Some optimal inapproximability results

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

February 18, 2002

Abstract

We prove optimal, up to an arbitrary $\epsilon > 0$, inapproximability results for Max-Ek-Sat for $k \geq 3$, maximizing the number of satisfied linear equations in an over-determined 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.

1 Introduction

Many natural optimization problems are NP-hard which implies that they are probably hard to solve exactly in the worst case. In practice, however, it is many times sufficient to get reasonably 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. For a maximization problem we say that an algorithm is a C-approximation algorithm if it, for each instance, produces an solution whose objective value is at least OPT/C where OPT is the global optimum. A similar definition applies to minimization problems.

A 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 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 [25] and one of the main results of this paper is that this is optimal to within an arbitrary additive constant $\epsilon > 0$.

A problem that in many respects is as basic as satisfiability is that of solving a system of linear equations over a field. If all equations can be satisfied simultaneously then a satisfying assignment can be found in polynomial time by Gaussian elimination. 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", where we interpret "best" as satisfying the maximal number of equations. This problem is NP-complete over the field of two elements since already the special case of having equations only 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 assigning random values to the variables. 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. This is true even if each equation only contains exactly three variables.

Other results included in this paper are similar results for linear equations over an arbitrary Abelian group Γ and set splitting of sets of size 4. By reductions we get improved constants for Max-2-Sat, Max-Cut and Max-di-Cut and Vertex Cover. These reductions are all from the problem of satisfying the maximal number of equations in system of linear equations over the field of two elements.

1.1 Short history and our contribution

The question of proving NP-hardness of approximation problems was discussed at length already in the book by Garey and Johnson [19], but really strong results were not obtained until the connection with multiprover interactive proofs was discovered in the seminal paper of Feige et al. [16]. There are a number of variants of multiprover interactive proofs and the two proof models that we use in this paper are that of two-prover interactive proofs and that of probabilistically checkable proofs.

The first model was introduced by Ben-Or et al. [12] 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 studied by Fortnow et al. [18], and it was given its current name in the paper by Arora and Safra [4]. In a PCP the verifier does (few) random spot-checks in a (large) written proof. Note that a two-prover interactive proof can be turned into a PCP simply by writing down the answers of both provers to all possible questions. The verifier would then simply check the answers to the questions it intended to pose. For a complete account of history of the entire area we refer to [9], but let us here give a short account of the path leading to the current results.

The surprising power of multiprover interactive proofs was first established by Babai, Fortnow and Lund [5] by showing that multiprover proofs with a polynomial time verifier could recognize all of NEXPTIME. This was scaled down to give very efficient verifiers for simpler predicates by Babai et al. [6] and the connection to approximability was discovered by Feige et al. [16] in 1990.

To obtain stronger bounds, to weaken assumptions, and to widen the range of problems for which the methods applied, more efficient proofs were sought. Arora and Safra [4] discovered proof composition and were the first to construct PCPs for NP-hard problems with a verifier that used logarithmic randomness and sub-logarithmic query complexity.

The first result proving hardness for the problems we are discussing here was obtained in the fundamental paper by Arora et al. [3] which establishes the celebrated PCP-theorem that states that each language in 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 cannot be approximated within C unless NP=P. The first explicit constant was given by Bellare et al. [10] and based on a slightly stronger hypothesis they achieved the constant 94/93. Bellare and Sudan [11] improved this to 66/65 and the strongest result prior to our results here is by Bellare, Goldreich and Sudan [9] obtaining the bound $80/77 - \epsilon$ for any $\epsilon > 0$.

The last two papers, [11, 9], use a similar approach to ours and let us describe this approach. The starting point is an efficient multiprover protocol, which in our case and in [9] comes naturally from a combination of the basic PCP by Arora et al. mentioned above and the wonderful parallel repetition theorem of Raz [32]. Bellare and Sudan [11] used a different protocol since the theorem by Raz was not known at that point in time.

The multiprover protocol is turned into a PCP by writing down the answers of the provers in coded form. The main source of the improvements of Bellare et al. [9] was the invention of a new code, the marvelous long code. The long code of an input $x \in \{0,1\}^u$ 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 if u is a constant it is of constant size and it is hence, at least in theory, affordable.

When a multiprover protocol is transformed to a PCP by writing down coded versions of the prover's answers the verifier can, if the coding is suitable, perform its verification in the multiprover protocol much more efficiently. The freedom to code the answers might, however, also help a cheating prover in that it can write down a string that is not a correct codeword and the verifier has to make sure that such behavior does not destroy the soundness of the new PCP. This forced previous verifiers under these circumstances to perform two tasks, to check to original conditions of the multiprover protocol and to check that the coding is correct.

We use the same written proof as did Bellare et al. [9] and our improvement comes from the ability to completely integrate the two tasks of checking acceptance in the two-prover protocol with checking that we have a correct coding of the prover's answers. We do not really check that the coding is correct in that all we need is that it is possible, given a written proof for the PCP that convinces the verifier to accept with high probability, to extract a strategy for the provers in the two-prover game. Before such strategies were extracted by looking at the legitimate codewords that were close, (i.e., agreed for more than half the inputs) to the codewords presented by the prover. In our case we instead extract the strategies by looking at discrete Fourier transform of these given codewords.

The written proof is the same in most of our tests yielding inapproximability result for the various problems we study. The acceptance criteria are however, specially designed to suit the targeted optimization problem. For example, for the result for linear equations the verifier decides whether to accept based solely on the exclusive-or of three bits. This philosophy of designing special purpose PCPs for each optimization problem was first done on a major scale by Bellare et al. [9]. It seems like this is required to obtain tight result for the problems discussed in this paper. This special design might make some of our tests seem awkward but this is probably inevitable.

For some other problems, most notably clique [23] (and almost for its relative chromatic number [17]), the optimal results are established by looking at natural parameters of the PCP and in particular by studying the number of free bits read by the verifier. Informally, assuming that a verifier always accepts a correct proof of a correct statement, this number is defined as follows. A bit read in a PCP is not free if, at the time of reading, the verifier will always reject unless it has a prespecified value. If this is not the case the bit is free.

The only problem in our paper that relates in a straightforward way to such natural parameters of a PCP is vertex cover. A PCP that uses f free bits, has completeness c and soundness s gives an inapproximability factor of

$$\frac{2^f - s}{2^f - c}$$

for vertex cover. Our proof system giving the result for linear equations has $f = 2$, $c = 1 - \epsilon$ and $s = 1/2 + \epsilon$ yielding an inapproximability factor arbitrarily close to $7/6$. As this is our only use of free bits we do not define it explicitly but rather refer to [9] for its formal definition as well as a thorough discussion of the free bit concept and its applications to inapproximability results and to the theory of PCPs in general.

1.2 Summary of results

For easy reference we here state most of our results in tabular form. We also compare to the best previous lower bounds as well as the performance ratio of the best polynomial time approximation algorithms. In most cases, the previously best result was obtained by Bellare et al. [9] and for a detailed account of the earlier history of each problem we refer to this paper. For formal definitions of the stated problems we refer to Section 2.2.

The number δ below has the meaning "a positive but unspecified constant" while ϵ can be replaced by any positive constant. The assumption used in all the lower bounds is P$\neq$ NP.

	Upper		Prev. best lower		Our lower
	Constant	Source	Constant	Source	
E3-LIN-2	2	folklore	$\frac{8}{7}-\epsilon$	[9]	$2-\epsilon$
E3-LIN-p	p	folklore	p^{δ}	[1]	$p-\epsilon$
E3-LIN-Γ	$\lvert\Gamma\rvert$	folklore	-		$\lvert\Gamma\rvert-\epsilon$
E2-LIN-2	1.1383	[20]	-		$\frac{12}{11}-\epsilon$
E3-SAT	$\frac{8}{7}$	[25]	$\frac{80}{77}-\epsilon$	[9]	$\frac{8}{7}-\epsilon$
E2-SAT	1.0741	[15]	$\frac{220}{217}-\epsilon$	[9]	$\frac{22}{21}-\epsilon$
E4-Set Splitting	$\frac{8}{7}$	folklore	$1+\delta$	[27]	$\frac{8}{7}-\epsilon$
Max-Cut	1.1383	[20]	$\frac{72}{71}-\epsilon$	[9]	$\frac{17}{16}-\epsilon$
Max-di-Cut	1.164	[20]	$\frac{72}{71}-\epsilon$	[9]	$\frac{12}{11}-\epsilon$
Vertex cover	2	[19, 7, 22]	$\frac{233}{218}-\epsilon$	[9]	$\frac{7}{6}-\epsilon$

Our lower bounds using gadgets (E2-SAT, E2-LIN-2, Max-Cut, Max-di-Cut) rely on the gadgets produced by Trevisan et al. [35] and since the prior published work in some cases depended on worse gadgets the improvement are not only due to our results.

The 2-approximation algorithm for vertex cover is an unpublished result due to Gavril that is given in [19]. The case of weighted graphs was treated by Bar-Yehuda and Even [7] and Hochbaum [22].

The inapproximability result for linear systems of equations mod p of Amaldi and Kann [1] needed arbitrary systems of linear equations mod p and hence did not, strictly speaking, apply to Max-E3-Lin-p.

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 our PCPs use the same written proof and in Section 3 we describe this proof. In Section 4 we describe tests for being a correct long code. These tests are presented for pedagogical reasons but are in Section 5 naturally extended to give the results for linear equations. In Section 6 we give the results on Max-k-Sat and in Section 7 we give corresponding results for Set Splitting. We obtain some results for other problems in Section 8. We finally briefly discuss how to make our arbitrary constants be functions of the input-length in Section 9 and end by some concluding remarks.

This is the complete version of the results announced in [24].

2 Notation and some essential previous results

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

2.1 Basic notation

All logarithms in this paper are to the base 2. We use vertical bars $|\cdot|$ to denote the size of an object. For a real or complex number it is the absolute value, for strings it is the length and for sets the size. We use the notation $\alpha \Delta \beta$ for two sets α and β to denote the symmetric difference, i.e., the elements that appear in exactly one of the sets α and β. The notation $\alpha \setminus \beta$ denotes the elements in α but not in β.

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 range 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 an empty product takes the value 1. The expected value of a variable X is denoted 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 in our notation the same as 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 the 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 occurring in U by $x|_U$. 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 a mod 2-projection and we let $x \in \pi_2^U(\alpha)$ iff α contains and odd number of elements y such that $y|_U = x$. When the identity of the set U is evident from the context the superscript of π is omitted.

For a 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 *[9] The* long code *of an assignment* $x \in \{-1,1\}^U$ *is the mapping* $A_x : \mathcal{F}_U \mapsto \{-1,1\}$ *where* $A_x(f) = f(x)$.

We identify a function with its truth-table and thus a long code is a string of length $2^{2^{|U|}}$ where we use an arbitrary, but fixed convention to order the elements of $\mathcal{F}_U$.

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 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 the clause.

Definition 2.2 *Let* $e \in [0,1]$ *be a real number. A CNF-formula* φ *with* m *clauses is* e*-satisfiable, iff some assignment satisfies* em *clauses and no assignment satisfies more than* em *clauses.*

Using the natural 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 2.3 *Let* k *be an integer. A CNF-formula is a E*k*-CNF-formula iff each clause is of length exactly* k.

For a CNF-formula φ and an assignment x let $N(\varphi, x)$ be the number of clauses of φ satisfied by x.

Definition 2.4 *Max-E*k*-Sat is the optimization problem of, given a E*k*-CNF formula* φ*, to find* x *that maximizes* $N(\varphi, x)$.

We are also interested in the problem of solving systems of linear equations over the finite field with 2 elements. Let us denote a typical system of linear equations L and, similarly to above, for an assignment x let $N(L, x)$ be the number of equations of L satisfied by x.

Definition 2.5 *Max-E*k*-Lin-2 is the problem of, given a system* L *of linear equations over* $\mathbb{Z}_2$*, with exactly* k *variables in each equation, to find* x *that maximizes* $N(L, x)$.

Definition 2.6 *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* $\{u, v\} \cap V_1$ *and* $\{u, v\} \cap V_2$ *are both nonempty is maximized.*

Definition 2.7 *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 2.8 *Vertex Cover is the problem of, given an undirected graph G with edges E and vertices V, to find a $V_1 \subseteq V$ with $|V_1|$ minimal such that V_1 intersects each edge.*

Definition 2.9 *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 i with both $S_i \cap V_1$ and $S_i \cap V_2$ nonempty.*

Note that E2-Set Splitting is exactly Max-Cut and that E3-Set Splitting is very related to E2-Set Splitting in that 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 really new problem is E4-Set Splitting.

Several of the above problems are special cases of a general class of problems called constraint satisfaction problems, from now on abbreviated CSP.

Let k be an integer and let P be a predicate $\{-1,1\}^k \mapsto \{-1,1\}$. An instance of CSP-$P$ is given by a collection $(C_i)_{i=1}^m$ of k-tuples of literals. For an assignment to the variables, a particular k-tuple is satisfied if P, when applied to values of the literals, returns -1. For an instance I and an assignment x we let $N(I, x, P)$ be the number of constraints of I satisfied by x under the predicate P.

Definition 2.10 *Max-CSP-P is the problem of, given an instance I, to find the assignment x that maximizes $N(I, x, P)$.*

It is straightforward to check that Max-Ek-Sat, Max-Ek-Lin, Max-Cut, and Max-di-Cut are all CPSs for particular predicates P. We are also interested in cases when negation is not allowed. We call such *monotone CSP* and one particular case is given by Ek-Set Splitting.

A key parameter for a CSP is the number of assignments that satisfy the defining predicate P.

Definition 2.11 *The* weight, *$w(P, k)$, of a CSP problem given by a predicate P on k Boolean variables is defined as $p2^{-k}$ where p is the number of assignments in $\{-1,1\}^k$ that satisfies P.*

The weight of Ek-Max-Lin-2 is $1/2$ for any k, it is $1 - 2^{-k}$ for Ek-Max-Sat, $1/2$ for Max-Cut, $1/4$ for Max-di-Cut and $1 - 2^{1-k}$ for Ek-Set Splitting. We note that the concept extends in the obvious way to non-Boolean domains.

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). The lower bounds we prove apply to the simpler variant, i.e., the variant where the algorithm is supposed to supply the numerical answer. Since we are proving inapproximability results, this only makes our results stronger.

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

Definition 2.12 *Let O be a maximization problem and let $C \geq 1$ be a real number. For an instance x of O let $OPT(x)$ be the optimal value. A C-approximation algorithm is an algorithm that on each input x outputs a number V such that $OPT(x)/C \leq V \leq OPT(x)$.*

Definition 2.13 *Let O be a minimization problem and let $C \geq 1$ be a real number. For an instance x of O let $OPT(x)$ be the optimal value. A C-approximation algorithm is an algorithm that on each input x outputs a number V such that $OPT(x) \leq V \leq C \cdot OPT(x)$.*

Definition 2.14 *An efficient C-approximation algorithm is a C-approximation algorithm that runs in worst case polynomial time.*

The formulation "having performance ratio C" is sometimes used as an alternative to saying "being a C-approximation algorithm".

Any Max-CSP-problem has an approximation algorithm with constant performance.

Theorem 2.15 *A Max-CSP given by predicate P on k variables admits a polynomial time approximation algorithm with performance ratio $w(P,k)^{-1}$.*

Proof: A random assignment satisfies a given k-tuple with probability $w(P,k)$. It is not difficult to find an assignment that satisfies this fraction of the given k-tuples by the method of conditional expected values. We omit the details. ▪

The main point of this paper is to establish that for many CPSs, Theorem 2.15 is in fact the best possible for a polynomial time approximation algorithm.

Definition 2.16 *A Max-CSP given by predicate P on k variables is* non-approximable beyond the random assignment threshold *iff, provided that NP$\neq$ P, for any $\epsilon > 0$, it does not allow a polynomial time approximation algorithm with performance ratio $w(P,k)^{-1} - \epsilon$.*

Some CSP become easier if you only consider satisfiable instances but some do not. We formalize also this notion.

Definition 2.17 *A Max-CSP given by predicate P on k variables is* non-approximable beyond the random assignment threshold on satisfiable instances *iff, for any $\epsilon > 0$ it is NP-hard to distinguish instances where all constraints can be simultaneously satisfied from those where only a fraction $w(P,k)+\epsilon$ of the constraints can be simultaneously satisfied.*

2.3 Proof systems

We define proofs systems 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. In different variants of proof systems, the notions of provers and written proofs are discussed. Written proofs are in fact identical with proofs using oracles where reading the i'th bit corresponds to asking the oracle the question "i ?". Provers, in general, are more powerful than oracles in that they are allowed to be randomized and history dependent. We discuss these complications in connection with the definition of two-prover protocols below.

Definition 2.18 *An* oracle *is a function* $\Sigma^* \mapsto \{0,1\}$.

A typical verifier $V^\pi(x,r)$ is a probabilistic Turing machines where π is the oracle, x the input and r the (internal) random coins of V. We say that the verifier *accepts* if it outputs 1 (written as $V^\pi(x,r)=1$) and otherwise it *rejects*.

Definition 2.19 *Let c and s be real numbers such that $1 \geq c > s \geq 0$. 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 exists an oracle π such that $Pr_r[V^\pi(x,r)=1] \geq c$.*
- *For $x \notin L$, for all π $Pr_r[V^\pi(x,r)=1] \leq s$.*

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 2.20 *The verifier V uses* logarithmic randomness *if there is an absolute constant c such that on each input x and proof π, the length of the random string r used by V^π is bounded by $c \log |x|$.*

Using logarithmic randomness makes the total number of possible sets of coin flips for V polynomial in $|x|$ and hence all such sets can be enumerated in polynomial time.

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

Definition 2.21 *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.*

The surprising power of interactive proofs was first established in the case of one prover by Lund et al. [29], and Shamir, [33] and then for many provers by Babai et al. [5]. After the fundamental connection with approximation was discovered by Feige et al. [16] the parameters of the proofs improved culminating in the following result [4, 3].

Theorem 2.22 *[3] There is a universal integer c such that any language in NP has a PCP with soundness $1/2$ and completeness 1 where V uses logarithmic randomness and reads at most c bits of the proof.*

Remark 2.23 *Although the number of bits read is independent of which language in NP we are considering, this is not true for the amount of randomness. The number of random bits is $d \log n$ for any language L, but the constant d depends on L.*

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 where the variables correspond to bits of the proof. The acceptance probability of a proof is then closely related to the number of clauses satisfied by the corresponding assignment and we obtain an inapproximability result for Max-3Sat. There is an approximation-preserving reduction [30] reducing general 3-CNF formulas to 3-CNF formulas in which each variable appears a bounded number of times. It has later been established [14] that we can make each variable appear exactly 5 times even if we require each clause to be of length exactly 3. These properties ensure that choosing a clause uniformly at random and a variable, uniformly at random, in this clause is the same as choosing a variable uniformly at random variable and then, uniformly at random, a clause containing this variable.

Theorem 2.24 *[3] 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 $|x|$ 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, each variable appears exactly 5 times.*

We next describe a two-prover one-round interactive proof. The verifier in such a proof has access to 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 but since the verifier runs in polynomial time it will not read more than a polynomial number of bits. We call the two oracles P_1 and P_2 and the two questions q_1 and q_2. Since the oracles are only accessed through these questions we refer to the fact that V accepts as $V(x, r, P_1(q_1), P_2(q_2)) = 1$.

