Some optimal inapproximability results

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

September 4, 1998

Abstract

We prove optimal, up to an arbitrary $\epsilon > 0$, inapproximability results for Max-Ek-Sat[1] for $k \geq 3$, maximizing the number of satisfied linear equations in an overdetermined system of linear equations modulo a prime p and Set Splitting. As a consequence of these results we get improved lower bounds for the efficient approximability of many optimization 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 variables. For simplicity, let us assume that each clause contains exactly

[1] Max-Ek-Sat is the variant of CNF-Sat where each clause is of length exactly k.

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 [17] and one of the main results of this paper is that this is optimal to within an arbitrary 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 over-determined 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 over an arbitrary Abelian group G and set splitting of sets of size 4. 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 [13], but really strong results were not obtained until the connection with multiprover interactive proofs, introduced for a different reason by [9], was discovered in the seminal paper of Feige et al. [11] in 1990. There are a number of variants of the multiprover interactive proofs and the two proof models that we need in this paper are that of two-prover interactive proofs and that of probabilistically checkable proofs.

The first model was introduced in the original paper [9] and here the verifier interacts with two provers who cannot communicate with each other. Probabilistically checkable proofs, which we from here on abbreviate PCPs, correspond to oracle proof systems of [12], but where introduced and studied in a way relevant to this paper in [2]. In a PCP the verifier does random 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] where it was established that NP has a PCP where the verifier reads only a constant number of bits and uses 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 [7] and based on slightly stronger hypothesis they achieved the constant 94/93. Bellare and Sudan [8] improved this to 66/65 and the strongest result prior to our results here are by Bellare, Goldreich and Sudan [6] obtaining the bound 27/26. This paper [6] also studied the problem of linear equations mod 2 and obtained inapproximability constant 8/7.

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. The verifier V chooses a clause at random and then randomly a variable appearing in that clause and asks one prover, P_1, for the values of all variables in the clause while the other prover, P_2, is asked for chosen variable. V 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. The first property is that the total size of the answers of the provers is bounded by the constant and the second property is that the answers by P_2 is simply used to check the answer by P_1. More formally, we need that the V acceptance criteria is that the bit(s) sent by P_2 is equal to some specific bit(s) sent by P_1 together with some condition that only depends on the answer sent by P_1.

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. We use the strong results by Raz [22], saying that, if we run u copies in parallel, the error probability decreases as c^u 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, a great discovery of [6], of an input is simply $x \in \{0,1\}^u$ is a string of length 2^{2^u}. The coordinates correspond to all possible functions $f : \{0,1\}^u \mapsto \{0,1\}$ and the coordinate corresponding to f takes the value $f(x)$. It is a very wasteful encoding but in our case u is a constant and hence 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[2] verifier. Thus we once more display the power of proof composition introduced in [2] although in an unusually simple form. Also note that the idea of using a multi-prover proof system and coding the answers is not novel to this paper and was used in more or less explicit form in [1, 7, 8, 6]. The power of this approach was increased and the proof complexity decreased by the wonderful parallel repetition theorem [22] of Raz.

Our main results are obtained by designing a PCP where the verifier reads bits, each random but correlated, of the supposed long codes of the answers of the provers in the two-prover protocols, 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 only 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 [6], that to get strong inapproximability results one needs to design a PCP with the intended application in mind.

The result that, for any $\epsilon > 0$ it is NP-hard to approximate Max-E3-Sat within $8/7 - \epsilon$ follows in a straightforward way 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 [21] we want the correct gap location which in this case means that it is NP-hard to distinguish the set of satisfiable formulas from the set of formulas for which it is only possible to satisfy a a fraction $7/8 + \epsilon$ of the clauses. We establish this by a separate, more complicated proof.

An outline of the paper is as follows. In Section 2 we introduce notation, give definitions and state some needed results from earlier papers. Most of your PCPs use the same written proof and in Section 3 we describe this proof. In Section 4 we give the results for linear equations. It is the simplest result to prove and it gives most of our corollaries. In Section 5 we give the results on Max-k-Sat and in Section 6 we give corresponding results for Set Splitting. We obtain some results for other problems in Section 7. We finally briefly discuss how to make our arbitrary constants be functions of the inputs length in Section 8 and end by some concluding remarks.

This is the complete version of the results announced in [16].

2 Notation and some essential previous results

In this section we give basic notation and collect the needed result from earlier papers.

[2]The concept of inner verifier was introduced by [6] and we refer the interested reader to that paper since we do not need to formalize this concept in the current paper

2.1 Basic notation

All logarithms in this paper are to the base 2. We use absolute value signs to denote the absolute value of a real or complex number. The notation $size(x)$ is used to denote the size of x in some obvious measure. If x is a bit-string then $size(x)$ is its length while if it is a set it is the number of elements. We use the notation $x\Delta y$ for two sets x and y to denote the symmetric difference, i.e. the elements that appear in exactly one of the sets x and y.

In sums and products we always indicate the variable over which the sum/product is taken. Sometimes, however, we do not explicitly give the range. This happens when this ranges is considered obvious and it is usually the case that we are summing over all objects of the given kind. An empty sum is taken to be 0 and and empty product takes the value 1. The expected value of a variable X is denote by $E_f[X]$ assuming we are taking the expected value over a random f. We do not give the distribution of this f which is supposed to be clear from the context.

For most of the paper we work with binary valued objects, but for a number of reasons it is more convenient for us to work over $\{-1,1\}$ rather than the standard $\{0,1\}$. We let -1 correspond to true and our most important Boolean operation is exclusive-or which is then simply ordinary multiplication. We also need other Boolean operations like $\wedge$ which is defined in the usual way using true and false and the fact that -1 is short for "true" and 1 is short for "false". Thus in particular $-1 \wedge 1 = 1$ and $-1 \wedge -1 = -1$.

We do not distinguish a set of variables and the set of indices of these variables. For a set U of variables we let $\{-1,1\}^U$ be set of all possible assignments to these variables and we use $\{-1,1\}^n$ instead of $\{-1,1\}^{[n]}$. Suppose $U \subseteq W$, then for $x \in \{-1,1\}^W$ we denote its restriction to the variables in U by $x|_U$ which is then en element of $\{-1,1\}^U$. This is essentially projection and we need this also to operate on set. For a set $\alpha \subseteq \{-1,1\}^W$ we define $\pi^U(\alpha)$ by letting $x \in \{-1,1\}^U$ belong to $\pi^U(\alpha)$ if $x = y|_U$ for some $y \in \alpha$.

We also need more general projections and in particular we need a mod p-projection of multisets. We view a multiset, α, with elements in $\{-1,1\}^U$ as a mapping $\{-1,1\}^U \mapsto \mathrm{Z}$ where $\alpha(y)$ is the number of occurrences of y. It is a mod p-multiset if we only care about $\alpha(y)$ as a member of Z_p. We have a projection of such multisets by letting $\pi_p^U(\alpha)$ be the multiset that contains the element x

$$\sum_{y|y|_U=x} \alpha(y) \tag{1}$$

times where the sum is computed mod p. We do not really distinguish a mod 2 multiset and an ordinary set and thus we consider π_2^U to be defined also on sets. Even more generally we allow "multisets" $\{-1,1\}^U \mapsto G$ for an Abelian group G. By letting the summation in (1) be in G we get a projection π_G^U.

For any set U we let $\mathcal{F}_U$ be the set of all functions $f : \{-1,1\}^U \mapsto \{-1,1\}$. A central point in this paper is to study functions $A : \mathcal{F}_U \mapsto \{-1,1\}$. One particular type of such functions is given by the long codes of assignments.

Definition 2.1 *[6] The* long code *of an assignment $x \in \{-1,1\}^U$ is a mapping $A_x : \mathcal{F}_U \mapsto \{-1,1\}$ where $A_x(f) = f(x)$.*

We also identify a function with it set of values and thus writing down a long code simply means writing down a string of length $2^{2^{size(U)}}$ where we use an arbitrary, but fixed convention to order the elements of $\mathcal{F}_U$. We also need an extension of the long code to a more general situation.

Let G be an Abelian group and thus G can be represented as a direct product of cyclic groups, $G = C_{i_1} \times C_{i_2} \times C_{i_3} \dots C_{i_k}$. The number of elements, $size(G)$, of G is $\prod_{l=1}^{k} i_l$. We represent a cyclic group C_i as the i'th roots of unity and an element g of G is thus a k-tuple of complex numbers and the group operation is coordinate-wise multiplication. Finally, let $\mathcal{F}_U^G$ be the set of functions $f : \{-1,1\}^U \mapsto G$. Then

Definition 2.2 *The* long G code *of an assignment $x \in \{-1,1\}^U$ is a mapping $A_x : \mathcal{F}_U^G \mapsto G$ where $A_\alpha(f) = f(x)$.*

Note that the standard long code corresponds to G being Z_2. We also write long p code rather than long Z_p code.

A CNF-formula is a formula φ of n Boolean variables $(x_i)_{i=1}^n$ given by m clauses $(C_j)_{j=1}^m$. A standard clause contains a number of literals, i.e. variables or their negations, and it is true if at least one of the literals is true. The number of literals in a clause is the length of clause. We also, for counting purposes, allow non-standard clauses, which we denote by $T(x,y,z)$, which are always true independent of the values of the variables in T.

Definition 2.3 *A CNF-formula φ with m clauses is e-satisfiable, iff the assignment that satisfies the largest number of clauses satisfies em clauses.*

Using the obvious extension of this we say that φ is at most e-satisfiable if it is d-satisfiable for some $d \leq e$.

2.2 Problems considered

Let us give formal definitions of the problems we consider in this paper.

Definition 2.4 *A CNF-formula is a Ek-CNF-formula iff each clause is of length exactly k.*

We have a natural problem corresponding to Ek-CNF formulas.

Definition 2.5 *Max-Ek-Sat is the optimization problem of given a Ek-CNF formula φ, find the maximal number of clauses of φ that can be satisfied by any assignment x.*

We also want to study the problem of satisfying the maximal number of linear equations in an Abelian group G.

Definition 2.6 *Max-Ek-Lin-G is the problem of given a system of m linear equations over an Abelian group G, with exactly k variables in each equation, find the maximal number of equations that can be satisfied by any assignment x.*

Most of our interest centers around the case when G is Z_p for some prime p. For simplicity we write this problem as Max-Ek-Lin-p. The simplest case is $p = 2$ and this is the first problem we study in Section 4 below.

We also are interested in some standard optimization problems and we state them here for definiteness.

Definition 2.7 *Max-Cut is the problem of given an undirected graph G with vertices V to find a partition V_1, V_2 of V such that the number of edges (u, v) such that $u \in V_1$ and $v \in V_2$ or $v \in V_1$ and $u \in V_2$ is maximized.*

Definition 2.8 *Max-di-Cut is the problem of given a directed graph G with vertices V to find a partition V_1, V_2 of V such that the number of directed edges (u, v) such that $u \in V_1$ and $v \in V_2$ is maximized.*

Definition 2.9 *Vertex Cover is the problem of given an undirected graph G with edges E and vertices V to find a subset V_1 of the vertices of minimal size such that for each $e \in E$ one of its endpoints is contained in V_1.*

Definition 2.10 *Ek-Set Splitting. Given a ground set V and a number of sets $S_i \subset V$ each of size exactly k. Find a partition V_1, V_2 of V to maximize the number of S_i which contain at least one element from each of V_1 and V_2.*

Note that E2-Set Splitting is exactly Max-cut and that E3-Set Splitting is in fact equivalent to E2-Set Splitting since the set (x, y, z) is split exactly when two of the three pairs (x, y), (x, z) and (y, z) are split. Thus the first new problem is E4-Set Splitting.

Note that for each of the above problems we could think of both finding the numerical answer (e.g. the size of a certain cut) or the object that gives this answer (e.g. the partition giving the numerical answer). We are in general only concerned with the former problem.

Finally we define what it means to C-approximate an optimization problem.

Definition 2.11 *Let O be a maximization problem. For an instance x of O let $OPT(x)$ be the optimal value. A C-approximation algorithm is an algorithm that on any input x outputs a number V such that $OPT(x)/C \leq V \leq OPT(x)$.*

Definition 2.12 *Let O be a minimization problem. For an instance x of O let $OPT(x)$ be the optimal value. A C-approximation algorithm is an algorithm that on any input x outputs a number V such that $OPT(x) \leq V \leq C \cdot OPT(x)$.*

Definition 2.13 *An efficient C-approximation algorithm is a C-approximation algorithm that runs in worst case polynomial time.*

2.3 Proof systems

We define proofs systems by the properties of the verifier. This is reflected in the below definition but also in later descriptions of proof systems which are defined by the properties of the verifier.

The verifier needs help to verify a statement and we allow a verifier to have access to one or more oracles. These oracles can, in a different vocabulary, be thought of as provers or as written proofs. For a written proof, reading the i'th bit simply corresponds to asking the oracle the question "i ?".

Definition 2.14 *A probabilistic polynomial time Turing machine V is a verifier in a* Probabilistically Checkable Proof (PCP) *with soundness s and completeness c for a language L iff*

- *For $x \in L$ there is an oracle π such that V^π outputs 1 on input x with probability at least c.*
- *For $x \notin L$, for all π the probability that V^π outputs 1 on input x is bounded by s.*

Each question to an oracle π is answered by one bit.

We use the statement "V accepts" as a rewording of the fact that V outputs 1.

We are interested in a number of properties of the verifier and one property that is crucial to us is that V does not use too much randomness.

Definition 2.15 *The verifier V uses* logarithmic randomness *if on each input x and proof π, V^π flips $O(\log |x|)$ random coins.*

This makes the total number of possible sets of coin flips for V polynomial and hence we can go over all such coin flips and still remain polynomial time.

We also care about the number of bits V reads from the proof.

Definition 2.16 *The verifier V reads c bits in a PCP if for each outcome of its random coins and each proof π, V^π asks at most c questions to the oracle. If c is independent of the size of the input x we say that V reads a constant number of bits.*

The surprising power of interactive proofs were first established in the case of one prover [19], [23] and then for many provers [3]. After the fundamental connection with approximation was discovered [11] the parameters of the proofs improved culminating in the following result [2, 1].

Theorem 2.17 *[1] There is a universal integer c such that any language in NP has an PCP with soundness $1/2$ and completeness 1 where V uses logarithmic randomness and reads at most c bits of the proof.*

The soundness can be improved by repeating the protocol a constant number of times. The number of bits can be reduced to 3 but this pushes the soundness towards 1, although it remains a constant below one. Properties described by reading 3 bits of the proof can be coded by a 3-CNF formula (i.e. a CNF-formula with clauses of length at most 3). The acceptance probability of a proof is then closely related to the number of clauses satisfied by the corresponding assignment and we have.

Theorem 2.18 *[1] Let L be a language in NP and x be a string. There is a universal constant $c < 1$ such that, we can in time polynomial in $size(x)$ construct a 3-CNF formula $\varphi_{x,L}$ such that if $x \in L$ then $\varphi_{x,L}$ is satisfiable while if $x \notin L$, $\varphi_{x,L}$ is at most c-satisfiable.*

For later convenience we want to massage φ obtained in this formula to make it look more uniform. A key property we want is that each variable appears in the same number of clauses and all clauses are of length exactly 3. These properties ensures that choosing a random clause and a random variable in this clause is the same as choosing a random variable and then a random clause containing this variable. This property makes arguments simpler in many cases.

By [20] we can apply an additional transformation such that each variable (counted both in positive and negative form) in the resulting φ appears at most 6 times. As stated above we also want make sure that each clause is of length exactly 3 and that each variable appears in exactly the same number of clauses. The first property is achieved by replacing a clause $(l_1 \vee l_2)$ of length 2 by two clauses $(l_1 \vee l_2 \vee z)$ and $(l_1 \vee l_2 \vee \bar{z})$ for a new variable z. We can assume that there are no clauses of length one since variables in such clauses can be assigned the only truth value satisfying the clause. This transformation at most doubles the number of occurrences of each variable and thus each variable now appears at most 12 times. The second property is achieved by adding non-standard clauses $T(x, y, z)$ for variables x, y and z. These are , as stated in Section 2.1, always true so they do not change the properties of φ very much but it simplifies later arguments as discussed above. The two transformations taken together change the value of c in Theorem 2.18 but the resulting formula remains at most c-satisfiable for some $c < 1$. We state this as a formal theorem.

Theorem 2.19 *[1] Let L be a language in NP and x be a string. There is a universal constant $c < 1$ such that, we can in time polynomial in $size(x)$ construct a E3-CNF formula $\varphi_{x,L}$ such that if $x \in L$ then $\varphi_{x,L}$ is satisfiable while if $x \notin L$, $\varphi_{x,L}$ is at most c-satisfiable. Furthermore, we can assume that each variable appears exactly 12 times.*

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

Definition 2.20 *A probabilistic polynomial time Turing machine V with two oracles is a verifier in a* two-prover one-round proof system *with soundness s and completeness c for a language L if on input x it produces, without interacting with its oracles, two strings q_1 and q_2, such that*

- *For $x \in L$ there are two oracles P_1 and P_2 such that when the queries q_1 and q_2 are answered according to P_1 and P_2 the probability that V accepts x is at least c.*
- *For $x \notin L$, for any two oracles P_1 and P_2 the probability that V accepts x given that q_1 and q_2 are answered according to P_1 and P_2 is bounded by s.*

The questions q_1 and q_2 are in both cases the only question V asks the oracles.

We think of P_1 and P_2 of two actual dynamic provers rather than written proofs. They are infinitely powerful and are cooperating. They can make any agreement before the interaction with V starts but then they cannot communicate during the run of the protocol. Thus it makes sense to ask P_1 and P_2 for the same information in different contexts.

In the case of two-prover protocols we only consider the case of perfect completeness, i.e. $c = 1$ in the above definition. Given such a one-round protocol with soundness s we can repeat it twice in sequence improving the soundness to s^2. Similarly repeating the protocol k times in sequence gives soundness s^k. This creates many round protocols and we need our protocols to remain one-round. This can be done by what has become known as parallel repetition and this simply means that V repeats his random choices to choose k pairs of questions $(q_1^{(i)}, q_2^{(i)})_{i=1}^k$ and sends $(q_1^{(i)})_{i=1}^k$ to P_1 and $(q_2^{(i)})_{i=1}^k$ to P_2, all at once. V then receives k answers from each prover and accepts if it would have accepted in all k protocols given each individual answer. The soundness of such a protocol can be greater than s^k, but when the answer size is small, Raz [22] proved that soundness is exponentially decreasing with k.

Theorem 2.21 *[22] For all integers d and $s < 1$, there exists $c_{d,s} < 1$ such that given a two-prover one-round proof system with soundness s and answer sizes bounded by d, then for all integers k the soundness of k protocols run in parallel is bounded by $c_{d,s}^k$.*

Since we not limit the answer size of the provers they can of course misbehave by sending long answers which always causes V to reject. Thus, by answer size, we mean the maximal answer size in any interaction where V accepts.

