Optimal Inapproximability with Universal Factor Graphs

PER AUSTRIN, KTH Royal Institute of Technology, Sweden
JONAH BROWN-COHEN, Chalmers University of Technology, Sweden
JOHAN HÅSTAD, KTH Royal Institute of Technology, Sweden

The factor graph of an instance of a constraint satisfaction problem (CSP) is the bipartite graph indicating which variables appear in each constraint. An instance of the CSP is given by the factor graph together with a list of which predicate is applied for each constraint. We establish that many Max-CSPs remain as hard to approximate as in the general case even when the factor graph is fixed (depending only on the size of the instance) and known in advance.

Examples of results obtained for this restricted setting are:

(1) Optimal inapproximability for Max-3-Lin and Max-3-Sat (Håstad, J. ACM 2001).
(2) Approximation resistance for predicates supporting pairwise independent subgroups (Chan, J. ACM 2016).
(3) Hardness of the "$(2+\varepsilon)$-Sat" problem and other Promise CSPs (Austrin et al., SIAM J. Comput. 2017).

The main technical tool used to establish these results is a new way of folding the long code which we call "functional folding".

CCS Concepts: • **Theory of computation → Problems, reductions and completeness**.

Additional Key Words and Phrases: hardness of approximation, factor graphs

ACM Reference Format:
Per Austrin, Jonah Brown-Cohen, and Johan Håstad. 2022. Optimal Inapproximability with Universal Factor Graphs. *ACM Trans. Algor.* 1, 1 (October 2022), 39 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Constraint Satisfaction Problems (CSPs), such as k-Lin and k-Sat, are some of the most well-studied problems in computational complexity. Already when considered as decision problems most CSPs are NP-complete and their maximization versions are hence NP-hard. In order to further investigate the computational complexity of these problems it is interesting to study restricted versions.

The factor graph of an instance of a CSP is the bipartite graph that connects u_i to v_j iff variable x_i appears in the jth constraint. The description of the instance is completed by indicating for each constraint which predicate is applied (in the case of e.g. Max-3-Sat this simply means specifying which variables are negated and which appear in their positive form). It is not difficult to find examples of factor graphs where the resulting Max-CSP instances are always easy; one example would be that the graph has bounded tree-width as such instances can be solved efficiently by dynamic programming. More generally, there has been extensive research on the tractability of CSPs with "left-hand side restrictions", i.e., restrictions on the structure of the factor graph [9, 13].

Authors' addresses: Per Austrin, austrin@kth.se, KTH Royal Institute of Technology, Stockholm, Sweden; Jonah Brown-Cohen, jonahb@chalmers.se, Chalmers University of Technology, Gothenburg, Sweden; Johan Håstad, johanh@kth.se, KTH Royal Institute of Technology, Stockholm, Sweden.

1549-6325/2022/10-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

In this paper we are interested in the other end of the spectrum, namely to see if there is a sequence of factor graphs, one for each length of the input, such that the underlying Max-CSP remains hard if we restrict the instances to use these factor graphs. This general class of questions was first systematically considered by Feige and Jozeph [10] who coined the term *universal factor graphs* for such sequences of factor graphs.

There are several reasons for studying this setting. On a fundamental level it is interesting to understand whether this separation into the factor graph and the predicate types applied can help us better understand the problem. In some situations one might have many instances with the same factor graph and it is interesting to study whether this is beneficial for a solver. This leads to the concept of "CSPs with preprocessing". A particularly interesting case of this is for a linear problem such as Max-k-Lin where a factor graph with n variables and m constraints determines a fixed linear subspace of $\mathbb{F}^m$ of dimension at most n, the negations give a point in $\mathbb{F}^m$, and the problem is equivalent to finding a point in this fixed subspace that is close to the given point. This is a decoding problem for a linear code, but note that, in this case, the distances between the points supplied and the linear code is much larger than the minimal distance of the code. Hence, this problem is not really a traditional decoding problem for error correcting codes.

It is not surprising that some problems remain NP-hard in this setting. For instance, it is not difficult to see that if some basic parameters, such as the alphabet and the number of tapes for the involved Turing machine, are fixed then the standard proof of the Cook-Levin theorem produces an instance that can easily be made to have a universal factor graph.

Using a different approach Feige and Jozeph [10] proved that universal factor graphs exist for the problem of deciding k-Sat. Furthermore, they also showed that the PCP Theorem can be established in the universal factor graph setting, opening up the possibility to proving hardness of approximation results for universal factor graphs. They went on to show that universal factor graphs exist for the problem of approximating Max-k-Sat to within some constant factor, which for $k = 3$ was $77/80 + \varepsilon$ for any $\varepsilon > 0$ [1]. Since general Max-3-Sat is well-known to be NP-hard to approximate within any constant larger than $7/8 + \varepsilon$ [15] this left open the problem whether fixing the factor graph might make Max-3-Sat easier to approximate.

In a follow-up work, Jozeph [18] showed that there are universal factor graphs for every NP-hard Boolean CSP and for every APX-hard Boolean Max-CSP, but with possibly weaker approximation gaps than in the standard setting.

1.1 Our Results

We show that the hardness ratio of Max-3-Sat as well as those of many other problems carry over to the universal factor graph setting. Essentially all our results are stated in the form of *approximation resistance*, i.e., that it is hard to approximate a certain Max-CSP significantly better than picking an assignment at random.

We show that the following problems are all approximation resistant also in the universal factor graph setting.

1. The Max-k-Sat problem on satisfiable instances for all $k \geq 3$.
2. The Max-TSA problem (each constraint is a Tri-Sum-And constraint, i.e., of the form $x_1 + x_2 + x_3 + x_4 \cdot x_5 = b \pmod 2$).
3. The Max-Not-2 problem on satisfiable instances (The Boolean predicate which is one if and only if the input does not contain two bits set to one).
4. Any predicate supporting a pairwise independent subgroup such as linear equations modulo q for any $q \geq 2$, and the very sparse "Hadamard predicate".

[1] From now on we use the convention that ε is used for a positive constant that can be chosen to be arbitrarily small

Approximation resistance for the Hadamard predicate is of particular interest because this gives optimal hardness of approximation for the general Max-k-CSP problem (up to a constant factor independent of k) [7]. This result only applies to problems for which solving satisfiable instances is easy. Therefore we also show that, in the universal factor graph setting, the general Max-K-CSP problem is hard to approximate on satisfiable instances to within a factor of $K^{cK^{1/3}}/2^K$ for some constant c [17].

In fact, we also prove *uselessness* in the sense of [3]. This means that it is NP-hard to find an assignment such that when picking a random constraint, the resulting distribution of values of the k-tuples of literals that are fed into the predicate deviates significantly from uniform, even under the promise that there exists an assignment that satisfies almost all or even all of the constraints. This concept only applies to CSPs in which we have free negations of variables and as such does not apply to the Max-TSA problem, but for the other problems we do obtain uselessness.

Hardness results for Max-TSA and similar problems are somewhat related to the one-way functions and pseudorandom generators proposed by Goldreich [12]. For instance the Tri-Sum-And predicate is one of the current standard candidate predicates for his construction, and our results show that there is a fixed set system such that it is NP-hard to distinguish strings that are in the image of the one-way function from strings that are at almost maximal distance from any output of the function. Of course in a cryptographic situation one is interested in an average case results while we are in a worst case setting. In any case we feel that our results give at least moral support to the conjecture that the functions proposed by Goldreich are indeed good pseudorandom generators.

We also consider hardness of Promise CSPs (PCSPs) as studied in several recent works [2, 5, 6]. In this setting we are given a satisfiable instance of some hard CSP (e.g. a 3-colorable graph) and are asked to find an assignment where each constraint is replaced by a weaker constraint (e.g. finding a c-coloring for some large c). One example of this is the "$(2+\varepsilon)$-Sat" problem, in which we are given a $(2k+1)$-Sat instance where it is promised that there is an assignment that satisfies at least k literals in every clause, and the goal is to find an assignment that satisfies at least 1 literal in every clause. A fairly general hardness result of Brakensiek and Guruswami [5] states that if all the so-called polymorphisms of a PCSP satisfy a property called "C-fixing" then the PCSP is NP-hard. They further showed that this result is sufficient to establish a complete dichotomy for PCSPs where all constraints are symmetric functions. We were not able to obtain this result under universal factor graphs, but we can obtain an earlier result of Austrin et al. [2], which applies if all the polymorphisms are C-juntas and the problem allows negations of variables. This in particular shows that the "$(2+\varepsilon)$-Sat" problem remains NP-hard in the universal factor graph setting.