Definition 2.25 *Let c and s be real numbers such that $1 \geq c > s \geq 0$. 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 $Pr_r[V(x, r, P_1(q_1), P_2(q_2)) = 1] \geq c$.*
- *For $x \notin L$, for any two oracles P_1 and P_2, $Pr_r[V(x, r, P_1(q_1), P_2(q_2)) = 1] \leq s$.*

The questions q_1 and q_2 are in both cases the only questions V asks the oracles. $P_1(q_1)$ depends on x, but may not depend on q_2 and similarly P_2 is independent of q_1.

It is many times convenient to think of P_1 and P_2 as 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.

Provers are in general allowed to be both history dependent and randomized. Since we are here only considering one-round protocols, there is no history and hence the question whether the provers are history dependent plays no role. As with randomization, it is easy[1] to see that for any x, the provers P_1 and P_2 maximizing $Pr_r[V(x, r, P_1(q_1), P_2(q_2)) = 1]$ can be made deterministic without decreasing the acceptance probability. When proving the existence of good strategies for the provers we will, however, allow ourselves to design probabilistic strategies, which then can be converted to deterministic strategies yielding ordinary oracles.

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 u times in sequence gives soundness s^u. 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 where V repeats his random choices to choose u independent pairs of questions $(q_1^{(i)}, q_2^{(i)})_{i=1}^u$ and sends $(q_1^{(i)})_{i=1}^u$ to P_1 and $(q_2^{(i)})_{i=1}^u$ to P_2, all at once. V then receives u answers from each prover and accepts if it would have accepted in all u protocols given each individual answer. The soundness of such a protocol can be greater than s^u, but when the answer size is small, Raz [32] proved that soundness is exponentially decreasing with u.

Theorem 2.26 *[32] 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 u, the soundness of u protocols run in parallel is bounded by $c_{d,s}^u$.*

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

[1] Fix an optimal strategy, which might be randomized, of P_1. Now, for each q_2, P_2 can consider all possible r of V producing q_2, compute q_1 and then, since the strategy of P_1 is fixed, exactly calculate the probability that V would accept for each possible answer. P_2 then answers with the lexicographically first string achieving the maximum. This gives an optimal deterministic strategy for P_2. We can then proceed to make P_1 deterministic by the symmetric approach.

2.4 Fourier Transforms

Our proofs depend heavily on Fourier analysis of functions $A : \mathcal{F}_U \mapsto \mathbb{R}$ where $\mathbb{R}$ is the set of real numbers. We recall some basic facts. For notational convenience let u denote $|U|$. 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 product 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 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

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

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

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

This sum is, in this paper, usually 1 since we mostly study A with range $\{-1,1\}$.

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 difference is that $\{-1,1\}^U$ takes the place of $[n]$. The inputs to "ordinary" functions are n bit strings which can be thought of as mappings from $[n]$ to $\{-1,1\}$. The inputs to our functions are mappings from $\{-1,1\}^U$ to $\{-1,1\}$ and this explains the change from $[n]$ to $\{-1,1\}^U$.

Suppose A is the long code of an input x_0. By definition, the basis function $\chi_{\{x_0\}}$ is exactly this long code. Thus, the Fourier transform satisfies $\hat{A}_{\{x_0\}} = 1$ while all the other Fourier coefficients are 0.

A significant part of this paper consists of manipulations of Fourier expansions and let us state a couple of basic facts for future reference. The proofs of the first two are straightforward and are left to the reader.

Lemma 2.27 *For any $f, g \in \mathcal{F}_U$ and $\alpha \subseteq \{-1,1\}^U$ we have $\chi_\alpha(fg) = \chi_\alpha(f)\chi_\alpha(g)$.*

Lemma 2.28 *For any $f \in \mathcal{F}_U$ and $\alpha, \beta \subseteq \{-1,1\}^U$ we have $\chi_\alpha(f)\chi_\beta(f) = \chi_{\alpha\Delta\beta}(f)$.*

Lemma 2.29 *Let k be an integer and suppose that for each $1 \leq i \leq k$ we have a random variable f_i whose range is $\mathcal{F}_{U_i}$ and that we are given $\alpha_i \subseteq \{-1,1\}^{U_i}$. Suppose that there is an i_0 and $x_0 \in \{-1,1\}^{U_{i_0}}$ such that $x_0 \in \alpha_{i_0}$ and that $f_{i_0}(x_0)$ is random with the uniform distribution and independent of $f_i(x)$ for all $(i,x) \neq (i_0, x_0)$ with $x \in \alpha_i$. Then*

$$E[\prod_{i=1}^{k} \chi_{\alpha_i}(f_i)] = 0,$$

where the expectation is taken over a random selection of $(f_i)_{i=1}^k$. In particular, $E[\chi_\alpha(f)] = 0$ when f is chosen randomly with the uniform probability and $\alpha \neq \emptyset$.

Proof: By the independence condition we have, with $\alpha'_i = \alpha_i$ for $i \neq i_0$ and $\alpha'_{i_0} = \alpha_{i_0}\Delta\{x_0\}$,

$$E[\prod_{i=1}^{k} \chi_{\alpha_i}(f_i)] = E[f_{i_0}(x_0)]E[\prod_{i=1}^{k} \chi_{\alpha'_i}(f_i)] = 0$$

since the first factor is 0. ■

In many cases we have $U \subseteq W$ and we have $f \in \mathcal{F}_U$ that we want to interpret as a function on $\{-1,1\}^W$. We do this by ignoring the coordinates not belonging to U. We use the same symbol f, but we write $f(y|_U)$ to make the restriction of the domain explicit. We have the following basic fact.

Lemma 2.30 *Assume $U \subset W$ and $f \in \mathcal{F}_U$. The for any $\beta \subseteq \{-1,1\}^W$ we have $\chi_\beta(f) = \chi_{\pi_2^U(\beta)}(f)$.*

Proof: We use the definition

$$\chi_\beta(f) = \prod_{y \in \beta} f(y|_U).$$

The number of times a value x appears in this product is exactly the number of $y \in \beta$ such that $\pi^U(y) = x$. Since we only care whether the sum is even or odd the product equals

$$\prod_{x \in \pi_2^U(\beta)} f(x)$$

and this is exactly $\chi_{\pi_2^U(\beta)}(f)$. ■

2.5 Folding and conditioning of long codes

It is many times convenient to make sure that $A(f) = -A(-f)$ is true for all f. The mechanism to achieve this was introduced by Bellare et al. [9] and was called "folding over 1" since 1 was used to denote true. Here we are folding over -1 but to emphasize that we are using the same notion we call it "folding over true".

Definition 2.31 *Given a function $A : \mathcal{F}_U \mapsto \{-1, 1\}$. The function A_{true},* folding A over true *is defined by for each pair $(f, -f)$ selecting one of the two functions. If f is selected then $A_{true}(f) = A(f)$ and $A_{true}(-f) = -A(f)$. If $-f$ is selected then $A_{true}(f) = -A(-f)$ and $A_{true}(-f) = A(-f)$.*

Note that the definition implies that $A_{true}(f) = -A_{true}(-f)$ is always true. The function A_{true} depends on the selection function but since this dependence is of no importance we leave it implicit.

Lemma 2.32 *If $B = A_{true}$ then for all α with $\hat{B}_\alpha \neq 0$ we have that $|\alpha|$ is odd and in particular α is not empty.*

Proof: By definition:

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

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

We sometimes know that the input x for which a given table A is supposed to be the long code should satisfy $h(x) = -1$ (i.e., h is true) for some function h. This was also needed in the paper by Bellare et al. [9] where they defined "folding over h" analogously to folding over 1. We need a stronger property which we call "conditioning upon h".

Definition 2.33 *For functions $A : \mathcal{F}_U \mapsto \{-1, 1\}$ and $h \in \mathcal{F}_U$, we define the function $A_h : \mathcal{F}_U \mapsto \{-1, 1\}$, that we call A* conditioned upon h *by setting, for each f, $A_h(f) = A(f \wedge h)$.*

We have the following lemma

Lemma 2.34 *Let $B = A_h$ where $A : \mathcal{F}_U \mapsto \{-1, 1\}$ and $h \in \mathcal{F}_U$ are arbitrary. Then for any α such that there exists $x \in \alpha$ with $h(x) = 1$ we have $\hat{B}_\alpha = 0$.*

Proof: Let us first note that the conclusion is natural since $B(f)$, by definition, only depends on the value of f at points such that $h(x) = -1$ and hence these are the only inputs that should appear in the Fourier expansion. Formally, we use the definition

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

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 is divided into $2^{2^{u}-1}$ pairs (f, f') and since $B(f) = B(f')$ 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 and thus the sum evaluates to 0. ■

We can apply folding over true and conditioning upon h simultaneously by defining a pairing of all functions of the type $(g \wedge h)$. Note that unless h is identically true not both f and $-f$ can be of this form and we pair $(g \wedge h)$ with $((-g) \wedge h)$ and define $A_{h,true}(f)$ as $A(f \wedge h)$ if $f \wedge h$ is chosen in its pair and as $-A((-f) \wedge h)$ if $(-f) \wedge h$ is chosen. It is easy to verify that $A_{h,true}(f) = -A_{h,true}(-f)$ and that $A_{h,true}(f)$ only depends on $f \wedge h$.

2.6 Extensions to arbitrary Abelian groups

We extend some results to arbitrary Abelian groups and hence we extend the definitions in the previous section, which applied to the group with 2 elements, to general Abelian groups.

Let Γ be an Abelian group. By the structure theorem [26] of Abelian groups Γ can be represented as a direct product of cyclic groups, $\Gamma = C_{i_1} \times C_{i_2} \times C_{i_3} \dots C_{i_k}$. The number of elements, $|\Gamma|$, of Γ is $\prod_{l=1}^{k} i_l$. We represent a cyclic group C_i as the i'th roots of unity and an element γ of Γ is thus a k-tuple of complex numbers and the group operation is coordinate-wise multiplication.

We also need the dual group Γ^* of Γ which is the group of homomorphisms of Γ into the complex numbers. We need very few properties of the dual group and we refer to [26] for a general discussion. For Abelian groups, Γ^* is isomorphic to Γ but we choose to represent it as elements of $\mathbb{Z}_{i_1} \times \mathbb{Z}_{i_1} \dots \mathbb{Z}_{i_k}$ where the group operation is component-wise addition. With $\gamma = (\gamma_1, \gamma_2 \dots, \gamma_k) \in \Gamma$ and $\gamma^* = (\gamma_1^*, \gamma_2^* \dots, \gamma_k^*) \in \Gamma^*$ we let γ^{γ^*} be the complex number

$$\prod_{i=1}^{k} \gamma_i^{\gamma_i^*}.$$

We are here using slightly nonstandard terminology. As defined above the dual group should really be functions mapping Γ to the complex numbers $\mathbb{C}$. An element of the dual group is given by γ^* and the associated function, in our notation, is $\gamma \mapsto \gamma^{\gamma^*}$. We do not here make the distinction between γ^* and the function it represents.

We let $\mathcal{F}_U^\Gamma$ be the set of functions $f : \{-1,1\}^U \mapsto \Gamma$ and we have a generalization of the long code.

Definition 2.35 *The* long Γ-code *of an assignment $x \in \{-1,1\}^U$ is the mapping $A_x : \mathcal{F}_U^\Gamma \mapsto \Gamma$ where $A_x(f) = f(x)$.*

We next define the Fourier transform. We study function $A : \mathcal{F}_U^\Gamma \mapsto \mathbb{C}$ where $\mathbb{C}$ are the complex numbers. The basis functions are given by functions

$\alpha : \{-1,1\}^U \mapsto \Gamma^*$ and are defined by

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

We have the inner product defined by

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

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

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

which is inverted by

$$A(f) = \sum_\alpha \hat{A}_\alpha \chi_\alpha(f). \tag{2}$$

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

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

if we are working with a function satisfying $|A(f)| = 1$ for all f.

We have 3 lemmas extending Lemma 2.27, Lemma 2.28, and Lemma 2.29. The first two follow in a straightforward way from the definitions.

Lemma 2.36 *For any $f, g \in \mathcal{F}_U^\Gamma$ and $\alpha : \{-1,1\}^U \mapsto \Gamma^*$ we have $\chi_\alpha(fg) = \chi_\alpha(f)\chi_\alpha(g)$.*

Lemma 2.37 *For any $f \in \mathcal{F}_U^\Gamma$ and $\alpha, \beta : \{-1,1\}^U \mapsto \Gamma^*$ we have $\chi_\alpha(f)\chi_\beta(f) = \chi_{\alpha+\beta}(f)$.*

Lemma 2.38 *Let k be an integer and suppose that for $1 \le i \le k$ we have a random variable f_i whose range is $\mathcal{F}_{U_i}^\Gamma$ and we are given $\alpha_i : \{-1,1\}^{U_i} \mapsto \Gamma^*$. Suppose that there is an i_0 and $x_0 \in \{-1,1\}^{U_{i_0}}$ such that $\alpha_{i_0}(x_0) \neq 0^k$ and that $f_{i_0}(x_0)$ is random with the uniform distribution and independent of $f_i(x)$ for all $(i, x) \neq (i_0, x_0)$ with $\alpha_i(x) \neq 0^k$. Then*

$$E[\prod_{i=1}^k \chi_{\alpha_i}(f_i)] = 0.$$

In particular for any α which is not identically 0^k we have $E[\chi_\alpha(f)] = 0$ when f is chosen uniformly at random.

Proof: By definition

$$\prod_{i=1}^{k} \chi_{\alpha_i}(f_i) = \prod_{i=1}^{k} \prod_{x \in \{-1,1\}^{U_i}} f_i(x)^{\alpha_i(x)}.$$

Now $f_{i_0}(x_0)^{\alpha_{i_0}(x_0)}$ appears in this product and is by assumption independent of all other factors. We need just observe that $E_\gamma[\gamma^\alpha] = 0$ for γ chosen uniformly in Γ and $\alpha \neq 0^k$. This follows since γ^α ranges over a full set of roots of unity. ∎

We have a natural extension of π_2^U.

Definition 2.39 *Let $U \subseteq W$ and $\beta : \{-1,1\}^W \mapsto \Gamma^*$. Then $\pi_\Gamma^U(\beta) = \alpha$ where $\alpha(x) = \sum_{y:y|_U = x} \beta(y)$.*

We next have the analogue of Lemma 2.30. Also the proof is analogous to and we omit it.

Lemma 2.40 *Assume $U \subset W$ and $f \in \mathcal{F}_U^\Gamma$. Then for any $\beta : \{-1,1\}^W \mapsto \Gamma^*$ we have $\chi_\beta(f) = \chi_{\pi_\Gamma^U(\beta)}(f)$.*

When working with long Γ-codes we need to fold over Γ.

Definition 2.41 *Given a function $A : \mathcal{F}_U^\Gamma \mapsto \Gamma$. The function A_Γ,* folding A over Γ *is defined by for each set of functions $(\gamma f)_{\gamma \in \Gamma}$ selecting one function. If $\gamma_0 f$ is selected then $A_\Gamma(\gamma f) = \gamma\gamma_0^{-1} A(\gamma_0 f)$ for all $\gamma \in \Gamma$.*

A long Γ-code A has range Γ and since we want to study functions with a range in $\mathbb{C}$, typically we study A^{γ^*} for some $\gamma^* \in \Gamma^*$. Multiplying such a function by the group element γ should multiply the result by γ^{γ^*} and thus the below definition is natural.

Definition 2.42 *Given a function $A : \mathcal{F}_U^\Gamma \mapsto \mathbb{C}$ and $\gamma^* \in \Gamma^*$. The function A is* γ^*-homogeneous *if for each $f \in \mathcal{F}_U^\Gamma$ and $\gamma \in \Gamma$ we have $A(\gamma f) = \gamma^{\gamma^*} A(f)$.*

We have the following consequence of the definitions.

Lemma 2.43 *Given a function $A : \mathcal{F}_U^\Gamma \mapsto \Gamma$, and $\gamma^* \in \Gamma^*$. Then if $B = A_\Gamma$ we have that B^{γ^*} is γ^*-homogeneous.*

Proof: From the definition of folding it follows that $B(\gamma f) = \gamma B(f)$. The lemma is now immediate. ∎

We need the consequence for the Fourier transform.

Lemma 2.44 *Given $A : \mathcal{F}_U^\Gamma \mapsto \mathbb{C}$, and $\gamma^* \in \Gamma^*$ and assume that A is γ^*-homogeneous. Then for all α with $\hat{A}_\alpha \neq 0$ we have $\sum_x \alpha(x) = \gamma^*$. In particular if γ^* is nonzero, there is some x with $\alpha(x) \neq 0^k$.*

Proof: Assume that $\sum_x \alpha(x) \neq \gamma^*$ and take some $\gamma \in G$ with $\gamma^{\gamma^* - \sum_x \alpha(x)} \neq 1$. We have

$$\hat{A}_\alpha = \sum_f A(f)\overline{\chi_\alpha(f)} = \sum_f A(\gamma f)\overline{\chi_\alpha(\gamma f)} \tag{3}$$

Now using

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

and the assumption of being γ^*-homogeneous we see that the right hand side of (3) equals

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

We conclude that $\hat{A}_\alpha = 0$. ∎

The notion of conditioning extends without virtually any changes.

Definition 2.45 *From a function $A : \mathcal{F}_U^\Gamma \mapsto R$ for any range R and $h \in \mathcal{F}_U$ we construct a function A_h, called A* conditioned upon *h by for each f, $A_h(f) = A(f \wedge h)$. Here $f \wedge h$ is defined by $f \wedge h(x) = f(x)$ when $h(x) = -1$ and $f \wedge h(x) = 1^k$ otherwise.*

We state the corresponding lemma without a proof.

Lemma 2.46 *Let $A : \mathcal{F}_U^\Gamma \mapsto \mathbb{C}$ and $h \in \mathcal{F}_U$ be arbitrary and set $B = A_h$. Then for any α such that there exists x with $\alpha(x) \neq 0^k$ and $h(x) = 1$ we have $\hat{B}_\alpha = 0$.*

The computational problem we study is given by systems of linear equations in the group Γ. If L is such a system we let $N(L, x)$ be the number of equations satisfied by x.

Definition 2.47 *Max-Ek-Lin-Γ is the problem of given a system L of linear equations over an Abelian group Γ, with exactly k variables in each equation, find x that maximizes $N(L, x)$.*

3 The basic two-prover protocol and the corresponding PCP

To get our inapproximability results we construct a range of different PCPs. Most of these PCPs have the same written proof and only the method to check this proof is 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 a 3-CNF formula φ given by Theorem 2.24. 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 also have the property that each clause is of length exactly 3 and each variable appears in exactly 5 clauses.

Basic 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 iff the two values for x_k agree and C_j is satisfied.

We have:

Lemma 3.1 *If φ is c-satisfiable then for any P_1 and P_2, V accepts in the basic two-prover protocol with probability at most $(2+c)/3$.*

Proof: The answers by P_2 define an assignment α_0 to all variables. Whenever V chooses a clause not satisfied by α_0, either P_1 answers with an unsatisfying assignment, causing V to reject outright or has at least probability $1/3$ of being caught for not being consistent with P_2. Since α_0 satisfies at most a fraction c of the clauses the probability of V rejecting is at least $(1-c)/3$. ∎

The basic 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 rather close to 1. We improve this second parameter by running the protocol in parallel.