2.4 Fourier Transforms

Our proofs depend heavily and Fourier analysis of functions $A : \mathcal{F}_U \mapsto \mathrm{R}$ where R is the set of real numbers. We recall some basic facts. For notational convenience let u denote $size(U)$. The set of basis functions used to define the Fourier

transforms are $\chi_\alpha(f) = \prod_{x\in\alpha} f(x)$ where $\alpha \subseteq \{-1,1\}^U$. The inner products of two functions A and B is given by

$$(A,B) = 2^{-2^u} \sum_{f\in\mathcal{F}_U} A(f)B(f).$$

Under this inner product the basis functions form a complete orthonormal system of functions and the Fourier coefficients of A are defined as the inner products with the basis functions χ_α. In other words, for each $\alpha \subseteq \{-1,1\}^U$

$$\hat{A}_\alpha = (A,\chi_\alpha) = 2^{-2^u} \sum_f A(f) \prod_{x\in\alpha} f(x).$$

We also have the Fourier inversion formula

$$A(f) = \sum_{\alpha\subseteq\{0,1\}^U} \hat{A}_\alpha \chi_\alpha(f) = \sum_{\alpha\subseteq\{0,1\}^U} \hat{A}_\alpha \prod_{x\in\alpha} f(x). \tag{2}$$

The Fourier coefficients are real numbers and we have Parseval's identity

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

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

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

and

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

Pattern matching tells us that the main difference is that $\{-1,1\}^U$ takes the place of $[n]$. Normal n bit strings can be thought of as mappings from $[n]$ to $\{-1,1\}$ while our basic unit now is mappings from $\{-1,1\}^U$ to $\{-1,1\}$ and thus nothing essentially new has happened.

As a very simple example let us just consider the Fourier transform when A is the long code of an input x_0. We simply observe that by definition the basis function $\chi_{\{x_0\}}$ is this very long codes. Thus the Fourier transform satisfies $\hat{A}_{\{x_0\}} = 1$ while all the other Fourier coefficients are 0. In general, the coefficient $A_{\{x_0\}}$ is a measure of the correlation of A with the long code for x_0 since the probability that $A(f) = \chi_{\{x_0\}}(f)$ is $(1 + \hat{A}_{\{x_0\}})/2$ and we say that A is close to a long code for x_0 is $\hat{A}_{\{x_0\}}$ is large.

In general a basis function χ_α is the exclusive-or of $\chi_{\{x\}}$ of all $x \in \alpha$ and hence it is an exclusive-or of long codes. Again we can think of the coefficient $\hat{A}_\alpha$ as the correlation between A and this exclusive-or of long codes and we may speak of the property that A is close to an exclusive or of long codes.

The word "close" has to be interpreted in the correct sense. Since $\sum_\alpha \hat{A}_\alpha^2 = 1$ and we have 2^{2^u} different α in this sum we would expect a random $\hat{A}_\alpha$ to have absolute value about $2^{-2^{u-1}}$. Thus an α with $\hat{A}_\alpha = .01$ has to be considered a very large coefficient and hence A is considered close to the corresponding exclusive-or even though they only agree for a fraction .505 of the inputs.

When studying the long G code we need a more general Fourier transform. Remember that G is an Abelian group given by $G = C_{i_1} \times C_{i_2} \times C_{i_3} \dots C_{i_k}$ where C_{i_j} is a cyclic group with i_j elements and that we represent an element $g \in G$ as a k-tuple $(g^{(j)})_{j=1}^k$ where $g^{(j)}$ is an i_j'th root of unity. In this we case study function $A : \mathcal{F}_U^G \mapsto$ C where C are the complex numbers. The basis functions are given by k-tuples $(\alpha^{(j)})_{j=1}^k$ where $\alpha^{(j)} : \{-1,1\}^U \mapsto \{0, 1 \dots i_j - 1\}$, where we can think of α as an element of G only that it this time is more convenient to represent C_{i_j} as Z_{i_j} under addition. In particular this means that we can add coefficients in a natural way. If the components of a function $f \in \mathcal{F}_U^G$ are given by $(f^{(j)})_{j=1}^k$ then our basis functions are defined by

$$\chi_\alpha(f) = \prod_{x \in \{-1,1\}^U} \prod_{j=1}^{k} f^{(j)}(x)^{\alpha^{(j)}(x)}.$$

We have the inner product formed by

$$(A, B) = size(G)^{-2^u} \sum_f A(f)\overline{B(f)},$$

where $\overline{B(f)}$ denotes complex conjugation. The basis functions again form an orthonormal system and we can again define the Fourier coefficients by

$$\hat{A}_\alpha = (A, \chi_\alpha),$$

which is inverted by

$$A(f) = \sum_\alpha \hat{A}_\alpha \chi_\alpha(f). \tag{3}$$

The numbers $\hat{A}_\alpha$ are complex numbers and Parseval's identity gives

$$\sum_\alpha |\hat{A}_\alpha|^2 = size(G)^{-2^u} \sum_f |A(f)|^2 = 1.$$

2.5 Testing a long code

Having collected all the important tools we are now ready to describe the first interesting test, namely to test whether a given function $A : \mathcal{F}_U \mapsto \{-1, 1\}$ is a long code of some input x. This test has no consequences for the optimization problems we want to study and the reason to present it is largely pedagogical. It is easy to analyze given the correct tools but still gives a nontrivial conclusion. Rather than jumping to the correct test immediately we first describe a simpler test.

Long code test, first attempt

Written proof. A string of length 2^{2^u}, to be thought of as a function $A : \mathcal{F}_U \mapsto \{-1, 1\}$.
Desired property. The function A should be a long code, i.e. $A(f) = f(x)$ for some $x \in \{-1, 1\}^U$.
Verifier.

1. Choose f_0 and f_1 from $\mathcal{F}_U$ with the uniform probability.
2. Set $f_2 = f_0 f_1$, i.e. define f_2 by for each $x \in \{-1, 1\}^U$, $f_2(x) = f_0(x)f_1(x)$.
3. Accept if $A(f_0)A(f_1)A(f_2) = 1$.

First note that V always accept a correct proof since if A is the correct long code for x_0 then

$$A(f_0)A(f_1)A(f_2) = f_0(x_0)f_1(x_0)f_2(x_0) = f_0^2(x_0)f_1(x_0)^2 = 1,$$

and we need to analyze the acceptance probability when A is not a correct long code.

By definition $A(f_0)A(f_1)A(f_2)$ is one when the test accepts and negative one when it fails and thus we want to evaluate the expected value, taken over the choice of f_0 and f_1, of $A(f_0)A(f_1)A(f_2)$ which is the *advantage* of the test, i.e. the probability of accept minus the probability of reject. We replace $A(f_i)$ by is Fourier expansion, i.e. using (2) and see that $E_{f_0,f_1}[A(f_0)A(f_1)A(f_2)]$ equals

$$E_{f_0,f_1}\left[\sum_{\alpha_0,\alpha_1,\alpha_2} \hat{A}_{\alpha_0}\hat{A}_{\alpha_1}\hat{A}_{\alpha_2} \prod_{x\in\alpha_0} f_0(x) \prod_{x\in\alpha_1} f_1(x) \prod_{x\in\alpha_2} f_2(x)\right]. \tag{4}$$

Using the linearity of expectation (4) equals

$$\sum_{\alpha_0,\alpha_1,\alpha_2} \hat{A}_{\alpha_0}\hat{A}_{\alpha_1}\hat{A}_{\alpha_2} E_{f_0,f_1}\left[\prod_{x\in\alpha_0} f_0(x) \prod_{x\in\alpha_1} f_1(x) \prod_{x\in\alpha_2} f_2(x)\right]. \tag{5}$$

The key is now that if not $\alpha_0 = \alpha_1 = \alpha_2$ then

$$E_{f_0,f_1}\left[\prod_{x\in\alpha_0} f_0(x) \prod_{x\in\alpha_1} f_1(x) \prod_{x\in\alpha_2} f_2(x)\right] = 0.$$

This follows since

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

and the fact that

$$E_f\left[\prod_{x\in\alpha} f(x)\right] = 0$$

unless α is empty when f is a random function. Thus the expression (5) simplifies to

$$\sum_{\alpha} \hat{A}_{\alpha}^3.$$

Now since $\sum_{\alpha} \hat{A}_{\alpha}^2 = 1$ we have that

$$\sum_{\alpha} \hat{A}_{\alpha}^3 \leq \max_{\alpha} \hat{A}_{\alpha} \sum_{\alpha} \hat{A}_{\alpha}^2 = \max_{\alpha} \hat{A}_{\alpha}.$$

Thus a function A that passes this test with probability $(1+p)/2$ needs to have some coefficient $\hat{A}_{\alpha}$ at least p and thus, in our sense of the word, it is close to a linear combination of long codes. Note however also that if A is one of our basis functions χ_{α} then A passes this test with probability one and thus we are very far from a test of the long code since the test accepts arbitrary exclusive-ors of long codes and in fact what we have presented is the linearity test of [5].

To make the test closer to a test of the long code we give up perfect completeness and allow for a small probability of rejecting a correct long code.

Long code test, second attempt, parameterized by ϵ

Written proof. A string of length 2^{2^u}, to be thought of as a function $A : \mathcal{F}_U \mapsto \{-1, 1\}$.
Desired property. The function A should be a long code, i.e. $A(f) = f(x)$ for some $x \in \{-1, 1\}^U$.
Verifier.

1. Choose f_0 and f_1 from $\mathcal{F}_U$ with the uniform probability.
2. Choose a function $\mu \in \mathcal{F}_U$ by setting $\mu(x) = 1$ with probability $1 - \epsilon$ and $\mu(x) = -1$ otherwise, independently for each $x \in \{-1, 1\}^U$.
3. Set $f_2 = f_0 f_1 \mu$, i.e. define f_2 by for each $x \in \{-1, 1\}^U$, $f_2(x) = f_0(x) f_1(x) \mu(x)$.
4. Accept if $A(f_0)A(f_1)A(f_2) = 1$.

This time V accepts a correct long code for an input x_0 exactly iff $\mu(x_0) = 1$ which, by definition, happens with probability $1 - \epsilon$. Now let us analyze the general case. We again want to calculate $E_{f_0,f_1,\mu}[A(f_0)A(f_1)A(f_2)]$ and the expansion upto (7) is still valid and we need to consider

$$E_{f_0,f_1,\mu}\left[\prod_{x\in\alpha_0} f_0(x) \prod_{x\in\alpha_1} f_1(x) \prod_{x\in\alpha_2} f_2(x)\right] =$$

$$E_{f_0,f_1,\mu}\left[\prod_{x\in\alpha_0\Delta\alpha_2} f_0(x) \prod_{x\in\alpha_1\Delta\alpha_2} f_1(x) \prod_{x\in\alpha_2} \mu(x)\right]. \tag{6}$$

Since f_0, f_1 and μ are independent we again see that unless unless $\alpha_0 = \alpha_1 = \alpha_2$ this expected value is 0. Since $E_\mu[\mu(x)] = 1 - 2\epsilon$ and $\mu(x)$ is independent for different x we have

$$E_\mu\left[\prod_{x\in\alpha_2} \mu(x)]\right] = (1-2\epsilon)^{size(\alpha_2)}$$

and thus

$$E_{f_0,f_1,\mu}\left[A(f_0)A(f_1)A(f_2)\right] = \sum_\alpha \hat{A}_\alpha^3 (1-2\epsilon)^{size(\alpha)} \leq \max_\alpha \hat{A}_\alpha (1-2\epsilon)^{size(\alpha)},$$

where the inequality again follows from Parseval's identity.

Again to pass the test with probability $(1+p)/2$ we need $\hat{A}_\alpha \geq p$ for some α but the point is that we can add the condition that $size(\alpha) \leq \epsilon^{-1}\log p^{-1}$. 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.

2.6 Folding and conditioning of long codes

An observant reader might have noted that if A is the constant function 1 then it always passes also the second test of the long code. 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 was introduced in [6] and was called "folding over 1"[3] and we define it formally as follows:

Definition 2.22 *A function $A : \mathcal{F}_U \mapsto \{-1,1\}$ is* folded over 1 *if for each f, $A(f) = -A(-f)$.*

When a folded function is specified by a table, we store for each pair $(f,-f)$ one entry in the table. We have some, arbitrary but fixed, convention which entry to store for each pair. When we want to access the other entry we simply change the sign of the read value. We have the following immediate observation:

Lemma 2.23 *If the function A is folded over 1 then for all α with $size(\alpha)$ even, $\hat{A}_\alpha = 0$.*

Proof: Consider the definition.

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

Since $A(f) = -A(-f)$ while $\prod_{x\in\alpha} f(x) = \prod_{x\in\alpha}(-f(x))$ when $size(\alpha)$ is even the two terms corresponding to f and $-f$ cancel each other and hence the sum is 0. ∎

[3] We should really call our folding "folding over -1". However, [6] called it "folding over 1" which was correct since that paper used $\{0,1\}$ notation. We keep the same name to emphasize that we are using the same concept

We sometimes have the additional property that we are interested in a supposed long code where we know that the input x for which it is supposed to be the long code satisfies $h(x) = -1$ (i.e. h is true) for some function h. This was also needed in [6] where they defined "folding over h" analogously to folding over 1. We need a stronger property which we call "conditioning upon h".

Definition 2.24 *A function $A : \mathcal{F}_U \mapsto \{-1, 1\}$ is* conditioned upon h *if for each f, $A(f)$ only depends on $f \wedge h$.*

Again this is easy to implement in a PCP since we simply use a table which contains the value of A only for functions of the form $g \wedge h$ and whenever we need to access $A(f)$ we first compute $f \wedge h$. We have the following straightforward lemma

Lemma 2.25 *If the function $A : \mathcal{F}_U \mapsto \{-1, 1\}$ is conditioned upon h, then for any α such that there exists $x \in \alpha$ with $h(x) = 1$ we have $\hat{A}_\alpha = 0$.*

Proof: Informally this is obvious since $A(f)$, by definition, only depends on the value of f at point such that $h(x) = -1$ and hence these are the only values that should appear in the Fourier expansion. Formally, we just use the definition

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

Now suppose there is $x_0 \in \alpha$ such that $h(x_0) = 1$. Then for any f consider f' where $f'(x_0) = -f(x_0)$ while $f'(x) = f(x)$ for $x \neq x_0$. The set of all functions gets divided into $2^{2^{u-1}}$ pairs (f, f') and since $A(f) = A(f')$ while $\prod_{x \in \alpha} f(x) = -\prod_{x \in \alpha} f'(x)$ the elements of a pair cancel each other in the above sum an thus the sum evaluates to 0. ∎

Note that there is no problem in applying folding over 1 and conditioning upon h simultaneously. As an example with could apply the second test of the long code to a supposed long code that was folded over 1 and conditioned upon h. In such a case the end result would be that if the test accepts with high probability then the function A is close to exclusive-or of small, odd, size of long codes of inputs x_i which satisfy $h(x_i) = -1$.

When working with long G codes where $G = C_{i_1} \times C_{i_2} \times C_{i_3} \ldots C_{i_k}$ we need to fold over G.

Definition 2.26 *A function $A : \mathcal{F}_U^G \mapsto G$ is* folded over G *if for each $f \in \mathcal{F}_U^G$ and $g \in G$, $A(gf) = gA(f)$.*

Note that folding over 1 is the special case "folded over Z_2".

We are interested in analyzing various object by using the Fourier expansion. Now Fourier transforms are defined for functions $A : \mathcal{F}_U^G \mapsto \mathrm{C}$ while our long G codes are functions $A : \mathcal{F}_U^G \mapsto \mathrm{C}^k$ and thus have a different range. We take care of the situation as follows. Let $A^{(j)}$ be the j'th component of as supposed long

G code A and let $\gamma = (\gamma_j)_{j=1}^k$, be vector where $\gamma_j \in [i_j]$. Define A^γ by

$$A^\gamma(f) = \prod_{j=1}^{k} A^{(j)}(f)^{\gamma_j}.$$

Similarly for $g \in G$ we let g^γ denote the complex number $\prod_{j=1}^k (g^{(j)})^{\gamma_j}$. Thus given a function $A : \mathcal{F}_U^G \mapsto G$ we get $size(G)$ different functions $A^\gamma : \mathcal{F}_U^G \mapsto \mathrm{C}$ and we can say some useful about their Fourier transforms. Remember that the Fourier coefficient are indexed by $\alpha = (\alpha^{(j)})_{j=1}^k$ where $\alpha^{(j)} : \{-1,1\}^U \mapsto\in \mathrm{Z}_{i_j}$. We have

Lemma 2.27 *If the function A is folded over G then for all α with $\hat{A}_\alpha^\gamma \neq 0$ we have $\sum_x \alpha(x) = \gamma$.*

Proof: By definition

$$\hat{A}_\alpha = \sum_f A^\gamma(f)\overline{\chi_\alpha(f)}.$$

Now partition the sum into disjoint sums of size $size(G)$ where one such sum is over $(gf)_{g\in G}$ for some $f \in \mathcal{F}_U^G$. We have

$$\chi_\alpha(gf) = \chi_\alpha(f)\prod_x g^{\alpha(x)} = g^{\sum_x \alpha(x)}\chi_\alpha(f)$$

and thus the total contribution from such a sum of size $size(G)$ is using the definition of folded over G

$$\sum_g g^\gamma A^\gamma(f)\overline{g^{\sum_x \alpha(x)}\chi_\alpha(f)} = A^\gamma(f)\overline{\chi_\alpha(f)}\sum_g g^{\gamma - \sum_x \alpha(x)}$$

and this is zero unless $\sum_x \alpha(x) = \gamma$. The lemma follows. ∎

When working over with complex number, complex conjugation comes in very natural.

Definition 2.28 *A function $A : \mathcal{F}_U^G \mapsto G$ is conjugation correct if for each $f \in \mathcal{F}_U^G$, $A(\overline{f}) = \overline{A(f)}$.*

Also this property is easy to implement when accessing the long G code and it has the following nice consequence.

Lemma 2.29 *If A is conjugation correct then for all α, it is the case that $\hat{A}_\alpha$ is real.*

Proof: This is again straightforward from the definition.

$$\overline{\hat{A}_\alpha} = \overline{(A,\chi_\alpha)} = size(G)^{-2^u}\sum_f \overline{A(f)}\prod_{j=1}^{k}\overline{\prod_{x\in\{-1,1\}^U}(f^{(j)}(x))^{\alpha^{(j)}(x)}} =$$

$$size(G)^{-2^u} \sum_f A(\overline{f}) \prod_{j=1}^{k} \prod_{x \in \{-1,1\}^U} (\overline{f^{(j)}(x)})^{\alpha^{(j)}} =$$

$$size(G)^{-2^u} \sum_f A(f) \prod_{j=1}^{k} \prod_{x \in \{-1,1\}^U} (f^{(j)}(x))^{\alpha^{(j)}} = \hat{A}_\alpha,$$

where we simply changed the summation variable from f to $\overline{f}$. We conclude that $\hat{A}_\alpha$ is real. ∎

The notion of conditioning extends without virtually any changes.