Finally we also note that many typical gadget reductions carry over without modification to the universal factor graph setting and as a consequence our optimal hardness for Max-3-Lin also implies hardness of, e.g., $(11/12+\varepsilon)$-approximating Max-2-Lin and $(21/22+\varepsilon)$-approximating Max-2-Sat.

Our results generally apply to any domain, but to keep the notation as simple as possible in order to focus on the main new ideas, most parts of the paper present the details only for the Boolean case. We then briefly comment towards the end of the paper on the (minor) modifications needed for general domains.

1.2 Overview of Proof Techniques

The first results on hardness of approximation for universal factor graphs are those of Feige and Jozeph [10]. They begin with a (possibly folklore) reduction showing that the decision version of 3-Sat is NP-hard with universal factor graphs (i.e., NP-hard to solve even when the factor graph is fixed in advance).

Much of the rest of the work in that paper is then dedicated to showing how to make classical reductions from hardness of approximation *factor graph-preserving*. That is, to ensure that applying the reduction to two 3-Sat instances with the same factor graph results in two instances of the target Max-CSP with the same factor graph. Such a reduction then implies that the target Max-CSP is hard to approximate with universal factor graphs. Indeed, the authors show that Dinur's proof of the PCP theorem can be made factor graph-preserving. They also successfully follow the work of Bellare et al. [4] and show how their reductions can be made factor graph-preserving, enabling them to obtain for instance a hardness of approximation of $77/80 + \varepsilon$ for Max-3-Sat. Feige and Jozeph were not, however, able to make the result of Håstad [15] factor graph-preserving. As a consequence, they do not get the optimal hardness $7/8 + \varepsilon$ for Max-3-Sat and they do not get inapproximability for Max-3-Lin with any fixed constant soundness and arbitrarily good completeness.

Therefore, to establish strong hardness results in the universal factor graph framework the key is to combine the universal factor graph property with an, at least somewhat, more modern soundness analysis of the underlying PCPs. It turns out that the main obstacle to achieving this combination arises in the folding of the long code. In the remainder of this section we first explain why the folding used in Håstad's reduction is not factor graph-preserving while that used by Bellare et al. is. We then introduce our new folding of the long code and give some intuition for why it is both factor graph preserving and able to achieve optimal inapproximability.

The classical "parallel repetition + long coding" paradigm begins with some Max-CSP, often Max-k-Sat, that is known to be hard to approximate within some small constant factor. Then, in the intended solution to the target Max-CSP, satisfying assignments to subsets of the variables of the original Max-CSP are encoded with the long code, i.e., by encoding $\mathbf{x} \in \{0,1\}^r$ by the string $A_{\mathbf{x}} \in \{0,1\}^{2^{2^r}}$ consisting of the evaluation of every Boolean function on r bits at the point $\mathbf{x}$. To prove that such a reduction is sound with optimal approximation ratio, one then constructs what is known as a long code testing gadget. This gadget has for each Boolean function f on r bits a variable $A(f)$ intended to take the value $A_{\mathbf{x}}(f)$ of the long code of some good assignment $\mathbf{x}$. On these variables constraints of the target Max-CSP are added, designed so that given an assignment satisfying many of the gadget constraints it is possible to "decode" this assignment to a valid code word $A_{\mathbf{x}}$ where $\mathbf{x}$ is a satisfying assignment to a subset of the variables of the original Max-CSP.

Note that there are two distinct tasks: (a) decoding to a code word, and (b) checking that $\mathbf{x}$ satisfies the constraints of the original Max-CSP. In order to achieve both at the same time, a key step used in classical reductions is *folding* of the long code. Folding identifies coordinates (possibly with negations) of the long code in an attempt to implicitly enforce the second condition (b), i.e. that $\mathbf{x}$ satisfies the original constraints.

The folding used by Håstad (sometimes also called *conditioning*) can be described as follows. If for some constant $b \in \{0,1\}$ it holds that $f(\mathbf{x}) = g(\mathbf{x}) + b$ for all $\mathbf{x}$ satisfying the original constraints, then we identify the corresponding long code entries by $A(f) = A(g) + b$. Notice that this folding depends heavily on precisely which constraints appear on the subset of original variables $\mathbf{x}$. For example, if our reduction started with an instance of Max-k-Sat, changing the negation on one of the original variables in $\mathbf{x}$ will completely change which coordinates are identified under this strong folding. Since the gadget we construct has these coordinates as variables, this will in turn change the factor graph of the gadget output by the reduction. Thus, this folding does not lead to factor graph preserving reductions. However, it was—until now—the only way known to obtain optimal hardness of approximation results for many fundamental problems.

In the earlier and weaker type of folding used by Bellare et al. we apply a similar folding, but we fold over each constraint on $\mathbf{x}$ individually instead of over all of them at once. Specifically, for each individual original constraint $h(\mathbf{x}) = b$ on $\mathbf{x}$, we identify all pairs of functions f and $g = f + h$

by $A(f) = A(g) + b$. Notice that if we change the value of $b \in \{0,1\}$, instead of changing which coordinates are identified, all that changes is the negation applied to the coordinate. Thus, if the constraint functions h are fixed in advance and only the right-hand sides b vary, this folding does produce factor graph preserving gadgets. This fact was the key to the reduction in [10]. We refer to CSPs of this type, where all constraints are of the form $h(\mathbf{x}) = b$ where the function h is fixed in advance and $b \in \{0,1\}$, as *equational CSPs*. The issue with this approach is that this weaker folding does not sufficiently enforce that the decoded long code word $A_{\mathbf{x}}$ corresponds to an $\mathbf{x}$ satisfying the original constraints. For this reason, the long code testing gadget constructed in the reduction contains extra constraints to explicitly enforce this, leading to sub-optimal approximation ratios.

Our solution to the problems mentioned above is a new method of folding the long code which we call *functional folding*. It can be viewed as a strengthening of the folding of Bellare et al. Let $\{h_i(\mathbf{x}) = b_i\}_{i=1}^{t}$ be the set of original constraints on the subset of variables $\mathbf{x}$. Our folding identifies the coordinates f and g which differ by $F(h_1(\mathbf{x}), \ldots, h_t(\mathbf{x}))$ where F is *any* Boolean function on t bits. We then identify the corresponding long code entries by $A(g) = A(f) + F(b_1, \ldots, b_t)$. Notice that just as in the case of the weaker folding above, changing the value of b_i in one of the original constraints will possibly change the negations on the coordinates of the long code testing gadget, but will not change which coordinates are identified. Therefore, this folding is factor graph-preserving when we start with an equational CSP. Furthermore, we show that our new folding does enforce the constraints on $\mathbf{x}$ strongly enough to allow for optimal approximation ratios.

In more detail, the folding of Håstad yields a gadget which, from a proof accepted with high probability, allows us to find a set of long code words, all of which correspond to satisfying assignments to the original constraints. Our new folding does not quite allow for this, but instead guarantees that we find a set of long codes words, *at least one of which* satisfies the original constraints. This is formalized in Lemma 3.4.

Having only this weaker guarantee, it turns out to be possible to reprove many earlier results in the universal factor graph setting after some modifications in the constructions and the proofs. The most important modification is that, even in the easiest cases of proving that Max-3-Lin is NP-hard to approximate within $\frac{1}{2} + \varepsilon$ or that $(2 + \varepsilon)$-Sat is NP-hard, we need to use a *smooth* parallel repetition as the starting point. This allows us to cope with the issue that only some of the set of long code words we find correspond to satisfying assignments.

Another difference, albeit a purely notational one, is the following. Because of the property of functional folding that only some of the coordinates correspond to feasible assignments in the parallel repeated game, we need to keep track for each long code of how the coordinates represent partial assignments to the underlying equational CSP. Due to this, we cannot view the parallel repeated game as an abstract label cover instance and in the process we lose most of the simplified notation afforded by this "modern" view. For example we have to view the purported long codes given to the PCP verifier as Boolean functions of Boolean functions rather than simply Boolean functions of generic bit strings.

The fact that we need a smooth parallel repetition implies that if we look more closely at the parameters of our results we do not get as strong results as in the general case. In the black and white world of polynomial vs non-polynomial time we match the general results, but in more fine-grained measures our results are weaker than the general case. Here we are referring to (i) the question of how small ε can be as a function of n, and (ii) the question of what quantitative lower bounds can be given on the running time assuming the Exponential Time Hypothesis. For these questions, our results for universal factor graphs do not match the state of art for the general case. We elaborate a bit more on this in the concluding remarks towards the end of the paper.