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 iff the two values for x_{k_i} agree and C_{j_i} is satisfied for all $1 \le i \le u$.

By applying Theorem 2.26 and Lemma 3.1 and using the honest strategy when φ is satisfiable 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 at most 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 this u-parallel two-prover protocol into a PCP. We write down, for each possible question, the long code of the answer. We call this proof SWP for Standard Written Proof.

Definition 3.3 *A* Standard Written Proof *with parameter u or SWP(u), contains for each set $V \subset [n]$ of size at most $3u$ a string of length $2^{2^{|V|}}$ which we interpret as the table of a function $A_V : \mathcal{F}_V \mapsto \{-1, 1\}$.*

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 $3u$.*

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

When accessing a long code on a set of inputs for which we have some information (like a set of clauses on these inputs being true) we use conditioning. We use the notation $A_{V,h}$ rather than $(A_V)_h$ for the table A_V conditioned upon h. We often fold the tables over true yielding a function called $A_{V,h,true}$.

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

The standard proof strategy for establishing that such a 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 convince the verifier in u-parallel two-prover protocol to accept with high probability.

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

Definition 3.5 *A* Standard Written Γ-Proof *with parameter u or SWΓP(u) contains for each set $V \subset [n]$ of size at most $3u$ a string of length $|\Gamma|^{2^{|V|}}$ which we interpret as the table of a function $A_V : \mathcal{F}_V^{\Gamma} \mapsto \Gamma$. The symbols of the proof represent elements of Γ.*

Definition 3.6 *A SWΓP(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 Γ-code of $x|_V$ for any V of size at most $3u$.*

4 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 we present it for pedagogical reasons. It is easy to analyze given the correct tools but still gives a nontrivial conclusion.

In most previous code-testing situations [3, 4, 9] the key parameter that has been analyzed is the distance from a given word to different code words. This is a natural parameter but considering only distances turns out to be too restrictive. We follow the path of [23] and use a strategy not only based on distances but on the complete Fourier transform that, for any A that passes the test with high probability, associate a small set of inputs. These inputs can later be used as strategies in the underlying two-prover interactive proof.

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., there exists an $x \in \{-1, 1\}^U$ such that for all f, $A(f) = f(x)$.
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 iff $A(f_0)A(f_1)A(f_2) = 1$.

First note that V always accepts 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. A random A is accepted with probability $1/2$ and thus this acceptance probability does not have any implications on the structure of A. We will establish, however, that any A that is accepted with probability $(1 + \delta)/2$ for $\delta > 0$ must have some special structure.

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 under the above assumption

$$E_{f_0,f_1}[A(f_0)A(f_1)A(f_2)] = \delta. \tag{4}$$

We replace $A(f_i)$ by its Fourier expansion, i.e. using (1) and see that (4) equals

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

Using the linearity of expectation (5) equals

$$\sum_{\alpha_0,\alpha_1,\alpha_2} \hat{A}_{\alpha_0}\hat{A}_{\alpha_1}\hat{A}_{\alpha_2} E_{f_0,f_1}\left[\chi_{\alpha_0}(f_0)\chi_{\alpha_1}(f_1)\chi_{\alpha_2}(f_2)\right]. \tag{6}$$

By the definition of f_2, Lemma 2.27, and Lemma 2.28 we have

$$\begin{aligned} \chi_{\alpha_0}(f_0)\chi_{\alpha_1}(f_1)\chi_{\alpha_2}(f_2) &= \chi_{\alpha_0}(f_0)\chi_{\alpha_1}(f_1)\chi_{\alpha_2}(f_0 f_1) = \\ \chi_{\alpha_0}(f_0)\chi_{\alpha_1}(f_1)\chi_{\alpha_2}(f_0)\chi_{\alpha_2}(f_1) &= \chi_{\alpha_0 \Delta \alpha_2}(f_0)\chi_{\alpha_1 \Delta \alpha_2}(f_1). \end{aligned}$$

Since f_0 and f_1 are independent (6) equals

$$\sum_{\alpha_0,\alpha_1,\alpha_2} \hat{A}_{\alpha_0}\hat{A}_{\alpha_1}\hat{A}_{\alpha_2} E_{f_0}\left[\chi_{\alpha_0\Delta\alpha_2}(f_0)\right] E_{f_1}\left[\chi_{\alpha_1\Delta\alpha_2}(f_1)\right]. \tag{7}$$

By Lemma 2.29 we to see that unless $\alpha_0 = \alpha_2$, we have $E_{f_0}[\chi_{\alpha_0\Delta\alpha_2}(f_0)] = 0$. Similarly, unless $\alpha_1 = \alpha_2$, $E_{f_1}[\chi_{\alpha_1\Delta\alpha_2}(f_1)] = 0$. Using these two facts we see that (7) simplifies to

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

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

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

We conclude that this maximum is at least δ. Thus we have proved that there is at least one α such that $\hat{A}_\alpha \ge \delta$ and by Parseval's equality there can be at most δ^{-2} such α. Thus with any A that causes the test to accept with high probability we can associate a small number of sets but since each set might be large we have failed to find a small set of inputs.

Since the test accepts with probability one if $A = \chi_\alpha$ we cannot do much better with the current test. In fact what we have presented is the linearity test of Blum et al. [13] and one part of the analysis given in the paper by Bellare et al. [8].

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 iff $A(f_0)A(f_1)A(f_2) = 1$.

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 (6) is still valid and we need to consider

$$E_{f_0,f_1,\mu}\left[\chi_{\alpha_0}(f_0)\chi_{\alpha_1}(f_1)\chi_{\alpha_2}(f_2)\right] =$$
$$E_{f_0,f_1,\mu}\left[\chi_{\alpha_0\Delta\alpha_2}(f_0)\chi_{\alpha_1\Delta\alpha_2}(f_1)\chi_{\alpha_2}(\mu)\right], \quad (8)$$

where we used the definition of f_2, Lemma 2.27, and Lemma 2.28.

Since f_0, f_1 and μ are independent we can use Lemma 2.29 to see that unless $\alpha_0 = \alpha_1 = \alpha_2$ the above expected value is 0. Since $E_\mu[\mu(x)] = 1 - 2\epsilon$ for each x and $\mu(x)$ are independent for different x we have

$$E_\mu\left[\chi_{\alpha_2}(\mu)\right] = (1-2\epsilon)^{|\alpha_2|}$$

and thus

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

where the inequality follows from Parseval's identity.

This time, we can conclude that for some α we have $\hat{A}_\alpha(1-\epsilon)^{|\alpha|} \ge \delta$. Since this inequality implies that $|\alpha| \le \epsilon^{-1}\log\delta^{-1}$ we have identified large Fourier coefficients that correspond to sets of limited size. This implies that we get the small set of inputs we were aiming for. These inputs can then be used as strategies in the two-prover protocol.

5 Linear equations

We first study the optimization problem Max-E3-Lin-2 and for natural reasons we want to design a test for SWP(u) that accepts depending only on the exclusive-or of three bits of the proof. It turns out that we can take the second long code test on W and simply move one of the three questions to the smaller set U. This allows us to test consistency at the same time as we are testing that the tables are correct long codes. The existence of the clauses are handled by using conditioning when accessing the long code on the set W.

Test $L_2^\epsilon(u)$

Written proof. A SWP(u).
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 choose, uniformly at random, a variable x_{k_i} occurring in C_{j_i}. Set $U = \{x_{k_1}, x_{k_2} \ldots x_{k_u}\}$, W to be the set of all variables occurring in the chosen clauses, and $h = \wedge_{i=1}^u C_{j_i}$.

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 iff $A_{U,true}(f) A_{W,h,true}(g_1) A_{W,h,true}(g_2) = 1$.

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

Lemma 5.1 *The completeness of Test $L_2^\epsilon(u)$ is at least $1 - \epsilon$.*

Proof: Fix a correct SWP(u) obtained from an assignment x satisfying φ We claim that V accepts unless $\mu(x|_W) = -1$. This follows since for a correct SWP(u) encoding x, folding over true and conditioning upon h is of no consequence and hence $A_{U,true}(f) = f(x|_U)$, $A_{W,h,true}(g_1) = g_1(x|_W)$ and $A_{W,h,true}(g_2) = g_2(x|_W) = f(x|_U) g_1(x|_W) \mu(x|_W)$ and the claim follows. 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 5.2 *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 $4\epsilon\delta^2$.*

Proof:

Let us first fix U, W, and h and for notational convenience we denote the function $A_{U,true}$ by A and the function $A_{W,h,true}$ by B. As in the tests for the long code we want to consider

$$E_{f,g_1,\mu}[A(f)B(g_1)B(g_2)] \tag{9}$$

since, by the assumption of the lemma,

$$E_{U,W,h,f,g_1,\mu}[A_{U,true}(f) A_{W,h,true}(g_1) A_{W,h,true}(g_2)] = \delta. \tag{10}$$

We replace each function by its Fourier expansion transforming (9) to

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

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

$$\sum_{\alpha,\beta_1,\beta_2} \hat{A}_\alpha \hat{B}_{\beta_1} \hat{B}_{\beta_2} E_{f,g_1,\mu}\left[\chi_\alpha(f) \chi_{\pi_2(\beta_2)}(f) \chi_{\beta_1}(g_1) \chi_{\beta_2}(g_1) \chi_{\beta_2}(\mu)\right] =$$

$$\sum_{\alpha,\beta_1,\beta_2} \hat{A}_\alpha \hat{B}_{\beta_1} \hat{B}_{\beta_2} E_f\left[\chi_{\alpha \Delta \pi_2(\beta_2)}(f)\right] E_{g_1}\left[\chi_{\beta_1 \Delta \beta_2}(g_1)\right] E_\mu\left[\chi_{\beta_2}(\mu)\right], \tag{11}$$

where we used Lemma 2.27, Lemma 2.28, Lemma 2.30, and the fact that f, g_1 and μ are chosen independently. By Lemma 2.29 unless $\beta_1 = \beta_2$ and $\alpha = \pi_2^U(\beta)$ the corresponding term in (11) equals 0. Finally, since

$$E_\mu[\chi_\beta(\mu)] = (1-2\epsilon)^{|\beta|}$$

we have reduced (9) to

$$\sum_\beta \hat{A}_{\pi_2^U(\beta)} \hat{B}_\beta^2 (1-2\epsilon)^{|\beta|}. \tag{12}$$

We want to design strategies for P_1 and P_2 in the two-prover game. Before doing this let us just summarize the work upto this point by the equality

$$E_{U,W,h}\left[\sum_\beta \hat{A}_{\pi_2(\beta)} \hat{B}_\beta^2 (1-2\epsilon)^{|\beta|}\right] = \delta, \tag{13}$$

where we for, readability reasons, have dropped the superscript of π.

We define randomized strategies for P_1 and P_2. These can, as discussed earlier, be converted to optimal deterministic strategies that do at least as well.

- P_2, upon receiving the set U, selects a random α with probability $\hat{A}_\alpha^2$ and then returns a random $x \in \alpha$ chosen with the uniform probability.
- P_1, upon receiving h and W, selects a random β with probability $\hat{B}_\beta^2$ and then returns a random $y \in \beta$.

Note that, by Lemma 2.32 any set selected by either prover is nonempty. Furthermore, by Lemma 2.34 every y sent by P_1 satisfies the selected clauses. Thus to analyze the probability that the verifier in the two-prover protocol accepts we need only estimate the probability that the answers are consistent, i.e., that $y|_U = x$.

We claim that this probability is at least $|\beta|^{-1}$ times the probability that for the selected 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 selecting a specific pair α and β is $\hat{A}_\alpha^2 \hat{B}_\beta^2$ and thus the overall success-rate for a fixed choice of U, W and h is at least

$$\sum_\beta \hat{A}_{\pi_2(\beta)}^2 \hat{B}_\beta^2 |\beta|^{-1}. \tag{14}$$

and the overall success-probability is the expected value of this expression with respect to random U, W and h. To compare this sum to (13) the following lemma is useful.

Lemma 5.3 *For $x, s > 0$, $x^{-s} \geq e^{-sx}$.*

Proof: Since the inequality for a general s is the s'th power of the inequality for $s = 1$ we only need to establish the lemma for $s = 1$. Since $xe^{-x} = e^{\ln x - x}$ 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 differentiation, $\ln x - x$ is decreasing for $x > 1$. ∎

Returning to the main path we see, using Cauchy-Schwartz' inequality, that

$$\sum_\beta \hat{A}_{\pi_2(\beta)} \hat{B}_\beta^2 |\beta|^{-1/2} \leq \left(\sum_\beta \hat{A}^2_{\pi_2(\beta)} \hat{B}^2_\beta |\beta|^{-1} \right)^{1/2} \left(\sum_\beta \hat{B}^2_\beta \right)^{1/2} \leq$$
$$\left(\sum_\beta \hat{A}^2_{\pi_2(\beta)} \hat{B}^2_\beta |\beta|^{-1} \right)^{1/2}.$$

This implies

$$E_{U,W,h} \left[\sum_\beta \hat{A}^2_{\pi_2(\beta)} \hat{B}^2_\beta |\beta|^{-1} \right] \geq E_{U,W,h} \left[(\sum_\beta \hat{A}_{\pi_2(\beta)} \hat{B}^2_\beta |\beta|^{-1/2})^2 \right] \geq$$
$$\left(E_{U,W,h} \left[\sum_\beta \hat{A}_{\pi_2^U(\beta)} \hat{B}^2_\beta |\beta|^{-1/2} \right] \right)^2, \tag{15}$$

where we have used that $E[X^2] \geq E[X]^2$.

Now by Lemma 5.3 with $s = 1/2$,

$$(4\epsilon|\beta|)^{-1/2} \geq e^{-2\epsilon|\beta|} \geq (1 - 2\epsilon)^{|\beta|}, \tag{16}$$

where we used $e^{-x} \geq 1 - x$ which is true for all $x \geq 0$. We conclude that $|\beta|^{-1/2} \geq (4\epsilon)^{1/2}(1 - 2\epsilon)^{|\beta|}$ and combining this with (15) and (13) we see that

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

As established above this is a lower bound for the probability that the verifier accepts in the u-parallel two-prover protocol and hence the proof of Lemma 5.2 is complete. ∎

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

Theorem 5.4 *For any $\epsilon > 0$ it is NP-hard to approximate Max-E3-Lin-2 within a factor $2 - \epsilon$. Said equivalently Max-E3-Lin-2 is non-approximable beyond the random assignment threshold.*

Proof: Set δ to a negative power of two 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 side which is a product of variables and right hand side which is either 1 or -1.

Let L be an arbitrary language in NP and suppose we are given an input x and we are trying to decide whether $x \in L$. By Theorem 2.24 we can, in polynomial time, create a E3-CNF formula φ with each variable occurring exactly 5 times such that if $x \in L$ then φ is satisfiable and if $x \notin L$ then φ is at most c-satisfiable where c is some definite constant less than 1. Now choose a u such that $4\delta^3 > c_c^u$ 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

$$b_{U,f} b_{W,h,g_1} b_{W,h,g_2} = b'$$

where $b_{U,f}$, b_{W,h,g_1} and b_{W,h,g_2} are the bits in the proof corresponding to $A_{U,true}(f)$, $A_{W,h,true}(g_1)$ and $A_{W,h,true}(g_2)$, respectively and b' is a constant. One might think that the right hand side would always be 1, but because of folding over true the bit corresponding to $A_{U,true}(f)$ in the proof might actually give the value of $A_{U,true}(-f)$. Thus the value of b' depends on our mechanism for folding and, of course, the identities of f, g_1 and g_2.

Let us now write down a set of linear equations with weights. Write down the equation

$$x_{b_{U,f}} x_{b_{W,h,g_1}} x_{b_{W,h,g_2}} = b'$$

where b' 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,h,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 $x \in L$ the maximal weight of simultaneously satisfiable equations is at least $1-\delta$ while if $x \notin L$, it is in view of Lemma 5.2 and the choice of u, at most $(1+\delta)/2$. The number of different equations is limited by the number of different choices of the verifier V. There are at most m^u choices for W and once W is chosen, at most 3^u choices for U. Given U 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 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 can determine the maximal total weight of simultaneously satisfiable equation within a factor smaller than

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

can be used to determine whether $x \in L$ and hence this task must be NP-hard. This proves the theorem if we allow weighted equations.

As is standard, the weights can be eliminated by duplicating each equation a suitable number of times. We leave the details to the interested reader. ∎

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 5.4 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 5.5 *For any $\epsilon > 0$, $k \geq 3$ it is NP-hard to approximate Max-Ek-Lin-2 within a factor $2-\epsilon$. Said equivalently, Max-Ek-Lin-2 is non-approximable beyond the random assignment threshold.*

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 $(x_i)_{i=1}^n$. Add the same $k-3$ new variables $(y_i)_{i=1}^{k-3}$ in every 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 of the smaller system. Thus we have a correct reduction from the case $k = 3$ to the case with $k > 3$. ∎

Sometimes it is useful to have systems of equations 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 side since it is determined by the folding convention. We next establish that if we have four variables in each equation then we can have the right hand side -1 in all equations. Note that we cannot have right hand side 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 satisfies all equations.

Theorem 5.6 *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. Said equivalently, Max-E4-Lin-2 with right hand side -1 is non-approximable beyond the random assignment threshold.*

Proof: We construct a special purpose PCP. Since we want to control the right hand side of the obtained equations we do not use folding over true.

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

Written proof. A SWP(u).
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 choose, uniformly at random, a variable x_{k_i} occurring in C_{j_i}. Set $U = \{x_{k_1}, x_{k_2} \ldots x_{k_u}\}$, W to be the set of all variables occurring in the chosen clauses, and $h = \wedge_{i=1}^{u} C_{j_i}$.

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 iff $A_U(f_1)A_U(f_2)A_{W,h}(g_1)A_{W,h}(g_2) = -1$.

We have

Lemma 5.7 *The completeness of Test $L^{\epsilon}_{2,-1}(u)$ is at least $1-\epsilon$.*

Proof: It is not difficult to see that the verifier accepts a correct SWP unless $\mu(y|_W) = -1$ where y is the satisfying assignment defining the proof. ∎

We turn to establishing the soundness of $L^{\epsilon}_{2,-1}$.

