log n \log\log n)}$). There is a constant $c' > 0$ such for $\epsilon^{-1} = c' \log\log\log\log n$, satisfiable E3-SAT formulas cannot be distinguished from those where only a fraction $7/8 + \epsilon$ of the clauses can be satisfied in polynomial time.*

7 Concluding remarks

The technique of using Fourier transforms to analyze PCPs seems very strong (see also [12]). It seems like the long code mixes well with Fourier transforms. The main components that make the proof work is that it is easy to control the appropriate test when the supposed long codes are exclusive-ors of long codes and some property that makes the different Fourier-coefficients independent. The latter is the more complicated property and in particular this seems to be a problem when analyzing adaptive tests, i.e. tests where future questions depends on answers given. It is our hope, however, that more results will be obtained using the same approach as this paper. There are many MAX-SNP optimization problems for which one would like to know the correct constant of approximation and in many cases it might be hard to obtain sharp results by reduction. In those cases special purpose PCP's would have to be constructed and analyzed.

As a particular instance one could consider the case of linear equations over more general structures than considered in Section 2. It is our belief that the results should extend to many domains and this will be investigated in the final version of this paper.

Acknowledgements: I am most grateful for discussions on these subjects with Oded Goldreich and Madhu Sudan. They have also made numerous comments improving the quality of this writeup. Muli Safra and Sajeev Arora made several helpful comments which simplified the proof. Gunnar Andersson and Lars Engebretsen gave many comments on a previous version of this manuscript. I am also grateful to Greg Sorkin for constructing the gadget for linear equations with two variables in each equation and to David Karger for coming up with the problem of Sat-problems of mixed sizes.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof verification and intractability of approximation problems. 33rd FOCS, 1992, pp 14-23.

[2] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. 33rd FOCS, 1992 2-13.

[3] R. Bar-Yehuda and S. Even A linear time approximation algorithm for the weighted vertex cover algorithm, Journal of Algorithms, 1991, Vol 2, pp 198-210.

[4] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi, and M. Sudan, Linearity testing in characteristic two, IEEE Transactions on Information, Vol 42, No 6, November 1996, pp 1781-1796.

[5] M. Bellare, O. Goldreich and M. Sudan Free Bits, PCPs and Non-Approximability—Towards tight Results. 36th FOCS, 1995, pp 422-431. See also a more complete version available from ECCC.

[6] M. Bellare, S. Goldwasser, C. Lund and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. 25th STOC, 1993, pp 294-304. (See also Errata sheet in STOC94).

[7] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson Multi-prover interactive proofs: How to remove intractability assumptions. 20th STOC, 1988, pp 113-131.

[8] U. Feige and M. Goemans Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT, 3rd ISTCS, 1995, pp 182-189.

[9] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. Journal of the ACM, 1996, Vol 43:2, pp 268-292.

[10] M.R. Garey and D.S. Johnsson, Computers and Intractability, W.H. Freeman and Company, 1979.

[11] M. Goemans and D. Williamson .878-approximation algorithms for Max-Cut and Max-2-SAT, 25th STOC, 1994, pp 422-431.

[12] J. Håstad Clique is hard to approximate within $n^{1-\epsilon}$, 37th FOCS, 1996, pp 627-636. There is also an updated version with the same name and a simpler proof.

[13] D.S. Johnsson Approximation algorithms for combinatorial problems, J. Computer and System Sciences, 1974, Vol 9, pp 256-278.

[14] C. Papadimitriou and M. Yannakakis Optimization, approximation and complexity classes. Journal of Computer and System Sciences, Vol 43, 1991, pp 425-440.

[15] E. Petrank The hardness of approximation, 2nd ISTCS, 1993, pp 275-284.

[16] R. Raz A parallel repetition theorem, 27th STOC, 1995, pp 447-456.

[17] L. Trevisan, G. Sorkin, M. Sudan, and D. Williamson Gadgets, approximation and linear programming, 37th FOCS, 1996, pp 617-626.

[18] O. Verbitsky Towards the parallel repetition conjecture, Theoretical Computer Science, Vol 157, 1996, pp 277-282.