Definition 2.30 *A function $A : \mathcal{F}_U^G \mapsto G$ is* conditioned upon h *if for each f, $A(f)$ only depends on $f \wedge h$.*

We state the corresponding lemma without a proof.

Lemma 2.31 *If the function $A : \mathcal{F}_U^G \mapsto G$ is conditioned upon h, then for any α such that there exists x with $\alpha(x) \neq 0^k$ with $h(x) = 1$ we have $\hat{A}_\alpha = 0$.*

3 The basic two-prover protocol and the corresponding PCP

To get our inapproximability results we construct different PCPs. Most of these PCPs, however, have the same written proof and only the method to check this proof customized to fit the combinatorial problem in mind. In this section we show how to construct this written proof by going through a two-prover protocol.

We start with an E3-SAT formula φ as given by Theorem 2.19. Thus, either φ is satisfiable or it is at most c-satisfiable for some $c < 1$ and it is NP-hard to distinguish the two cases. We have the property that each clause is of length exactly 3 and each variable appears exactly 12 times.

Simple two prover protocol

Input. A 3-CNF formula , $\varphi = C_1 \wedge C_2 \ldots C_m$ where C_j contains the variables x_{a_j}, x_{b_j} and x_{c_j}.

Verifier.

1. Choose $j \in [m]$ and $k \in \{a_j, b_j, c_j\}$ both uniformly at random and send j to P_1 and k to P_2.

2. Receive values for x_{a_j}, x_{b_j} and x_{c_j} from P_1 and for x_k from P_2. Accept if the two values for x_k agree and C_j is satisfied.

We have.

Lemma 3.1 *If φ is c-satisfiable then the optimal strategy for P_1 and P_2 causes V to accept in the simple two prover protocol with probability $(2+c)/3$.*

Proof: The answers by P_2 defines an assignment α_0 to all variables. Since the provers coordinate their strategies, P_1 knows α_0 and it is now not hard to determine the optimal strategy for P_1. Whenever V chooses a clause that is satisfied by α_0, P_1 answers according to α_0. Whenever V chooses a clause not satisfied by α_0, to have any probability of V accepting P_1, should not answer according to α_0 and to have minimal probability of his answer being found inconsistent with the answer of P_2 he should change the value of exactly one variable. The probability of V rejecting in this case is exactly $1/3$. Thus if α_0 satisfies a fraction c' of the clauses then V rejects under this prover strategy with probability $1/3(1-c')$. Thus it is optimal for P_2 to choose α_0 to satisfy the largest possible number of clauses and the lemma follows. ∎

The simple two prover protocol is good in that V only asks for the value of four bits but it is bad in that the acceptance probability is always rather close to 1. We improve this second parameter by running the protocol in parallel a constant number of times.

u parallel two prover protocol

Input. A 3-CNF formula , $\varphi = C_1 \wedge C_2 \ldots C_m$ where C_j contains the variables x_{a_j}, x_{b_j} and x_{c_j}.
Verifier.

1. For $i = 1, 2 \ldots u$, choose $j_i \in [m]$ and $k_i \in \{a_{j_i}, b_{j_i}, c_{j_i}\}$ all uniformly at random and independently and send $(j_i)_{i=1}^u$ to P_1 and $(k_i)_{i=1}^u$ to P_2.
2. Receive values for $x_{a_{j_i}}, x_{b_{j_i}}$ and $x_{c_{j_i}}$ from P_1 and for x_{k_i} from P_2 for $i = 1, 2 \ldots u$. Accept if the two values for x_{k_i} agree and C_{j_i} is satisfied for all $1 \leq i \leq u$.

By applying Theorem 2.21 and Lemma 3.1 we get.

Lemma 3.2 *If φ is c-satisfiable where $c < 1$ then there is a constant $c_c < 1$ such that for any integer u, the optimal strategy for P_1 and P_2 causes V to accept in the u-parallel two prover protocol with probability c_c^u. If φ is satisfiable then V can be made to always accept.*

To simplify notation we denote a set of variables $(k_i)_{i=1}^u$ sent to P_2 by U and a set $(x_{a_{j_i}}, x_{b_{j_i}}, x_{c_{j_i}})_{i=1}^u$ sent to P_1 by W. Thus typically a set U is of size u and a set W is of size $3u$.

Now we want to convert the this u-parallel two-prover protocol into a PCP. We would do it by simple writing down the prover answers but this is not very efficient and we instead write down, for each possible question, the long of the possible answer. We call proof simply SWP for Standard Written Proof.

Definition 3.3 *A* Standard Written Proof *with parameter u or simply SWP(u) contains for each set $U \subset [n]$ of size at most u a string of length $2^{2^{size(U)}}$ which we interpret as the table of a function $A_U : \mathcal{F}_U \mapsto \{-1, 1\}$. It also contains for each set W constructed as the set of variables in u clauses a function $A_W : \mathcal{F}_W \mapsto \{-1, 1\}$.*

Definition 3.4 *A SWP(u) is a* correct proof *for a formula φ of n variables if there is an assignment x which satisfies φ such that A_V is the long code of $x|_V$ for any V of size at most u or any V constructed as the set of variables of u clauses.*

Many times we will fold the supposed long codes over 1 and all times we condition the long code over W upon $\wedge_{i=1}^{u} C_{j_i}$[4].

The size of a SWP(u) is about $n^u 2^{2^{3u}}$ and thus as long as u is a constant this is polynomial size.

The general strategy for proving inapproximability for a our optimization problem is to design a test of SWP(u) that closely mimics the optimization problem.

The standard proof strategy for establishing that such PCP has small soundness is to prove that if a specific SWP(u) passes a particular test with high probability then we can use this proof to create strategies for P_1 and P_2 to pass the u-parallel two prover test with high probability.

Finally, we generalize the notation to deal with long-G-codes.

Definition 3.5 *A* Standard Written G-Proof *with parameter u or simply SWGP(u) contains for each set $U \subset [n]$ of size at most u a string of length $size(G)^{2^{size(U)}}$ which we interpret as the table of a function $A_U : \mathcal{F}_U^G \mapsto G$. It also contains for each set W constructed as the set of variables in u clauses a function $A_W : \mathcal{F}_W^G \mapsto G$.*

Definition 3.6 *A SWGP(u) is a* correct proof *for a formula φ of n variables if there is an assignment x which satisfies φ such that A_V is the long G code of $x|_V$ for any V of size at most u or any V constructed as the set of variables of u clauses.*

4 Linear equations

We first study the optimization problem Max-E3-Lin-2 and for reasons that becomes obvious in the proof of Theorem 4.4 we want to design a test for SWP(u) that accepts depending only on the exclusive-or of three bits of the proof. It seems natural to use a variant of the simple test for the long code and this is in fact what we do.

[4]To be very exact we need to take care of the fact that the same set W can be obtained from several ways of choosing u clauses. One way to do this is to let W be defined by the u clauses rather than the variables it contains. This means that when we are conditioning upon $\wedge_{i=1}^{u} C_{j_i}$ we have several different tables for the same W. Another way is to condition the table on all conditions supported on W. Both of these methods work.

Test $L_2^\epsilon(u)$

Written proof. A SWP(u) where all tables are folded over 1 and tables A_W, for sets W defined by clause $(C_{i_j})_{i=1}^u$ conditioned upon $\wedge_{i=1}^u C_{i_j}$.
Desired property. To check that it is a correct SWP(u) for a given formula $\varphi = C_1 \wedge C_2 \ldots C_m$.
Verifier.

1. Choose u random clauses $(C_{j_i})_{i=1}^u$ with uniform probability and for each i a random variable x_{k_i} occurring in C_{j_i} again with uniform probability. Set $U = \{x_{i_1}, x_{i_2} \ldots x_{i_u}\}$ and W to be the set of all variables in the chosen clauses.

2. Choose $f \in \mathcal{F}_U$ with the uniform probability.

3. Choose $g_1 \in \mathcal{F}_W$ with the uniform probability.

4. Choose a function $\mu \in \mathcal{F}_W$ by setting $\mu(y) = 1$ with probability $1 - \epsilon$ and $\mu(y) = -1$ otherwise, independently for each $y \in \{-1, 1\}^W$.

5. Set $g_2 = f g_1 \mu$, i.e. define g_2 by for each $y \in \{-1, 1\}^W$, $g_2(y) = f(y|_U) g_1(y) \mu(y)$.

6. Accept if $A_U(f) A_W(g_1) A_W(g_2) = 1$.

We need to analyze this test and it is not difficult to establish a good bound for the completeness.

Lemma 4.1 *The completeness of Test* $L_2^\epsilon(u)$ *is at least* $1 - \epsilon$.

Proof: Fix a correct SWP(u) obtained from the assignment x satisfying φ We claim that V accepts unless $\mu(x|_W) = -1$. This follows since for a correct SWP(u) the folding over 1 and conditioning upon $\wedge_{i=1}^u C_{j_i}$ do not affect the long codes and hence $A_U(f) = f(x|_U)$, $A_W(g_1) = g_1(x|_W)$ and $A_W(g_2) = g_2(x|_W) = f(x|_U) g_1(x|_W) \mu(x|_W)$. The probability that $\mu(x|_W) = -1$ is, by definition, ϵ and the lemma follows. ∎

The main problem is therefore to establish the soundness and to this end we have.

Lemma 4.2 *For any* $\epsilon > 0$, $\delta > 0$, *suppose that the probability that the verifier of Test* $L_2^\epsilon(u)$ *accepts is* $(1 + \delta)/2$. *Then there is a strategy for* P_1 *and* P_2 *in the* u*-parallel two prover protocol that makes the verifier of that protocol accept with probability at least* $\epsilon\delta^2/2$.

Proof:

Let us first fix U and W and for notational convenience we denote the function A_U by A and the function A_W by B.

As in the tests for the long code we want to consider

$$E_{f,g_1,\mu}[A(f)B(g_1)B(g_2)] \tag{7}$$

since by the assumption of the lemma

$$E_{U,W,f,g_1,\mu}[A_U(f)A_W(g_1)A_W(g_2)] = \delta. \tag{8}$$

We again want to use the Fourier expansion and (7) equals

$$E_{f,g_1,\mu}\left[\sum_{\alpha,\beta_1,\beta_2}\hat{A}_\alpha\hat{B}_{\beta_1}\hat{B}_{\beta_2}\prod_{x\in\alpha}f(x)\prod_{y\in\beta_1}g_1(y)\prod_{y\in\beta_2}g_2(y)\right] =$$

$$\sum_{\alpha,\beta_1,\beta_2}\hat{A}_\alpha\hat{B}_{\beta_1}\hat{B}_{\beta_2}E_{f,g_1,\mu}\left[\prod_{x\in\alpha}f(x)\prod_{y\in\beta_1}g_1(y)\prod_{y\in\beta_2}g_2(y)\right] =$$

$$\sum_{\alpha,\beta_1,\beta_2}\hat{A}_\alpha\hat{B}_{\beta_1}\hat{B}_{\beta_2}E_{f,g_1,\mu}\left[\prod_{x\in\alpha}f(x)\prod_{y\in\beta_1}g_1(y)\prod_{y\in\beta_2}(f(y|_U)g_1(y)\mu(y))\right]. \tag{9}$$

If $\beta_1 \neq \beta_2$ then since $g_1(y)$ for $y \in \beta_1\Delta\beta_2$ is random and independent of everything else

$$E_{g_1}\left[\prod_{x\in\alpha}f(x)\prod_{y\in\beta_1}g_1(y)\prod_{y\in\beta_2}(f(y|_U)g_1(y)\mu(y))\right] = 0$$

thus (9) reduces to

$$\sum_{\alpha,\beta}\hat{A}_\alpha\hat{B}_\beta^2E_{f,\mu}\left[\prod_{x\in\alpha}f(x)\prod_{y\in\beta}(f(y|_U)\mu(y))\right]. \tag{10}$$

Now consider a term in the above sum and let s_x be number of $y \in \beta$ such that $y|_U = x$. Since $f(x)$ is random and independent for different x, unless for every x either $x \in \alpha$ and s_x is odd or $x \notin \alpha$ and s_x is even

$$E_f\left[\prod_{x\in\alpha}f(x)\prod_{y\in\beta}(f(y|_U)\mu(y))\right] = 0.$$

These conditions imply that we only keep terms with $\pi_2^U(\beta) = \alpha$ and finally since

$$E_\mu[\prod_{y\in\beta}\mu(y)] = (1-2\epsilon)^{size(\beta)}$$

we have reduced the sum (7) to

$$\sum_\alpha\sum_{\beta|\pi_2^U(\beta)=\alpha}\hat{A}_\alpha\hat{B}_\beta^2(1-2\epsilon)^{size(\beta)}. \tag{11}$$

Since we from now on only will project onto the set U we drop the index U on the operation π. We want to prove that if the expected value of this (over

random choices of U and W) is at least δ then we have a good strategy of the provers.

We first identify parts of the sum that are always small. Since each β appears for exactly one α

$$\sum_{\alpha|\hat{A}_\alpha \leq \delta/2} \sum_{\beta|\pi_2(\beta)=\alpha} \hat{A}_\alpha \hat{B}_\beta^2 \leq \frac{\delta}{2} \sum_\beta \hat{B}_\beta^2 \leq \frac{\delta}{2}.$$

This implies that

$$E_{U,W}\left[\sum_{\alpha|\hat{A}_\alpha > \delta/2} \sum_{\beta|\pi_2(\beta)=\alpha} \hat{A}_\alpha \hat{B}_\beta^2 (1-2\epsilon)^{size(\beta)}\right] \geq \frac{\delta}{2}. \tag{12}$$

Now we define good randomized strategies for P_1 and P_2. 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 and $\hat{A}_\alpha \geq \delta/2$. The probability of picking α is defined to be proportional to $\hat{A}_\alpha$. It is easy to see that it is at least $\delta/2\hat{A}_\alpha$ since

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

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 U since we only have that the expected value (over the choice of U and W) of the sum (11) is large. For U 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 probability $\hat{B}_\beta^2$ and then answering with a random $y \in \beta$.

Let us evaluate the success-rate of this strategy. Note that, by Lemma 2.25 every y sent by P_1 satisfies the corresponding clauses and thus we only need to look at the probability that the answers are consistent. This probability is at least $size(\beta)^{-1}$ times the probability that for the picked α and β we have $\alpha = \pi_2(\beta)$. This follows since in this case for each $x \in \alpha$ there is at least one $y \in \beta$ such that $y|_U = x$. The probability of picking a specific pair α and β is, provided $\hat{A}_\alpha > \delta/2$, at least $\hat{A}_\alpha \hat{B}_\beta^2 \delta/2$ and thus the overall successrate for a fixed choice of U and W is at least

$$\delta/2 \sum_{\alpha|\hat{A}_\alpha \geq \delta/2} \sum_{\beta|\pi_2(\beta)=\alpha} \hat{A}_\alpha \hat{B}_\beta^2 size(\beta)^{-1}. \tag{13}$$

To compare this sum to (12) we use the following elementary lemma.

Lemma 4.3 *For each $x > 0$, $x^{-1} \geq e^{-x}$.*

Proof: We have that $xe^{-x} = e^{\ln x - x}$ and thus we need to prove that $\ln x - x \leq 0$ for each $x > 0$. This is certainly true for $x \leq 1$ since neither term is positive and, as can be seen from differentiating $\ln x - x$ is decreasing for $x > 1$. ∎

From this lemma, applied to $x = 2\epsilon size(\beta)$, it follows that

$$size(\beta)^{-1} \geq 2\epsilon e^{-2\epsilon size(\beta)} \geq 2\epsilon(1-2\epsilon)^{size(\beta)},$$

where the second inequality follows from $e^{-x} \geq (1-x)$ which is true for any $x > 0$. To sum up the overall success rate of the defined strategies for P_1 and P_2 is at least

$$\delta/2 E_{U,W}\left[\sum_{\alpha|\hat{A}_\alpha|\geq\delta/2}\sum_{\beta|\pi_2(\beta)=\alpha}\hat{A}_\alpha B_\beta^2 size(\beta)^{-1}\right] \tag{14}$$

which by the above argument is at least

$$\delta\epsilon E_{U,W}\left[\sum_{\alpha|\hat{A}_\alpha|\geq\delta/2}\sum_{\beta|\pi_2(\beta)=\alpha}\hat{A}_\alpha B_\beta^2 (1-2\epsilon)^{size(\beta)}\right]$$

which by (12) is at least $\delta^2\epsilon/2$ and the proof of Lemma 4.2 is complete. ∎

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

Theorem 4.4 *For any $\epsilon > 0$ it is NP-hard to approximate Max-E3-Lin-2 within a factor $2 - \epsilon$.*

Proof: Set δ to a constant greater than 0 such that

$$\frac{1-\delta}{(1+\delta)/2} \geq 2 - \epsilon.$$

Remember also that since we are working of $\{-1, 1\}$ a linear equation mod 2 has a left hand size which is product of variables.

Let L be an arbitrary language in NP and given an input x. By Theorem 2.19 we can in polynomial time create a E3-CNF formula φ with each variable occurring exactly 12 times such that if $x \in L$ then φ is satisfiable and if $x \notin L$ when φ is at most c-satisfiable where c is some definite constant less than 1. Now choose a u such that $\delta^3/2 > c_c^u$ where c_c is the constant from Lemma 3.2 and consider applying test $L_2^\delta(u)$ to φ.

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

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

where $b_{U,f}$, b_{W,g_1} and b_{W,g_2} are the bits in the proof corresponding to $A_U(f)$, $A_W(g_1)$ and $A_W(g_2)$, respectively. One might think that the right hand size

would always be 1, but because of our folding of the long codes the bit corresponding to $A_U(f)$ in the proof might actually give the value of $A_U(-f)$. Thus the value of c depends on our mechanism for folding and, of course, the identity of f, g_1 and g_2. Let us now write down a set of linear equations with weights. Write down the equation

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

where c is defined as above. The weight of this equation is the probability that the verifier in test $L_2^\delta(u)$ chooses the tuple (U, W, f, g_1, g_2). Now each proof corresponds to an assignment to the variables x_b and the total weight of all satisfied equations is exactly the probability that this proof is accepted. This implies that if $x \in L$ this maximal weight is $1 - \delta$ while if $x \notin L$, it is in view of Lemma 4.2 and the choice of u, at most $(1 + \delta)/2$. The number of different equations is limited by the number of different choices of V. There are at most m^u choices for W and once W is chosen at most 3^u choices for V. Given V and W the number of choices for f is at most 2^{2^u} and for g_1 and g_2 $2^{2^{3u}}$ each. Thus the total number of choices is at most

$$m^u 2^{2u+2^u+2^{3u+1}}$$

which is polynomial since u is a constant. For each choice it is also not difficult to compute the corresponding weight (given as a rational number). Thus we can produce this set of equations in polynomial time.

It follows that any algorithm that could determine the maximal total weight of satisfiable equation within a factor smaller then

$$\frac{1-\delta}{(1+\delta)/2}$$

can be used to determine whether $x \in L$ and hence this must be NP-hard. Our choice of δ then would have proved the theorem if we would have considered weighted sets of equations.