1.3 Organization

An outline of the paper is as follows. In Section 2 we cover some background material. In Section 3 we define our new notion of functional folding and set up the general framework for our hardness reductions. Section 4 is devoted to the reproving the results of Håstad [15] for basic problems such as Max-3-Lin and Max-3-Sat (and also Max-TSA and Max-Not-2 for which the hardness is proved in the same way). In Section 5 we show how to adapt the results of Chan [7], and in Section 6 we do the same with the results of Huang [17]. In Section 7 we give results for Promise CSPs. Then in Section 8 we discuss small extensions such as gadget reductions and larger domains, and finally in Section 9 we round off with some remaining open questions.

2 PRELIMINARIES

For a vector $\mathbf{v} \in \Sigma^I$ indexed by I over some set Σ, and a vector $S = (i_1, \ldots, i_k) \in I^k$ of indices, we write $\mathbf{v}_S$ for the vector $(v_{i_1}, \ldots, v_{i_k})$. For a set β of vectors we let $\beta_S = \{\, \mathbf{v}_S \mid \mathbf{v} \in \beta \,\}$ (viewed as a set and not a multiset).

For any event $E(x)$ we let $\mathbf{1}(E(x))$ denote the function that is one exactly when $E(x)$ is true and zero otherwise. In a similar vein for a set $\beta \subseteq I$ of indices $\mathbf{1}_\beta \in \{0,1\}^I$ is the vector that is one exactly on the set β and zero outside this set. If β is a single index, this is a unit vector.

For an index set I of coordinates we write $\mathcal{F}_I = \{\, f : \{0,1\}^I \to \{0,1\} \,\}$ for the set of all Boolean functions on I, and for an integer n we write $\mathcal{F}_n = \mathcal{F}_{[n]}$.

When arguing about Boolean functions we let "+" denote exclusive-or. We also have addition of real numbers but hopefully the meaning of each + is clear from the context.

2.1 Constraint Satisfaction Problems

We begin by introducing notation for CSPs. As this paper focuses on the case of Boolean inputs we only give formal definitions in this special case. We expect that the reader is able to guess the more general definitions needed for our brief discussion of larger domains in Section 8.2.

Definition 2.1. A k-ary constraint language Γ is a finite set of functions $f : \{0,1\}^k \to \{0,1\}$.

Different constraint languages give rise to different CSPs as formalized in the following definition.

Definition 2.2. An instance of the Max-CSP(Γ) problem is of the form $I = (X, C)$ where X is a set of n variables and C is a set of m constraints. Each constraint $c \in C$ is a pair $c = (f, S)$ where $S \in X^k$ (the *scope* of c) is a tuple of k distinct variables and $f \in \Gamma$ (the *constraint type* of c). An assignment $\mathbf{a} \in \{0,1\}^X$ of values to the variables satisfies the constraint $c = (f, S)$ if $f(\mathbf{a}_S) = 1$.

The objective is to find an assignment to the variables satisfying as many constraints as possible. An instance of Max-CSP(Γ) is α-satisfiable if there is an assignment that satisfies an α fraction of the constraints.

For every Max-CSP and some approximation parameters, there is a naturally associated approximation problem, which we generally phrase as the following promise decision problem.

Definition 2.3. For parameters $0 \le s \le c \le 1$, the problem of (c, s)*-approximating* Max-CSP(Γ) is the promise problem where the goal is to distinguish between instances of Max-CSP(Γ) that are at least c-satisfiable, and those that are at most s-satisfiable.

For example, Max-3-Lin is the problem Max-CSP(Γ) where Γ consists of the two functions $f_0(x,y,z) = x + y + z$ and $f_1(x,y,z) = x + y + z + 1$. In fact Max-3-Lin is a prototypical example of the following important subclass of CSPs which plays a critical role in our reductions.

Definition 2.4. An *equational CSP* is a CSP where the constraint language Γ contains exactly two functions: f and $\neg f$ for some $f : \{0,1\}^k \to \{0,1\}$. Equivalently, an equational CSP is one where each constraint c has the form either $f(\mathbf{x}_S) = 0$ or $f(\mathbf{x}_S) = 1$.

Definition 2.5. For a predicate $f : \{0,1\}^k \to \{0,1\}$, Max-CSP$(f^{\pm})$ is the CSP where each constraint is of the form $f(\mathbf{x}_S + \mathbf{b}) = 1$ for some scope S and vector $\mathbf{b} \in \{0,1\}^k$.

We now formally define the CSPs of interest in this paper.

Definition 2.6. The Tri-Sum-And predicate $f_{\mathrm{TSA}} : \{0,1\}^5 \to \{0,1\}$ is given by

$$f_{\mathrm{TSA}}(x, y, z, u, v) = x + y + z + uv.$$

We write Max-TSA for the (equational) Max-CSP$(\{f_{\mathrm{TSA}}, \neg f_{\mathrm{TSA}}\})$ problem which is a well known NP-hard problem.

Definition 2.7. Max-k-Sat is the Max-CSP$(\vee_k^{\pm})$ problem, where $\vee_k : \{0,1\}^k \to \{0,1\}$ is the OR function on k Boolean variables.

Definition 2.8. Max-k-Lin is the (equational) Max-CSP$(\{+_k, \neg +_k\})$ problem, where $+_k : \{0,1\}^k \to \{0,1\}$ is the parity function on k Boolean variables.

Definition 2.9. For k of the form $2^\ell - 1$ for some integer ℓ, $\mathrm{Had}_k : \{0,1\}^k \to \{0,1\}$ is the predicate where the indices are in one-to-one correspondence with the nonempty subsets of $[\ell]$ and a string $\mathbf{x}$ satisfies Had_k iff $x_\alpha + x_\beta = x_\gamma$ whenever $\alpha \triangle \beta = \gamma$.[2]

It is easy to see that an accepted $\mathbf{x}$ is determined by the ℓ singleton variables $x_{\{i\}}$ and that Had_k accepts exactly $2^\ell = k + 1$ strings. We write Max-Had$_k$ for the Max-CSP$(\mathrm{Had}_k^{\pm})$ problem.

2.2 Factor Graphs and Preprocessing

Next we define the factor graph of a CSP.

Definition 2.10. The factor graph of an instance $I = (X, C)$ of Max-CSP(Γ) is the bipartite graph $G = (X, Y, E)$, where $Y = \{\, S \mid (f, S) \in C \,\}$ is the multiset of scopes of the constraints of I, and there is an edge between a variable $x \in X$ and scope $S \in Y$ whenever $x \in S$.

Note that the factor graph precisely describes the scopes of the constraints of I but is independent of the constraint types f used for each constraint. Our results are about hardness of approximation for Max-CSPs where the factor graph is fixed in advance and the instance only consists of the constraint types. To this end, we make the following definitions.

Definition 2.11. A family $\mathcal{G} = \{\mathcal{G}_n\}_{n>0}$ of factor graphs parameterized by n is *explicit* if $\mathcal{G}_n$ can be constructed in time poly(n).

Definition 2.12. We say that Max-CSP(Γ) is (c, s)*-UFG-NP-hard* if there is an explicit family $\{\mathcal{G}_n\}$ of factor graphs and a polynomial time reduction R from 3-Sat instances I on n variables to Max-CSP(Γ) instances $R(I)$ having factor graph $\mathcal{G}_n$ such that

(1) If I is satisfiable, then $R(I)$ is c-satisfiable.
(2) If I is unsatisfiable, then $R(I)$ is not s-satisfiable.

We often say that a problem "has universal factor graphs" when it is UFG-NP-hard.

[2] Here $\alpha \triangle \beta$ is the symmetric difference of the two sets α and β, i.e., the set of elements that appear in exactly one of the two sets.

Remark 2.13. As a 3-Sat instance might have up to n^3 clauses this definition is rather relaxed in how it handles size parameters. In this paper this does not matter as we do not keep track of degrees of various polynomials appearing in our proofs. If a more fine-grained theory was desired it would be useful to introduce also a parameter m for the number of clauses in the 3-Sat formula and see how this parameter enters into the size of the resulting factor graph family.

Our definitions here have minor technical differences with those in [10]. This was done for reasons of presentation, and it is easy to check that all of the results in both this paper and [10] hold for both (very similar) sets of definitions for universal factor graphs. In particular, it is not difficult to see that if Max-CSP(Γ) is (c, s)-UFG-NP-hard using a family $\mathcal{G}$ then there is no poly(n)-size circuit family $\{C_n\}$ that (c, s)-approximates Max-CSP(Γ) on instances with factor graph $\mathcal{G}_n$ unless NP $\subseteq$ P/poly (in other words, Max-CSP(Γ) is hard to (c, s)-approximate with preprocesssing of the factor graph).