Lemma 5.8 *For any $\epsilon > 0$, $\delta > 0$, suppose that the probability that the verifier of Test $L^{\epsilon}_{2,-1}(u)$ 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 $2\epsilon\delta$.*

Proof: Fix U, W and h and let $A = A_U$ and $B = A_{W,h}$. Since $-A(f_1)A(f_2)B(g_1)B(g_2)$ is one if the test accepts and negative one if it rejects we want to analyze

$$E_{f_1,f_2,g_1,\mu}[-A(f_1)A(f_2)B(g_1)B(g_2)]. \qquad (17)$$

We replace each term by its Fourier expansion and using the linearity of expectation and the definition of g_2 we arrive at

$$-\sum_{\alpha_1,\alpha_2,\beta_1,\beta_2} \hat{A}_{\alpha_1}\hat{A}_{\alpha_2}\hat{B}_{\beta_1}\hat{B}_{\beta_2}E_{f_1,f_2,g_1,\mu}\left[\chi_{\alpha_1}(f_1)\chi_{\alpha_2}(f_2)\chi_{\beta_1}(g_1)\chi_{\beta_2}(-f_1f_2g_1\mu)\right]. \quad (18)$$

By Lemma 2.27, Lemma 2.28, and Lemma 2.30 we see that

$$\chi_{\alpha_1}(f_1)\chi_{\alpha_2}(f_2)\chi_{\beta_1}(g_1)\chi_{\beta_2}(-f_1f_2g_1\mu) =$$
$$\chi_{\alpha_1\Delta\pi_2(\beta_2)}(f_1)\chi_{\alpha_2\Delta\pi_2(\beta_2)}(f_2)\chi_{\beta_1\Delta\beta_2}(g_1)\chi_{\beta_2}(-\mu).$$

Since f_1, f_2, g_1 and μ are chosen independently we can calculate the expected value of each factor separately. By Lemma 2.29 we see that unless $\beta_1 = \beta_2 = \beta$ and $\alpha_1 = \alpha_2 = \pi_2^U(\beta)$ the expected value is 0. Finally, since

$$E_\mu[\chi_\beta(\mu)] = (1-2\epsilon)^{|\beta|}$$

we have reduced (17) to

$$\sum_\beta \hat{A}^2_{\pi_2(\beta)}\hat{B}^2_\beta(-1)^{|\beta|+1}(1-2\epsilon)^{|\beta|}. \quad (19)$$

The expected value of (19) over random U, W, and h is, by the assumption of the lemma, δ. Since we are not folding over true it might be that the term corresponding to $\beta = \emptyset$ is nonzero. It is, however, non-positive and hence dropping this term only increases the sum and hence by the assumption of the lemma we conclude that

$$\sum_{\beta\neq\emptyset} E_{U,W,h}\left[\hat{A}^2_{\pi_2(\beta)}\hat{B}^2_\beta(1-2\epsilon)^{|\beta|}\right] \geq \delta. \quad (20)$$

Now we define a strategy for P_1 and P_2. Given U, P_2 chooses α with probability $\hat{A}^2_\alpha$ and returns a random $x \in \alpha$ while P_1 when asked W and h chooses a β with probability $\hat{B}^2_\beta$ and returns a random $y \in \beta$. If either α or β is empty the corresponding prover gives up. Note that by Lemma 2.34 any y returned by P_1 satisfies the chosen clauses and reasoning as in the proof of Lemma 5.2 we see that the success probability of the strategy of the provers is at least

$$E_{U,W,h}\left[\sum_{\beta\neq\emptyset}\hat{A}^2_{\pi_2(\beta)}\hat{B}^2_\beta|\beta|^{-1}\right]. \quad (21)$$

By Lemma 5.3 with $s = 1$,

$$(2\epsilon|\beta|)^{-1} \geq e^{-2\epsilon|\beta|} \geq (1-2\epsilon)^{|\beta|}.$$

Comparing with (20) we conclude that the success probability is at least $2\delta\epsilon$ and the proof is complete. ∎

Now, Theorem 5.6 follows from Lemma 5.7 and Lemma 5.8 very much as Theorem 5.4 followed from Lemma 5.1 and Lemma 5.2. Essentially the only difference is that since we are not folding over true all right hand sides are -1. We leave the details to the reader. ■

Theorem 5.6 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 in a general Abelian group. Note that a particularly interesting case is that of linear equations mod p, but since the proof in this special case is essentially identical to the case for general Abelian groups we only give the general case. It might be constructive to think of Γ as $\mathbb{Z}_p$ at the first reading of the proof below.

Theorem 5.9 *For any $\epsilon > 0$ and any Abelian group Γ, it is NP-hard to approximate Max-E3-Lin-Γ within a factor $|\Gamma| - \epsilon$. Said equivalently, Max-E3-Lin-Γ is non-approximable beyond the random assignment threshold.*

Proof: We use the notation given in Section 2.6. Remember that Γ is written multiplicatively and each element is a k-tuple of complex numbers. The identity is denoted by 1^k.

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

Written proof. A SWΓP(u).
Desired property. To check that it is a correct SWΓP(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 choose, uniformly at random, a variable x_{k_i} occurring in C_{j_i}. Set $U = \{x_{k_1}, x_{k_2} \ldots x_{k_u}\}$, W to be the set of all variables occurring in the chosen clauses, and $h = \wedge_{i=1}^{u} C_{j_i}$.
2. Choose $f \in \mathcal{F}_U^{\Gamma}$ with the uniform probability.
3. Choose $g_1 \in \mathcal{F}_W^{\Gamma}$ with the uniform probability.
4. Choose a function $\mu \in \mathcal{F}_W^{\Gamma}$ by setting $\mu(y) = 1^k$ with probability $1 - \epsilon$ and otherwise $\mu(y) = \gamma$ where γ is chosen randomly and uniformly in Γ. This is done independently for each $y \in \{-1,1\}^W$.
5. Set $g_2 = (fg_1\mu)^{-1}$, 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 iff $A_{U,\Gamma}(f)A_{W,h,\Gamma}(g_1)A_{W,h,\Gamma}(g_2) = 1^k$.

We leave to the reader to verify that the verifier accepts a correct SWP when μ takes the value 1^k on the satisfying assignment. From this we conclude:

Lemma 5.10 *The completeness of Test $L_\Gamma^\epsilon(u)$ is at least $1-\epsilon$.*

Lemma 5.11 below analyzes the soundness of Test $L_\Gamma^\epsilon(u)$. Theorem 5.9 follows from Lemma 5.10 and Lemma 5.11 in the same way as Theorem 5.4 followed from Lemma 5.1 and Lemma 5.2. We omit the details. ∎

Lemma 5.11 *For any $\epsilon > 0$, $\delta > 0$, suppose that the probability that the verifier of Test $L_\Gamma^\epsilon(u)$ accepts is $(1+\delta)/|\Gamma|$. 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 $2\delta^2\epsilon|\Gamma|^{-2}$.*

Proof: The test succeeds if $A_{U,\Gamma}(f)A_{W,h,\Gamma}(g_1)A_{W,h,\Gamma}(g_2)$ is 1^k and fails otherwise. To evaluate the general performance we want to convert this to a rational number and we consider

$$\sum_{\gamma^*:\gamma^*\in\Gamma^*,\gamma^*\neq 0^k} (A_{U,\Gamma}(f)A_{W,h,\Gamma}(g_1)A_{W,h,\Gamma}(g_2))^{\gamma^*},$$

where Γ^* is the dual group of Γ written additively.

The reason to study this number is given by the following lemma.

Lemma 5.12 *Suppose $\gamma \in G$, and consider*

$$\sum_{\gamma^*:\gamma^*\in\Gamma^*,\gamma^*\neq 0^k} \gamma^{\gamma^*}.$$

This is sum is $|\Gamma|-1$ if $\gamma = 1^k$ and otherwise it is -1.

Proof: The statement of the lemma is equivalent to the statement that if we sum over all γ^* in Γ^*, then the sum is $|\Gamma|$ and 0 in the two cases, respectively.

The first part of the lemma follows from the fact that each term is 1 and there are $|\Gamma|$ terms. For the other part take any j, $1 \le j \le k$ such that $\gamma_j \neq 1$. Then, as we vary γ_j^* over all its possible values $\gamma_j^{\gamma_j^*}$ varies over a complete set of roots of unity. It follows that $\sum_{\gamma^*} \gamma^{\gamma^*} = 0$, which, as observed above, implies the lemma. ∎

By Lemma 5.12 and the assumption of Lemma 5.11, we have

$$E_{U,W,h,f,g_1,\mu}\left[\sum_{\gamma^*\neq 0^k} (A_{U,\Gamma}(f)A_{W,h,\Gamma}(g_1)A_{W,h,\Gamma}(g_2))^{\gamma^*}\right] = \delta. \tag{22}$$

Now, fix U, W, h and γ^* and set $A = A_{U,\Gamma}^{\gamma^*}$ and $B = A_{W,h,\Gamma}^{\gamma^*}$. Let us analyze the corresponding term in (22) by replacing each function by the Fourier expansion.

$$E_{f,g_1,\mu}\left[A_{U,\Gamma}(f)A_{W,h,\Gamma}(g_1)A_{W,h,\Gamma}(g_2))^{\gamma^*}\right] =$$

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

$$\sum_{\alpha,\beta_1,\beta_2}\hat{A}_\alpha\hat{B}_{\beta_1}\hat{B}_{\beta_2}E_{f,g_1,\mu}\left[\chi_\alpha(f)\chi_{\beta_1}(g_1)\chi_{\beta_2}((fg_1\mu)^{-1})\right] =$$

$$\sum_{\alpha,\beta_1,\beta_2}\hat{A}_\alpha\hat{B}_{\beta_1}\hat{B}_{\beta_2}E_f\left[\chi_{\alpha-\pi_\Gamma(\beta_2)}(f)\right]E_{g_1}\left[\chi_{\beta_1-\beta_2}(g_1)\right]E_\mu\left[\chi_{\beta_2}((\mu)^{-1})\right].$$

The first equality is obtained from the Fourier expansion while the second equality follows from the definition of g_2 and the linearity of expectation. The third inequality follows by Lemma 2.36, Lemma 2.37, and Lemma 2.40 and the fact that f, g_1 and μ are independent.

By Lemma 2.38 one the first two expected values is 0 unless $\alpha = \pi_\Gamma(\beta_2)$ and $\beta_1 = \beta_2$. Finally, 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 (22) equals

$$E_{U,W,h}\left[\sum_\beta \hat{A}_{\pi_\Gamma(\beta)}\hat{B}_\beta^2(1-\epsilon)^{s(\beta)}\right]. \tag{23}$$

We conclude that the there is a γ_0^* such that the absolute value of (23) is at least $\delta|\Gamma|^{-1}$. Fix the value of this γ_0^*. We are now ready to define the strategies of the provers.

On receiving W and h, P_1 considers the table $B = A_{W,h,\Gamma}^{\gamma_0^*}$, selects a β with probability $|\hat{B}_\beta|^2$ and returns a random y subject to $\beta(y) \neq 0^k$.

On receiving U, P_2 considers the table $A = A_{U,\Gamma}^{\gamma_0^*}$ and selects an α with probability $|\hat{A}_\alpha|^2$ and returns a random x subject to $\alpha(x) \neq 0^k$.

Note that by Lemma 2.44 the set of candidates for x and y are always nonempty and by Lemma 2.46 any y returned by P_1 always satisfies the selected clauses. Thus we need only analyze the probability that $y|_U = x$. This happens with probability at least

$$E_{U,W,h}\left[\sum_\beta |\hat{A}^2_{\pi_\Gamma(\beta)}\hat{B}_\beta^2|s(\beta)^{-1}\right].$$

Using (15) we see that this is bounded from below by

$$\left(E_{U,W,h}\left[\sum_\beta \left|\hat{A}_{\pi_\Gamma(\beta)}\hat{B}_\beta^2\right| s(\beta)^{-1/2}\right]\right)^2,$$

and by Lemma 5.3 with $s = 1/2$,

$$(2\epsilon s(\beta))^{-1/2} \geq e^{-\epsilon s(\beta)} \geq (1-\epsilon)^{s(\beta)}. \tag{24}$$

These facts combine with the fact that (23) is at least $\delta|\Gamma|^{-1}$ to show that the probability of the verifier accepting the given prover strategy is at least $2\epsilon\delta^2|\Gamma|^{-2}$ and Lemma 5.11 follows. ∎

Theorem 5.9 can be extended to more variables in each equation yielding similar results as Theorem 5.5. We omit the details.

It remains to study the case of 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 $x_i x_j = 1$ makes the problem more general, but it does not prevent the use of semidefinite programming (as in [20]) 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, by a construction usually referred to as a gadget and we proceed as follows.

Given an equation $xyz = c$ we construct a constant number of equations in two variables involving the variables x, y, z and some new auxiliary variables. These constraints come with weights. It is an α-gadget, iff for any x, y, z that satisfies $xyz = c$ one can adjust the auxiliary variables to satisfy constraints of total weight α while if $xyz \neq c$ then the maximum obtainable is exactly $\alpha - 1$. For a more thorough discussion of gadgets we refer to the paper by Trevisan et al. [35].

We have the following

Lemma 5.13 *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: This is Lemma 2.8 of [35] and we only sketch the proof.

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 $w\alpha + (1-w)(\alpha-1)$. Since it is NP-hard to distinguish the two cases when $w = 1-\delta$ and $w = \frac{1}{2}+\delta$, if we could determine the optimum of the transformed problem to a better accuracy than

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

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

Using this we have

Theorem 5.14 *For any $\epsilon > 0$ it 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 [34] using the techniques of [35]. We start with an equation of the form $x_1x_2x_3 = 1$. 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, α') of the cube we have the equation

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

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. Since negating all variables does not change a solution to Max-E2-Lin-2 we can assume that z takes the value 1.

Let us consider an assignment that satisfies $x_1x_2x_3 = 1$. 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 y_α 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 $x_1x_2x_3 = -1$ an enumeration establishes that it is only possibly to satisfy 10 equations. Thus we have constructed a 6-gadget and Theorem 5.14 follows from Lemma 5.13. ∎

5.1 Extensions to other CSPs

In this section we prove that any CSP where the predicate P is implied by the predicate of linearity inherits non-approximability. Note that negating one of the inputs to P does not change the corresponding CSP and hence exchanging $xyz = 1$ to $xyz = -1$ below gives an equivalent theorem.

Theorem 5.15 *Let P be a predicate on 3 bits such that $P(x,y,z) = -1$ for any x,y,z satisfying $xyz = 1$, then the CSP given by P is non-approximable beyond the random assignment threshold.*

Proof: We establish this by using a slight modification of $L_2^\epsilon(u)$, in that we change the acceptance criteria to requiring that $(A_{U,true}(f), A_{W,h,true}(g_1), A_{W,h,true}(g_2))$ satisfies P. This condition is strictly more generous than that of $L_2^\epsilon(u)$ and thus completeness does not decrease and remains at least $1 - \epsilon$.

Let us look at the soundness. Consider the special case when P is the predicate "not one", i.e. it accepts unless exactly one input is true. We later show how to extend the result to other predicates. The multilinear expression

$$\frac{5 - x - y - z + xy + xz + yz + 3xyz}{8} \tag{25}$$

is one if $P(x, y, z)$ is true and 0 otherwise. Thus we analyze the expected value of (25) with $x = A_{U,true}(f)$, $y = A_{W,h,true}(g_1)$ and $z = A_{W,h,true}(g_2)$. Folding over true implies that $E[A_{U,true}(f)] = 0$ for a random function f and similarly the other terms of degree one in (25) have expected value 0. The pairs (f, g_1) and (f, g_2) are pairs of independent functions and thus

$$E[A_{U,true}(f)A_{W,h,true}(g_i)] = 0$$

for $i = 1, 2$. Finally, since the triplets (f, g_1, g_2) and $(-f, g_1, -g_2)$ are equally likely to be selected by the test

$$E[A_{W,h,true}(g_1)A_{W,h,true}(g_2)] = 0.$$

This implies that if the test accepts with probability $(5 + \delta)/8$ then

$$E[A_{U,true}(f)A_{W,h,true}(g_1)A_{W,h,true}(g_2)] = \delta/3$$

and we have obtained the basic relation (10) that we, in the proof of Lemma 5.2, proved implied the existence of successful strategies for P_1 and P_2. Since any predicate P can be written as a multilinear function the same analysis applies to all the predicates mentioned in the lemma. ∎

In our definition of CSP negating an input to P or permuting the inputs does not change the problem. Thus, in fact Theorem 5.15 only applies to 3 essentially different predicates accepting 5, 6 and 7 inputs respectively. For these three predicates Zwick [36] established Theorem 5.15 by giving reductions from the inapproximability result for linear equations given in Theorem 5.4. It is curious to note that Zwick proved that these are the only predicates on 3 variables that give CSPs which are non-approximable beyond the random assignment threshold.

Theorem 5.15 extends to predicates on an arbitrary number of bits.

Theorem 5.16 *Let P be a predicate on k bits where $k \geq 3$ such that $P(x_1, x_2, \ldots x_k) = -1$ for any $(x_i)_{i=1}^k$ satisfying $\prod_{i=1}^k x_i = 1$, then the CSP given by P is non-approximable beyond the random assignment threshold.*

Proof: The proof is very similar to the proof given above but since we proved the inapproximability of linear equations with k variables in each equation by a reduction we have to design a new PCP. In view of Theorem 5.15 we can clearly assume that $k \geq 4$. The PCP is as follows.

Test $L_2^{P,k,\epsilon}(u)$

Written proof. A SWP(u).
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 choose, uniformly at random, a variable x_{k_i} occurring in C_{j_i}. Set $U = \{x_{k_1}, x_{k_2} \ldots x_{k_u}\}$, W to be the set of all variables occurring in the chosen clauses, and $h = \wedge_{i=1}^u C_{j_i}$.
2. Choose $(f_i)_{i=1}^{k-2} \in \mathcal{F}_U$ independently each 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 $g_2 = g_1 \mu \prod_{i=1}^{k-2} f_i$.
6. Accept iff the vector $(A_{U,true}(f_i))_{i=1}^{k-2}$ concatenated with $(A_{W,h,true}(g_1), A_{W,h,true}(g_2))$ satisfies P.

Since P is true whenever the product of the input bits is 1, we conclude that the verifier always accepts the proof when $\mu(y|_W) = 1$ where y is the assignment coding a correct SWP(u). Thus the completeness of $L_2^{P,k,\epsilon}(u)$ is at least $1 - \epsilon$.

To analyze the soundness we write P as a multilinear function. Using an argument similar to that used in the proof of Theorem 5.15 we see that the only term of a multilinear expansion that is not 0 is

$$E[B(g_1)B(g_2)\prod_{i=1}^{k-2} A(f_i)].$$

This is analyzed as in the proof of Lemma 5.2 by the Fourier expansion and the result is

$$E_{U,W,h}\left[\sum_\beta \hat{A}_{\pi(\beta)}^{k-2} \hat{B}_\beta^2 (1-2\epsilon)^{|\beta|}\right].$$

The same strategy of P_1 and P_2 as in the proof of Lemma 5.2 can now be seen to make the verifier in the two-prover protocol accept with probability at least $2\delta\epsilon$. We omit the details. ∎

6 Satisfiability problems

We start with a direct consequence of Theorem 5.4.

Theorem 6.1 *For any $\epsilon > 0$ it is NP-hard to approximate Max-E3-Sat within a factor $8/7 - \epsilon$. Said equivalently, Max-E3-Sat is non-approximable beyond the random assignment threshold.*

Proof: This is a special case of Theorem 5.15 but let us also give the immediate 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 4-gadget and the result follows by Lemma 5.13. ∎

We want to extend Theorem 6.1 to prove that Max-E3-Sat is non-approximable beyond the random assignment threshold on satisfiable instances. The proof of this is rather complicated. To establish that Max-E4-Sat has the same property is more straightforward and since it presents most of the ideas involved we present this theorem first.

Theorem 6.2 *For any $\epsilon > 0$ it is NP-hard to distinguish satisfiable E4-Sat formulas from $(15/16+\epsilon)$-satisfiable E4-Sat formulas. Said equivalently, Max-4-Sat is non-approximable beyond the random assignment threshold on satisfiable instances.*

Proof: We first define the test.