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

Suppose we have M equations. Let k be a number to be determined. Suppose equation i has weight w_i, then we simply write $\lceil kw_iM \rceil$ copies of equation i, each of weight one. Note that since w_i are probabilities $\sum_{i=1}^M w_i = 1$. This creates a system of at most

$$\sum_{i=1}^{M} \lceil kw_iM \rceil \leq \sum_{i=1}^{M} (1 + kw_iM) = (k+1)M$$

equations. Now when $x \in L$ we can satisfy at least

$$\sum kw_iM = (1-\delta)kM$$

equations where the sum is over all equations satisfied by the assignment corresponding to a correct proof. If $x \notin L$ no assignment satisfies weighted equations of total weight more than $(1+\delta)/2$ and summing over those equations we have

$$\sum \lceil kw_i M \rceil \leq \sum (1 + kw_i M) \leq M + kM(1+\delta)/2$$

Thus in this case any algorithm that determines the optimal number of equations that can be satisfied with a ratio better than

$$\frac{(1-\delta)kM}{M + kM(1+\delta)/2} = \frac{1-\delta}{\frac{1}{k} + (1+\delta)/2}$$

solves an NP-hard problems. For any $\epsilon > 0$ we can choose $\delta > 0$ and a constant k such that this is larger than $2 - \epsilon$. We then choose u as before. 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 4.4 could all be satisfied simultaneously. However, to decide whether a set of linear equations have a common solution can be done in polynomial time by Gaussian elimination and thus perfect completeness would have implied P=NP.

It is not hard to extend the result to more variables in each equation.

Theorem 4.5 *For any $\epsilon > 0$, $k \geq 3$ it is NP-hard to approximate Max-Ek-Lin-2 within a factor $2 - \epsilon$.*

Proof: We have a straightforward reduction from the case $k = 3$ to arbitrary k. Given a system of equations with 3 variables in each equation in the variables $(x_i)_{i=1}^n$. Simply add the same $k-3$ new variables $(y_i)_{i=1}^{k-3}$ in very equation to make them all have k variables. Consider any assignment of the variables of this larger system and consider the same assignment to $(x_i)_{i=1}^n$ in the smaller system. If $\prod_{i=1}^{k-3} y_i = 1$ then it satisfies exactly the same equations while if $\prod_{i=1}^{k-3} y_i = -1$ it satisfies exactly the equations not satisfied in the larger system. Changing every x_i to its negation, however, now satisfies the equations satisfied in the larger system.

From the above argument we see that the maximal number of equations satisfied by the system is preserved and that it is easy to translate a solution of the larger system to an equally good solution to the small system. Thus we have a correct reduction from the case $k = 3$ to the case with $k > 3$. ∎

Sometimes it is useful to have a linear system of equation of a special type. Our systems are very uniform and the only part of the equations we do not control explicitly is the right hand size since it is determined by the folding convention. We below prove that if we have four variables in each equation than in fact we can have the right hand side -1 in all equations. Note that we cannot have right hand size 1 in all equations since in this case we can satisfy all equations by giving the value 1 to all variables. Similarly we cannot hope to have an odd number of variables in all equations since in this case giving -1 to all variables satisfy all equations.

Theorem 4.6 *For any $\epsilon > 0$, it is NP-hard to approximate Max-E4-Lin-2 within a factor $2 - \epsilon$ even in the case when all right hand sides are equal to -1.*

Proof: The only way we know how to prove this is construct a special purpose PCP. We want to avoid folding to be able to control the right hand side of the equations. Apart from this fact, the setup is very much similar to Test $L_2^\epsilon(u)$ and we skip some details in the proofs.

Test $L_{2,-1}^\epsilon(u)$

Written proof. A SWP(u) without folded tables but A_W, for sets W defined by clause $(C_{i_j})_{i=1}^u$ conditioned upon $\wedge_{i=1}^u C_{i_j}$.
Desired property. To check that it is a correct SWP(u) for a given formula $\varphi = C_1 \wedge C_2 \ldots C_m$.
Verifier.

1. Choose u random clauses $(C_{j_i})_{i=1}^u$ with uniform probability and for each i a random variable x_{k_i} occurring in C_{j_i} again with uniform probability. Set $U = \{x_{i_1}, x_{i_2} \ldots x_{i_u}\}$ and W to be the set of all variables in the chosen clauses.

2. Choose $f_1 \in \mathcal{F}_U$ and $f_2 \in \mathcal{F}_U$ independently with the uniform probability.

3. Choose $g_1 \in \mathcal{F}_W$ with the uniform probability.

4. Choose a function $\mu \in \mathcal{F}_W$ by setting $\mu(y) = 1$ with probability $1 - \epsilon$ and $\mu(y) = -1$ otherwise, independently for each $y \in \{-1, 1\}^W$.

5. Set $g_2 = -f_1 f_2 g_1 \mu$, i.e. define g_2 by for each $y \in \{-1, 1\}^W$, $g_2(y) = -f_1(y|_U) f_2(y|_U) g_1(y) \mu(y)$.

6. Accept if $A_U(f_1) A_U(f_2) A_W(g_1) A_W(g_2) = -1$.

We have

Lemma 4.7 *The completeness of Test $L_{2,-1}^\epsilon(u)$ is at least $1 - \epsilon$.*

We skip the proof since it is essentially identical to the simple proof of Lemma 4.1.

The main problem is therefore to establish the soundness and to this end we have.

Lemma 4.8 *For any $\epsilon > 0$, $\delta > 0$, suppose that the probability that the verifier of Test $L_{2,-1}^\epsilon(u)$ accepts is $(1 + \delta)/2$. Then there is a strategy for P_1 and P_2 in the u-parallel two prover protocol that makes the verifier of that protocol accept with probability $2\epsilon\delta$.*

Proof: As before, fix U and W and let A and B the supposed long codes on these sets. We want to analyze

$$E_{f_1,f_2,g_1,\mu}[-A(f_1)A(f_2)B(g_1)B(g_2)]. \tag{15}$$

Expanding all terms by the Fourier transform and using the linearity of expectation we arrive at the equivalent of (9) which is

$$-\sum_{\alpha_1,\alpha_2,\beta_1,\beta_2} \hat{A}_{\alpha_1}\hat{A}_{\alpha_2}\hat{B}_{\beta_1}\hat{B}_{\beta_2}$$

$$E_{f_1,f_2,g_1,\mu}\left[\prod_{x\in\alpha_1} f_1(x)\prod_{x\in\alpha_2} f_2(x)\prod_{y\in\beta_1} g_1(y)\prod_{y\in\beta_2}(-f_1(y|_U)f_2(y|_U)g_1(y)\mu(y))\right]. \tag{16}$$

Reasoning as in the proof of Lemma 4.2 we first see that unless $\beta_1 = \beta_2 = \beta$ the expected value over g_1 yields 0. Then, we again observer that unless $\alpha_1 = \pi_2(\beta)$ the expected value over f_1 yields 0 and similarly we conclude that we only need to care about terms with $\alpha_2 = \pi_2(\beta)$. Since μ is chosen with the same distribution as in test $L_2^\epsilon(u)$ (16) reduces to

$$\sum_{\alpha}\sum_{\beta|\pi_2(\beta)=\alpha} \hat{A}_\alpha^2\hat{B}_\beta^2(-1)^{size(\beta)+1}(1-2\epsilon)^{size(\beta)}. \tag{17}$$

The expected value of this sum over random U and W is by the condition of the lemma at least δ. Observe that if $\alpha = \pi_2(\beta)$ then the $size(\beta)$ is odd iff $size(\alpha)$ is odd. Also note that terms in (17) are positive iff $size(\beta)$ is odd and hence summing only over those terms makes the sum larger. Remember that since we are not folding over 1, also β and α of even size occur in this sum.

Now we define a strategy for P_1 and P_2 very much like before. Given U, P_2 chooses α of odd size with probability proportional to $\hat{A}_\alpha^2$ and returns a random $x \in \alpha$ while P_1 when asked W chooses a β also of odd size with probability proportional to $\hat{B}_\beta^2$ and returns a random $y \in \beta$. The success probability of this strategy is at least

$$E_{U,W}\left[\sum_{\alpha|size(\alpha)}\sum_{\text{odd }\beta|\pi_2(\beta)=\alpha} \hat{A}_\alpha^2\hat{B}_\beta^2 size(\beta)^{-1}\right] \tag{18}$$

and by using Lemma 4.3 and comparing to (17) this is at least $2\delta\epsilon$ and the lemma is proved. ∎

Now, Theorem 4.6 follows from Lemma 4.7 and Lemma 4.8 very much as Theorem 4.4 followed from Lemma 4.1 and Lemma 4.2. Essentially the only difference is that since we are not folding over 1 we make sure that all right hand sides are -1. We leave the details to the reader. ∎

Theorem 4.6 extends to the case of having exactly $2k$ variables in each equation for any $k \geq 2$. If we allow the same variable twice in the same equation there is an obvious reduction. If this is not allowed one can prove the result by modifying Test $L^{\epsilon}_{2,-1}(u)$ by choosing $2(k-1)$ random functions $f_i \in \mathcal{F}_U$ and then making the obvious changes.

We next turn to the question of linear equations mod p and here we only give the basic theorem.

Theorem 4.9 *For any $\epsilon > 0$ it is NP-hard to approximate Max-E3-Lin-p within a factor $p - \epsilon$.*

Proof: We construct a proof-system similar to the one considered in the proof of Theorem 4.4. We replace a SWP(u) by a SWpP(u) and proceed to analyze it by using the Fourier expansion. As discussed before we represent Z_p by the p'th roots of unity. Let ζ be such a root. We first define the test

Test $L^{\epsilon}_{p}(u)$

Written proof. A SWpP(u) where all tables are folded over Z_p and correct with respect to conjugation and tables A_W, for sets W defined by clause $(C_{i_j})_{i=1}^{u}$ conditioned upon $\wedge_{i=1}^{u} C_{i_j}$.
Desired property. To check that it is a correct SWpP(u) for a given formula $\varphi = C_1 \wedge C_2 \ldots C_m$.
Verifier.

1. Choose u random clauses $(C_{j_i})_{i=1}^{u}$ with uniform probability and for each i a random variable x_{k_i} occurring in C_{j_i} again with uniform probability. Set $U = \{x_{i_1}, x_{i_2} \ldots x_{i_u}\}$ and W to be the set of all variables in the chosen clauses.

2. Choose $f \in \mathcal{F}^p_U$ with the uniform probability.

3. Choose $g_1 \in \mathcal{F}^p_W$ with the uniform probability.

4. Choose a function $\mu \in \mathcal{F}^p_W$ by setting $\mu(y) = 1$ with probability $1 - \epsilon$ and otherwise $\mu(y) = \zeta^i$ where i is chosen randomly from $\{0, 1, \ldots p-1\}$. This is done independently for each $y \in \{-1, 1\}^W$.

5. Set $g_2 = (f g_1 \mu)^{-1}$, i.e. define g_2 by for each $y \in \{-1, 1\}^W$, $g_2(y) = (f(y|_U) g_1(y) \mu(y))^{-1}$.

6. Accept if $A_U(f) A_W(g_1) A_W(g_2) = 1$.

Completeness follows along the same lines as before and we omit the proof.

Lemma 4.10 *The completeness of Test $L^{\epsilon}_{p}(u)$ is at least $1 - \epsilon$.*

In fact completeness is even slightly better since we allow the choice of $i = 0$ in the definition of μ. Thus a correct proof is in fact accepted with probability $1 - \frac{p-1}{p}\epsilon$.

The interesting lemma for the soundness is given by.

Lemma 4.11 *For any $\epsilon > 0$, $\delta > 0$, suppose that the probability that the verifier of Test $L_p^\epsilon(u)$ accepts is $(1+\delta)/p$. Then there is a strategy for P_1 and P_2 in the u-parallel two prover protocol that makes the verifier of that protocol accept with probability $\delta^2\epsilon/(2p^2)$.*

Proof: The proof follows along the same lines as the proof of Lemma 4.2. The first difference to take care of is that when $A_U(f)A_W(g_1)A_W(g_2) \neq 1$ it can be ζ^i for any $1 \leq i \leq p-1$. This means that we cannot simply compute the expected value of $A_U(f)A_W(g_1)A_W(g_2)$ and hope that it gives all the information that we need. The solution is, however, not difficult. Let σ_i $i = 1 \ldots p-1$ be the automorphisms of Z_p sending ζ to ζ^i (i.e. in the notation given by $\{0, 1, \ldots p-1\}$ it is simply multiplication by i). Then the quantity

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

is $p-1$ if the test succeeds and -1 otherwise. Thus the key is to consider

$$E_{U,W,f,g_1,\mu}\left[\sum_{i=1}^{p-1} \sigma_i(A_U(f)A_W(g_1)A_W(g_2))\right] \tag{19}$$

which when the test accepts with probability $(1+\delta)/p$ is δ. We again fix U and W, call the supposed long p codes on these sets A and B and expand these functions using the Fourier expansions. Remember that in this case we have

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

and

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

where $\alpha(x)$ and $\beta(y)$ are elements in Z_p.

The conventions on how to access the long-p-codes implies, by Lemma 2.27 that $\sum_x \alpha(x) \equiv 1$ and $\sum_y \beta_y \equiv 1 \bmod p$ for all nonzero coefficients. The property of being correct with respect to conjugation implies, by Lemma 2.29 that all coefficients are real. Finally, by Lemma 2.31 it follows that if $\hat{B}_\beta \neq 0$ then each element y such that $\beta(y) \neq 0$ satisfies the clauses used to define W.

Now let us evaluate the sum (19), when U and W are fixed. Using the Fourier expansion and the linearity of expectation we obtain.

$$\sum_{i=1}^{p-1} \sum_{\alpha,\beta_1,\beta_2} \sigma_i\left(\hat{A}_\alpha \hat{B}_{\beta_1} \hat{B}_{\beta_2}\right) E_{f,g_1,\mu}\left[\sigma_i(\prod_x f(x)^{\alpha(x)} \prod_y g_1(y)^{\beta_1(y)} g_2(y)^{\beta_2(y)})\right]. \tag{20}$$

The inner expected value equals, using the definition of g_2.

$$E_{f,g_1,\mu}\left[\sigma_i(\prod_x f(x)^{\alpha(x)} \prod_y g_1(y)^{\beta_1(y)} (f(y|_U) g_1(y) \mu(y))^{-\beta_2(y)})\right]. \tag{21}$$

Now, note that the triplet $(\sigma_i(f), \sigma_i(g_1), \sigma_i(g_2))$ has the same distribution as (f, g_1, g_2) and hence we can ignore σ_i when investigating the inner expected value. Note also that since all $\hat{A}_\alpha$ and $\hat{B}_\beta$ are real we can in fact eliminate σ_i all together. If $\beta_1(y) \neq \beta_2(y)$ for some y, then (21) is zero and finally, since μ and f are independent we can treat the expected values over those variables separately. Since $\sum_{i=0}^{p-1} \zeta^i = 0$ we have that

$$E[\mu(y)^t] = (1-\epsilon) + \frac{\epsilon}{p}\sum_{i=0}^{p-1}\zeta^i = 1-\epsilon$$

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

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

is 0 if not

$$\prod_{y\in\pi^{-1}(x)} \beta_2(y) \equiv \alpha(x) \bmod p$$

for all x in which case it is 1. This condition is just the definition of $\pi_p(\beta) = \alpha$. Summing up, if we let $s(\beta)$ be the number of nonzero components of β then we have reduced (20) to

$$(p-1)\sum_\alpha \hat{A}_\alpha \sum_{\beta \mid \pi_p(\beta)=\alpha} \hat{B}_\beta^2 (1-\epsilon)^{s(\beta)}. \tag{22}$$

We now discard all $\hat{A}_\alpha$ with $\hat{A}_\alpha < \frac{\delta}{2p}$. This reduces the expected value of the sum by at most $\delta/2$. We are now ready to define a strategy for provers. As before P_2 chooses a random α such that $\hat{A}_\alpha > \frac{\delta}{2p}$ with probability proportional to $\hat{A}_\alpha$. P_2 then picks a random x such that $\alpha(x) \neq 0$ with uniform probability over such x. In a similar manner P_1 chooses a β with probability $\hat{B}_\beta^2$ and returns a random y such that $\beta(y) \neq 0$. The success probability of this strategy is at least

$$\frac{\delta}{2p} \sum_{\alpha \mid \hat{A}_\alpha > \frac{\delta}{2p}} \hat{A}_\alpha \sum_{\beta \mid \pi_p(\beta)=\alpha} \hat{B}_\beta^2 s(\beta)^{-1}.$$

Comparing this to (22) doing the same calculations as in the proof of Lemma 4.2, and taking expected value over U and W we see that the success probability of the strategy is at least

$$\frac{\delta^2\epsilon}{2p(p-1)}$$

and Lemma 4.11 follows. ∎

Theorem 4.9 follows from Lemma 4.10 and Lemma 4.11 in the standard way. ∎

Having done the mod p case, we proceed to handle general Abelian groups.

Theorem 4.12 *For any $\epsilon > 0$ and any Abelian group G, it is NP-hard to approximate Max-E3-Lin-G within a factor $size(G) - \epsilon$.*

Proof: We have $G = C_{i_1} \times C_{i_2} \times C_{i_3} \ldots C_{i_k}$ where each component is thought of as a complex number which is a i_j'th root of unity.

The test which is completely analogous to the mod p case.

Test $L^{\epsilon}_{G}(u)$

Written proof. A SWGP(u) where all tables are folded over G and correct with respect to conjugation and tables A_W, for sets W defined by clause $(C_{i_j})_{i=1}^{u}$ conditioned upon $\wedge_{i=1}^{u} C_{i_j}$.
Desired property. To check that it is a correct SWGP(u) for a given formula $\varphi = C_1 \wedge C_2 \ldots C_m$.
Verifier.

1. Choose u random clauses $(C_{j_i})_{i=1}^{u}$ with uniform probability and for each i a random variable x_{k_i} occurring in C_{j_i} again with uniform probability. Set $U = \{x_{i_1}, x_{i_2} \ldots x_{i_u}\}$ and W to be the set of all variables in the chosen clauses.
2. Choose $f \in \mathcal{F}_U^G$ with the uniform probability.
3. Choose $g_1 \in \mathcal{F}_W^G$ with the uniform probability.
4. Choose a function $\mu \in \mathcal{F}_W^G$ by setting $\mu(y) = 1^k$ with probability $1 - \epsilon$ and otherwise $\mu(y) = g$ where g is chosen randomly in G. This is done independently for each $y \in \{-1, 1\}^W$.
5. Set $g_2 = (f g_1 \mu)^{-1}$, i.e. define g_2 by for each $y \in \{-1, 1\}^W$, $g_2(y) = (f(y|_U) g_1(y) \mu(y))^{-1}$.
6. Accept if $A_U(f) A_W(g_1) A_W(g_2) = 1^k$.