To prove our hardness results, we start with some problem already known to have universal factor graphs, and then do a reduction to a CSP achieving the optimal approximation ratio. The key difference from the standard version of such reductions is that, given two instances with the same factor graph, our reductions must produce two instances with the same factor graphs.

Definition 2.14. A reduction R from Max-CSP(Γ) to Max-CSP(Λ) is *factor graph-preserving* if, whenever two Max-CSP(Γ) instances I and I' have the same factor graph, then $R(I)$ and $R(I')$ also have the same factor graph.

The key property of factor graph-preserving reductions is the following immediate fact.

Fact 2.15. *If* Max-CSP(Γ) *is* (c, s)*-UFG-NP-hard and there is a factor graph-preserving polynomial time reduction from* (c, s)*-approximating* Max-CSP(Γ) *to* (c', s')*-approximating* Max-CSP(Λ)*, then* Max-CSP(Λ) *is* (c', s')*-UFG-NP-hard.*

The starting point for our reductions is the following hardness result for Max-TSA.

Theorem 2.16 ([18]). *There is a constant* $s < 1$ *such that Max-TSA is* $(1, s)$*-UFG-NP-hard.*

2.3 Analysis of Boolean Functions

We use standard notation, but in a slightly non-standard set-up. As mentioned in Section 1.2 we are mostly concerned with analysing tables which are Boolean functions of Boolean functions, rather than functions taking generic bit strings as inputs. Of course there is no real difference between a Boolean function $f \in \mathcal{F}_n$ and a bit string of length 2^n as long as we identify $\{0, 1\}^n$ and $[2^n]$ but the notation looks slightly different. Furthermore, our choice to make Boolean functions take value in $\{0, 1\}$ rather than $\{-1, 1\}$ causes us to many times replace what would have been $A(f)$ in the latter notation by $(-1)^{A(f)}$ in our current notation. Let us turn to some definitions.

Definition 2.17. For $\alpha \subseteq \{0, 1\}^n$ we have a character $\chi_\alpha : \mathcal{F}_n \to \{-1, 1\}$ defined by

$$\chi_\alpha(f) = (-1)^{\sum_{\mathbf{x} \in \alpha} f(\mathbf{x})}.$$

Definition 2.18. For a Boolean function $A : \mathcal{F}_n \to \{0, 1\}$ we define the Fourier coefficients by

$$\hat{A}_\alpha = \mathop{\mathbb{E}}_{f}[(-1)^{A(f)} \chi_\alpha(f)].$$

We have the Fourier inversion formula

$$(-1)^{A(f)} = \sum_{\alpha} \hat{A}_\alpha \chi_\alpha(f)$$

and Plancherel's identity $\sum_\alpha \hat{A}_\alpha^2 = 1$. The Boolean-valued function $A(f)$ and the real-valued function $(-1)^{A(f)}$ are of course just different views of the same mathematical object.

2.4 Parallel Repetition

An instance $I = (X, C)$ of a Max-CSP can be naturally associated with a basic two-prover game. The verifier picks a random constraint $(S, f) \in C$ and a uniformly random variable x_i from the tuple $\mathbf{x}_S$. It sends x_i to prover P_1 and $\mathbf{x}_S$ to prover P_2. P_1 responds with a value for x_i and P_2 responds with values for all the variables in $\mathbf{x}_S$. The verifier accepts if and only if the values given to x_i by the two provers are the same, and the value given to $\mathbf{x}_S$ satisfies f.

If the instance I is satisfiable then the provers can win this game with probability 1 (perfect completeness), and if I is at most $(1 - \delta)$-satisfiable then they can win with probability at most $1 - \delta/k$ where k is the arity of each constraint and thus we preserve soundness strictly smaller than one.

In the r-wise parallel repetition of this game the verifier chooses r random constraints which it sends to P_2 and randomly one variable from each constraint which it sends to P_1. The provers respond with values for all the variables they are sent. The verifier accepts if an only if the values from P_2 satisfy the constraints and match those sent by P_1 on the common variables. The r-parallel repeated game has perfect completeness and soundness c^r for some $c < 1$ [23].

The r-parallel repeated game corresponds naturally to the constraint satisfaction problem known as *label cover* where the answers of the two provers are the assignments (known as labels) to collections of variables, and the constraints are the tests made by the prover. In classical optimal inapproximability it is often not necessary for subsequent reductions to remember the details of how labels and constraints correspond to collections of variables for the original problem. Thus many standard reductions ignore these details and instead focus only on the abstract label cover problem.

In our reductions we need a variant called *smooth* parallel repetition. In this version an extra set of t constraints are sent to both provers. Of course for this to be useful, P_2 does not know which t of its $t + r$ constraints are sent to P_1. The verifier now also checks that it gets the same values for the tk values requested from both provers. It is easy to see that these extra variables sent to both provers do not increase soundness which remains at most c^r. As with standard parallel repetition, one may view the smooth parallel repeated game as an instance of a constraint satisfaction problem known as *smooth label cover*, where again assignments to sets of variables sent to each prover are known as labels.

We use smooth parallel repetition in order to ensure that for any two distinct answers, a_1 and a_2 sent by P_2 it is unlikely that there is one answer by P_1 that is accepting for both a_i. This is ensured by setting t significantly larger than r, while keeping both parameters constant independent of the number of variables in the CSP. Let us give a formal description.

Definition 2.19. Given a CSP instance $I = (X, C)$ the (r, t)-smooth parallel repetition is the following two-prover game.

(1) For $j = 1, \ldots, t + r$ choose a constraint $c_j = (S_j, f_j) \in C$ uniformly at random.
(2) For $j = 1, \ldots, r$ choose a variable $x_{i_j} \in \mathbf{x}_{S_{t+j}}$ uniformly at random.
(3) Send $(\mathbf{x}_{S_j})_{j=1}^{t+r}$ in random order to P_2, and send both $(\mathbf{x}_{S_j})_{j=1}^{t}$ and $(x_{i_j})_{j=1}^{r}$ to P_1.
(4) Receive values for the variables sent to each prover, and check that for each j the values $\mathbf{a}_{S_j}$, given by P_2 to $\mathbf{x}_{S_j}$ satisfy c_j, and that the two values given to each of $(\mathbf{x}_{S_j})_{j=1}^{t}$ and $(x_{i_j})_{j=1}^{r}$ by the two provers agree.

We denote the set of variables sent to P_1 by U and the set sent to P_2 by W.

The smoothness property of the repeated game is quantified by the following claim.

Claim 2.20. For a fixed set of variables W sent to P_2 and any set of possible answers $S \subseteq \{0,1\}^W$ from P_2 in the (r,t)-smooth parallel repetition, the probability (over the choice of U conditioned on W) that there exists two different answers $\mathbf{a}$ and $\mathbf{a}'$ in S such that $\mathbf{a}_U = \mathbf{a}'_U$ is at most $\frac{|S|^2 r}{t+r}$.

Proof. For any two fixed elements $\mathbf{a}$ and $\mathbf{a}'$ in S, the probability that $\mathbf{a}_U = \mathbf{a}'_U$ is at most $\frac{(k-1)r}{k(t+r)} \leq \frac{r}{t+r}$. This follows as they differ in at least one coordinate, the variables of W are sent in random order to P_2, and only $(k-1)r$ of the $k(t+r)$ coordinates are projected away. The claim follows by a union bound over all $\binom{|S|}{2}$ pairs of elements of S. □

3 FUNCTIONAL FOLDING AND REDUCTION TEMPLATE

Most parts of our proofs are standard. We apply parallel repetition to a constraint-versus-variable two-prover proof and then code the answers of the provers by the long code. The main novelty is a new way of folding the long code in a factor graph-preserving way and we now describe this mechanism.

3.1 Factor Graph-Preserving Folding of Long Codes

The full long code of a string $\mathbf{x} \in \{0,1\}^\ell$ is a table $A : \mathcal{F}_\ell \rightarrow \{0,1\}$ indexed by the set $\mathcal{F}_\ell$ of all functions $f : \{0,1\}^\ell \rightarrow \{0,1\}$, where $A(f)$ is supposed to take the value $f(\mathbf{x})$.

When giving a long code of a string $\mathbf{x}$ that is supposed to satisfy some conditions, say $h_i(\mathbf{x}) = b_i$ for $i = 1, 2, \ldots, r$, it turns out to be essential to incorporate these conditions directly into the long code. The idea is to divide the functions into equivalence classes such that the value on one function in the equivalence class determines the value of all other functions in the same equivalence class. It is then sufficient to give only one bit for each equivalence class of functions. This has, in the past, been done in two different ways.