Test 4S(u)

Written proof. A SWP(u).
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 choose, uniformly at random, a variable x_{k_i} occurring in C_{j_i}. Set $U = \{x_{k_1}, x_{k_2} \ldots x_{k_u}\}$, W to be the set of all variables occurring in the chosen clauses, and $h = \wedge_{i=1}^u C_{j_i}$.
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 $A_{U,true}(f) = A_{W,h,true}(g_1) = A_{W,h,true}(g_2) = A_{W,h,true}(g_3) = 1$.

Before analyzing the test, let us intuitively discuss its design. Since we want to obtain a result for E4-Sat we want to read 4 bits. The first choice is how to divide these between A_U and A_W. How to do this is far from obvious and is difficult to motivate at this point. It turns out that the complications in the proof comes from the correlation that appears among the chosen functions in $\mathcal{F}_W$ and to make this as small as possible we choose to read 3 bits in A_W. To get a reduction to Max-E4-Sat we need that the acceptance criteria should be $A_{U,true}(f) \vee A_{W,h,true}(g_1) \vee A_{W,h,true}(g_2) \vee A_{W,h,true}(g_3)$. Since we want perfect completeness we need to make sure that $f(y|_U) \vee g_1(y) \vee g_2(y) \vee g_3(y)$ is true for any y. Furthermore, to make a successful analysis by Fourier transforms it is important that each function is chosen with the uniform distribution. The reason for this is that the Fourier coefficients are averages and thus are most informative when inputs are chosen with the uniform probability. Giving these considerations, the goal of the design was to make the functions as independent as possible.

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

Lemma 6.3 *The completeness of test 4S(u) is 1.*

The lemma analyzing the soundness is given below. Theorem 6.2 follows from Lemma 6.3 and Lemma 6.4 in the same way as Theorem 5.4 followed from Lemma 5.1 and Lemma 5.2. We omit the details. ∎

Lemma 6.4 *If Test $4S(u)$ accepts with probability $(15+\epsilon)/16$, 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 $\epsilon^2/4$.*

Proof: We have that

$$1-\frac{1}{16}(1+A_{U,true}(f))(1+A_{W,h,true}(g_1))(1+A_{W,h,true}(g_2))(1+A_{W,h,true}(g_3)) \tag{26}$$

is 1 if the test accepts and 0 otherwise. This follows since each of $A_{U,true}(f)$, $A_{W,h,true}(g_1)$, $A_{W,h,true}(g_2)$ and $A_{W,h,true}(g_3)$ is either 1 or -1 and unless they are all 1, one factor in the product is 0 and the expression evaluates to 1. If all numbers are 1 the expression evaluates to 0.

We need to estimate the expected value of (26) which gives the probability of success. Fix U, W and h and let $A = A_{U,true}$ and $B = A_{W,h,true}$. We expand the product in (26) and estimate the expected value of each term separately. The only terms that can have a nonzero expected value are terms containing both $B(g_2)$ and $B(g_3)$. This follows since the collections (f, g_1, g_2) and (f, g_1, g_3) form independent random variables and, because of folding over true, the expected value of each single factor is 0. Thus the expected value of (26) equals

$$\begin{aligned}\frac{15}{16}-\frac{1}{16}(&E[B(g_2)B(g_3)]+E[A(f)B(g_2)B(g_3)]\\ &+E[B(g_1)B(g_2)B(g_3)]+E[A(f)B(g_1)B(g_2)B(g_3)]).\end{aligned} \tag{27}$$

Test 4S(u) is equally likely to produce the set (f, g_1, g_2, g_3) and $(-f, g_1, g_2, -g_3)$ and since both A and B are folded over true this implies that $E[B(g_2)B(g_3)] = 0$ and $E[B(g_1)B(g_2)B(g_3)] = 0$. Of the two remaining terms 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 linearity of expectation to obtain

$$\sum_{\alpha,\beta_1,\beta_2\beta_3} \hat{A}_\alpha \hat{B}_{\beta_1} \hat{B}_{\beta_2} \hat{B}_{\beta_3} E_{f,g_1,g_2,g_3}\left[\chi_\alpha(f)\chi_{\beta_1}(g_1)\chi_{\beta_2}(g_2)\chi_{\beta_3}(g_3)\right]. \tag{28}$$

Any term with $\beta_2 \neq \beta_3$ has expected value 0. This follows by Lemma 2.29 since if $y \in \beta_2 \Delta \beta_3$ then $g_2(y)$ (or $g_3(y)$ if $y \in \beta_3$) fulfills the conditions of that lemma. Thus we can assume that $\beta_2 = \beta_3 = \beta$ when studying the remaining terms.

If $y \in \beta_1 \setminus \beta$ then $g_1(y)$ is independent of all other factors and thus we can again apply Lemma 2.29. Since elements with different projections onto U are independent we need to estimate

$$E_{f,g_1,g_2,g_3}\left[\prod_{y\in\beta_1} g_1(y) \prod_{y\in\beta} g_2(y)g_3(y)\right] \tag{29}$$

and

$$E_{f,g_1,g_2,g_3}\left[f(x)\prod_{y\in\beta_1} g_1(y) \prod_{y\in\beta} g_2(y)g_3(y)\right] \tag{30}$$

where $\beta_1 \subseteq \beta$ and all elements of β project onto a fixed element, x, of U. We consider different cases depending on g_1. If $g_1(y) = -1$ for some $y \in \beta$ the expected value over the rest is 0 and thus we can concentrate on the case when $g_1(y) = 1$ for all $y \in \beta$. This happens with probability $2^{-|\beta|}$ and then (29) is equal to $(-f(x))^{|\beta|}$ while (30) equals $f(x)(-f(x))^{|\beta|}$. This means that (29) equals $2^{-|\beta|}$ when $|\beta|$ is even and 0 otherwise while (30) equals $-2^{-|\beta|}$ when $|\beta|$ is odd and 0 otherwise. Repeating this argument for all x in U we see that the terms are nonzero only when $\pi_2(\beta) = \alpha$ and hence (28) equals

$$\sum_{\beta} A_{\pi_2(\beta)} \hat{B}_\beta^2 (-1)^{|\beta|} 2^{-|\beta|} \sum_{\beta_1\subseteq\beta} \hat{B}_{\beta_1}. \tag{31}$$

The inner sum is, using Cauchy-Schwartz inequality, bounded by

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

and substituting this into (31), we get the upper bound

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

for the absolute value of (28).

Before continuing, let us consider $E[A(f)B(g_2)B(g_3)]$. We can repeat the calculations performed above with the only difference that there is no sum of β_1. We get the equality

$$E_{U,W,h}[A(f)B(g_2)B(g_3)] = \sum_{\beta} A_{\pi_2(\beta)} \hat{B}_\beta^2 (-1)^{|\beta|} 2^{-|\beta|}.$$

Summing up, for fixed U, W and h the probability of acceptance is at most

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

By the assumption of Lemma 6.4 we conclude that

$$E_{U,W,h}\left[\sum_{\beta} |A_{\pi_2(\beta)}|\hat{B}_\beta^2 2^{-|\beta|/2}\right] \geq \epsilon/2. \tag{32}$$

We are ready to define the strategy of the provers.

On receiving W and h, P_1 selects a random β with probability proportional to $\hat{B}_\beta^2$ and then a random $y \in \beta$. Similarly P_2 selects a random α with probability $\hat{A}_\alpha^2$ and then a random $x \in \alpha$.

By Lemma 2.32 both α and β are always nonempty and by Lemma 2.34, y always satisfies the selected clauses and thus we need only estimate the probability that $y|_U = x$. This is true with probability at least

$$E_{U,W,h}\left[\sum_{\beta} \hat{A}_{\pi_2(\beta)}^2 \hat{B}_\beta^2 |\beta|^{-1}\right].$$

By (15) this is at least

$$\left(E_{U,W,h}\left[\sum_{\beta} |\hat{A}_{\pi_\Gamma(\beta)}| B_\beta^2 |\beta|^{-1/2}\right]\right)^2,$$

and since $x^{-1/2} \geq 2^{-x/2}$ is true for all integers x we conclude, by (32), that the verifier in two-prover game accepts with probability at least $(\epsilon/2)^2$ and Lemma 6.4 follows. ∎

We turn to the more difficult problem of establishing that Max-E3-Sat is non-approximable beyond the random assignment threshold on satisfiable instances.

Theorem 6.5 *For any $\epsilon > 0$ it is NP-hard to distinguish satisfiable E3-CNF formulas from $(7/8+\epsilon)$-satisfiable E3-CNF formulas. Said equivalently, Max-E3-Sat is non-approximable beyond the random assignment threshold on satisfiable instances.*

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

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

Written proof. A SWP(u).
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 choose, uniformly at random, a variable x_{k_i} occurring in C_{j_i}. Set $U = \{x_{k_1}, x_{k_2} \ldots x_{k_u}\}$, W to be the set of all variables occurring in the chosen clauses, and $h = \wedge_{i=1}^u C_{j_i}$.

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 $A_{U,true}(f) = A_{W,h,true}(g_1) = A_{W,h,true}(g_2) = 1$.

The intuition of the construction is similar to the intuition for Test $4S$.

It is easy to see that we get perfect completeness and we omit the proof of the below lemma.

Lemma 6.6 *The completeness of Test $3S^\epsilon(u)$ is 1.*

To estimate the soundness we first write the acceptance criteria as

$$1 - \frac{1}{8}(1 + A_{U,true}(f))(1 + A_{W,h,true}(g_1))(1 + A_{W,h,true}(g_2)). \tag{33}$$

Fix U, W, and h and define $A = A_{U,true}$ and $B = A_{W,h,true}$. We expand the product and estimate the expected value of each term separately. Since both pairs (f, g_1) and (f, g_2) are pairs of random independent functions and the tables are folded over true, the only expected values that might be nonzero are the ones containing both $B(g_1)$ and $B(g_2)$. Thus the expected value of (33) equals

$$\frac{7}{8} - \frac{1}{8}(E[B(g_1)B(g_2)] + E[A(f)B(g_1)B(g_2)]). \tag{34}$$

We consider each term separately. Expanding $E[B(g_1)B(g_2)]$ by the Fourier expansion yields

$$\sum_{\beta_1,\beta_2} \hat{B}_{\beta_1}\hat{B}_{\beta_2} E\left[\prod_{y\in\beta_1} \chi_{\beta_1}(g_1)\chi_{\beta_2}(g_2)\right]. \tag{35}$$

By Lemma 2.29, any term with $\beta_1 \neq \beta_2$ has expected value 0. Furthermore, the parts of the product with different projections onto U are independent. Thus, we need to study

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

where all y project onto the same element x. It is not hard to calculate 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 $1 - s\epsilon$ 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 a value around $-s\epsilon$ for small s.

For $x \in \pi(\beta)$ let s_x denote the number of elements of β that project onto x. Then, by the above reasoning, (35) equals

$$\sum_\beta \hat{B}_\beta^2 \prod_{x\in\pi(\beta)} (\frac{1}{2}((-1)^{s_x} + (1-2\epsilon)^{s_x})). \tag{36}$$

One could have hoped to estimate this sum as a function of ϵ tending to 0 with ϵ, but unfortunately this is not true in general. To help the reader's intuition at this point let us sketch an example illustrating the problems.

Given any $\epsilon > 0$, we define W and β such that there is a constant $c > 0$, independent of ϵ such that

$$\left| E_U \left[\prod_{x\in\pi(\beta)} (\frac{1}{2}((-1)^{s_x} + (1-2\epsilon)^{s_x})) \right] \right| \geq c, \tag{37}$$

where the expectation is over a random $U \subset W$ selected with the induced probability, i.e., each element in each clause is selected with probability $1/3$.

Let W be defined by the triplets $(3i-2, 3i-1, 3i)$ for $i = 1, 2 \ldots k+1$ where $k = \lceil \epsilon^{-1} \rceil$ and let

$$\beta = 1^{3k+3} \cup_{i=2}^{k+1} \{e_{1,3i}, e_{2,3i}\},$$

where $e_{i,j}$ is the assignment giving the value -1 to x_i and x_j while giving the value 1 to all other variables. Let us analyze (37).

If U contains 1 or 2 then there will be many (about $k/3$) x such that $s_x = 1$ and thus the contribution to (37) is very small from these U.

On the other hand if U chooses 3 from the first triplet the elements pair up since $\pi_U(e_{1,3i}) = \pi_U(e_{2,3i})$ for all i. We expect that around $2k/3$ of these pairs project onto the all 1 assignment while the other, roughly $k/3$, pairs project onto distinct elements due to $3i$ being placed into U. Thus, in this case

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

is about

$$1/2(-1 + (1-2\epsilon)^{1+4k/3})(1 - 2\epsilon + 2\epsilon^2)^{k/3},$$

which is a negative value with absolute value bounded from below by an absolute constant, and this completes our example.

The set β in the above example is of size around ϵ^{-1} and it turns out that much larger and much smaller sets β are easy to control. This is useful since we can later vary ϵ.

Lemma 6.7 *There is a constant $c>0$ such that when g_1 and g_2 are chosen as in Test $3S^\epsilon(u)$ we have*

$$|E_{U,f,g_1,g_2}\left[(B(g_1)B(g_2)\right]| \leq 3\delta + \sum_{\beta \ | \ \delta\epsilon^{-1}\leq|\beta|\leq(2\delta^{-2})^{1/c}\epsilon^{-1}} \hat{B}_\beta^2.$$

The constant c is the constant from Lemma 6.9 below.

Proof: We split the sum (36) into three pieces depending on in which of the intervals $[1,\delta\epsilon^{-1}]$, $[\delta\epsilon^{-1},(2\delta^{-2})^{1/c}\epsilon^{-1}]$, and $[(2\delta^{-2})^{1/c}\epsilon^{-1},\infty]$, $|\beta|$ belongs. The sum over the middle interval need not be estimated since it appears on the right hand of the estimate in the lemma. For the small sets we have

Lemma 6.8

$$\left| \sum_{\beta \ | \ |\beta|\leq\delta\epsilon^{-1}} \hat{B}_\beta^2 E\left[\prod_{x\in\pi(\beta)} (\frac{1}{2}((-1)^{s_x}+(1-2\epsilon)^{s_x}))\right]\right| \leq \delta$$

Proof: As B is folded over true, we can assume that $|\beta|$ is odd and hence some $x\in\pi(\beta)$ must have an odd value of s_x. For this x

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

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

The part

$$\sum_{\beta \ | \ |\beta|\geq(2\delta^{-2})^{1/c}\epsilon^{-1}} \hat{B}_\beta^2 E\left(\prod_{x\in\pi(\beta)} (\frac{1}{2}((-1)^{s_x}+(1-2\epsilon)^{s_x}))\right) \tag{38}$$

requires some more work to estimate. In order to study this quantity let us define $S_\epsilon^U(\beta)$ by

$$S_\epsilon^U(\beta) = \epsilon\sum_x \min(s_x,\epsilon^{-1}).$$

One cay view $S_\epsilon^U(\beta)$ as a generalization of $|\pi(\beta)|$ and the key connection to the product above is given by (40) below. It is important to us to prove that $S_\epsilon^U(\beta)$ is likely to be large when β is large. The key lemma is given below. We postpone the proof of this lemma to an appendix.

Lemma 6.9 *Suppose $|\beta| \geq \epsilon^{-1}$. There is a constant $c > 0$ such that, if the clauses defining W are disjoint,*

$$E_U[1/S_\epsilon^U(\beta)] \leq (\epsilon|\beta|)^{-c}.$$

where the expected value is taken over a random set U constructed by selecting one variable from each of the clauses of W. One acceptable value for c is $\frac{1}{35}$.

Since the clauses defining W are disjoint with probability $1 - O(\frac{1}{n})$ we will when applying Lemma 6.9, for notational simplicity, ignore this condition. The way we use Lemma 6.9 is to apply Markovs inequality to it. Let us state this variant for easy reference.

Corollary 6.10 *Let c be the constant of Lemma 6.9. For $a, b > 1$ suppose $|\beta| = (ab)^{1/c}\epsilon^{-1}$. If the clauses defining W are disjoint, then except with probability a^{-1} it is true that $S_\epsilon^U(\beta) \geq b$. The probility is taken over a random U.*

Next we claim that

$$|\frac{1}{2}((-1)^s + (1-2\epsilon)^s)| \leq e^{-\min(1, s\epsilon)/2}. \tag{39}$$

To establish this we can assume $0 \leq s \leq \epsilon^{-1}$ and that s is even since other cases follow from this. We have $(1-2\epsilon)^s \leq e^{-2\epsilon s}$ and setting $f(x) = 2e^{-x/2} - (1+e^{-2x})$ we need to prove that $f(x) \geq 0$ for $x \in [0,1]$. We have $f''(x) = \frac{1}{2}e^{-x/2} - 4e^{-2x}$ and hence $f''(x) \leq 0$ in the interval in question and we only have to check the inequality at the end points. We have $f(0) = 0$ and $f(1) = 2e^{-1/2} - (1+e^{-2}) > 0$.

From (39) it follows that

$$\prod_{x \in \pi(\beta)} (\frac{1}{2}((-1)^{s_x} + (1-2\epsilon)^{s_x})) \leq e^{-S_\epsilon^U(\beta)/2}. \tag{40}$$

Thus (38) is bounded from above by

$$\sum_{\beta \mid |\beta| \geq (2\delta^{-2})^{1/c}\epsilon^{-1}} \hat{B}_\beta^2 E[e^{-S_\epsilon^U(\beta)/2}].$$

From Corollary 6.10 it follows that except with probability at most δ, it is true that $S_\epsilon^U(\beta) \geq 2\delta^{-1}$. This implies that the expected value of the term corresponding to β in (38) is at most $(\delta + e^{-\delta^{-1}})\hat{B}_\beta^2 \leq 2\delta\hat{B}_\beta^2$, where we used Lemma 5.3 with $s = 1$ and $x = \delta^{-1}$. Summing over β finishes the proof of Lemma 6.7. ∎

Let us now consider

$$E_{U,W,h,f,g_1,g_2}[A(f)B(g_1)B(g_2)] \tag{41}$$

in more detail.

Lemma 6.11 *If*

$$|E_{U,W,h,f,g_1,g_2}[A(f)B(g_1)B(g_2)]| \geq \delta$$

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 at least $\epsilon(\delta^2/64)^{1+1/c}$. Here c is the constant from Lemma 6.9.

Proof: Fix U, W, and h. We use the Fourier expansion to obtain

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

If $\beta_1 \neq \beta_2$ then we can apply Lemma 2.29 with $y \in \beta_1 \Delta \beta_2$ to see that the expected value is 0 and thus we can assume $\beta_1 = \beta_2 = \beta$. Lemma 2.29 also applies to the case $\alpha \not\subseteq \pi(\beta)$ and thus we assume $\alpha \subseteq \pi(\beta)$. A calculation shows that

$$E_{f,g_1,g_2}\left[\chi_\alpha(f)\chi_\beta(g_1 g_2)\right] = \prod_{x\in\alpha\cap\pi(\beta)} (\frac{1}{2}((-1)^{s_x} - (1-2\epsilon)^{s_x})) \prod_{x\in\pi(\beta)\setminus\alpha} (\frac{1}{2}((-1)^{s_x} + (1-2\epsilon)^{s_x})),$$

where $s_x = |\pi^{-1}(x) \cap \beta|$. Let us denote this value by $p(\alpha, \beta)$. The assumption of the lemma implies

$$|E_{U,W,h} \sum_{\beta,\alpha\subseteq\pi(\beta)} \hat{A}_\alpha \hat{B}_\beta^2 p(\alpha,\beta)| \geq \delta. \tag{42}$$

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 $\hat{A}_\alpha^2$ and returns a random $x \in \alpha$. Similarly P_1 chooses a random β with probability $\hat{B}_\beta^2$ and returns a random $y \in \beta$. By Lemma 2.32 both α and β are always nonempty and by Lemma 2.34, y always satisfies the selected clauses and thus we need only estimate the probability that $y|_U = x$. This happens with probability at least

$$E_{U,W,h}\left[\sum_{\beta,\alpha\subseteq\pi(\beta)} \hat{A}_\alpha^2 \hat{B}_\beta^2 |\beta|^{-1}\right]. \tag{43}$$

We need to prove that this is large based on (42) and we proceed to establish the connection.