Completeness follows along the same lines as before.

Lemma 4.13 *The completeness of Test $L^{\epsilon}_{G}(u)$ is at least $1 - \epsilon$.*

In fact the verifier accepts a correct proof with probability $1 - \frac{size(G)-1}{size(G)}\epsilon$.

The soundness lemma is also more or less the usual and is given below. The theorem follows from the two lemmas in the standard way. ∎

Lemma 4.14 *For any $\epsilon > 0$, $\delta > 0$, suppose that the probability that the verifier of Test $L^{\epsilon}_{G}(u)$ accepts is $(1 + \delta)/size(G)$. Then there is a strategy for P_1 and P_2 in the u-parallel two prover protocol that makes the verifier of that protocol accept with probability $\delta^2 \epsilon / (2 size(G)^2)$.*

Proof: We very much follow the same approach as in the mod p case. Remember that for $\gamma = (\gamma_j)_{j=1}^k$ and $g \in G$, g^γ is the complex number $\prod_{j=1}^k (g^{(j)})^{\gamma_j}$.

The test succeeds if $A_U(f)A_W(g_1)A_W(g_2)$ is 1^k and fails otherwise. To evaluate the general performance we want to convert this to a rational number and thus we consider

$$\sum_{\gamma \neq 0^k} (A_U(f)A_W(g_1)A_W(g_2))^\gamma$$

which is $|G|-1$ if the test accepts and -1 otherwise. By the assumption of the lemma we have

$$E_{U,W,f,g_1,\mu}\left[\sum_{\gamma \neq 0^k} (A_U(f)A_W(g_1)A_W(g_2))^\gamma\right] \geq \delta. \tag{23}$$

Now we can investigate the Fourier transform of A_U^γ and A_W^γ and we are on familiar ground. The functions A_U^γ are related but this does not play any role in the argument. Fix U and W and let A denote the function A_U^γ and B denote the function A_W^γ, then the term corresponding to γ in (23) is in fact given by

$$E_{U,W,f,g_1,\mu}\left[\sum_{\alpha,\beta_1,\beta_2} \hat{A}_\alpha \hat{B}_{\beta_1}\hat{B}_{\beta_2}\chi_\alpha(f)\chi_{\beta_1}(g_1)\chi_{\beta_2}(g_2)\right] =$$

$$\sum_{\alpha,\beta_1,\beta_2} \hat{A}_\alpha \hat{B}_{\beta_1}\hat{B}_{\beta_2} E_{U,W,f,g_1,\mu}\left[\prod_x f(x)^{\alpha(x)} \prod_y g_1(y)^{\beta_1(y)} g_2(y)^{\beta_2(y)}\right] =$$

$$\sum_{\alpha,\beta_1,\beta_2} \hat{A}_\alpha \hat{B}_{\beta_1}\hat{B}_{\beta_2} E_{U,W,f,g_1,\mu}\left[\prod_x f(x)^{\alpha(x)} \prod_y g_1(y)^{\beta_1(y)} (f(y|_U)g_1(y)\mu(y))^{-\beta_2(y)}\right].$$

The inner expected value is again 0 unless $\beta_1 = \beta_2 = \beta$. This follows since $E_g[g^\gamma] = 0$ for any $\gamma \neq 0^k$. Since f and μ are independent we can study

$$E\left[\prod_x f(x)^{\alpha(x)} \prod_y f(y|_U)^{-\beta(y)}\right] \tag{24}$$

and

$$E\left[\prod_y \mu(y)^{-\beta(x)}\right]$$

separately. Since the values of f at different points are independent (24) is zero unless

$$\sum_{y \in \pi^{-1}(x)} \beta(y) = \alpha(x)$$

which is simply saying that $\pi_G(\beta) = \alpha$.

If we let $s(\beta)$ denote the number of y such that $\beta(y) \neq 0^k$ then

$$E\left[\prod_y \mu(y)^{-\beta(y)}\right] = (1-\epsilon)^{s(\beta)}.$$

Summing up, the term corresponding to γ in the expression (23) equals

$$\sum_{\alpha,\beta,\pi_G(\beta)=\alpha} \hat{B}_\beta^2 \hat{A}_\alpha (1-\epsilon)^{s(\beta)},$$

where $\hat{A}_\alpha$ and $\hat{B}_\beta$ are the Fourier coefficients of A_U^γ and A_W^γ respectively. Now simply choose the γ that maximizes this sum. We proceed to define successful strategies for the provers in a way similar to before. We omit the details. ∎

Before we continue let us just note that Theorem 4.9 and Theorem 4.12 can be extended to more variables in each equation yielding similar results as Theorem 4.5. We omit the details.

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 [14]) to get an approximation algorithm that performs as well as for Max-Cut. To get an improved lower bound we give a reduction from Max-E3-Lin-2.

The most common type of reduction is to use a gadget which is a way to transform one local condition from one problem to one or more local conditions of the other. In our case we reduce from Max-E3-Lin-2 and thus we proceed as follows. Given an equation $x + y + z = c$ we construct a constant number of constraints of the other problem involving the variables x, y, z and possibly some new auxiliary variables. These constraints come with weights. It is an α-gadget if for any x, y, z that satisfies $x+y+z = c$ one can adjust the auxiliary variables to satisfy constraints of total weight α while if $x+y+z \neq c$ then the maximum obtainable is $\alpha - 1$. For a more thorough discussion of gadgets we refer to [25].

We have the following

Lemma 4.15 *Suppose there is an α-gadget reducing Max-E3-Lin-2 to an optimization problem O. Then, unless NP=P, for any ϵ, O cannot be approximated within $\frac{2\alpha}{2\alpha-1} - \epsilon$ in polynomial time.*

Proof: (Sketch) This is Lemma 2.8 of [25] and but since the proof is simple let us, for completeness, ate least give a sketch.

We use the gadget to construct an instance of O. If the total weight of the Max-E3-Lin-2 instance is 1 then for any solution that satisfies equations of total weight w, the corresponding solution of the transformed problem satisfies constraints of total weight $w\alpha + (1-w)(\alpha - 1)$. Now since it is NP-hard to

distinguish the two cases when $w = 1 - \epsilon$ and $w = \frac{1}{2} + \epsilon$, if we could determine the optimum of the transformed problem to a better accuracy than

$$\frac{(1-\epsilon)\alpha + \epsilon(\alpha - 1)}{(1/2+\epsilon)\alpha + (1/2-\epsilon)(\alpha-1)}$$

we would solve an NP-hard problem. Since ϵ was arbitrary, the lemma follows. ∎

Using this we have

Theorem 4.16 *For any $\epsilon > 0$ is is NP-hard to approximate Max-E2-Lin-2 within a factor $12/11 - \epsilon$.*

Proof: This follows from a reduction from Max-E3-Lin-2. We use a gadget constructed by Sorkin [24] using the techniques of [25]. We start with an equation of the form $x_1 x_2 x_3 = 1$. The set of equations we construct have variables which are best imagined as sitting at the corners of a three-dimensional cube. For each $\alpha \in \{0,1\}^3$ we have a variable y_α. For each edge (α_1, α_2) of the cube we have the equation

$$y_{\alpha_1} y_{\alpha_2} = -1$$

and for each main diagonal (α, α') we have the equation

$$y_\alpha y_{\alpha'} = 1.$$

Since a cube has 12 edges and 4 main diagonals we get a total of 16 equations each of which we give weight 1/2. We let x_1 take the place of y_{011}, x_2 the place of y_{101} and x_3 the place of y_{110}. The variable y_{000} is replaced by z which is the same variable for all local reductions, while all the other variables are distinct in the different gadgets. The role of the variable z is to define 1. The reason for this is that solutions to Max-E2-Lin-2 problems come in pairs in that complementing all variables does not change the set of satisfied equations. Thus the solution we choose is the one with $z = 1$.

Let us consider the assignments that satisfy $x_1 x_2 x_3 = 1$. Either the variables all take the value 1 or exactly two take the value -1. In the former case we assign y_α the value $(-1)^{\alpha_1+\alpha_2+\alpha_3}$. While in the second case, assuming $x_1 = 1$ the other cases being symmetric we assign it the value $(-1)^{\alpha_2+\alpha_3}$. In the first case we satisfy all the "edge equations" while in the second case we satisfy 8 "edge equations" and all "diagonal equations" and thus in either case we satisfy 12 equations. When $x_1 x_2 x_3 = -1$ an enumeration establishes that we can only satisfy 10 equations. Thus we have constructed a 6-gadget and the lemma follows from Lemma 4.15. ∎

5 Satisfiability problems

We start with a direct consequence of Theorem 4.4.

Theorem 5.1 *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 Max-E3-Lin-2. An equation $xyz = 1$ 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. Thus we have constructed a trivial 4-gadget and the result follows by Lemma 4.15. ∎

Something that is not desirable in the above reduction is that we show that it is hard to distinguish at least $1 - \epsilon$-satisfiable formulas from at most $7/8 + \epsilon$ satisfiable formulas. We would prefer to have the first class be the set of satisfiable formulas. The proof of this is more complicated. Since the proof is far simpler for the case of 4-SAT formulas we first establish this result.

Theorem 5.2 *For any $\epsilon > 0$ it is NP-hard to distinguish satisfiable E4-SAT formulas from $15/16 + \epsilon$ satisfiable E4-SAT formulas.*

Proof: We first define the test.

Test 4S(u)

Written proof. A SWP(u) where all tables are folded over 1 and tables A_W, for sets W defined by clause $(C_{i_j})_{i=1}^{u}$ conditioned upon $\wedge_{i=1}^{u} C_{i_j}$.
Desired property. To check that it is a correct SWP(u) for a given formula $\varphi = C_1 \wedge C_2 \ldots C_m$.
Verifier.

1. Choose u random clauses $(C_{j_i})_{i=1}^{u}$ with uniform probability and for each i a random variable x_{k_i} occurring in C_{j_i} again with uniform probability. Set $U = \{x_{i_1}, x_{i_2} \ldots x_{i_u}\}$ and W to be the set of all variables in the chosen clauses.

2. Choose $f \in \mathcal{F}_U$ with the uniform probability.

3. Choose $g_1, g_2 \in \mathcal{F}_W$ independently with the uniform probability.

4. Choose function $g_3 \in \mathcal{F}_W$ by for each $y \in \{-1, 1\}^W$ independently doing the following. If $g_1(y) = -1$ then set $g_3(y)$ randomly while if $g_1(y) = 1$ set $g_3(y) = -f(y|_U)g_2(y)$.

5. Accept unless $A_U(f) = A_W(g_1) = A_W(g_2) = A_W(g_3) = 1$.

It is not hard to see that we get perfect completeness and we omit the proof of the lemma below.

Lemma 5.3 *The completeness of test 4S(u) is 1.*

The key lemma of the usual type proving Theorem 5.2 is given below. We again omit the details how to prove the theorem. ∎

Lemma 5.4 *If Test $4S(u)$ accepts with probability $15/16 + \epsilon$, then there is a strategy for P_1 and P_2 in the u-parallel two prover protocol that makes the verifier of that protocol accept with probability $8\epsilon^2$.*

Proof: We have that

$$1 - \frac{1}{16}(1 + A_U(f))(1 + A_W(g_1))(1 + A_W(g_2))(1 + A_W(g_3)) \tag{25}$$

is 1 if the test accepts and 0 otherwise. We need to estimate the expected value of (25) which gives the probability of success. We expand the product and estimate the expected value of each term separately. The only terms that can have a nonzero expected values are terms containing both $A_W(g_2)$ and $A_W(g_3)$. This follows since the collections (f, g_1, g_2) and (f, g_1, g_3) form independent random variables and the expected value of each single factor in a term is 0. Fix U and W and let the corresponding tables be denoted by A and B. Thus the expected value of (25) is in fact equal to

$$\begin{aligned} &\frac{15}{16} - \frac{1}{16}(E[B(g_2)B(g_3)] + E[A(f)B(g_2)B(g_3)] \\ &+E[B(g_1)B(g_2)B(g_3)] + E[A(f)B(g_1)B(g_2)B(g_3)]), \end{aligned} \tag{26}$$

where the expected values are taken over the choice of f, g_1, g_2 and g_3. Test 4S(u) is equally likely to produce the set (f, g_1, g_2, g_3) and $(-f, g_1, g_2, -g_3)$. Since both A and B are folded over 1 this implies that $E[B(g_2)B(g_3)] = 0$ and the same applies to $E[B(g_1)B(g_2)B(g_3)]$. Of the two remaining term let us first consider $E[A(f)B(g_1)B(g_2)B(g_3)]$ which is the most difficult to estimate. We substitute the Fourier expansion and use the linearity of expectation to obtain

$$\sum_{\alpha,\beta_1,\beta_2\beta_3} \hat{A}_\alpha \hat{B}_{\beta_1} \hat{B}_{\beta_2} \hat{B}_{\beta_3} E_{f,g_1,g_2,g_3} \left[\prod_{x\in\alpha} f(x) \prod_{y\in\beta_1} g_1(y) \prod_{y\in\beta_2} g_2(y) \prod_{y\in\beta_3} g_3(y) \right].$$

Any term with $\beta_2 \neq \beta_3$ has expected value 0. Also if β_1 is not a subset of $\beta = \beta_2 = \beta_3$ the corresponding term also has expected value 0. This follows since for $x \in \beta_1 - \beta$, $g_1(x)$ is random and independent of all other variables. Since elements with different projection onto U are independent we need to estimate

$$E_{f,g_1,g_2,g_3} \left[\prod_{y\in\beta_1} g_1(y) \prod_{y\in\beta} g_2(y)g_3(y) \right] \tag{27}$$

and

$$E_{f,g_1,g_2,g_3} \left[f(x) \prod_{y\in\beta_1} g_1(y) \prod_{y\in\beta} g_2(y)g_3(y) \right] \tag{28}$$

where $\beta_1 \subseteq \beta$ and all elements of β project onto a fixed element, x, of U. Let us divide this into cases depending on the value of g_1. If $g_1(y) = -1$ for some

$y \in \beta$ the expected value over the rest is 0 and thus we only care about the case when $g_1(y) = 1$ for all $y \in \beta$. This happens with probability $2^{-size(\beta)}$ and then the first expression is equal to $(-f(x))^{size(\beta)}$ while the second is equal to $f(x)(-f(x))^{size(\beta)}$. This means that (27) equals $2^{-size(\beta)}$ when $size(\beta)$ is even and 0 otherwise while (28) equals $-2^{-size(\beta)}$ when $size(\beta)$ is odd and 0 otherwise. This implies that we get the total estimate

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

The inner sum is bounded using Cauchy-Schwartz inequality as

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

and substituting this we get the bound

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

Before continuing, let us also consider $E[A(f)B(g_1)B(g_2)]$. The calculations are essentially the same except that we do not need to sum over β_1 and thus we get the estimate

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

Thus when fixating U and W the probability of acceptance is at most

$$\frac{15}{16} + \frac{1}{8} \sum_{\alpha} |A_\alpha| \sum_{\beta, \pi_2(\beta)=\alpha} \hat{B}_\beta^2 2^{-size(\beta)/2}.$$

Now define the standard strategy for the provers. P_1 simply picks a random β with probability proportional to $\hat{B}_\beta^2$ and then a random $y \in \beta$. Similarly P_2 picks a random α subject to $|\hat{A}_\alpha| \geq 4\epsilon$ with probability proportional to $|\hat{A}_\alpha|$ and then a random $x \in \alpha$. Thus success of this strategy is at least

$$E_{U,W} \left[4\epsilon \sum_{\alpha, |\hat{A}_\alpha| \geq 4\epsilon} |A_\alpha| \sum_{\beta, \pi_2(\beta)=\alpha} \hat{B}_\beta^2 size(\beta)^{-1}. \right]$$

We have that

$$\sum_{\alpha, |\hat{A}_\alpha| < 4\epsilon} |A_\alpha| \sum_{\beta, \pi_2(\beta)=\alpha} \hat{B}_\beta^2 2^{-size(\beta)/2} < 4\epsilon$$

and since $size(\beta)^{-1} > \frac{1}{2} 2^{-size(\beta)/2}$ for all β,

$$E_{U,W} \left[\sum_{\alpha, |\hat{A}_\alpha| \geq 4\epsilon} |A_\alpha| \sum_{\beta, \pi_2(\beta)=\alpha} \hat{B}_\beta^2 size(\beta)^{-1} \right] \geq$$

$$E_{U,W}\left[\frac{1}{2}\sum_{\alpha,|\hat{A}_\alpha|\geq 4\epsilon}|A_\alpha|\sum_{\beta,\pi_2(\beta)=\alpha}\hat{B}_\beta^2 2^{-size(\beta)/2}\right]\geq 2\epsilon$$

we have that the success rate of this strategy is at least $8\epsilon^2$. ∎

Note that for the constructed functions the quadruple $(f(y|_U), g_1(y), g_2(y), g_3(y))$ never takes any of the values $(1,1,1,1), (1,1,-1,-1), (-1,1,1,-1), (-1,1,-1,1)$. This implies that essentially without changes to the proof we get a more general theorem. Instead of letting the condition on clauses we try to satisfy be conjunction we could look at more general constraints. In particular for any predicate P on $\{-1,1\}^4$ we get a corresponding optimization problem that given a set of quadruples of literals to to get an assignment for which the number of quadruples that satisfy P is maximal. This gives 2^{16} possible problems, many of which are equivalent due to symmetries. For a discussion of this type of problems we refer to [26, 18]. Here we just note that we get optimal results for a number of predicates.

Theorem 5.5 *Let P be a predicate on $\{-1,1\}^4$ such that*

$$P^{-1}(1)\subseteq\{(1,1,1,1),(1,1,-1,-1),(-1,1,1,-1),(-1,1,-1,1)\}$$

is of size f. Then for any $\epsilon>0$, for the corresponding constraint satisfaction problem it is NP-hard to distinguish satisfiable formulas from at most $1+\epsilon-\frac{f}{16}$ satisfiable formulas.

Proof: The test is again $4\mathrm{S}(u)$ and it is obvious that we get perfect completeness. For the soundness we simply write the acceptance criteria as a multilinear expression in $A_U(f)$ and $A_W(g_i)$. We have already established that each multilinear term has small expected value and unless there is a good strategy for P_1 and P_2. We omit the details. ∎

Let us now do the stronger result establishing tight approximability for Max-E3-Sat with perfect completeness.

Theorem 5.6 *For any $\epsilon>0$ it is NP-hard to distinguish satisfiable E3-SAT formulas from $7/8+\epsilon$ satisfiable E3-SAT formulas.*

Proof: While the overall structure of the proof is similar to the previous cases a number of complications arise. The written proof is, however, the same as in the proofs of Theorem 4.4 and Theorem 5.2. We first describe a test with a parameter $\epsilon<1/2$.