(1) Put f and g in the same equivalence class if and only if $f(\mathbf{x}) = g(\mathbf{x}) + \sum_{i=1}^r \sigma_i h_i(\mathbf{x})$ for some constants σ_i.
(2) Put f and g in the same equivalence class if $f(\mathbf{x}) = g(\mathbf{x})$ for all $\mathbf{x}$ that satisfy $h_i(\mathbf{x}) = b_i$ for all i. In many situations we also allow for the possibility that $f(\mathbf{x}) = \neg g(\mathbf{x})$ for all such $\mathbf{x}$.

In particular, the former method, which we call linear folding, was used by Bellare et al. [4] and the latter, which we call conditioning, by Håstad [15]. The second method creates fewer equivalence classes.

In the current situation we want the construction to be factor graph-preserving and this turns out to be equivalent to the property that the equivalence classes do not depend on the unknown negations. If each constraint comes from a CNF clause on some known set of variables S with unknown negations then it is easy to see that not even linear folding has this property. Once we write such a condition on the form $h(\mathbf{x}_S) = b$ then the identity of h and hence the folding depends on the negations.

On the other hand if we start with an equational CSP, linear folding does have the desired property. Linear folding was sufficient in the case when the soundness analysis of the PCP protocol used established that a table given by the prover was close in Hamming distance to a correct long code. In most sharp inapproximability results one is not able to establish such a strong property and in such situations linear folding is not sufficient for the analysis of the protocol. However, even in the situation of equational CSPs it is easy to see that conditioning creates equivalence classes that are dependent on the right hand sides and thus it seems too strong to use in our setting.

The main new technical tool of this paper is to define an intermediate type of folding that is factor graph-preserving, but is still strong enough to be used instead of conditioning in many situations where a PCP is analyzed using Fourier analysis and/or the Invariance principle. We proceed to define this folding which is based on the following equivalence relation.

Definition 3.1. Let $H = \{h_1, \dots, h_r\} \subseteq \mathcal{F}_\ell$. Two functions $f, g \in \mathcal{F}_\ell$ are *H-equivalent* if there exists $F : \{0,1\}^r \longrightarrow \{0,1\}$ such that

$$f(\mathbf{x}) = g(\mathbf{x}) + F(h_1(\mathbf{x}), \dots, h_r(\mathbf{x}))$$

for all $\mathbf{x} \in \{0,1\}^\ell$.

It is easy to check that this definition gives an equivalence relation, and we refer to the classes of this relation as H-equivalence classes. Now we can define the folding of A over all functions on the constraints.

Definition 3.2. Let $A : \mathcal{F}_\ell \longrightarrow \{0,1\}$ be a supposed long code, $\{h_i(\mathbf{x}) = b_i\}_{i=1}^r$ be a set of constraints, $\mathbf{b} = (b_1, \dots, b_r) \in \{0,1\}^r$, and $H = \{h_1, \dots, h_r\}$. For each class of H-equivalent functions we choose one representative. The folding of A over all functions on H with respect to $\mathbf{b}$, denoted by $A_{H,b}$, is now defined as follows. If g is the representative for an H-equivalence class containing some function f, then we define

$$A_{H,b}(f) \stackrel{\text{def}}{=} A(g) + F(b_1, b_2, \dots b_r).$$

where $f = g + F(h_1, h_2, \dots h_r)$.

We call this folding *functional folding*, and it is easy to see that if $A_{H,b}$ is functionally folded, then changing the choice of representative for each H equivalence class does not change the function. Note that even if $r = 0$ we always fold over the function that is identically true.

Given A, we simulate queries to $A_{H,b}$ in the standard way. Whenever we want to read the value of $A_{H,b}(f)$, we find the representative $g = f + F(h_1, \dots, h_r)$ of the H-equivalence class for f, and return the value $A(g) + F(b_1, b_2, \dots b_r)$. Since the equivalence classes depend only on the h_i and not on the b_i, we immediately have that this folding is factor graph-preserving when we start with an equational CSP.

Fact 3.3. *Let A,H and* $\mathbf{b}$ *be as above. If a PCP verifier simulates queries to $A_{H,b}$ by querying the table A, then the query locations depend only on H.*

While this folding is factor graph-preserving, it is not yet clear that it is actually useful in enforcing the constraints. The next lemma shows that the folding does indeed enforce the constraints in the sense that all non-zero Fourier coefficients correspond to sets of assignments with an odd number of them satisfying the constraints.

Lemma 3.4. *Let $A : \mathcal{F}_\ell \longrightarrow \{0,1\}$ be a supposed long code, $\{h_i(\mathbf{x}) = b_i\}_{i=1}^r$ be a set of constraints,* $\mathbf{b} = (b_1, \dots, b_r) \in \{0,1\}^r$*, and $H = \{h_1, \dots, h_r\}$. Let $A_{H,b}$ be A folded over all functions on H with respect to* $\mathbf{b}$*. If $\hat{A}_{H,b}(\alpha) \neq 0$ then the number of* $\mathbf{x} \in \alpha$ *that simultaneously satisfy $h_i(\mathbf{x}) = b_i$ for all i is odd.*

Proof. Let $h(\mathbf{x}) = (h_1(\mathbf{x}), \dots, h_r(\mathbf{x}))$. Suppose that the number of $\mathbf{x} \in \alpha$ that satisfy all of the constraints $h_i(\mathbf{x}) = b_i$ is even. Recall that

$$\hat{A}_{H,b}(\alpha) = \mathop{\mathbb{E}}_{f \in \mathcal{F}_\ell}\left[(-1)^{A_{H,b}(f)} \chi_\alpha(f)\right] = \mathop{\mathbb{E}}_{f \in \mathcal{F}_\ell}\left[(-1)^{A_{H,b}(f)} \prod_{\mathbf{x} \in \alpha} (-1)^{f(\mathbf{x})}\right]. \tag{3.1}$$

Consider the function $\mathbf{1}_b : \{0,1\}^r \longrightarrow \{0,1\}$ which is one only at the point $\mathbf{b}$. Now we pair up the functions f and $f + \mathbf{1}_b(h)$ in (3.1). By the folding we have

$$A_{H,b}(f + \mathbf{1}_b(h)) = A_{H,b}(f) + \mathbf{1}_b(\mathbf{b}) = A_{H,b}(f) + 1.$$

In particular this implies that $(-1)^{A_{H,b}(f+\mathbf{1}_b(h))} = -((-1)^{A_{H,b}(f)})$. On the other hand, since the number of $\mathbf{x} \in \alpha$ satisfying $h(\mathbf{x}) = \mathbf{b}$ is even we have

$$\prod_{\mathbf{x}\in\alpha}(-1)^{f(\mathbf{x})+\mathbf{1}_b(h(\mathbf{x}))} = \prod_{\mathbf{x}\in\alpha}(-1)^{f(\mathbf{x})}(-1)^{\mathbf{1}_b(h(\mathbf{x}))} = \prod_{\mathbf{x}\in\alpha}(-1)^{f(\mathbf{x})}$$

Thus the terms corresponding to f and $f+\mathbf{1}_b(h)$ cancel in pairs and the expectation (3.1) is zero. □

Remark. By a similar argument, for any fixed choice of $\mathbf{c} \neq \mathbf{b}$, if $\hat{A}_{H,b}(\beta) \neq 0$ the number of elements $\mathbf{x} \in \beta$ such that $h(\mathbf{x}) = \mathbf{c}$ is even. As this is not needed in our arguments we leave the verification of this to the reader.

Lemma 3.4 shows that if we use functional folding then for any non-zero Fourier coefficient at least one element satisfies all the constraints. This should be compared with conditioning where *all* elements in a non-zero Fourier coefficient satisfies all the constraints. On the other hand with linear folding where we have no such guarantees and it is difficult to find an assignment that satisfies the constraints given a non-zero Fourier coefficient.

3.2 Basic Setup of Hardness Reductions

It turns out that by using some minor modifications, the guarantee of Lemma 3.4 is sufficient to analyze many of the classical protocols.

We follow the standard setup: start with a Max-CSP having some hardness of approximation, apply parallel repetition to create a two-prover game with perfect completeness and arbitrarily small soundness, and then long code the answers of the two provers and test the long codes using constraints from the target CSP. In order to apply functional folding to the long codes, we start with an equational CSP and in particular use the hardness of the Max-TSA problem (Theorem 2.16), and in order to be able to use the weaker guarantee of functional folding as compared to conditioning we use smooth parallel repetition (Definition 2.19).

As all the reductions follow the same initial steps and only the exact choice of constraints varies from CSP to CSP, we summarize these first steps and the notation used in the following definition.