The quantity that multiplies $\hat{B}_\beta^2$ in (42) is

$$\begin{aligned}\sum_{\alpha\subseteq\pi(\beta)} \hat{A}_\alpha p(\alpha,\beta) &\leq \left(\sum_{\alpha\subseteq\pi(\beta)} \hat{A}_\alpha^2\right)^{1/2} \left(\sum_{\alpha\subseteq\pi(\beta)} p^2(\beta,\alpha)\right)^{1/2} \\ \leq \left(\sum_{\alpha\subseteq\pi(\beta)} p^2(\beta,\alpha)\right)^{1/2} &\leq e^{-S_\epsilon^U(\beta)/4}.\end{aligned} \tag{44}$$

To see the last inequality in (44) note that the sum equals

$$\prod_{x\in\pi(\beta)}((\frac{1}{2}((-1)^{s_x}-(1-2\epsilon)^{s_x}))^2+(\frac{1}{2}((-1)^{s_x}+(1-2\epsilon)^{s_x}))^2). \tag{45}$$

The factor corresponding to x in (45) is of the form a^2+b^2 where $|a|+|b|=1$ and, by (39), $\max(|a|,|b|)\le e^{-\min(1,s_x\epsilon)/2}$, and hence it is bounded by $e^{-\min(1,s_x\epsilon)/2}$. Multiplying over x gives the last inequality in (44).

Define S to be $(64\delta^{-2})^{1/c}\epsilon^{-1}$ and consider any term with $|\beta|\ge S$. By Corollary 6.10, except with probability $\delta/4$ we have $S_\epsilon^U(\beta)\ge 16\delta^{-1}$. We conclude that for such terms in (42) are bounded from above by

$$E_{h,W}\left[\hat{B}_\beta^2 E_U[e^{-S_\epsilon^U(\beta)/4}]\right]\le E_{h,W}\left[\hat{B}_\beta^2(\delta/4+e^{-4\delta^{-1}})\right]\le E_{h,W}\left[\hat{B}_\beta^2\delta/2\right].$$

This implies that if we discard all terms in (42) with $|\beta|\ge S$, the remaining expected value is at least $\delta/2$. Returning to the analysis of (43) we see that it is at least

$$S^{-1}E_{U,W,h}\left[\sum_{\beta,\alpha\subseteq\pi(\beta),|\beta|\le S}\hat{B}_\beta^2\hat{A}_\alpha^2\right].$$

Now by the above reasoning we have

$$\begin{aligned}(\delta/2)^2\le\left(E_{U,W,h}\left[\sum_{\beta,\alpha\subseteq\pi(\beta),|\beta|<S}\hat{B}_\beta^2\hat{A}_\alpha p(\alpha,\beta)\right]\right)^2 &\le\\ E_{U,W,h}\left[\left(\sum_{\beta,\alpha\subseteq\pi(\beta),|\beta|<S}\hat{B}_\beta^2\hat{A}_\alpha p(\alpha,\beta)\right)^2\right] &\le\\ E_{U,W,h}\left[\left(\sum_{\beta,\alpha\subseteq\pi(\beta),|\beta|<S}\hat{B}_\beta^2\hat{A}_\alpha^2\right)\left(\sum_{\beta,\alpha\subseteq\pi(\beta),|\beta|<S}\hat{B}_\beta^2 p^2(\alpha,\beta)\right)\right] &\le\\ E_{U,W,h}\left[\sum_{\beta,\alpha\subseteq\pi(\beta),|\beta|<S}\hat{B}_\beta^2\hat{A}_\alpha^2\right],&\end{aligned}$$

where the last inequality follows from

$$\sum_\beta\sum_{\alpha\subseteq\beta}\hat{B}_\beta^2p^2(\alpha,\beta)\le\sum_\beta\hat{B}_\beta^2\le 1.$$

where we again used the last inequality of (44). We conclude that the verifier in the two-prover protocol accepts with the given strategies with probability at least

$$S^{-1}(\delta/2)^2\ge\epsilon(\delta^2/64)^{1+1/c},$$

and the proof is complete. ∎

We are now in position to prove Theorem 6.5. We describe the appropriate test. Given a positive $\delta < 1/2$ we proceed as follows, where c is the constant of Lemma 6.9.

Test F3S$^\delta(u)$

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

From Lemma 6.6 we conclude that we have perfect completeness.

Lemma 6.12 *The completeness of Test F3S$^\delta(u)$ is 1.*

On the soundness side we have the crucial lemma below. Using Lemma 6.12 and Lemma 6.13 we complete the proof of Theorem 6.5 in a similar way that Theorem 5.4 followed from Lemma 5.1 and Lemma 5.2. We omit the details. ∎

Lemma 6.13 *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 $2^{-O(\delta^{-1}\log \delta^{-1})}$.*

Proof: As given by (34) the probability that the verifier accepts is

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

where f, g_1 and g_2 are chosen as described in the test. By Lemma 6.7 we have for fixed W and h,

$$|E_{U,f,g_1,g_2}[A_{W,h,true}(g_1) A_{W,h,true}(g_2))| \le \frac{1}{t} \sum_{i=1}^{t} (3\delta + \sum_{\beta \mid \delta\epsilon_i^{-1} \le |\beta| \le (2\delta^{-2})^{1/c}\epsilon_i^{-1}} \hat{B}_\beta^2) \le$$

$$3\delta + \frac{1}{t} \le 4\delta$$

since the intervals of summations are disjoint. Thus, from the assumption of the lemma there must be some j such that

$$E_{U,W,h,f,g_1,g_2}[A_{U,true}(f) A_{W,h,true}(g_1) A_{W,h,true}(g_2)] \ge \delta$$

when f, g_1 and g_2 are chosen as in test $3S^{\epsilon_j}(u)$. For this j we get by Lemma 6.11 a strategy for P_1 and P_2 with success probability $\epsilon_j^{O(1)} \delta^{O(1)}$. Now $\epsilon_j = \delta^{O(j)}$ and since $j \le t = \lceil \delta^{-1} \rceil$ the lemma follows. ∎

It is not hard to extend Theorem 6.5 to longer clauses.

Theorem 6.14 *For any $\epsilon > 0$ and any $k \geq 4$ it is NP-hard to distinguish satisfiable Ek-CNF formulas from at most $1 - 2^{-k} + \epsilon$ satisfiable Ek-CNF formulas. Said equivalently, Max-Ek-Sat is non-approximable beyond the random assignment threshold on satisfiable instances.*

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 $N + O$. A small calculation yields the desired result. ∎

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 6.15 *Consider the CSP where each constraint is a disjunction of literals of size at least 3. This problem is non-approximable beyond the random assignment threshold on satisfiable instances.*

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

Theorem 6.16 *For any $\epsilon > 0$ it is NP-hard to approximate Max-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 [9, 35] and Lemma 5.13. ∎

6.1 Implied results for other CSPs

As in the case of Theorem 5.4, the constructed PCPs can be used, with only minor modifications, to obtain results for other CSPs. This time we look at constraints that are more restrictive than the originally intended constraints. First we derive a consequence of the proof of Theorem 6.2.

Theorem 6.17 *Let P be a predicate on 4 variables such that*

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

Then the CSP given by P on 4 variables is non-approximable beyond the random assignment threshold on satisfiable instances.

Proof: The test is given by 4S(u) except that the final test is replaced by requiring that $(A_{U,true}(f), A_{W,h,true}(g_1), A_{W,h,true}(g_2), A_{W,h,true}(g_3))$ satisfies the predicate P. From the definition of g_3 it follows that $(f(y|_U), g_1(y), g_2(y), g_3(y))$

never takes any of the values that falsifies P and thus this modification does not cause the verifier to reject a correct PCP and we still have perfect completeness.

To analyze the soundness we write the acceptance criteria as a multilinear expression in $A_{U,true}(f)$ and $A_{W,h,true}(g_i)$. We have already established that each multilinear term has small expected value unless there is a good strategy for P_1 and P_2 in the two prover multiprover game. We omit the details. This is sufficient to prove Theorem 6.17. ∎

In the test $3S^{\epsilon}(u)$ (and hence $F3S^{\delta}$) we constructed functions f, g_1 and g_2 such that the triples $(f(y|_U), g_1(y), g_2(y))$ never take the values $(1,1,1)$ or $(1,-1,-1)$. In our original application to Max-E3-Sat we just needed that $(1,1,1)$ was avoided. If we replace the acceptance criteria by the predicate

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

we get, by a very similar proof, the following theorem. We omit the details.

Theorem 6.18 *The CSP on 3 variables problem given by the predicate OXR is non-approximable beyond the random assignment threshold on satisfiable instances.*

7 Set splitting

The verifier that gives a result for set splitting must be a bit different from previous verifiers for some basic reasons. Firstly, there is no negation present in set splitting and hence we cannot fold over true. Secondly, we cannot have the bipartite situation when we ask some questions in A_U and then some questions in $A_{W,h}$. If this was the case a cheating prover could fool the verifier by setting $A_U(f) = 1$ 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 non-approximable beyond the random assignment threshold. As in the case for Max-E3-Sat it is more complicated to get the result for satisfiable instances.

Theorem 7.1 *For any $\epsilon > 0$, it is NP-hard to approximate E4-Set splitting within a factor $8/7 - \epsilon$. Said equivalently, E4-Set splitting is non-approximable beyond the random assignment threshold.*

Proof: We first give the test. Assume $\epsilon < \frac{1}{2}$.

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

Written proof. A SWP(u).
Desired property. To check that it is a correct SWP(u) for a given formula $\varphi = C_1 \wedge C_2 \dots C_m$.
Verifier.

1. Choose u variables x_{k_i}, uniformly at random and set $U = \{x_{k_1}, x_{k_2}, \ldots, x_{k_u}\}$. 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 appearing in these clauses and $h^1 = \wedge_{i=1}^{u} C_{j_i^1}$. By a similar and independent procedure produce W^2 and h^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 iff $A_{W^1,h^1}(g_1^1)$, $A_{W^1,h^1}(g_2^1)$, $A_{W^2,h^2}(g_1^2)$, and $A_{W^2,h^2}(g_2^2)$ are not all equal.

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

Lemma 7.2 *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 $A_{W^2,h^2}(g_1^2) \neq A_{W^2,h^2}(g_2^2)$. Similarly if $f(x|_U) = -1$, unless $\mu^1(x|_{W^1}) = -1$ we have $A_{W^1,h^1}(g_1^1) \neq A_{W^1,h^1}(g_2^1)$. Thus in either case we accept with probability $1 - \epsilon$. ∎

For the soundness we have the corresponding lemma below. Theorem 7.1 follows from Lemma 7.2 and Lemma 7.3 in similar way to which Theorem 5.4 followed from Lemma 5.1 and Lemma 5.2. We omit the details. ∎

Lemma 7.3 *If Test $SS^\epsilon(u)$ accepts with probability $(7 + \delta)/8$, 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 $2\delta\epsilon$.*

Proof: Fix U, W^1, h^1, W^2, and h^2 and set $A = A_{W^1,h^1}$ and $B = A_{W^2,h^2}$. The expression

$$\begin{aligned} 1 - \frac{1}{16}&((1 + A(g_1^1))(1 + A(g_2^1))(1 + B(g_1^2))(1 + B(g_2^2))) \\ -\frac{1}{16}&((1 - A(g_1^1))(1 - A(g_2^1))(1 - B(g_1^2))(1 - B(g_2^2))) \end{aligned} \tag{46}$$

is 1 if the test accepts and 0 otherwise. Expanding (46) we get

$$\frac{7}{8}-\frac{A(g_1^1)B(g_1^2)+A(g_2^1)B(g_1^2)+A(g_1^1)B(g_2^2)+A(g_2^1)B(g_2^2)}{8}$$
$$-\frac{A(g_1^1)A(g_2^1)+B(g_1^2)B(g_2^2)+A(g_1^1)A(g_2^1)B(g_1^2)B(g_2^2)}{8}. \tag{47}$$

The expectation of (47) gives the probability that the verifier accepts the proof and we estimate the expectation of each term separately. 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 thus we bound the terms from below. First we have

Lemma 7.4 *For $1 \le i,j \le 2$ we have*

$$E_{W^1,W^2,h^1,h^2,g_1^1,g_1^2,\mu^1,\mu^2}[A(g_i^1)B(g_j^2)] \ge 0.$$

This is true for any fixed choices of U and f.

Proof: First note that g_1^k and g_2^k have the same distribution and hence we only need to consider the case $i=j=1$. Once U is fixed, g_1^1 and g_1^2 are selected independently with the same distribution and hence the two numbers $A(g_1^1)$ and $B(g_1^2)$ are also independent and selected with the same distribution. It follows that

$$E_{W^1,h^1,g_1^1,W^2,h^2,g_1^2}[A(g_1^1)B(g_1^2)] = E_{W^1,h^1,g_1^1}[A(g_1^1)]^2 \ge 0.$$

■

We proceed to handle two more terms.

Lemma 7.5 *We have $E_{f,g_1^1,\mu^1}[A(g_1^1)A(g_2^1)] \ge 0$ This is true for any fixed choice of U, W^1, and h^1. The similar statement is true for $B(g_1^2)B(g_2^2)$.*

Proof: We replace $A(g_1^1)$ by the Fourier expansion and then using the linearity of expectation, Lemma 2.27, Lemma 2.28, Lemma 2.30, and the fact that f, g_1^1 and μ^1 are independent to obtain

$$\begin{aligned} E_{f,g_1^i,\mu^i}[A(g_1^1)A(g_2^1)] &= \\ \sum_{\alpha_1,\alpha_2} \hat{A}_{\alpha_1}\hat{A}_{\alpha_2} E_{f,g_1^1,\mu^1}\left[\chi_{\alpha_1}(g_1^1)\chi_{\alpha_2}(g_2^1)\right] &= \\ \sum_{\alpha_1,\alpha_2} \hat{A}_{\alpha_1}\hat{A}_{\alpha_2} E_{f,g_1^1,\mu^1}\left[\chi_{\alpha_1}(g_1^1)\chi_{\alpha_2}(fg_1^1\mu^1)\right] &= \\ \sum_{\alpha_1,\alpha_2} \hat{A}_{\alpha_1}\hat{A}_{\alpha_2} E_{g_1^1}\left[\chi_{\alpha_1\Delta\alpha_2}(g_1^1)\right] E_f\left[\chi_{\pi_2(\alpha_2)}(f)\right] E_\mu\left[\chi_{\alpha_2}(\mu^1)\right]. & \end{aligned} \tag{48}$$

By Lemma 2.29, if $\alpha_1 \neq \alpha_2$ the first expected value is 0 and unless $\pi_2(\alpha_2)=0$ so is the second. The third expected value is easily seen to be equal to $(1-2\epsilon)^{|\alpha|}$ and thus we get the total result

$$\sum_{\alpha|\pi_2(\alpha)=0} \hat{A}_\alpha^2(1-2\epsilon)^{|\alpha|}, \tag{49}$$

which is clearly positive. The proof of the other part of the lemma is identical except for notation. ∎

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

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

We replace each factor by its Fourier expansion and use linearity of expectation, Lemma 2.27, Lemma 2.28, and Lemma 2.30 to arrive at

$$\sum \hat{A}_{\alpha_1}\hat{A}_{\alpha_2}\hat{B}_{\beta_1}\hat{B}_{\beta_2}E\left[\chi_{\alpha_1}(g_1^1)\chi_{\alpha_2}(g_2^1)\chi_{\beta_1}(g_1^2)\chi_{\beta_2}(g_2^2)\right] =$$
$$\sum \hat{A}_{\alpha_1}\hat{A}_{\alpha_2}\hat{B}_{\beta_1}\hat{B}_{\beta_2}E\left[\chi_{\alpha_1\Delta\alpha_2}(g_1^1)\chi_{\pi_2(\alpha_2)\Delta\pi_2(\beta_2)}(f)\chi_{\alpha_2}(\mu^1)\chi_{\beta_1\Delta\beta_2}(g_1^2)\chi_{\beta_2}(-\mu^2)\right], \quad (50)$$

where the sums are over α_1, α_2, β_1, and β_2 and the expectations are taken over f, g_1^1, μ^1, g_1^2, and μ^2. Using that f, g_1^1 and g_2^1 are independent we see, by Lemma 2.29, that unless $\alpha_1 = \alpha_2$, $\beta_1 = \beta_2$ and $\pi_2(\alpha_2) = \pi_2(\beta_2)$ the expected value is 0. Using this, and the fact that $E_\mu[\chi_\alpha(\mu)] = (1-2\epsilon)^{|\alpha|}$ we see that (50) equals

$$\sum_{\alpha,\beta|\pi_2(\beta)=\pi_2(\alpha)} \hat{A}_\alpha^2\hat{B}_\beta^2(-1)^{|\beta|}(1-2\epsilon)^{|\alpha|+|\beta|}. \quad (51)$$

Note that $\pi_2(\beta) = \pi_2(\alpha)$ ensures that $(-1)^{|\beta|} = (-1)^{|\alpha|}$ and thus the result is, as expected, symmetric in A and B. Since the terms corresponding to $\pi(\beta) = \emptyset$ are positive we have established that, based on the hypothesis of the lemma,

$$E_{U,W^1,h^1,W^2,h^2}\left[\left|\sum_{\alpha,\beta:\pi_2(\beta)=\pi_2(\alpha),\pi(\beta)\neq\emptyset} \hat{A}_\alpha^2\hat{B}_\beta^2(1-2\epsilon)^{|\alpha|+|\beta|}\right|\right] \geq \delta. \quad (52)$$

The strategies of P_1 and P_2 in the two prover game are now as follows.

P_1 on input W and h looks at $A_{W,h}$ and chooses a α with probability $\hat{A}_\alpha^2$ and returns a random $x \in \alpha$. The strategy for P_2 is, given U, to choose a random set of clauses giving W and h. This gives a random table B and P_2 chooses a random β with probability $\hat{B}_\beta^2$ and then returns a random $x \in \pi_2(\beta)$. If either of the two sets is empty the corresponding prover gives up. Note that, by Lemma 2.34 each y satisfies the selected clauses and thus we only have to estimate the probability that $y|_U = x$. The probability of this is at least

$$E_{U,W^1,h^1,W^2,h^2}\Big[\sum_{\alpha,\beta|\pi_2(\alpha)=\pi_2(\beta),\pi_2(\beta)\neq\emptyset} \hat{A}_\alpha^2\hat{B}_\beta^2|\alpha|^{-1}\Big].$$

By Lemma 5.3, with $s = 1$, $|\alpha|^{-1} \geq 2\epsilon(1-2\epsilon)^{|\alpha|}$ and thus comparing the last sum to (52) we get that the probability of accepting is at least $2\delta\epsilon$. The proof of Lemma 7.3 is complete. ∎

We now turn to the case of satisfiable instances for Set splitting and we want to establish.