Test $\mathbf{3S}^\epsilon(u)$

Written proof. A SWP(u) where all tables are folded over 1 and tables A_W, for sets W defined by clause $(C_{i_j})_{i=1}^u$ conditioned upon $\wedge_{i=1}^u C_{i_j}$.
Desired property. To check that it is a correct SWP(u) for a given formula $\varphi=C_1\wedge C_2\ldots C_m$.
Verifier.

1. Choose u random clauses $(C_{j_i})_{i=1}^{u}$ with uniform probability and for each i a random variable x_{k_i} occurring in C_{j_i} again with uniform probability. Set $U = \{x_{i_1}, x_{i_2} \dots x_{i_u}\}$ and W to be the set of all variables in the chosen clauses.

2. Choose $f \in \mathcal{F}_U$ with the uniform probability.

3. Choose $g_1 \in \mathcal{F}_W$ with the uniform probability.

4. Choose function $g_2 \in \mathcal{F}_W$ by for each $y \in \{-1,1\}^W$ independently doing the following. If $f(y|_U) = 1$ then set $g_2(y) = -g_1(y)$ while if $f(y|_U) = -1$ set $g_2(y) = g_1(y)$ with probability $1-\epsilon$ and otherwise $g_2(y) = -g_1(y)$.

5. Accept unless $A_U(f) = A_W(g_1) = A_W(g_2) = 1$.

It is again easy to see that we get perfect completeness.

Lemma 5.7 *The completeness of Test 3S$^{\epsilon}$(u) is 1.*

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

$$1 - \frac{1}{8}(1 + A_U(f))(1 + A_W(g_1))(1 + A_W(g_2)).$$

We again fix U and W and call the two involved functions A and B. 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 $B(g_2)$ and we end up with

$$\frac{7}{8} - \frac{1}{8}(E[B(g_1)B(g_2)] + E[A(f)B(g_1)B(g_2)]).$$

We consider each term separately. Expanding the first by the Fourier expansion gives that we should estimate

$$\sum_{\beta_1,\beta_2} \hat{B}_{\beta_1}\hat{B}_{\beta_2} E\left[\prod_{y\in\beta_1} g_1(y) \prod_{y\in\beta_2} g_2(y)\right].$$

Now as before any term with $\beta_1 \neq \beta_2$ have expected value 0. The parts of the product with different projections onto U are independent. Thus we need to estimate values of expressions of the form

$$\prod_y g_1(y)g_2(y).$$

where all y project onto the same element x. It is not hard to calculated this expected value to be

$$\frac{1}{2}((-1)^s + (1-2\epsilon)^s).$$

where s is the number of elements the product. For even s this is a number between $1/2$ and 1 and decreasing as a function of s. For s small it is roughly

$1-2s\epsilon$ while if s is $\Omega(\epsilon^{-1})$ it is a constant tending to $1/2$. For odd s it is always between $-1/2$ and 0 and also here decreasing with s and taking the value around $-2s\epsilon$ for small s. For an $x \in \pi(\beta)$ let s_x denote the number of elements of β that project onto x. We thus need to estimate.

$$\sum_{\beta} \hat{B}_\beta^2 \prod_{x \in \pi(\beta)} (\frac{1}{2}((-1)^{s_x} + (1-2\epsilon)^{s_x})). \tag{29}$$

One could have hoped to estimate this sum as a function of ϵ tending to 0 with ϵ. This is not possible since it simply is not true. It turns out, however, that it is easy to control small β and large β and this is very useful since we can later vary ϵ. Let $c > 0$ be a small positive number to be determined, then

Lemma 5.8 *When g_1 and g_2 are chosen as in Test $3S^\epsilon(u)$ we have*

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

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

Proof: We split the sum (29) into three pieces depending on in which of the intervals $[1, \epsilon^{-1/2}]$, $[\epsilon^{-1/2}, \epsilon^{-4/c}]$, and $[\epsilon^{-4/c}, \infty]$ $size(\beta)$ belongs. The middle interval need not be estimated since it appears on the right hand of the estimate in the lemma. For the small β we have

Lemma 5.9

$$\left| \sum_{\beta \mid size(\beta) \leq \epsilon^{-1/2}} \hat{B}_\beta^2 E_{g_1,g_2} \left[\prod_{y \in \beta} g_1(y) g_2(y) \right] \right| \leq \epsilon^{1/2}$$

Proof: Since $size(\beta)$ is odd, some $x \in \pi(\beta)$ must have an odd value of s_x. Hence for this x

$$0 \geq \frac{1}{2}((-1)^{s_x} + (1-2\epsilon)^{s_x}) \geq \frac{1}{2}(-1 + (1-2s_x\epsilon)) = -s_x\epsilon \geq -\epsilon^{1/2}.$$

The absolute value of the other factors is bounded by one and since $\sum_\beta \hat{B}_\beta^2 \leq 1$, the lemma follows. ∎

The part

$$\sum_{\beta \mid size(\beta) \geq \epsilon^{-4/c}} \hat{B}_\beta^2 E \left(\prod_{x \in \pi(\beta)} (\frac{1}{2}((-1)^{s_x} + (1-2\epsilon)^{s_x})) \right)$$

requires some more work to estimate. The product is bounded in absolute value by $(1-\epsilon)^{|\pi(\beta)|}$ and hence it is useful to prove that it is quite unlikely (over the choice of U) that $\pi(\beta)$ is small. To this end we use the below lemma, the proof of which we postpone to an appendix.

Lemma 5.10 *There is a constant $c > 0$ such that*

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

where the expected value is taken over a random set U constructed by picking one variable from each of the clauses of W.

To use the lemma consider

$$\sum_{\beta \mid size(\beta) \geq \epsilon^{-4/c}} \hat{B}_\beta^2 E[(1-2\epsilon)^{|\pi(\beta)|}].$$

From Lemma 5.10 it follows that except with probability $size(\beta)^{-c/2} \leq \epsilon^2$, it is true that $|\pi(\beta)| \geq size(\beta)^{c/2} = \epsilon^{-2}$. This implies that the expected value of the term for β is at most $(\epsilon^2 + (1-\epsilon)^{\epsilon^{-2}})\hat{B}_\beta^2 \leq 2\epsilon\hat{B}_\beta^2$. Summing over β gives Lemma 5.8. ∎

Let us now consider

$$E_{U,W,f,g_1,g_2}[A(f)B(g_1)B(g_2)] \tag{30}$$

in more detail.

Lemma 5.11 *If $|E[A(f)B(g_1)B(g_2]| \geq \delta$ where the expected value is over choosing U, W, f, g_1 and g_2 as in test $3S^\epsilon(u)$, then there is a strategy for P_1 and P_2 in the u-parallel two prover protocol that makes the verifier of that protocol accept with probability $\delta^2(8\epsilon\delta/(8\log(4\delta^{-1})))^{1/c}/4$, where c is the constant from Lemma 5.10.*

Proof: Using the Fourier expansion we transform (30) to

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

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

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

The inner expected value is

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

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

$$\sum_{\beta,\alpha\subseteq\pi(\beta)} \hat{B}_\beta^2 \hat{A}_\alpha h(\alpha,\beta). \tag{31}$$

Now

$$\sum_{\alpha\subseteq\pi(\beta)} |h(\alpha,\beta)| =$$

$$\prod_{x\in\pi(\beta)} (|\frac{1}{2}((-1)^{s_x} - (1-2\epsilon)^{s_x})| + |\frac{1}{2}((-1)^{s_x} + (1-2\epsilon)^{s_x})|) = 1.$$

This means that if we drop all α with $|\hat{A}_\alpha| \leq \delta/4$ from the above sum, we only decrease the sum by at most $\delta/4$.

Before we continue let us just point out that the strategies of the two provers are the standard strategies. I.e. P_2 chooses an α with probability proportional to $|\hat{A}_\alpha|$ and conditioned upon this number being at least $\delta/4$ and return a random $x \in \alpha$. Similarly P_1 chooses a random β with probability proportional to $\hat{B}^2_\beta$ and returns a random $y \in \beta$. The successprobability of this strategy is at least

$$\frac{\delta}{4} \sum_{\beta,\alpha\subseteq\pi(\beta)} \hat{B}^2_\beta \hat{A}_\alpha size(\beta)^{-1}. \tag{32}$$

We have compare this to (31) and the problem is the extra factor which in (32) depends on the size of β while in (31) it depends on the size of $\pi(\beta)$. We do have a connection of these two quantities given by Lemma 5.10 and this is what we use.

The quantity that multiplies $\hat{B}^2_\beta$ in (31) is

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

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

It is not difficult to see that the last sum equals

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

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 $(1-\epsilon)$. Thus if fact we have the upper estimate

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

for (30). This implies that we can drop any term with $size(\pi(\beta)) > 2\epsilon^{-1}\ln(4\delta^{-1})$ on top of those with small $\hat{A}_\alpha$ and still maintain a sum that it at least $\delta/2$. But by Lemma 5.10 the probability that a β with $size(\beta) > (8\delta^{-1}\epsilon^{-1}\ln(4\delta^{-1}))^{1/c}$

will result in such a large projection is bounded by $\delta/4$. This implies that we can also erase all such β from the sum (31) and still maintain a sum that has expected value at least $\delta/4$. But when we restrict the summation over all such small β each term in (32) is at most a factor $(8\delta^{-1}\epsilon^{-1}\ln(4\delta^{-1}))^{1/c}$ smaller than the corresponding term in (31) and the lemma follows. ∎

We are now in position to prove Theorem 5.6. We describe the appropriate test. Given a positive δ which is bounded by $1/2$ we proceed as follows.

Test F3S$^{\delta}(u)$

1. Set $t = \lceil \delta^{-1} \rceil$, $\epsilon_1 = \delta^2$ and $\epsilon_i = \epsilon_{i-1}^{8/c}$ for $i = 2, 3, \ldots t$.
2. Choose a random j, $1 \leq j \leq t$ with uniform distribution. Run test 3S$^{\epsilon_j}(u)$.

First we note that we have perfect completeness.

Lemma 5.12 *The completeness of Test F3S$^{\delta}(u)$ is 1.*

This follows simply from Lemma 5.7. On the soundness side we have the crucial lemma below. We complete the proof proof of Theorem 5.6 in the usual way. We omit the details. ∎

Lemma 5.13 *If the test F3S$^{\delta}(u)$ accepts with probability $(7+5\delta)/8$ then there is a strategy for P_1 and P_2 in the u-parallel two prover protocol that makes the verifier of that protocol accept with probability $2^{-2^{O(\delta^{-1})}}$.*

Proof: The probability of accepting is

$$E_{f,g_1,g_2,U,W}\left[1 - \frac{1}{8}(1 + A_U(f))(1 + A_W(g_1))(1 + A_W(g_2))\right] =$$

$$7/8 - \frac{1}{8}E_{f,g_1,g_2,U,W}[A_W(g_1)A_W(g_2)] - \frac{1}{8}E_{f,g_1,g_2,U,W}[A_U(f)A_W(g_1)A_W(g_2)]$$

where f, g_1 and g_2 are chosen as described in the test. By Lemma 5.8 we have

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

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

since the intervals of summations are disjoint. Thus for some j we must have

$$E_{f,g_1,g_2,U,W}[A_U(f)A_W(g_1)A_W(g_2)] \geq \delta$$

when f, g_1 and g_2 are chosen as in test 3S$^{\epsilon_j}(u)$. For this j we get by Lemma 5.11 a strategy for P_1 and P_2 with success probability $\epsilon_j^{O(1)}$. Now $\epsilon_j = \delta^{2^{O(j)}}$ and since $j \leq t = \lceil \delta^{-1} \rceil$ the lemma follows. ∎

Note that also here we have, as in the proof of perfect completeness for Max-4-Sat, proved slightly more. The reason is that the triples $(f(y|_U), g_1(y), g_2(y))$ never take the values $(1,1,1)$ or $(1,-1,-1)$. Thus we can consider the constraint satisfaction problem associated with the predicate

$$OXR(x_1, x_2, x_3) = x_1 \vee (x_2 \oplus x_3).$$

We have

Theorem 5.14 *Consider the constraint satisfaction problem given by the predicate OXR. For any $\epsilon > 0$ it is NP-hard to distinguish satisfiable formulas from instances where only a fraction $\frac{3}{4} + \epsilon$ can be satisfied.*

Proof: We again do test $\mathrm{F3S}^{\delta}(u)$ and by the observation that $(f(y|U), g_1(y), g_2(y))$ never takes the values $(1,1,1)$ and $(1,-1,-1)$ we get perfect completeness. The proof of the soundness is based on the same observation as the proof of soundness in for Theorem 5.5. We omit the details. ∎

It is not hard to extend Theorem 5.6 to longer clauses.

Theorem 5.15 *For any $\epsilon > 0$ and any $k \geq 3$ it is NP-hard to distinguish satisfiable Ek-SAT formulas from at most $1 - 2^{-k} + \epsilon$ satisfiable Ek-SAT formulas.*

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 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 a slight extension stated below. We omit the straightforward proof.

Theorem 5.16 *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.*

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

Theorem 5.17 *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 Max-E3-Lin-2. Just use the 11-gadget of [6, 25]. ∎

6 Set splitting

The verifier that gives a result for set splitting must be a little bit different for some very basic reasons. Firstly, there is no negation present in set splitting and hence we cannot fold over 1. Secondly, we cannot have the bipartite situation when we ask some questions in A_U and then some questions in A_W. This follows since we need that the acceptance criteria is that not all answers are equal one can cheat such a verifier by making $A_U(f) = 1$ for all f and $A_W(g) = -1$ for all g. We remedy this situation by taking two different sets of type W. First we give the simpler version just establishing that E4-Set splitting is hard to approximate. As in the case for Max-E3-Sat it is more complicated to get perfect completeness.

Theorem 6.1 *For any $\epsilon > 0$, it is NP-hard to approximate E4-Set splitting within a factor $8/7 - \epsilon$.*

Note that this is optimal since a random assignment splits a fraction $7/8$ of the sets on average.

Proof: We first give the test. Assume that ϵ is bounded by $\frac{1}{2}$.

Test $SS^{\epsilon}(u)$

Written proof. A SWP(u) without folded tables but A_W, for sets W defined by clause $(C_{i_j})_{i=1}^{u}$ conditioned upon $\wedge_{i=1}^{u} C_{i_j}$.
Desired property. To check that it is a correct SWP(u) for a given formula $\varphi = C_1 \wedge C_2 \ldots C_m$.
Verifier.

1. Choose u random variables x_{k_i} with the uniform probability. Form W^1 by for each x_{k_i} choosing a random clause $C_{j_i^1}$ that contains x_{k_i} and then letting W^1 be the set of variables in these variables. By a similar and independent procedure produce W^2 by choosing clauses $C_{j_i^2}$.

2. Choose $f \in \mathcal{F}_U$ with the uniform probability.

3. Choose $g_1^1 \in \mathcal{F}_{W^1}$ and $g_1^2 \in \mathcal{F}_{W^2}$ independently with the uniform probability.

4. For $i = 1, 2$, choose a function $\mu^i \in \mathcal{F}_{W^i}$ by setting $\mu^i(y) = 1$ with probability $1 - \epsilon$ and $\mu^i(y) = -1$ otherwise, independently for each $y \in \{-1, 1\}^{W^i}$ and for $i = 1$ and $i = 2$.

5. Set $g_2^1 = f g_1^1 \mu^1$, i.e. define g_2^1 by for each $y \in \{-1, 1\}^{W^1}$, $g_2^1(y) = f(y|_U) g_1^1(y) \mu^1(y)$.

6. Set $g_2^2 = -f g_1^2 \mu^2$, i.e. define g_2^2 by for each $y \in \{-1, 1\}^{W^2}$, $g_2^2(y) = -f(y|_U) g_1^2(y) \mu^2(y)$.

7. Accept if $A_{W^1}(g_1^1)$, $A_{W^1}(g_2^1)$, $A_{W^2}(g_1^2)$, and $A_{W^2}(g_2^2)$ are not all equal.

We have the standard completeness lemma which we, since the situation has changed somewhat, even prove.

Lemma 6.2 *The completeness of Test $SS^\epsilon(u)$ is at least $1-\epsilon$.*

Proof: Assume we have a correct SWP(u). Then we have a global satisfying assignment x and all subtables are long codes of restrictions of x. Assume that $f(x|_U) = 1$, then unless $\mu^2(x|_{W^2}) = -1$ we have that $g_1^2(x|_{W^2}) \neq g_2^2(x|_{W^2})$ which is equivalent to saying that $A_{W^2}(g_1^2) \neq A_{W^2}(g_2^2)$. Similarly if $f(x|_U) = -1$, unless $\mu^1(x|_{W^2}) = -1$ we have $A_{W^1}(g_1^1) \neq A_{W^1}(g_2^1)$. Thus in either case we accept with probability $1-\epsilon$. ∎

For the soundness we have the corresponding lemma below. Theorem 6.1 follows by the standard argument. ∎

Lemma 6.3 *If Test $SS^\epsilon(u)$ accepts with probability $7/8+\delta$, then there is a strategy for P_1 and P_2 in the u-parallel two prover protocol that makes the verifier of that protocol accept with probability $8\delta\epsilon$.*

Proof: Fix U, W^1 and W^2. Denote the supposed long code on W^1 by B and the supposed long code on W^2 by C. Then the expression

$$\begin{aligned} 1 - \frac{1}{16}((1+B(g_1^1))(1+B(g_2^1))(1+C(g_1^2))(1+C(g_2^2))) \\ -\frac{1}{16}((1-B(g_1^1))(1-B(g_2^1))(1-C(g_1^2))(1-C(g_2^2))) \end{aligned} \tag{33}$$

is 1 if the test accepts and 0 otherwise. Expanding (33) we get

$$\begin{aligned} \frac{7}{8} - \frac{B(g_1^1)C(g_1^2)+B(g_2^1)C(g_1^2)+B(g_1^1)C(g_2^2)+B(g_2^1)C(g_2^2)}{8} \\ -\frac{B(g_1^1)B(g_2^1)+C(g_1^2)C(g_2^2)+B(g_1^1)B(g_2^1)C(g_1^2)C(g_2^2)}{8} \end{aligned} \tag{34}$$

and we are interested of the expected value of this. All expected values will not necessarily be small in absolute value, but we only have to worry about each term taking a negative value of large magnitude and it turns out that the expected value of most terms is positive. First we have

Lemma 6.4 *$E[B(g_1^1)C(g_1^2)] \geq 0$ where the expected value is taken over the choice of W^1, W^2, g_1^1, and g_1^2. It is true for any fixed choices of U and f.*

Proof: Once U is fixed, the choices that give $B(g_1^1)$ and $C(g_1^2)$ are independent and done with the same distribution. It follows that

$$E_{W^1,g_1^1,W^2,g_1^2}[B(g_1^1)C(g_1^2)] = E_{W^1,g_1^1}[B(g_1^1)]E_{W^2,g_1^2}[C(g_1^2)] = E_{W^1,g_1^1}[B(g_1^1)]^2 \geq 0.$$

∎

Since also g_2^1 and g_2^2 also are random functions and g_i^1 is independent of g_j^2 for all pairs i and j, Lemma 6.4 takes care of 4 terms in (34). We can handle another two (since f and $-f$ appear with the same probability) by the following lemma.