Definition 3.5 (Reduction template for Max-CSPs with universal factor graphs)**.** Given a Max-TSA instance $I = (X, C)$ and parameters r and t, we construct a new set X' of Boolean variables as follows.

First, form the (r, t)-smooth parallel repeated game of I as in Definition 2.19. For any set of variables $U \subseteq X$ that may be sent to P_1, we introduce a supposed long code $A_U : \mathcal{F}_W \to \{0, 1\}$ of the assignment to U. We write $A_U^{\boxplus}$ for A_U functionally folded over all constraints of I induced by U. For a set $W \subseteq X$ that may be sent to P_2 we define B_W and $B_W^{\boxplus}$ analogously.

The variables X' consist of all values of $A_U(f)$ and $B_W(g)$ over all U, W, $f \in \mathcal{F}_U$ and $g \in \mathcal{F}_W$.

In the concrete reductions in the rest of the paper, a set U (resp. W) always refers to a query to P_1 (resp. P_2) in the repeated game.

Note that, as per the definition of functional folding in the previous section, queries to $A_U^{\boxplus}$ and $B_W^{\boxplus}$ are made by querying fixed entries of A_U and B_W (depending only on the factor graph of I) and then possibly negating the result (depending on the right hand sides of I).

4 CLASSICAL OPTIMAL INAPPROXIMABILITY

Using the folding introduced in the previous section we prove classical optimal inapproximability results with universal factor graphs. The first two problems, Max-3-Sat and Max-3-Lin, were first proven to be approximation resistant in [15]. The third problem, Max-TSA, did not appear in that original paper, but the techniques used follow the standard dictatorship testing analysis via the Fourier transform introduced there. In each case, we start with the reduction template from Definition 3.5 and then construct a PCP verifier for the target problem.

In the hardness results for Max-3-Sat and Max-TSA, we use the notation $\beta_{\oplus U}$ to denote the set of vectors $\mathbf{x} \in \beta_U$ such that there is an odd number of $\mathbf{y} \in \beta$ with $\mathbf{y}_U = \mathbf{x}$. In the notation of [15], this is the π_2 operator.

Remark 4.1. In this section we only explicitly give proofs of approximation resistance and not uselessness. However the proofs implicitly yield the stronger property of uselessness. We elaborate on this in Section 5 where uselessness is defined and discussed in more detail.

4.1 Max-3-Lin

Theorem 4.2. *For any $\varepsilon > 0$, Max-3-Lin is $(1 - \varepsilon, \frac{1}{2} + \varepsilon)$-UFG-NP-hard.*

We remark that an a priori alternative route (and simpler, compared to what we do below) to obtaining this theorem would be to use as starting point $(1 - \varepsilon, 0.999)$-UFG-NP-hardness of Max-3-Lin. Starting with such a hardness result, one could derive Theorem 4.2 with a simpler (and more efficient) reduction based on Hadamard codes rather than long codes. However, prior to this paper, such a starting point was not known (for instance Feige and Jozeph, obtaining the hardness results of Bellare et al., prove a $(6/7, 3/4 + \varepsilon)$-UFG-NP-hardness for Max-3-Lin). The only way we know how to derive the starting point needed for such a Hadamard code-based reduction to go through is using our reduction here which already establishes Theorem 4.2.

The long code test we use to establish the result is exactly the same as the original test of Håstad in [15], but as explained in Section 3, we can only access the functional foldings of the long codes, and not condition them on the constraints of the underlying CSP.

Definition 4.3. The Max-3-Lin PCP verifier does the following:

(1) Pick a random pair of sets (U, W) sent to the two provers in the parallel game.
(2) Pick a uniform random function $f \in \mathcal{F}_U$ and a uniform random $g_1 \in \mathcal{F}_W$.
(3) Define $g_2 \in \mathcal{F}_W$ by setting $g_2(\mathbf{y}) = g_1(\mathbf{y}) + f(\mathbf{y}_U)$ with probability $1 - \varepsilon$ and the negation of this value otherwise, independently for each $\mathbf{y} \in \{0, 1\}^W$.
(4) Accept if and only if $A_U^{\boxplus}(f) + B_W^{\boxplus}(g_1) + B_W^{\boxplus}(g_2) = 0$.

The tests defined above correspond to checking three variable linear equations over $\mathbb{F}_2$, and so the PCP defines a Max-3-Lin instance. Also we have immediately by Fact 3.3 that the overall reduction from Max-TSA to Max-3-Lin given by this PCP is factor graph-preserving.

Now we analyze the completeness and soundness of the PCP.

Lemma 4.4. *The completeness of the Max-3-Lin verifier in Definition 4.3 is $1 - \varepsilon$.*

Proof. Let $\mathbf{a}$ be a satisfying assignment to the original Max-TSA constraints. Then we set $A_U(f) = f(\mathbf{a}_U)$ and $B_W(g) = f(\mathbf{a}_W)$ for all subsets of variables in the two prover game. Note that $B_W(g) = B_W^{\boxplus}(g)$ since functional folding does not affect a true long code of a satisfying assignment. Second, with probability $1 - \varepsilon$ we have $A_U^{\boxplus}(f) + B_W^{\boxplus}(g_1) + A_W^{\boxplus}(g_2) = f(\mathbf{a}_U) + g_1(\mathbf{a}_W) + g_2(\mathbf{a}_W) = 0$ and hence the test accepts with probability $1 - \varepsilon$. □

We now prove soundness for the test.

Lemma 4.5. *Let $\varepsilon, \delta > 0$ and set $C_{\varepsilon,\delta} \overset{\text{def}}{=} (2\varepsilon)^{-1} \log(\frac{2}{\delta})$. For $r > 0$ set $t = r \cdot \frac{4}{\delta^2} C_{\varepsilon,\delta}^2$. If the Max-3-Lin PCP verifier accepts with probability at least $\frac{1+\delta}{2}$ then there is a strategy for the provers in the (r, t)-smooth parallel repeated game that causes the verifier to accept with probability at least $\frac{\delta^4}{16 C_{\varepsilon,\delta}}$.*

Proof. For notational convenience, throughout this proof we write A_U and B_W instead of $A_U^{\boxplus}$ and $B_W^{\boxplus}$, but it is important to keep in mind that all long codes are folded. By construction, $(-1)^{A_U(f)}(-1)^{B_W(g_1)}(-1)^{B_W(g_2)}$ is 1 when the test accepts and -1 when it rejects. Now the standard analysis (see [15]) gives that, if the test accepts with probability $(1+\delta)/2$ then

$$\begin{aligned}\delta &= \mathop{\mathbb{E}}_{U,W}\left[\mathop{\mathbb{E}}_{f,g_1,g_2}\left[(-1)^{A_U(f)}(-1)^{B_W(g_1)}(-1)^{B_W(g_2)}\right]\right]\\ &= \mathop{\mathbb{E}}_{U,W}\left[\sum_{\alpha,\beta_1,\beta_2}\hat{A}_U(\alpha)\hat{B}_W(\beta_1)\hat{B}_W(\beta_2)\mathop{\mathbb{E}}_{f,g_1,g_2}\left[\chi_\alpha(f)\chi_{\beta_1}(g_1)\chi_{\beta_2}(g_2)\right]\right]\\ &= \mathop{\mathbb{E}}_{U,W}\left[\sum_{\beta}\hat{A}_U(\beta_{\oplus U})\hat{B}_W(\beta)^2\,\mathbb{E}\left[\chi_{\beta_{\oplus U}}(f)\chi_\beta(g_1+g_2)\right]\right]\\ &= \mathop{\mathbb{E}}_{U,W}\left[\sum_{\beta}\hat{A}_U(\beta_{\oplus U})\hat{B}_W(\beta)^2(1-2\varepsilon)^{|\beta|}\right]\end{aligned}$$

where the final equality follows from step (3) in Definition 4.3, in particular from the fact that $(-1)^{f(y_U)} = (-1)^{g_1(y)+g_2(y)}$ with probability $1-\varepsilon$, and is equal to the negation otherwise. An application of Cauchy-Schwartz to the sum over β yields

$$\delta^2 \leq \mathop{\mathbb{E}}_{U,W}\left[\left(\sum_{\beta}\hat{A}_U(\beta_{\oplus U})^2\hat{B}_W(\beta)^2(1-2\varepsilon)^{2|\beta|}\right)\left(\sum_{\beta}\hat{B}_W(\beta)^2\right)\right].$$

Plancherel applied to the second term inside the expectation gives

$$\delta^2 \leq \mathop{\mathbb{E}}_{U,W}\left[\sum_{\beta}\hat{A}_U(\beta_{\oplus U})^2\hat{B}_W(\beta)^2(1-2\varepsilon)^{2|\beta|}\right]. \tag{4.1}$$