Theorem 7.6 *For any $\epsilon > 0$, it is NP-hard to distinguish instances for E4-Set splitting where all sets can be split from instances where the best partition splits only a fraction $7/8 + \epsilon$ of the sets. Said equivalently, E4-Set splitting is non-approximable beyond the random assignment threshold on satisfiable instances.*

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).
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 variables x_{k_i}, uniformly at random and set $U = \{x_{k_1}, x_{k_2}, \ldots, x_{k_u}\}$. 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 appearing in these clauses, and $h^1 = \wedge_{i=1}^u C_{j_i^1}$. By a similar and independent procedure produce W^2 and h^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 iff $A_{W^1,h^1}(g_1^1)$, $A_{W^1,h^1}(g_2^1)$, $A_{W^2,h^2}(g_1^2)$, and $A_{W^2,h^2}(g_2^2)$ are not all equal.

Completeness is straightforward.

Lemma 7.7 *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 $A_{W^2,h^2}(g_1^2) \neq A_{W^2,h^2}(g_2^2)$ and otherwise $A_{W^1,h^1}(g_1^1) \neq A_{W^1,h^1}(g_2^1)$. ∎

Next we need to analyze the soundness and hence estimate (47).

Lemma 7.8 *For $1 \leq i, j \leq 2$ we have*

$$E_{W^1,W^2,h^1,h^2,g_1^1,g_2^1,g_1^2,g_2^2}[A(g_i^1)B(g_j^2)] \geq 0.$$

This is true for any fixed choices of U and f.

The proof is the same as that of Lemma 7.4 which just depended on the fact that $A(g_i^1)$ and $B(g_j^2)$ are identically distributed and independent. We omit it.

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

Lemma 7.9 *When g_1^1 and g_2^1 are chosen as in test $PSS_\epsilon(u)$ we have*

$$E_{U,f,g_1^1,g_2^1}[A(g_1^1)A(g_2^1)] \geq -3\delta - \sum_{\alpha \,\mid\, \delta\epsilon^{-1}\leq|\alpha|\leq(2\delta^{-2})^{1/c}\epsilon^{-1}} \hat{A}_\alpha^2,$$

where c is the constant of Lemma 6.9. The similar statement is true for $B(g_1^2)B(g_2^2)$.

Proof: We again prove the statement about the $A(g_1^1)A(g_2^1)$, the other part having an almost identical proof. As observed above, $(U, W^1, f, g_1^1, g_2^1)$ have the same distribution as the corresponding objects in test $3\mathrm{S}^\epsilon(u)$ and the only reason we cannot simply appeal to Lemma 6.7 is that we are not assuming that A is folded over true. The only place in the proof of Lemma 6.7 this fact is used is in the proof of Lemma 6.8 where we conclude that s_x is odd for some x. However, we need only observe that the terms in the expansion (36) 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

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

The expansion in the first half of (50) is still valid and terms where $\alpha_1 \neq \alpha_2$ or $\beta_1 \neq \beta_2$ evaluate to 0 also in this case. Thus we need to study

$$E_{f,g_1^1,g_2^1,g_1^2,g_2^2}\left[\chi_\alpha(g_1^1 g_2^1)\chi_\beta(g_1^2 g_2^2)\right]. \tag{53}$$

The pairs (g_1^1, g_2^1) and (g_1^2, g_2^2) are dependent through f. For each x let s_x be the number of $y \in \alpha$ with $y|_U = x$ and t_x the number of $z \in \beta$ with $z|_U = x$. A straightforward calculations shows that (53) equals

$$\prod_x \left(\frac{1}{2}((-1)^{s_x}(1-2\epsilon)^{t_x} + (1-2\epsilon)^{s_x}(-1)^{t_x})\right). \tag{54}$$

Thus we want to estimate

$$E_{U,W^1,h^1,W^2,h^2}\left[\sum_{\alpha,\beta}\hat{A}_\alpha^2\hat{B}_\beta^2\prod_x\left(\frac{1}{2}((-1)^{s_x}(1-2\epsilon)^{t_x} + (1-2\epsilon)^{s_x}(-1)^{t_x})\right)\right]. \tag{55}$$

To estimate this we divide the sum into three pieces, depending on the sizes of α and β.

Lemma 7.10 *If we restrict summation to terms where either α or β is of size at least $(2\delta^{-2})^{1/c}\epsilon^{-1}$, where $c > 0$ is the constant from Lemma 6.9, then*

$$\left| E_{U,W^1,h^1,W^2,h^2}\left[\sum_{\alpha,\beta}\hat{A}_\alpha^2\hat{B}_\beta^2\prod_x(\frac{1}{2}((-1)^{s_x}(1-2\epsilon)^{t_x}+(1-2\epsilon)^{s_x}(-1)^{t_x}))\right]\right| \leq 4\delta. \quad (56)$$

Proof: Let us analyze the terms with $|\beta| \geq (2\delta^{-2})^{1/c}\epsilon^{-1}$. Note that, by (39),

$$|(\frac{1}{2}((-1)^{s_x}(1-2\epsilon)^{t_x}+(1-2\epsilon)^{s_x}(-1)^{t_x}))| \leq (\frac{1}{2}((1-2\epsilon)^{t_x}+1) \leq e^{-\min(1,\epsilon t_x)/2}$$

and thus

$$|\prod_x(\frac{1}{2}((-1)^{s_x}(1-2\epsilon)^{t_x}+(1-2\epsilon)^{s_x}(-1)^{t_x}))| \leq e^{-S_\epsilon^U(\beta)/2}.$$

By Corollary 6.10, we know that the probability (over the choice of U) that $S_\epsilon^U(\beta)$ is smaller than $2\delta^{-1}$ is bounded by δ. This implies that the total contribution of all terms including β is

$$E_U\left[\hat{B}_\beta^2 e^{-S_\epsilon^U(\beta)}\right] \leq (\delta + e^{-\delta^{-1}})\hat{B}_\beta^2 \leq 2\delta\hat{B}_\beta^2$$

and the lemma follows by the linearity of expectation, that $\sum_{\alpha,\beta}\hat{A}_\alpha^2\hat{B}_\beta^2 = 1$, and a similar estimate when α is large. ∎

Next we have

Lemma 7.11 *If we restrict summation to terms where both α and β is of size at most $\delta\epsilon^{-1}$ then*

$$\sum_{\alpha,\beta}\hat{A}_\alpha^2\hat{B}_\beta^2\prod_x(\frac{1}{2}((-1)^{s_x}(1-2\epsilon)^{t_x}+(1-2\epsilon)^{s_x}(-1)^{t_x})) \geq -(\delta + \sum_{\alpha,\beta|\pi(\alpha)\cap\pi(\beta)\neq\emptyset}\hat{A}_\alpha^2\hat{B}_\beta^2), \quad (57)$$

where the sum on the right hand side is also over α and β of size at most $\delta\epsilon^{-1}$.

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 side 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 $-\delta$. Thus the terms we are interested in gets multiplied by a number of absolute value at most δ. Since $\sum_{\alpha,\beta}\hat{A}_\alpha^2\hat{B}_\beta^2 = 1$ the lemma follows. ∎

Now consider the following strategies for P_1 and P_2. P_2 chooses a random α with probability $\hat{A}_\alpha^2$ and answers with a random $y \in \alpha$. P_1 chooses a random W and h and then a random β with probability $\hat{B}_\beta^2$ and then responds with a random $x \in \pi(\beta)$. If either of the two sets is empty the corresponding prover gives up. Note that, by Lemma 2.34 each y satisfies the selected clauses and thus we only have to estimate the probability that $y|_U = x$. This happens with probability at least

$$\delta^{-2}\epsilon^2 E[\sum_{\alpha,\beta|\pi(\alpha)\cap\pi(\beta)\neq\emptyset} \hat{A}_\alpha^2 \hat{B}_\beta^2]$$

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

Lemma 7.12 *Let Acc be the accept probability of V in the u-parallel two-prover interactive proof with optimal P_1 and P_2 then*

$$E_{U,W^1,W^2,h^1,h^2,f,g_1^1,g_2^1,g_1^2,g_2^2}\left[A(g_1^1)A(g_2^1)B(g_1^2)B(g_2^2)\right] \geq$$
$$-\left(5\delta + \delta^2\epsilon^{-2}Acc + 2E_{W^1,h^1}[\sum_{\alpha|\delta\epsilon^{-1}\leq|\alpha|\leq(2\delta^{-2})^{1/c}\epsilon^{-1}} \hat{A}_\alpha^2]\right). \qquad (58)$$

Proof: We have seen that the left hand side equals (55). The terms when the size of both sets are bounded $\delta\epsilon^{-1}$ is bounded, by Lemma 7.11 and the prover strategy given after the lemma, from below by $-(\delta + \delta^2\epsilon^{-2}Acc)$. The case when either set is of size at least $(2\delta^{-2})^{1/c}\epsilon^{-1}$ is bounded, by Lemma 7.10, in absolute value by 4δ. Finally, terms with $\delta\epsilon^{-1} \leq |\alpha| \leq (2\delta^{-2})^{1/c}\epsilon^{-1}$, are bounded in absolute value, using $\sum_\beta \hat{B}_\beta^2 = 1$, by the sum on the right hand side of (58), and the same bound applies to terms with $\delta\epsilon^{-1} \leq |\beta| \leq (2\delta^{-2})^{1/c}\epsilon^{-1}$. ∎

We are now ready to give the test to prove Theorem 7.6.

Test FSS$^\delta(u)$

1. Set $t = \lceil\delta^{-1}\rceil$, $\epsilon_1 = \delta$ and $\epsilon_i = \delta^{1+2/c}2^{-1/c}\epsilon_{i-1}$ 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 7.13 *The completeness of test FSS$^\delta(u)$ is 1.*

For the soundness we have the crucial lemma below and this proves Theorem 7.6 by the standard argument. We omit the details. ∎

Lemma 7.14 *If the test FSS$^\delta(u)$ accepts with probability $(7+112\delta)/8$ 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 $2^{-O(\delta^{-1}\log\delta^{-1})}$.*

Proof: We have by Lemma 7.9 that when g_1 and g_2 are taken as in test FSS$^\delta$ then

$$E[A_{W,h}(g_1^1)A_{W,h}(g_2^1)] \geq -\frac{1}{t}\sum_{i=1}^{t}(3\delta + \sum_{\alpha \;|\; \delta\epsilon_i^{-1}\leq|\alpha|\leq(2\delta^{-2})^{1/c}\epsilon_i^{-1}} \hat{A}_\alpha^2) \geq$$

$$-(3\delta+\frac{1}{t}) \geq -4\delta$$

since the summation intervals are disjoint. Using Lemma 7.12 and doing a similar calculation we conclude that $E[A(g_1^1)A(g_2^1)B(g_1^2)B(g_2^2)]$ is at least

$$-(5\delta+\delta^2\epsilon_t^{-2}Acc+\frac{2}{t}) \geq -(7\delta+\delta^2\epsilon_t^{-2}Acc).$$

Since the probability of accept is given by the expectation of (47) and the four remaining terms can be ignored due to Lemma 7.8, we conclude that

$$\frac{7+12\delta}{8} \leq \frac{7+4\delta+7\delta+\delta^2\epsilon_t^{-2}Acc}{8},$$

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

7.1 Implied results for other CSPs

We first derive some consequences from the proof of Theorem 7.1.

Theorem 7.15 *Let P be a predicate on $\{-1,1\}^4$ such that*

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

and such that $P(x,y,z,w) = -1$ for any x,y,z,w satisfying $xyzw = -1$. Then the monotone CSP defined by P is non-approximable beyond the random assignment threshold.

Proof: By Theorem 7.1 we can assume that P is not the set-splitting predicate, and we can, since reordering the inputs does not disturb the theorem, assume that $(1,1,-1,-1)$ belongs to $P^{-1}(1)$. If P rejects at least 6 inputs it is easy to check that we can reorder the inputs so that also $(-1,-1,1,1)$ belongs to $P^{-1}(1)$.

Now consider Test SS$^\epsilon(u)$ with the change that the acceptance criteria is given by P. Then since

$$A_{W^1,h^1}(g_1^1)A_{W^1,h^1}(g_2^1)A_{W^2,h^2}(g_1^2)A_{W^2,h^2}(g_2^2) = -1$$

unless μ^1 or μ^2 takes the value -1 on the satisfying assignment we see that the completeness of the test is at least $1-2\epsilon$. We need to analyze the soundness.

As usual we write the acceptance criteria as a multilinear function. We need some properties of the multilinear expansion of P summarized in the lemma below. Let $Q = \frac{1-P}{2}$ which is 1 if P accepts and 0 if P rejects.

Lemma 7.16 *The multilinear expansion of $Q(x,y,z,w)$ has the following properties:*

1. *The sum of the coefficients of all degree 1 terms is 0.*
2. *The sum of the coefficients of the terms xz, xw, yz and yw is nonpositive.*
3. *The sum of the coefficients of the terms xy and zw is nonpositive.*
4. *The sum of the coefficients of all degree 3 terms is 0.*
5. *The coefficient of $xyzw$ is negative.*

Proof: We have

$$Q(x,y,z,w) = 1 - \frac{1}{16} \sum_{\alpha \in P^{-1}(1)} (1+\alpha_1 x)(1+\alpha_2 y)(1+\alpha_3 z)(1+\alpha_4 w).$$

Now 1. follows from the fact that

$$\sum_{\alpha \in P^{-1}(1)} \sum_{i=1}^{4} \alpha_i = 0$$

and similarly 4. follows from the fact that the sum of all products of triples also is 0. For 2. we need to study the quadratic form

$$-(\alpha_1+\alpha_2)(\alpha_3+\alpha_4)$$

which takes the value -4 on $(1,1,1,1)$ and $(-1,-1,-1,-1)$, 4 on $(1,1,-1,-1)$ and $(-1,-1,1,1)$ and 0 on the rest of the possible elements of $P^{-1}(1)$. Clearly the sum over the actual elements of $P^{-1}(1)$ is nonpositive and 2. follows.

To address 3. we need to study

$$-(\alpha_1\alpha_2 + \alpha_3\alpha_4)$$

which takes the value -2 on $(1,1,1,1)$, $(-1,-1,-1,-1)$, $(1,1,-1,-1)$, and $(-1,-1,1,1)$ and 2 on $(1,-1,1,-1)$, $(1,-1,-1,1)$, $(-1,1,1,-1)$, and $(-1,1,-1,1)$. By our assumption on P we have at least as many elements in $P^{-1}(1)$ of the former kind as of the latter and hence this sum of coefficient is nonpositive.

Finally 5. follows from the fact that $\alpha_1\alpha_2\alpha_3\alpha_4 = 1$ for any element in $P^{-1}(1)$. ∎

Let us now return to the analysis of the soundness of the test SS^ϵ when P is used as the acceptance predicate. As discussed above we use the multilinear extension of Q and we analyze terms collected into terms of the same types. Since each $A_{W^i,h^i}(g_j^i)$ has the same distribution independent of i and j, by 1. of Lemma 7.16 we see that the expected value of all degree one terms of Q give a total contribution of 0. Similarly, from 4. of the same lemma we see that the same is true for degree 3 terms.

From 2. of Lemma 7.16 and Lemma 7.4 we see that the mixed terms have a total contribution that is non-positive and finally by 3. of Lemma 7.16 and Lemma 7.5 the same can be said about the other terms of degree 2.

To finish the proof of Theorem 7.15 we just have to prove that a negative expected value of large magnitude of the product of all four factors imply a good strategy for P_1 and P_2 in the two prover protocol, but this was already done in the proof of Lemma 7.3. ∎

Note that Theorem 5.6 is a special case of Theorem 7.15 and that the constructed PCPs are in fact different and thus we have an alternative proof for this theorem.

Note also that the condition that both $(-1,-1,-1,-1)$ and $(1,1,1,1)$ belong to $P^{-1}(1)$ is necessary since a monotone CSP which does not reject both $(1,1,1,1)$ and $(-1,-1,-1,-1)$ can trivially be satisfied by a constant assignment.

Next we turn to studying inapproximability for satisfiable instances. Let us first discuss monotone CSPs.

Theorem 7.17 *Let P be a predicate on $\{-1,1\}^4$ such that*

$$\begin{aligned}&\{(1,1,1,1),(-1,-1,-1,-1)\} \subseteq P^{-1}(1) \subseteq \\ &\subseteq \{(1,1,1,1),(1,1,-1,-1),(-1,-1,1,1),(-1,-1,-1,-1)\}.\end{aligned}$$

Then the monotone CSP defined by P is non-approximable beyond the random assignment threshold on satisfiable instances.

Proof: The test we apply is $\mathrm{FSS}^\delta(u)$ with the acceptance criteria that P should hold for the quadruple $(A_{W^1,h^1}(g_1^1), A_{W^1,h^1}(g_2^1), A_{W^2,h^2}(g_1^2), A_{W^2,h^2}(g_2^2))$. It is not difficult to see that the verifier always accepts a correct proof.

For the soundness note that any P we study is covered by Lemma 7.16. The analysis of each set of terms is done as in the proof of Theorem 7.15. ∎

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

Theorem 7.18 *Let P be a predicate on $\{-1,1\}^4$ such that*

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

Then the CSP defined by P is non-approximable beyond the random assignment threshold on satisfiable instances.

Proof: Apply test $\mathrm{FSS}^{\delta}(u)$ except that tables are folded over true and that the final acceptance criteria is given by P. It is not difficult to check that we have perfect completeness.

For the soundness we again study the multilinear expression evaluating P. Folding over true lets us conclude that all terms except $A(g_1^1)A(g_2^1)$, $B(g_1^2)B(g_2^2)$ and $A(g_1^1)A(g_2^1)B(g_1^2)B(g_2^2)$ 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 7.6. ∎

8 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 8.1 *For any $\delta > 0$ it is NP-hard to approximate vertex cover within $7/6 - \delta$.*

Proof: This follows from Proposition 11.6 of [9] 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 $\mathrm{L}_2^{\epsilon}(u)$. We create a graph (as first done in [16]) 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 two nodes. 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 satisfiable the size of any 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 set we have vertex covers of sizes $2^r(3 + \epsilon)$ and $2^r(\frac{7}{2} - \epsilon)$ in the two cases respectively. This implies 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 [35], the optimal result for Max-E3-Lin-2 also give improved inapproximability results for a number of other problems.

Theorem 8.2 *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 $abc = 1$ and the 9-gadget for $abc = -1$. If there are more equations of the second type we complement all the variables. The result follows from a minor extension of Lemma 5.13. ∎

Theorem 8.3 *For any $\epsilon > 0$ it is NP-hard to approximate Max-di-Cut within $13/12 - \epsilon$.*

Proof: There is a 6.5-gadget [35] and we can apply Lemma 5.13. ∎

9 Getting nonconstant ϵ

There is nothing that prevents us from using ϵ and δ that are decreasing as functions of n in our proofs. The acceptance probability of the constructed strategy in the two-prover protocol would then also decrease with n. This, in its turn, implies that to get a contradiction we would need a value of u that is increasing with n and then the PCP would no longer be of polynomial size. If we are willing to assume a stronger hypothesis than NP$\neq$ P something can still be achieved.