Lemma 6.5 *$E[B(g_1^1)B(g_2^1)] \geq 0$ where the expected value is taken over the choice of f and g_1^1 and g_2^1. It is true for any fixed choice of U and W^1.*

Proof: By replacing $B(g_i^1)$ by the Fourier expansion and then using the linearity of expectation we see that we need to estimate

$$\sum_{\beta_1,\beta_2} \hat{B}_{\beta_1} B_{\beta_2} E_{g_1^1,\mu^1,f} \left[\prod_{y\in\beta_1} g_1^1(y) \prod_{y\in\beta_2} g_2^1(y) \right]. \tag{35}$$

If $\beta_1 \neq \beta_2$ then the expected value is 0 and thus we need to consider

$$\sum_{\beta} \hat{B}_{\beta_1}^2 E_{g_1^1,\mu^1,f} \left[\prod_{y\in\beta} g_1^1(y) g_2^1(y) \right] = \sum_{\beta} \hat{B}_{\beta_1}^2 E_{\mu^1,f} \left[\prod_{y\in\beta} (-f(y|_U)\mu^1(y)) \right]. \tag{36}$$

This is 0 unless for each $x \in \{-1,1\}^U$ the number of y with $y|_U = x$ is even. In this case the expected value is $(1-2\epsilon)^{size(\beta)}$. Thus the expected value is

$$\sum_{\beta|\pi_2(\beta)=0} \hat{B}_{\beta_1}^2 (1-2\epsilon)^{size(\beta)}, \tag{37}$$

which is clearly positive. This proves the lemma. ∎

All that remains is to analyze the "interesting term", i.e.

$$E_{f,g_1^1,g_2^1,g_1^2,g_2^2} \left[B(g_1^1)B(g_2^1)C(g_1^2)C(g_2^2) \right].$$

We again expand this by the Fourier expansion and thus the linearity of expectation to arrive at

$$\sum_{\beta_1,\beta_2,\gamma_1,\gamma_2} \hat{B}_{\beta_1} B_{\beta_2} \hat{C}_{\gamma_1} C_{\gamma_2} E_{f,g_1^1,g_2^1,g_1^2,g_2^2} \left[\prod_{y\in\beta_1} g_1^1(y) \prod_{y\in\beta_2} g_2^1(y) \prod_{z\in\gamma_1} g_1^2(z) \prod_{z\in\gamma_2} g_2^2(z) \right]. \tag{38}$$

Again we need $\beta_1 = \beta_2$ and $\gamma_1 = \gamma_2$ to have a nonzero expected value. Using this, (38) can be rewritten as

$$\sum_{\beta,\gamma} \hat{B}_{\beta}^2 \hat{C}_{\gamma}^2 E_{f,g_1^1,g_2^1,g_1^2,g_2^2} \left[\prod_{y\in\beta} g_1^1(y) g_2^1(y) \prod_{z\in\gamma} g_1^2(z) g_2^2(z) \right] =$$

$$\sum_{\beta,\gamma} \hat{B}_{\beta}^2 \hat{C}_{\gamma}^2 E_{f,\mu^1,\mu^2} \left[\prod_{y\in\beta} f(y|_U)\mu^1(y) \prod_{z\in\gamma} (-f(z|_U)\mu^2(z) \right]. \tag{39}$$

Now this expected value is 0 unless $\pi_2(\beta) = \pi_2(\gamma)$ in which case it is

$$(-1)^{size(\gamma)}(1-2\epsilon)^{size(\beta)+size(\gamma)}.$$

Thus we get the final result

$$\sum_{\beta,\gamma|\pi_2(\gamma)=\pi_2(\beta)} \hat{B}_\beta^2 \hat{C}_\gamma^2 (-1)^{size(\gamma)}(1-2\epsilon)^{size(\beta)+size(\gamma)}. \tag{40}$$

Note that $\pi_2(\gamma) = \pi_2(\beta)$ ensures that $(-1)^{size(\gamma)} = (-1)^{size(\beta)}$ and thus the result is, as expected, symmetric result in B and C. Since the terms corresponding to even size sets is positive we have now established that based on the hypothesis of the lemma

$$E_{U,W^1,W^2}\left[\sum_{\beta,\gamma:\pi_2(\gamma)=\pi_2(\beta),size(\gamma)\ \text{odd}} \hat{B}_\beta^2 \hat{C}_\gamma^2 (1-\epsilon)^{size(\beta)+size(\gamma)}\right] \geq 8\delta \tag{41}$$

and we are in a position to give the strategy of P_1 and P_2. The strategy for P_1 is the standard, he simply chooses a β of odd size with probability proportional to $\hat{B}_\beta^2$ and then a random element in β. The strategy for P_1 is to choose and α of odd size with probability proportional to

$$E_{W^2}[\sum_{\gamma|\pi_2(\gamma)=\alpha} \hat{C}_\gamma^2]$$

and then answers with a random $x \in \alpha$. The probability of success of this strategy is at least

$$E_{U,W^1,W^2}[\sum_{\beta,\alpha|\pi_2(\beta)=\alpha} \hat{B}_\beta^2 \sum_{\gamma|\pi_2(\gamma)=\alpha} \hat{C}_\gamma^2 size(\beta)^{-1}] =$$

$$E_{U,W^1,W^2}[\sum_{\beta,\gamma|\pi_2(\beta)=\pi_2(\gamma)} \hat{B}_\beta^2 \hat{C}_\gamma^2 size(\beta)^{-1}]$$

where all sums are over odd size sets. Comparing this last sum in the usual way to (41) we get that this is at least $8\delta\epsilon$. The proof of Lemma 6.3 is complete. ∎

We now turn to the case of perfect completeness and hence we want to establish.

Theorem 6.6 *For any $\epsilon > 0$, it is NP-hard to distinguish instances for E4-Set splitting when all sets can be split from instances where the best partition splits only a fraction $7/8 + \epsilon$ of the sets.*

Proof: The proof is, in many respects, very similar to the proof of the corresponding result for Max-E3-Sat and in particular we need a parameterized test. Assume $\epsilon < 1/2$.

Test $PSS^\epsilon(u)$

Written proof. A SWP(u) without folded tables but A_W, for sets W defined by clause $(C_{i_j})_{i=1}^u$ conditioned upon $\wedge_{i=1}^u C_{i_j}$.
Desired property. To check that it is a correct SWP(u) for a given formula $\varphi = C_1 \wedge C_2 \ldots C_m$.
Verifier.

1. Choose u random variables x_{k_i} with the uniform probability. Form W^1 by for each x_{k_i} choosing a random clause $C_{j_i^1}$ that contains x_{k_i} and then letting W^1 be the set of variables in these variables. By a similar and independent procedure produce W^2 by choosing clauses $C_{j_i^2}$.
2. Choose $f \in \mathcal{F}_U$ with the uniform probability.
3. Choose $g_1^1 \in \mathcal{F}_{W^1}$ and $g_1^2 \in \mathcal{F}_{W^2}$ independently with the uniform probability.
4. Define g_2^1 by the following procedure. If $f(y|_U) = -1$ then set $g_2^1(y) = -g_1^1(y)$ and otherwise set $g_2^1(y) = g_1^1(y)$ with probability $1 - \epsilon$ and otherwise $g_2^1(y) = -g_1^1(y)$.
5. Define g_2^2 by the following procedure. If $f(y|_U) = 1$ then set $g_2^2(y) = -g_1^2(y)$ and otherwise set $g_2^2(y) = g_1^2(y)$ with probability $1 - \epsilon$ and otherwise $g_2^2(y) = -g_1^2(y)$.
6. Accept if $A_{W^1}(g_1^1)$, $A_{W^1}(g_2^1)$, $A_{W^2}(g_1^2)$, and $A_{W^2}(g_2^2)$ are not all equal.

Completeness is straightforward.

Lemma 6.7 *The completeness of Test PSS$^\epsilon$(u) is* 1.

Proof: Assume we have a correct SWP(u). Then we have a global satisfying assignment x and all subtables are long codes of restrictions of x. If $f(x|_U) = 1$, then $A_{W^2}(g_1^2) \neq A_{W^2}(g_2^2)$ and otherwise $A_{W^1}(g_1^1) \neq A_{W^1}(g_2^1)$. ∎

Next we need to analyze the soundness and hence estimate (34). Some work follows from what we have done before.

Lemma 6.8 *$E[B(g_1^1)C(g_1^2)] \geq 0$ where the expected value is taken over the choice of W^1, W^2, g_1^1, and g_1^2. It is true for any fixed choices of U and f.*

The proof is virtually the same as that of Lemma 6.4 which just depends on the fact that $B(g_1^1)$ and $C(g_1^2)$ are identically distributed and independent. We omit it.

Next observe that g_1^1 and g_2^1 in test PSS$^\epsilon$(u) are taken with exactly the same distribution as g_1 and g_2 in test 3S$^\epsilon$(u). The lemma below should hence not come as a surprise.

Lemma 6.9 *When g_1^1 and g_2^1 are chosen as in test PSS$_\epsilon$(u) we have*

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

This is true for a fixed W^1 but it is expected value over the choice of U contained in W^1 as well as the choice of f, g_1^1 and g_2^1

Proof: As observed above $(U, W^1, f, g_1^1, g_2^1)$ have the same distribution as the corresponding objects in test $3S^\epsilon(u)$ and the only reason we cannot simply appeal to Lemma 5.8 is that we are not assuming that B is folded over 1. The only place in the proof of Lemma 5.8 this fact is in the proof of Lemma 5.9 where we use that one of the s_x is odd. However, we need only observe that the terms with all s_x even are positive and hence can safely be disregarded with the present statement of the lemma. ∎

It remains to estimate the most complicated term

$$E_{U,W^1,W^2,f,g_1^1,g_2^1,g_1^2,g_2^2}\left[B(g_1^1)B(g_2^1)C(g_1^2)C(g_2^2)\right].$$

We get again the expansion (38) and we can again erase terms where $\beta_1 \neq \beta_2$ or $\gamma_1 \neq \gamma_2$ since they evaluate to 0. Thus we need to compute

$$E_{U,W^1,W^2,f,g_1^1,g_2^1,g_1^2,g_2^2}\left[\prod_{y\in\beta} g_1^1(y)g_2^1(y)\prod_{z\in\gamma} g_1^2(z)g_2^2(z)\right]. \tag{42}$$

The only dependencies that exist are in the case of inputs with projection onto the same elements x. For each x let s_x be the number of $y \in \beta$ with $y|_U = x$ and t_x the number of $z \in \gamma$ with $z|_U = x$. A straightforward calculations shows that (42) equals

$$\prod_x (\frac{1}{2}((-1)^{s_x}(1-2\epsilon)^{t_x} + (1-2\epsilon)^{s_x}(-1)^{t_x})). \tag{43}$$

Thus we want to estimate

$$E_{U,W^1,W^2}\left[\sum_{\beta,\gamma} \hat{B}_\beta^2 \hat{C}_\gamma^2 \prod_x (\frac{1}{2}((-1)^{s_x}(1-2\epsilon)^{t_x} + (1-2\epsilon)^{s_x}(-1)^{t_x}))\right]. \tag{44}$$

To estimate this we again divide the sum into three pieces. The middle piece is left untouched while we treat the two endpieces.

Lemma 6.10 *If we restrict summation to terms where $size(\beta)$ or $size(\gamma)$ is at least $\epsilon^{-4/c}$, where c is the constant from Lemma 5.10, then*

$$\left|E_{U,W^1,W^2}\left[\sum_{\beta,\gamma} \hat{B}_\beta^2 \hat{C}_\gamma^2 \prod_x (\frac{1}{2}((-1)^{s_x}(1-2\epsilon)^{t_x} + (1-2\epsilon)^{s_x}(-1)^{t_x}))\right]\right| \leq 4\epsilon. \tag{45}$$

Proof: Suppose $size(\gamma) \geq \epsilon^{-4/c}$. First note that for any x with $t_x \neq 0$ the corresponding factor is bounded in absolute value by $(1-\epsilon)$ and hence the sum of the lemma is bounded by

$$\sum_{\beta,\gamma} \hat{B}_\beta^2 \hat{C}_\gamma^2 (1-\epsilon)^{size(\pi(\gamma))}.$$

Now by Lemma 5.10 we know that the probability (over the choice of U) that $size(\pi(\gamma))$ is smaller than ϵ^{-2} is bounded by ϵ^2. This implies that

$$E_U[\hat{B}_\beta^2\hat{C}_\gamma^2(1-\epsilon)^{size(\pi(\gamma))}] \leq (\epsilon^2 + (1-\epsilon)^{\epsilon^{-2}})\hat{B}_\beta^2\hat{C}_\gamma^2 \leq 2\epsilon\hat{B}_\beta^2\hat{C}_\gamma^2$$

and the lemma follows by the linearity of expectation, a similar calculation when $size(\beta)$ is large, and that $\sum_{\beta,\gamma}\hat{B}_\beta^2\hat{C}_\gamma^2 = 1$. ∎

Next we have

Lemma 6.11 *If we restrict summation to terms where $size(\beta)$ or $size(\gamma)$ is at most $\epsilon^{-1/2}$ then*

$$\sum_{\beta,\gamma}\hat{B}_\beta^2\hat{C}_\gamma^2\prod_x(\frac{1}{2}((-1)^{s_x}(1-2\epsilon)^{t_x} + (1-2\epsilon)^{s_x}(-1)^{t_x})) \geq$$
$$-(\sqrt{\epsilon} + \sum_{\beta,\gamma|\pi(\beta)\cap\pi(\gamma)\neq\emptyset}\hat{B}_\beta^2\hat{C}_\gamma^2), \quad (46)$$

where the sum on the right hand side has the size restriction on β and γ.

Proof: Any term with all s_x and t_x even is positive and any term with s_x and t_x both nonzero contributes to the sum on the right hand size of the inequality. We hence only have to bound terms not satisfying either of these properties. Assume without loss of generality that s_x is odd and t_x is 0. Then

$$\frac{1}{2}((-1)^{s_x}(1-2\epsilon)^{t_x} + (1-2\epsilon)^{s_x}(-1)^{t_x}) = \frac{1}{2}((1-2\epsilon)^{s_x} - 1),$$

which, since $(1-2\epsilon)^{s_x} \geq 1 - 2s_x\epsilon$, is a number between 0 and $-\sqrt{\epsilon}$. Thus the terms we are interested in gets multiplied by a number of absolute value at most $\sqrt{\epsilon}$. Since $\sum_{\beta,\gamma}\hat{B}_\beta^2\hat{C}_\gamma^2 = 1$ the lemma follows. ∎

Now consider the following strategies for P_1 and P_2. P_1 chooses a random β of size at most $\epsilon^{-1/2}$ with probability proportional to $\hat{B}_\beta^2$ and answers with a random $y \in \beta$. P_2 chooses a random α of size at most $\epsilon^{-1/2}$ with probability proportional to $E_{W^2}[\sum_{\gamma|\pi(\gamma)=\alpha} C_\gamma^2]$. The success probability of this strategy is at least

$$\epsilon E_{W^2}[\sum_{\beta,\gamma|\pi(\beta)\cap\pi(\gamma)\neq\emptyset}\hat{B}_\beta^2\hat{C}_\gamma^2]$$

where the sum is over sets β and γ of size at most ϵ^{-1}. The work done so far can be summarized as follows.

Lemma 6.12 *Let Acc be the accept probability of V in the u-parallel two two interactive proof with optimal P_1 and P_2 then*

$$E_{U,W^1,W^2,f,g_1^1,g_2^1,g_1^2,g_2^2}\left[B(g_1^1)B(g_2^1)C(g_1^2)C(g_2^2)\right] \geq$$
$$-(5\sqrt{\epsilon} + \epsilon^{-1}Acc + 2E_{W^1}[\sum_{\beta|\epsilon^{-1/2}\leq size(\beta)\leq\epsilon^{-4/c}}\hat{B}_\beta^2]. \quad (47)$$

Proof: We have seen that the left hand size equals (44). The terms when the size of both sets are bounded $\epsilon^{-1/2}$ is handled by Lemma 6.11 and the prover strategy given after the lemma. The case when either set is of size at least $\epsilon^{-4/c}$ is handled by Lemma 6.10 while the remaining terms are bounded, using $\sum_\gamma \hat{C}_\gamma^2 = 1$ by the sum on the right hand side of (47). ∎

We are now ready to give the test to prove Theorem 6.6.

Test FSS$^\delta(u)$

1. Set $t = \lceil 2\delta^{-1} \rceil$, $\epsilon_1 = \delta^2/25$ and $\epsilon_i = \epsilon_{i-1}^{8/c}$ for $i = 2, 3, \ldots t$.
2. Choose a random j, $1 \leq j \leq t$ with uniform distribution. Run test PSS$^{\epsilon_j}(u)$.

First we note that we have perfect completeness.

Lemma 6.13 *The completeness of test FSS$^\delta(u)$ is 1.*

For the soundness we have the crucial lemma below and this proves Theorem 6.6 by the standard argument. We omit the details. ∎

Lemma 6.14 *If the test FSS$^\delta(u)$ accepts with probability $(7+5\delta)/8$ then there is a strategy for P_1 and P_2 in the u-parallel two prover protocol that makes the verifier of that protocol accept with probability $2^{-2^{O(\delta^{-1})}}$.*

Proof: We have by Lemma 6.9 that when g_1 and g_2 are taken as in test FSS$^\delta$ then

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

$$-(3\epsilon_1^{1/2} + \frac{1}{t}) \geq -2\delta$$

since the summation intervals are disjoint. Using Lemma 6.12 and doing a similar calculation we conclude that $E[B(g_1^1)B(g_2^1)C(g_1^2)C(g_2^2)$ it at least

$$-(5\sqrt{\epsilon} + \epsilon_t^{-1}Acc + \frac{2}{t}) \geq -(2\delta + \epsilon_t^{-1}Acc).$$

We conclude that $Acc \geq \epsilon_j\delta$ and the lemma follows from using $\epsilon_j \geq 2^{-2^{O(\delta^{-1})}}$. ∎

Let us collect some additional theorems that we obtain without too much effort from the proof.

Theorem 6.15 *Let P be a predicate on $\{-1,1\}^4$ such that*

$$\{(1,1,1,1),(-1,-1,-1,-1)\} \subseteq P^{-1}(1) \subseteq$$

$$\subseteq \{(1,1,1,1),(1,1,-1,-1),(-1,-1,1,1),(-1,-1,-1,-1)\}$$

and this set is of size f. Then for any $\epsilon > 0$, for the corresponding constraint satisfaction problem for monotone formulas, it is NP-hard to distinguish satisfiable formulas from at most $1+\epsilon-\frac{f}{16}$ satisfiable formulas.

Proof: The proof is based on the same observation as the proof of Theorem 5.5 but since we did not have to consider all terms when we analyzed set splitting we give some details.