Note that the contribution to the inner sum from β that are larger than $C_{\varepsilon,\delta}$ is at most $\delta^2/4$. Thus we have

$$\frac{3}{4}\delta^2 \leq \mathop{\mathbb{E}}_{U,W}\left[\sum_{|\beta|\leq C_{\varepsilon,\delta}}\hat{A}_U(\beta_{\oplus U})^2\hat{B}_W(\beta)^2\right]. \tag{4.2}$$

The strategies for the two provers are as follows. With probability $\hat{A}_U(\alpha)^2$ prover P_1 outputs a random $\mathbf{x} \in \alpha$, and with probability $\hat{B}_W(\beta)^2$ prover P_2 outputs a random $\mathbf{y} \in \beta$ which satisfies all the constraints on W. By Lemma 3.4 for any β with $\hat{B}_W(\beta) \neq 0$ there are an odd number of assignments $\mathbf{y} \in \beta$ satisfying the constraints on W. In particular, there is at least one assignment $\mathbf{y}^* \in \beta$ satisfying all the constraints. However, if $\mathbf{y}^*$ and another assignment $\mathbf{y} \in \beta$ collide under the map $\mathbf{y} \rightarrow \mathbf{y}_U$, then there may be no corresponding assignment $\mathbf{x} = \mathbf{y}_U^*$ in $\beta_{\oplus U}$.

We now analyze the contribution to (4.2) of terms where such collisions occur. By Claim 2.20, we have that for each fixed W and β, the probability over the choice of U of a collision is at most $|\beta|^2\frac{r}{t+r}$. Let $S_{W,\beta}$ be the set of U which cause a collision. Then

$$\sum_{|\beta|\leq C_{\varepsilon,\delta}}\mathop{\mathbb{E}}_{U}\left[\mathbf{1}(U \in S_{W,\beta})\hat{A}_U(\beta_{\oplus U})^2\hat{B}_W(\beta)^2\right] \leq \sum_{|\beta|\leq C_{\varepsilon,\delta}}\mathop{\mathbb{E}}_{U}\left[\mathbf{1}(U \in S_{W,\beta})\right]$$

$$\le C^2_{\varepsilon,\delta}\frac{r}{t+r} \le \frac{\delta^2}{4}.$$

Combining this with (4.2) yields

$$\frac{\delta^2}{2} \le \mathop{\mathbb{E}}_{U,W}\left[\sum_{|\beta|\le C_{\varepsilon,\delta}} \mathbf{1}(U \notin S_{W,\beta})\hat{A}_U(\beta_{\oplus U})^2\hat{B}_W(\beta)^2\right].$$

Since the inner sum is non-negative and bounded by one, we have by Markov's inequality that with probability at least $\frac{\delta^2}{4}$ over the choice of U and W

$$\frac{\delta^2}{4} \le \sum_{|\beta|\le C_{\varepsilon,\delta}} \mathbf{1}(U \notin S_{W,\beta})\hat{A}_U(\beta_{\oplus U})^2\hat{B}_W(\beta)^2.$$

This implies that for each such pair (U, W), with probability at least $\frac{\delta^2}{4}$ the following events all occur:

- P_2 chooses some β with $|\beta| \le C_{\varepsilon,\delta}$.
- P_1 chooses $\alpha = \beta_{\oplus U}$.
- $U \notin S_{W,\beta}$ i.e. no two elements of β collide under projection to U.

If no two elements of β collide under the map $\mathbf{y} \mapsto \mathbf{y}_U$, then the projection $\mathbf{y}^*_U$ of the good assignment $\mathbf{y}^* \in \beta$ satisfying all the constraints on W is also an element of $\beta_{\oplus U}$. Therefore P_1 chooses $\mathbf{y}^*_U$ with probability at least $|\alpha|^{-1} \ge |\beta|^{-1} \ge C^{-1}_{\varepsilon,\delta}$. When this happens, the responses of P_1 and P_2 to U and W are accepted in the parallel repeated game. Thus, as these three events happen for a $\frac{\delta^2}{4}$ fraction of all queries (U, W), the strategy of the two provers is accepted with probability at least $\frac{\delta^4}{16}C^{-1}_{\varepsilon,\delta}$, as desired. □

The reduction can now be completed by recalling that the parallel repeated game has soundness c^r for some constant c. But by Lemma 4.5 (setting $\delta = 2\varepsilon$) the value of the repeated game is larger than c^r for sufficiently large r, unless fewer than a $\frac{1}{2} + \varepsilon$ fraction of equations in the Max-3-Lin instance can be satisfied.

4.2 Max-3-Sat

As with Max-3-Lin, we can use our new folding along with a variant of the Max-3-Sat test in [15] to obtain UFG-NP-hardness. Here we do not follow exactly the same long code test as [15] but instead follow a subsequent simplified test and analysis using smooth label cover [14, 19].

Theorem 4.6. *For any $\varepsilon > 0$, Max-3-Sat is $(1, \frac{7}{8} + \varepsilon)$-UFG-NP-hard.*

The PCP we construct for Max-3-Sat is described below.

Definition 4.7. The Max-3-Sat PCP verifier does the following:

(1) Pick a random pair of sets (U, W) sent to the two provers in the parallel game.
(2) Pick a uniform random function $f \in \mathcal{F}_U$ and a uniform random $g_1 \in \mathcal{F}_W$.
(3) Pick a function $g_2 \in \mathcal{W}$ as follows. For each $\mathbf{y} \in \{0,1\}^W$, if $f(\mathbf{y}_U) = 0$ then assign $g_2(\mathbf{y}) = g_1(\mathbf{y}) + 1$. If $f(\mathbf{y}_U) = 1$ then with probability $1 - \varepsilon$ assign $g_2(\mathbf{y}) = g_1(\mathbf{y})$ and otherwise assign $g_2(\mathbf{y}) = g_1(\mathbf{y}) + 1$.
(4) Accept unless $A^{\boxplus}_U(f) = B^{\boxplus}_W(g_1) = B^{\boxplus}_W(g_2) = 0$.

It is easy to prove that this test has perfect completeness, and as the proof is near-identical to that of Lemma 4.4 we omit it.

Lemma 4.8. *The completeness of the Max-3-Sat verifier in Definition 4.7 is* 1.

Lemma 4.9. *Let $\delta > 0$, and let $0 < \varepsilon < \left(\frac{\delta}{64}\right)^2 \log(\frac{16}{\delta})^{-2}$. Set $C_{\varepsilon,\delta} = \frac{2}{\varepsilon}\log(\frac{16}{\delta})$. For $r > 0$, let $t = r \cdot \frac{16}{\delta} C^2_{\varepsilon,\delta}$. If the Max-3-Sat PCP verifier accepts with probability at least $\frac{7+\delta}{8}$ then there is a strategy for the provers in the (r,t)-smooth parallel repeated game that causes the verifier to accept with probability at least $\frac{\delta^4}{64C_{\varepsilon,\delta}}$.*

Proof. As in the soundness analysis for Max-3-Lin, we write A_U and B_W instead of $A^{\boxplus}_U$ and $B^{\boxplus}_W$ throughout this proof in order to keep the notation manageable. The probability that the test accepts is given by

$$\mathop{\mathbb{E}}_{U,W}\left[\mathop{\mathbb{E}}_{f,g_1,g_2}\left[1 - \frac{1}{8}(1 + (-1)^{A_U(f)})(1 + (-1)^{B_W(g_1)})(1 + (-1)^{B_W(g_2)})\right]\right]. \tag{4.3}$$

Recall that the long codes A_U and B_W are both folded over true i.e. $A_U(f + \mathbf{1}) = A_U(f) + 1$ which can be seen by taking $F \equiv 1$ in Definition 3.1. Folding over true implies that $\mathbb{E}\left[(-1)^{A_U(f)}\right] = \mathbb{E}\left[(-1)^{B_W(g_i)}\right] = 0$. Furthermore since f and g_i are independent we have $\mathbb{E}\left[(-1)^{A_U(f)}(-1)^{B_W(g_i)}\right] = \mathbb{E}\left[(-1)^{A_U(f)}\right]\mathbb{E}\left[(-1)^{B_W(g_i)}\right] = 0$. In other words, after expanding (4.3), any non-constant term which does not contain $(-1)^{B_W(g_1)}(-1)^{B_W(g_2)}$ becomes 0, so the acceptance probability equals

$$\frac{7}{8} - \frac{1}{8}\mathop{\mathbb{E}}_{U,W}\left[\mathop{\mathbb{E}}_{f,g_1,g_2}\left[(-1)^{B_W(g_1)}(-1)^{B_W(g_2)} + (-1)^{A_U(f)}(-1)^{B_W(g_1)}(-1)^{B_W(g_2)}\right]\right] \geq \frac{7+\delta}{8} \tag{4.4}$$

We analyze the two terms in the above expectation one at a time.