Theorem 9.1 *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: Let c' be a constant to be determined. We apply the proof of Theorem 5.4 with $\epsilon = \delta = (\log n)^{-c'}$ and $u = dc' \log\log n$ for some absolute constant d chosen such that $c_c^u < 2\delta^3$, where c_c is the constant from Lemma 3.2. We get that unless φ is satisfiable, the maximal acceptance probability in the PCP is $(1+\delta)/2$ while if φ is satisfiable this acceptance probability of a correct proof is $(1-\epsilon)$.

Translating this to a linear system of equations we get a system in

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

variables with at most

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

equations such that determining the number of simultaneously satisfiable equations within a factor

$$\frac{2(1-\epsilon)}{1+\delta} = 2 - O((\log n)^{c'})$$

is NP-hard. Note that size of the system is, for a suitably small constant c', bounded by $2^{O(\log n \log\log n)}$.

Assume that there is an approximation algorithm running in polynomial time and having performance ratio $2 - (\log n)^{c'/2}$. Note that both performance ratio and running time are with respect to the size of the linear system constructed and not the original n used above. If N is the size of the system of linear equations described above, $\log N \in o(\log n)^2$ the hence assumed approximation algorithm would be able to tell whether the original formula is satisfiable. The running time would be

$$N^{O(1)} = 2^{O(\log n \log\log n)}.$$

The theorem follows. ∎

The proof of Theorem 6.2 has as good constants at that of Theorem 5.4 and hence we have

Theorem 9.2 *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 $(15+\epsilon)/16$ of the clauses can be satisfied in polynomial time.*

We omit the proof since the modifications needed over previous proofs are the same as those described in the proof of Theorem 9.1.

The situation for Theorem 6.5 is different in that the obtained acceptance probability in the two-prover game is much smaller as a function of δ.

Theorem 9.3 *Assume NP $\not\subseteq$ DTIME($2^{O(\log n \log\log n)}$). There is a constant $c > 0$ such for*

$$\epsilon = \frac{c \log\log\log n}{\log\log n},$$

satisfiable E3-CNF formulas cannot be distinguished from E3-CNF formulas where only a fraction $7/8 + \epsilon$ of the clauses can be satisfied in polynomial time.

Proof: Choose $u = c\log\log n$ and

$$\delta = \frac{c' \log\log\log n}{\log\log n},$$

in $\mathrm{FSS}^\delta(u)$ for constants c and c' to be determined. By Lemma 6.13 the success probability of P_1 and P_2 by the defined strategy is $2^{-O(\delta^{-1}\log\delta^{-1})}$. The soundness of the two-prover protocol is c_c^u and hence for any c there is a large enough c', that makes this smaller than the success-rate of the obtained strategy. Since the size of the obtained 3SAT formula is, for small enough c, $2^{O(\log n \log\log n)}$ a similar argument to that given in the proof of Theorem 9.1 establishes the theorem. ∎

Clearly we can study the extension to non-constant ϵ for all the problems we have encountered. Since the proofs are very similar this is a rather tedious exercise and we only state the result.

Theorem 5.5, Theorem 5.6, Theorem 5.9, Theorem 5.14, Theorem 5.15, Theorem 5.16, Theorem 6.1, Theorem 6.2, Theorem 6.14, Theorem 6.16, Theorem 6.17, Theorem 7.1, Theorem 8.1, Theorem 8.2, and Theorem 8.3 extend along the lines of Theorem 9.1 with $\epsilon = (\log n)^{-c'}$ and the assumption that NP $\not\subseteq$ DTIME($2^{O(\log n \log\log n)}$).

Theorem 6.15, Theorem 6.18, Theorem 7.6, Theorem 7.17, and Theorem 7.18 extend along the lines of Theorem 9.3 with $\epsilon = (c' \log\log\log n)(\log\log n)^{-1}$ and the same assumption.

10 Concluding remarks

The technique of using Fourier transforms to analyze PCPs based on the long code seems very strong (see also [23]). It does not, however, seem universal even limited to CSPs. In particular, an open question that remains is to the decide whether the predicate 'not two' is non-approximable beyond the random assignment threshold on satisfiable instances. This question is a special case of the more general program of trying to understand exactly which CSPs are non-approximable beyond the random assignment threshold. For predicates on three variables the situation is completely resolved by the paper of Zwick [36]. The result is that all predicates implied by a linear constraint are non-approximable beyond the random assignment threshold (as also proved in Theorem 5.15).

For predicates on four variables the situation is less clear and apart from the information in this paper we refer to the paper by Guruswami et al. [21]. We have, at this stage, not made an attempt to systematize the information, but this would clearly be a worthwhile effort.

It seems like predicates on two variables are not non-approximable beyond the random assignment threshold. In the Boolean case this follows from the approximation results obtained by semidefinite programming [20]. Over other ranges, less is known, but in the case of linear constrains, nontrivial approximation is obtained by Andersson et al. [2].

Acknowledgments: I am most grateful for discussions on these subjects with Oded Goldreich and Madhu Sudan. Oded and Madhu have also given numerous comments on earlier versions of this manuscript and this have helped improving the quality of this writeup. I am also grateful to two anonymous referees for comments that have had a very positive influence on the current version of the paper. Muli Safra and Sajeev Arora made several helpful comments which simplified some of the proofs. Gunnar Andersson, Lars Engebretsen and Venkatesan Guruswami gave many comments on versions of this manuscript. I am also grateful to Greg Sorkin for constructing the gadget for linear equations with two variables in each equation.

References

[1] E. Amaldi and V. Kann The complexity and approximability of finding feasible subsystems of linear relations. Theoretical Computer Science, Vol 147, 1995, pp. 181–210.

[2] G. Andersson, L. Engebretsen, and J. Håstad A New Way to Use Semidefinite Programming with Applications to Linear Equations mod p. Proceedings 10th Annual ACM-SIAM Symposium on Discrete Algorithms, 1999, pp 41–50.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof verification and intractability of approximation problems. Journal of the ACM, Vol 45, No. 3, May 1998, pp. 501–555.

[4] S. ARORA AND S. SAFRA. Probabilistic checking of proofs: a new characterization of NP. Journal of the ACM, Vol 45, No 1, 1998, pp 70-122.

[5] L. BABAI, L. FORTNOW, AND C. LUND. Non-deterministic exponential time has two-prover interactive protocols. Computational Complexity, Vol 1, 1991, pp 3–40.

[6] L. BABAI, L. FORTNOW, L. LEVIN, AND M. SZEGEDY. Checking computations in polylogarithmic time. Proceedings of the 23rd Annual Symposium on the Theory of Computation, ACM, New York, 1991, pp 21–31.

[7] R. BAR-YEHUDA AND S. EVEN. A linear time approximation algorithm for the weighted vertex cover algorithm. Journal of Algorithms, 1981, Vol 2, pp 198–210.

[8] 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.

[9] 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.

[10] 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).

[11] M. BELLARE AND M. SUDAN. Improved non-approximability results. Proceedings of 26th Annual ACM Symposium on Theory of Computation, Montreal, 1994, pp 184–193.

[12] M. BEN-OR, S. GOLDWASSER, J. KILIAN, AND A. WIGDERSON. Multiprover interactive proofs. How to remove intractability. Proceedings of the 20th Annual ACM Symposium on Theory of Computation, Chicago, 1988, pp 113–131.

[13] M. BLUM, M. LUBY AND R. RUBINFELD. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences* Vol. 47, 549–595, 1993.

[14] U. FEIGE. A threshold of $\ln n$ for approximating set cover. Journal of the ACM, vol 45, 1998, pp 634–652.

[15] 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.

[16] 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.

[17] U.Feige and J. Kilian. Zero-knowledge and the chromatic number. Journal of Computer and System Sciences, Vol 57:2, 1998, pp 187-200.

[18] L. Fortnow, J. Rompel, and M. Sipser. On the power of Multi-Prover Interactive Protocols. Theoretical Computer Science, Vol 134, No. 2, 1994, pp. 545–557.

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

[20] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM, Vol. 42, No 6, 1995, pp. 1115–1145.

[21] V. Guruswami, D. Lewin, M. Sudan, and L. Trevisan. A tight characterization of NP with 3 query PCPs. Proceedings of 39th Annual IEEE Symposium on Foundations of Computer Science, Palo Alto, 1998, pp. 8–17.

[22] D. Hochbaum. Efficient algorithms for the stable set, vertex cover and set packing problems. Discrete Applied Math, 6, 1983, pp. 243–254.

[23] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. Acta Mathematica, Vol 182, 1999 pp. 105–142.

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

[25] D.S. Johnson. Approximation algorithms for combinatorial problems. J. Computer and System Sciences, 1974, Vol 9, pp 256–278.

[26] T. Judson. Abstract Algebra, Theory and Applications, 1994, PWS Publishing Company, Boston.

[27] V. Kann, J. Lagergren and A. Panconesi Approximability of maximum splitting of k-sets and some other APX-complete problems. Information processing letters, Vol 58, 1996, pp. 105–110.

[28] 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.

[29] 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.

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

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

[32] R. Raz. A parallel repetition theorem. SIAM J. on Computing, Vol 27, No. 3, 1998, pp. 763–803.

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

[34] G. Sorkin. Personal communication.

[35] L. Trevisan, G. Sorkin, M. Sudan, and D. Williamson. Gadgets, approximation and linear programming. SIAM Journal on Computing, Vol 29, 2074-2097, 2000.

[36] U. Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. *Proceedings 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1998, pp 201–210.

A Proof of Lemma 6.9

Assume without loss of generality that W is generated by clauses $(C_i)_{i=1}^u$ where C_i contains the variables x_{3i-2}, x_{3i-1}, and x_{3i}. We think of selecting U as a process where at stage j we decide which variable from C_j to include into U. We denote the j'th chosen variable by k_j and thus $k_j \in \{3j-2, 3j-1, 3j\}$.

The elements of β are during this process naturally divided into groups defined by their values on the variables put into U at a given point. More formally, we let G_x^j, where $x \in \{-1,1\}^j$, be the set of elements $y \in \beta$ such that $y_{k_i} = x_i$ for $i = 1, 2, \ldots j$. Two elements in the same group G_x^j might eventually turn out to have different projections onto U due to the fact that they take different values on later coordinates being put into U.

For each G_x^j we have a weight $W(G_x^j)$ defined as follows. As long as G_x^j contains at least ϵ^{-1} elements with different projections onto the coordinates that has not yet been decided whether they will be put into U we set $W(G_x^j) = (\epsilon d_x^j)^c$ where d_x^j is the number of such different projections. If the number of different projections is smaller than ϵ^{-1} we define $W(G_x^j)$ to be the minimum of 1 and ϵs_x^j where s_x^j is the total number of elements (with or without different projections) in G_x^j. At times in our calculations when we are uncertain if a certain group has ϵ^{-1} distinct projections we use the latter definition of $W(G)$. This results in a lower weight since the first definition gives a value that is at least 1 and the second a value is at most 1.

We study the quantity

$$F^j = \left(\sum_{x \in \{-1,1\}^j} W(G_x^j) \right)^{-1}$$

and we prove that

$$E[F^{j+1}|F^j] \leq F^j \tag{59}$$

for each j. Since $F^0 = (\epsilon|\beta|)^{-c}$ and $F^u = S_\epsilon^U(\beta)^{-1}$ the lemma follows from (59).

Let $X^j = (F^j)^{-1}$. Our method to prove (59) is to introduce a random variable Y^j which is coupled with X^j, such that $Y^j \leq X^j$ is always true and furthermore we establish that $E[(Y^{j+1})^{-1}|F^j] \leq F^j$.

For a string x of length j let x^+ and x^- be the two strings of length $j+1$ obtained from concatenating x with 1 and -1 respectively. We want to analyze $W(G_{x^+}^{j+1})$ and $W(G_{x^-}^{j+1})$ in terms of $W(G_x^j)$.

The elements of G_x^j naturally fall into 8 classes, $D_{x,\alpha}^j$, where $\alpha \in \{-1,1\}^3$ gives the values of a particular element on the variables $(x_{3j+1}, x_{3j+2}, x_{3j+3})$. Once k_{j+1} is chosen it determines which classes form $G_{x^+}^{j+1}$ and which form $G_{x^-}^{j+1}$. As an example if x_{3j+2} is chosen then $G_{x^+}^{j+1}$ is the union of $D_{x,111}^j$,$D_{x,11-1}^j$,$D_{x,-111}^j$, and $D_{x,-11-1}^j$, while $G_{x^-}^{j+1}$ is $G_x^j \setminus G_{x^+}^{j+1}$. Let $d_{x,\alpha}^j$ be the number distinct projections in $D_{x,\alpha}^j$. We have

$$d_x^j = \sum_\alpha d_{x,\alpha}^j.$$

Obviously

$$d_{x^+}^{j+1} \geq \max_\alpha d_{x,\alpha}^j,$$

where the maximum ranges over all α used to form $G_{x^+}^{j+1}$ and a similar inequality holds for $d_{x^-}^{j+1}$.

The interesting case is when G_x^j has at least ϵ^{-1} different projections since otherwise

$$W(G_x^j) \leq W(G_{x^+}^{j+1}) + W(G_{x^-}^{j+1})$$

and such groups add at least as much to X^{j+1} as to X^j. Since, by convexity, $E[\frac{1}{X}] \leq \frac{1}{a}$ implies $E[\frac{1}{X+b}] \leq \frac{1}{a+b}$ for a positive random variable X and positive numbers a and b, we can simply ignore these terms.

Suppose, without loss of generality, that $D_{x,111}^j$ is the largest of the 8 groups. We have two cases depending on whether $d_{x,111}^j \geq \epsilon^{-1}$. Suppose first that $d_{x,111}^j \geq \epsilon^{-1}$. For notational simplicity let d denote ϵd_x^j and let d' be the size of the second largest group multiplied by ϵ.

We analyze what happens to the two largest groups. We say that these two classes are separated iff one becomes part of $G_{x^+}^{j+1}$ and the other becomes part of $G_{x^-}^{j+1}$. Define a random variable Y_x^{j+1} by

- If the two largest classes are not separated, then Y_x^{j+1} equals $\max(1, (d - 7d')^c)$ if $d' \leq d/8$ and $\max(1, (d/8)^c)$ otherwise.
- If the two largest classes are separated, then Y_x^{j+1} equals $\max(1, (d - 7d')^c) + W(d')$ if $d' \leq d/8$, $\max(1, (d/8)^c) + W(d/8)$ otherwise. Here $W(d') = d'^c$ for $d' \geq 1$ and $W(d') = d'$ otherwise.

In the case when $d^j_{x,111} < \epsilon^{-1}$ we define $d'' = \min(d-1, d', 1)$ and define Y_x^{j+1} as follows

- If the two largest classes are not separated, then Y_x^{j+1} equals 1.
- If the two largest classes are separated, then Y_x^{j+1} equals $1 + d''$.

Finally we set $Y^{j+1} = \sum_x Y_x^{j+1}$. We claim that $X^{j+1} \geq Y^{j+1}$ and this follows from

$$Y_x^{j+1} \leq W(G^{j+1}_{x^+}) + W(G^{j+1}_{x^-}). \tag{60}$$

To see (60) when $d^j_{x,111} \geq \epsilon^{-1}$ note that the group to which $D^j_{x,111}$ joins has a number of distinct elements which is at least the maximum of ϵ^{-1}, $\epsilon^{-1}(d-7d')$ and $\epsilon^{-1}d'$. Furthermore, in the case the two largest classes are separated, the number of distinct element (and hence also the number of elements) in the group in which the second largest class ends up is of size at least $\epsilon^{-1}d'$.

When $d^j_{x,111} < \epsilon^{-1}$, then since $s^{j+1}_{x+} + s^{j+1}_{x-} = d\epsilon^{-1}$, if $\min(s^{j+1}_{x+}, s^{j+1}_{x-}) = \delta\epsilon^{-1}$ then $W(G^{j+1}_{x+}) + W(G^{j+1}_{x-}) \geq 1 + \min(\delta, d-1)$. Both cases follow from this fact. In the first case we have $\delta = 0$ and in the second we use $\delta \geq d'$.

We now establish that $E[1/Y^{j+1}|F_j] \leq F^j$. Let w_x be the value of Y_x^{j+1} if the two large groups are not separated and let $w_x + b_x$ be the value if they are separated. We have

$$\sum_x w_x \leq Y^{j+1} \leq \sum_x (w_x + b_x), \tag{61}$$

and since any two classes are separated with probability at least $1/3$ we also have

$$E[Y^{j+1}] \geq \sum_x (w_x + \frac{1}{3} b_x). \tag{62}$$

Since the function $f(z) = 1/z$ is concave and decreasing the random variable Z that satisfies (61) and (62) and which gives the largest value of $E[1/Z]$ is the one which takes the value $\sum_x w_x$ which probability $2/3$ and the value $\sum_x (w_x + b_x)$ with probability $1/3$. We conclude that

$$E[1/Y^{j+1}|F_j] \leq \frac{2}{3}\left(\sum_x w_x\right)^{-1} + \frac{1}{3}\left(\sum_x (w_x + b_x)\right)^{-1}.$$

Since $\sum_x b_x \leq \sum_x w_x$ (in fact $b_x \leq w_x$ for every x) and

$$\frac{2}{3}\frac{1}{w} + \frac{1}{3}\frac{1}{w+b} \leq \frac{1}{w + b/5}$$

for any $0 \leq b \leq w$ we conclude that

$$E[1/Y^{j+1}|F_j] \leq \left(\sum_x (w_x + b_x/5)\right)^{-1}. \tag{63}$$

Finally, we claim that $w_x + b_x/5 \geq W(G_x^j)$, for any x. To establish this let us go over all possible cases.

If $d_{111,x}^j \geq \epsilon^{-1}$ and $d' < 1$ then we need to establish

$$\max(1, (d - 7d')^c) + d'/5 \geq d^c.$$

Since the derivative of w^c is bounded by c when $w > 1$ this is true provided $c \leq \frac{1}{35}$.

If $d_{111,x}^j \geq \epsilon^{-1}$ and $d' \geq 1$ then we can assume that $d' \leq d/8$ since $d' > d/8$ is equivalent to $d' = d/8$. We need to establish

$$\max(1, (d - 7d')^c) + d'^c/5 \geq d^c.$$

To see this note that if $f(x) = (d - 7x)^c + x^c/5 - d^c$ then, assuming $0 < c < 1$, $f''(t) \leq 0$ in the entire interval and hence to check that f is nonnegative in the interval we only have to check this property at the end-points. Clearly $f(0) = 0$ and

$$f(d/8) = \frac{6}{5}(d/8)^c - d^c = d^c(\frac{6}{5}2^{-3c} - 1).$$

This last number is nonnegative for $0 < c \leq \frac{1}{3}\log_2(6/5)$ and hence for $c \leq \frac{1}{35}$.

Finally when $d_{111,x}^j < \epsilon^{-1}$ then we need to establish that

$$1 + d''/5 \geq d^c.$$

Now $d' \geq (d - 1)/7$ and since the derivative of d^c is bounded by c when $d \geq 1$ the inequality is true as long as $c \leq \frac{1}{35}$.

We conclude that (63) is bounded from above by

$$\left(\sum_x W(G_x^j)\right)^{-1} = F_j,$$

and the proof of the lemma is complete.