The test we apply is $\mathrm{FSS}^{\delta}(u)$ with the acceptance criteria that P should hold for the quadruple $(A_{W^1}(g_1^1), A_{W^1}(g_2^1), A_{W^2}(g_1^2), A_{W^2}(g_2^2))$ and it is easy to see that we get perfect completeness.

Let us first analyze the soundness in the case when

$$P^{-1}(1) = \{(1,1,1,1),(-1,-1,1,1),(-1,-1,-1,-1)\}.$$

When writing this acceptance criteria a multilinear expression (again with B and C for A_{W^1} and A_{W^2} respectively) we obtain

$$\begin{aligned} &\frac{13}{16} + \frac{B(g_1^1)+B(g_2^1)-C(g_1^2)-C(g_2^2)}{16} \\ &-\frac{B(g_1^1)C(g_1^2)+B(g_2^1)C(g_1^2)+B(g_1^1)C(g_2^2)+B(g_2^1)C(g_2^2)}{16} \\ &-\frac{3(B(g_1^1)B(g_2^1)+C(g_1^2)C(g_2^2))}{16} \\ &-\frac{B(g_1^1)B(g_2^1)(C(g_1^2)+C(g_2^2))-(B(g_1^1)+B(g_2^1))C(g_1^2)C(g_2^2)}{16} \\ &-\frac{B(g_1^1)B(g_2^1)C(g_1^2)C(g_2^2)}{8} \end{aligned} \tag{48}$$

Most of terms we have analyzed and we only have to observe that by symmetry $E[B(g_i^1)] = E[C(g_i^2)]$ and $E[B(g_1^1)B(g_2^1)C(g_1^2)] = E[B(g_1^1)C(g_1^2)C(g_2^2)]$ and hence the new terms cancel. The rest of the proof is unchanged. ∎

If we allow negation we can fold the tables over one and we obtain.

Theorem 6.16 *Let P be a predicate on $\{-1,1\}^4$ such that*

$$P^{-1}(1) \subseteq \{(1,1,1,1),(1,1,-1,-1),(-1,-1,1,1),(-1,-1,-1,-1)\}$$

and this set is of size f. Then for any $\epsilon > 0$, for the corresponding constraint satisfaction problem, it is NP-hard to distinguish satisfiable formulas from at most $1+\epsilon-\frac{f}{16}$ satisfiable formulas.

Proof: Apply test $\mathrm{FSS}^{\delta}(u)$ except that tables are folded over 1. Perfect completeness is obvious. Furthermore for any P we get an expression similar to (48) to describe when the test accepts. Folding lets us conclude that all terms except $B(g_1^1)B(g_2^1)$, $C(g_1^2)C(g_2^2)$ and $B(g_1^1)B(g_2^1)C(g_1^2)C(g_2^2)$ have expected value 0. We need just observe that these three terms appear with negative sign for any of the above P and the rest of the proof follows similarly to the proof of Theorem 6.6. ∎

7 Results for other problems

We use a general method for converting efficient PCPs for NP-problems to lower bounds for vertex cover and we get.

Theorem 7.1 *For any $\epsilon > 0$ it is NP-hard to approximate vertex cover with $7/6 - \epsilon$.*

Proof: This follows from Proposition 11.6 of [6] with $f = 2$ $c = 1 - \epsilon$ and $s = \frac{1}{2} + \epsilon$ and the fact that our PCP which gave the result for Max-E3-Lin-2 used 2 free bits, had completeness $1 - \epsilon$ and soundness $\frac{1}{2} + \epsilon$. For completeness we sketch the proof.

Start with test $\mathrm{L}^{\epsilon}(u)$. We create a graph (as first done in [11]) whose nodes are given by accepting views of the verifier. A view is determined by the random coins flipped by V and the bits read in the proof. If V flips r coins the total number of nodes is 2^{r+2} since the third bit read in the proof is determined by the previous two and the fact that the verifier should accept. Draw an edge between two nodes if they are conflicting in that the two views examine the same bit but this bit takes different values in the An independent set in this graph corresponds to a written proof and the size of this independent set is 2^r times the probability that the verifier accepts this proof. Thus when the formula φ is satisfiable there is an independent set of size $2^r(1 - \epsilon)$ while when it is not the size of the independent set is at most $2^r(\frac{1}{2} + \epsilon)$. Since a set of nodes is a vertex cover iff its complement is an independent sets, in the two cases we have vertex covers of sizes $2^r(3 + \epsilon)$ and $2^r(\frac{7}{2} - \epsilon)$ respectively. This means that a $7/6 - \delta$-approximation algorithm can, by choosing ϵ sufficiently small, be used to decide an NP-hard question. ∎

By using the gadgets of [25], the optimal result for Max-E3-Lin-2 also give improved inapproximability results for a number of other problems.

Theorem 7.2 *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 7.3 *For any $\epsilon > 0$ it is NP-hard to approximate max-di-cut within $13/12 - \epsilon$.*

Proof: There is a 6.5-gadget [25]. ∎

7.1 Gap between bounds

The results for linear equations, k-SAT, E$k \geq 3$ and for set splitting 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 [14, 10, 4].

	Appr. Upper	Appr. Lower
E2-SAT	1.0741	1.0476
Max-Cut	1.1383	1.0624
Max-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 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 [25] 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.

8 Getting nonconstant ϵ

There is nothing that prevents us from using ϵ and δ that are non-constant in our proofs. Since the acceptance probability of the constructed strategy in the two prover games is a function of ϵ and δ also this probability would decrease with n and hence to get a contradiction we would need a value of u that grows with n and hence the size of the result proofs would not be polynomial. If we are willing to assume a stronger hypothesis than NP$\neq$ P something can still be achieved.

Theorem 8.1 *Assume NP$\not\subseteq$ DTIME($2^{O(\log n \log\log n)}$). Then, there is a constant $c' > 0$ such for $\epsilon = (\log n)^{-c'}$, Max-E3-Lin-2 cannot be approximated within $2 - \epsilon$ in polynomial time.*

Proof: We just apply the proof of Theorem 4.4 with $\epsilon = \delta = (\log n)^{-c'}$. To get a contradiction we need $c_c^u < \delta^3/2$ which is equivalent to $u \geq dc' \log\log n$

for some absolute constant d. The resulting linear system has

$$m^u 2^{2^{3u}} + n^u 2^{2^u}$$

variables and this is $n^{O(\log\log n)}$ provided $3udc_1 < 1$. If this holds, a polynomial time approximation algorithm with approximation ration better than $2 - 2(\log n)^{-c_1}$ would, when applied to the resulting system, solve an NP-hard problem in time $n^{O(\log\log n)}$. The theorem follows. ∎

The proof of Theorem 5.2 has as good constants at that of Theorem 4.4 and hence we have

Theorem 8.2 *Assume NP$\not\subseteq$ 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.*

We omit the proof since it is almost identical to the proof of Theorem 8.1

The situation for Theorem 5.6 is, however, much worse since constants blow up to a much larger extent.

Theorem 8.3 *Assume NP$\not\subseteq$ DTIME($2^{O(\log n \log\log n)}$). There is a constant $c' > 0$ such for $\epsilon^{-1} = c' \log\log\log n$, 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.*

Proof: (Sketch) Choose $u = c \log\log n$ in FSS$^\delta(u)$. The success probability of P_1 and P_2 that is obtained is $2^{-2^{O(\delta)}}$. Provided that this is smaller than c_c^u we get a contradiction and if $\delta^{-1} \leq c' \log\log\log n$ for a suitable constant c' this is true. ∎

9 Concluding remarks

The technique of using Fourier transforms to analyze PCPs seems very strong (see also [15]). 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.

When it comes to constraint satisfaction problems it would be nice to characterize exactly for which problems the naive random algorithm is upto an arbitrary additive ϵ best possible. Zwick [26] has investigated this problems for predicates of three literals but the general situation is still not understood.

Acknowledgements: I am most grateful for discussions on these subjects with Oded Goldreich and Madhu Sudan. They, and also an anonymous referee, have numerous comments earlier versions of this manuscript and this have helped 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. Proceedings of 33rd Annual IEEE Symposium on Foundations of Computer Science, Pittsburgh, 1992, pp 14-23.

[2] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. Proceedings of 33rd Annual IEEE Symposium on Foundations of Computer Science, Pittsburgh, 1992, 2-13.

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

[4] 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.

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

[6] M. Bellare, O. Goldreich and M. Sudan. Free Bits, PCPs and Non-Approximability—Towards tight Results. SIAM Journal on Computing, Volume 27, 1998, pp 804–915.

[7] M. Bellare, S. Goldwasser, C. Lund and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. Proceedings of the 25th Annual ACM Symposium on Theory of Computation, San Diego, 1993, pp 294-304. (See also Errata sheet in Proceedings of the 26th Annual ACM Symposium on Theory of Computation, Montreal, 1994, pp 820).

[8] M. Bellare and M. Sudan. Improved non-approximability results. Proceedings of 26th Annual ACM Symposium on Theory of Computation, Montreal, 1994, pp 184-193.

[9] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs. How to remove intractability. Proceedings of the 20th Annual ACM Symposium on Theory of Computation, Chicago, 1988, pp 113-131.

[10] U. Feige and M. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. 3rd Israeli Symposium on the theory of Computing and Systems, 1995, pp 182-189.

[11] 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.

[12] L. Fortnow, J. Rompel, and M. Sipser. On the power of Multi-Prover Interactive Protocols. Proceedings 3rd IEEE Symposium on Structure in Complexity Theory, pp 156-161, 1988.

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

[14] M. Goemans and D. Williamson. .878-approximation algorithms for Max-Cut and Max-2-SAT. Proceedings of 26th Annual ACM Symposium on Theory of Computation, Montreal, 1994, pp 422-431.

[15] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. Proceedings of 37th Annual IEEE Symposium on Foundations of Computer Science, Burlington, 1996, pp 627-636. See also a more complete version available from ECCC, Electronic Colloquium on Computational Complexity (http://www.eccc.uni-trier.de/eccc).

[16] J. Håstad. Some optimal inapproximability results. Proceedings of 29th Annual ACM Symposium on Theory of Computation, El Paso, 1997, pp 1-10.

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

[18] S. Khanna, M. Sudan and D.P. Williamson A complete classification of the approximability of maximization problems derived from Boolean Constraint satisfaction. Proceedings of the 28th Annual ACM Symposium on Theory of Computing, El Paso, Texas, pp 11-20.

[19] C. Lund, L. Fortnow, H. Karloff and N. Nisan. Algebraic methods for interactive proof systems. Journal of the ACM, Vol 39, No 2, pp 859-868.

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

[21] E. Petrank. The hardness of approximation. 2nd Israeli Symposium on the Theory of Computing and Systems, 1993, pp 275-284.

[22] R. Raz. A parallel repetition theorem. Proceedings of 27th Annual ACM Symposium on Theory of Computation, Las Vegas, 1995, pp 447-456.

[23] A. Shamir. IP=PSPACE. Journal of the ACM, Vol 39, No 2, pp 869-877.

[24] G. Sorkin. Personal communication.

[25] L. Trevisan, G. Sorkin, M. Sudan, and D. Williamson. Gadgets, approximation and linear programming. Proceedings of 37th Annual IEEE Symposium on Foundations of Computer Science, Burlington, 1996, pp 617-626.

[26] U. Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. Accepted to SODA 1998.

A Proof of Lemma 5.10

Remember that we are here dealing with an expected value over the choice of U. We analyze a process where for the clauses of W we pick for one clause at the time which variable to put into U. During this process we keep track of a quantity which initially is $size(\beta)^{-c}$ and in the end takes the value $|\pi(\beta)|^{-1}$. We prove that this quantity is expected to decrease for each step. This is sufficient to prove the lemma.

After for some of the clauses it has been selected which variables to put into U the elements of β fall naturally into groups, where in each group all vectors have the same values on each coordinate put into U. During this procedure we also get duplication of some elements (because the only coordinate in which they differ are decided not to be put in U). We keep only one copy of each duplication. Number the groups in some arbitrary way and suppose that group G_i contains s_i elements. We have $\sum_i s_i \leq size(\beta)$. We want to study

$$F = \frac{1}{\sum_i s_i^c}.$$

As promised F is initially $size(\beta)^{-c}$ since we only have one group with $size(\beta)$ elements. At the end of the process we have $\pi(\beta)$ groups with one element each and thus in this case $F = |\pi(\beta)|^{-1}$, also as promised. Let F^j be the value of F after for the jth clause it has been decided which variable to put into U. We want to prove that $E[F^{j+1}|F^j] \leq F^j$.

Define X^j to be $1/F^j = \sum_i s_i^c$. When forming X^{j+1} we first (for technical reasons) form a smaller number Z^{j+1} as follows. Let Z_i^{j+1} be the contribution

from the elements that constituted the ith group at stage j and let $Z^{j+1} = \sum_i Z_i^{j+1}$. At the $j+1$ step the group G_i either remains one group of size $s_i' \leq s_i$ and in this case $Z_i^{j+1} = s_i'^c$ while if it turns into two groups of size $s_{i,1}$ and $s_{i,2}$ where $s_{i,1} \geq s_{i,2}$ then we set $Z_i^{j+1} = s_{i,1}^c + c_1 s_{i,2} s_i^{c-1}$. Here c_1 (as well as c_2 and c_3 to be introduced shortly) is a constant to be determined. We collect the conditions needed on these constants as we go along. Here we note that Z^{j+1} is a conservative estimate of X^{j+1} provided $c_1 \leq 1$.

We now claim that there is a Y such that $Y \leq c_2 X^j$ and

1. $E[Z^{j+1}|X^j] \geq X^j + Y$.
2. $|Z^{j+1} - X^j| \leq c_3 Y$ always.

Let us first see that this is sufficient. We start by the following easy lemma:

Lemma A.1 *Let Z be a random variable that always takes values in the interval $[1, 1+\delta]$, then $E[1/Z] = 1/E[Z] + h_\delta$ where $0 \leq h_\delta \leq \delta^2$.*

Proof: Let $t_\delta(x)$ be the linear function which gives the tangent to the function $1/x$ at $x = 1 + \delta/2$. Then

$$0 \leq 1/x - t_\delta(x) \leq \delta^2$$

for x in the interval $[1, 1+\delta]$. This implies that

$$E[1/Z] \leq \delta^2 + E[t_\delta(Z)] = \delta^2 + t_\delta(E[Z]) \leq \delta^2 + 1/E[Z].$$

The other inequality follows by convexity. ∎

Let us see how to apply Lemma A.1 to the current situation given that the two conditions above are satisfied. Setting $Z = Z^{j+1}/(X^j - c_3 Y)$ satisfies the hypothesis of Lemma A.1 with $\delta = 2c_3 Y/(X^j - c_3 Y) \leq 4c_3 Y/X^j$ provided $c_2 c_3 \leq 1/2$. We get

$$E[F^{j+1}|F^j] \geq E[1/Z^{j+1}] = (X^j - c_3 Y)^{-1} E[1/Z] \leq$$

$$\leq (X^j - c_3 Y)^{-1}(1/E[Z] + (4c_3 Y/X_j)^2) \leq$$

$$\frac{1}{X^j + Y} + \frac{32 c_3^2 Y^2}{(X^j)^3} \leq \frac{1}{X^j} - \frac{Y}{X^j(X^j + Y)} + \frac{32 c_3^2 Y^2}{(X^j)^3} \leq \frac{1}{X^j} = F^j$$

provided $32 c_3^2 Y (X^j + Y) \leq (X^j)^2$ which is true if $32 c_3^2 c_2 (1 + c_2) \leq 1$.

Thus it remains to establish the claim. Take one group G_i and we are about to decide which one of three coordinate x_{i_1}, x_{i_2} and x_{i_3} should go into U. The elements of G_i give one of eight values to these three coordinates. Assume that the most common and second most common values appear for m_i and f_i elements of G_i, respectively. Remember that Z_i^{j+1} is the contribution of the members of G_i to Z^{j+1} and remember that the contribution after the j'th step is s_i^c.

Let us first estimate the worst case deviation of Z_i^{j+1} from s_i^c. We now for sure that at least one group of size at least m_i remains. We use $m_i \geq \max(f_i, s_i - 7f_i)$ and consider the two cases giving the different alternatives in the maximum. For $f_i \leq s_i/8$ the decrease is, since the derivative of s_i^c is bounded by $8cs_i^{c-1}$ for values larger than $s_i/8$, bounded by $56cf_is_i^{c-1}$ while for $f_i \geq s_i/8$ the decrease is bounded by $(1-(1/8)^c)s_i^c \leq 8(1-(1/8)^c)f_is_i^{c-1}$.

To estimate the maximal increase, note that the size of the larger remaining group is bounded by s_i and the size of the smaller group is at most $4f_i$. Thus the maximal increase is bounded by $4c_1f_is_i^{c-1}$. Provided $56c \leq 4c_1$ and $8(1-(1/8)^c) \leq 4c_1$ the maximal change is always bounded by $4c_1f_is_i^{c-1}$.

Let us now analyze the expected change. With probability at least $1/3$ the two most common values go to different groups creating one group of size at least m_i and another of size at least f_i. If this does not happen we assume that we have the worst case scenario established above. This gives the lower bound for the expected increase when $f < s_i/8$ to be at least

$$(s_i - 7f_i)^c + \frac{1}{3}c_1f_is_i^{c-1} - s_i^c \geq \frac{1}{3}c_1f_is_i^{c-1} - 56f_ics_i^{c-1} \geq \frac{1}{4}c_1f_is_i^{c-1}$$

provided $c_1/12 \geq 56c$. On the other hand when $f_i > s_i/8$ we have an increase which is at least

$$f_i^c + \frac{1}{3}c_1f_is_i^{c-1} - s_i^c = s_i^c((1/8)^c - 1) + \frac{1}{3}c_1f_is_i^{c-1} \geq \frac{1}{4}c_1f_is_i^{c-1},$$

provided $c_1/12 \geq 8(1-(1/8)^c)$.

Setting $Y = \frac{c_1}{4}\sum_i f_is_i^{c-1}$ we achieve that the expected value increases by at least Y. The worst case change is bounded by $4c_1\sum_i f_is_i^{c-1}$ and thus we can use 16 for c_3. Clearly $Y \leq \frac{c_1}{4}\sum s_i^c = \frac{c_1}{4}X_j$ and thus we set $c_2 = c_1/4$. Let us finally collect the remaining desired relations between the chosen constants. We need

1. $c_1 \leq 1$.
2. $c_2c_3 \leq 1/2$.
3. $32c_3^2c_2(1+c_2) \leq 1$.
4. $56c \leq 4c_1$.
5. $8(1-(1/8)^c) \leq 4c_1$.
6. $c_1/12 \geq 56c$.
7. $c_1/12 \geq 8(1-(1/8)^c)$.

We have already determined that $c_3 = 16$ and $c_2 = c_1/4$. We may then choose c_1 small enough to satisfy conditions 1, 2, and 3. When we have chosen c_1 all the remaining conditions are of the form $c \leq a_i$ for some positive numbers a_i and we make simply take the minimum of the a_i. The proof of Lemma 5.10 is complete.