Claim 4.10.

$$\left|\mathop{\mathbb{E}}_{U,W}\left[\mathop{\mathbb{E}}_{g_1,g_2}\left[(-1)^{B_W(g_1)}(-1)^{B_W(g_2)}\right]\right]\right| \leq \delta/4. \tag{4.5}$$

Proof. For a fixed choice of U and W, writing out the Fourier expansion of the inner expectation of this term yields:

$$\begin{aligned}\mathop{\mathbb{E}}_{g_1,g_2}\left[(-1)^{B_W(g_1)}(-1)^{B_W(g_2)}\right] &= \sum_{\beta_1,\beta_2}\hat{B}_W(\beta_1)\hat{B}_W(\beta_2)\,\mathbb{E}\left[\chi_{\beta_1}(g_1)\chi_{\beta_2}(g_2)\right] \\ &= \sum_{\beta}\hat{B}_W(\beta)^2\,\mathbb{E}\left[\chi_\beta(g_1+g_2)\right],\end{aligned} \tag{4.6}$$

where we used the fact that for $\beta_1 \neq \beta_2$ the expectation inside the above sum is zero. Now observe that if $\mathbf{y}_U \neq \mathbf{y}'_U$, then $g_1(\mathbf{y}) + g_2(\mathbf{y})$ is independent of $g_1(\mathbf{y}') + g_2(\mathbf{y}')$. Using this fact we have

$$\mathbb{E}\left[\chi_\beta(g_1+g_2)\right] = \mathbb{E}\left[\prod_{\mathbf{y}\in\beta}(-1)^{g_1(\mathbf{y})+g_2(\mathbf{y})}\right] = \prod_{\mathbf{x}\in\beta_U}\mathbb{E}\left[\prod_{\mathbf{y}\in\beta:\mathbf{y}_U=\mathbf{x}}(-1)^{g_1(\mathbf{y})+g_2(\mathbf{y})}\right]$$

Let $s_{\mathbf{x}}$ be the number of $\mathbf{y}\in\beta$ with $\mathbf{y}_U = \mathbf{x}$. The expectation over $\mathbf{y}$ which project to the same $\mathbf{x}$ is equal to

$$\mathbb{E}\left[\prod_{\mathbf{y}\in\beta:\mathbf{y}_U=\mathbf{x}}(-1)^{g_1(\mathbf{y})+g_2(\mathbf{y})}\right] = \frac{1}{2}((-1)^{s_{\mathbf{x}}} + (1-2\varepsilon)^{s_{\mathbf{x}}}).$$

We bound such terms in two different ways depending on the size of β. First, observe that $\left|\frac{1}{2}((-1)^{s_{\mathbf{x}}} + (1-2\varepsilon)^{s_{\mathbf{x}}})\right| \leq 1-\varepsilon$ and so $\left|\mathbb{E}\left[\chi_\beta(g_1+g_2)\right]\right| \leq (1-\varepsilon)^{|\beta_U|}$. Applying Claim 2.20 to the first $C_{\varepsilon,\delta}$ elements of β for $|\beta| > C_{\varepsilon,\delta}$, we have $|\beta_U| \ge C_{\varepsilon,\delta}$ except with probability at most $\frac{r}{t+r}C_{\varepsilon,\delta}^2 \le \delta/16$ over the choice of U. Thus, averaging over U and W we get

$$\begin{aligned}\left|\mathop{\mathbb{E}}_{U,W}\left[\sum_{\beta:|\beta|>C_{\varepsilon,\delta}} \hat{B}_W(\beta)^2\,\mathbb{E}\left[\chi_\beta(g_1+g_2)\right]\right]\right| &\le \mathop{\mathbb{E}}_{U,W}\left[\sum_{\beta:|\beta|>C_{\varepsilon,\delta}} \hat{B}_W(\beta)^2(1-\varepsilon)^{|\beta_U|}\right]\\ &\le \mathop{\mathbb{E}}_{W}\left[\sum_{\beta:|\beta|>C_{\varepsilon,\delta}} \hat{B}_W(\beta)^2\left((1-\varepsilon)^{C_{\varepsilon,\delta}}+\delta/16\right)\right] \le \frac{\delta}{8},\end{aligned} \tag{4.7}$$

where the last inequality used our choice of $C_{\varepsilon,\delta}$ and Plancherel. We turn to β of size at most $C_{\varepsilon,\delta}$ and, again by Claim 2.20, except with probability at most $\frac{r}{t+r}|\beta|^2 \le \delta/16$ over the choice of U, there is at least one $\mathbf{y}$ in β that does not collide with any other $\mathbf{y}' \in \beta$. Letting $\mathbf{x} = \mathbf{y}_U$ we then have $s_{\mathbf{x}} = 1$. For such "good" choices of U we have $\frac{1}{2}((-1)^{s_{\mathbf{x}}} + (1-2\varepsilon)^{s_{\mathbf{x}}}) = -\varepsilon$. Thus after averaging over U and W we have that

$$\left|\mathop{\mathbb{E}}_{U,W}\left[\sum_{\beta:|\beta|\le C_{\varepsilon,\delta}} \hat{B}_W(\beta)^2\,\mathbb{E}\left[\chi_\beta(g_1+g_2)\right]\right]\right| \le \varepsilon + \delta/16 < \frac{\delta}{8}. \tag{4.8}$$

Adding up (4.7) and (4.8) we obtain (4.5) (via (4.6)). □

We now analyze the second term.

Claim 4.11.

$$\left|\mathop{\mathbb{E}}_{U,W}\left[\mathop{\mathbb{E}}_{g_1,g_2}\left[(-1)^{A_U(f)}(-1)^{B_W(g_1)}(-1)^{B_W(g_2)}\right]\right]\right| \le \frac{\delta}{4} + \mathop{\mathbb{E}}_{U,W}\left[\sum_{|\beta|\le C_{\varepsilon,\delta}} \hat{A}_U(\beta_U)^2\hat{B}_W(\beta)^2(1-\varepsilon)^{2|\beta_U|}\right]^{1/2} \tag{4.9}$$

Proof. For a fixed choice of U and W, writing the Fourier expansion of the inner expectation yields

$$\begin{aligned}\mathop{\mathbb{E}}_{g_1,g_2}\left[(-1)^{A_U(f)}(-1)^{B_W(g_1)}(-1)^{B_W(g_2)}\right] &= \sum_{\alpha,\beta_1,\beta_2} \hat{A}_U(\alpha)\hat{B}_W(\beta_1)\hat{B}_W(\beta_2)\,\mathbb{E}\left[\chi_\alpha(f)\chi_{\beta_1}(g_1)\chi_{\beta_2}(g_2)\right]\\ &= \sum_{\beta,\alpha\subseteq\beta_U} \hat{A}_U(\alpha)\hat{B}_W(\beta)^2\,\mathbb{E}\left[\chi_\alpha(f)\chi_\beta(g_1+g_2)\right]\end{aligned} \tag{4.10}$$

where we have used the fact that the expectation is zero unless $\beta_1 = \beta_2 = \beta$ and $\alpha \subseteq \beta_U$. Let us define $E(\alpha,\beta) \overset{\text{def}}{=} \mathbb{E}\left[\chi_\alpha(f)\chi_\beta(g_1+g_2)\right]$ to denote the inner expectation above (and note that this function depends on U and W). Next let $s_{\mathbf{x}}$ be the number of $\mathbf{y} \in \beta$ such that $\mathbf{y}_U = \mathbf{x}$. Then for $\alpha \subseteq \beta_U$

$$E(\alpha,\beta) = \prod_{\mathbf{x}\in\alpha}\frac{1}{2}((-1)^{s_{\mathbf{x}}} - (1-2\varepsilon)^{s_{\mathbf{x}}}) \prod_{\mathbf{x}\in\beta_U\setminus\alpha}\frac{1}{2}((-1)^{s_{\mathbf{x}}} + (1-2\varepsilon)^{s_{\mathbf{x}}}). \tag{4.11}$$

Observe that this implies $|E(\alpha,\beta)| \le (1-\varepsilon)^{|\beta_U|}$, since every factor in the product is bounded in magnitude by $1-\varepsilon$. Further note that

$$\begin{aligned}\sum_{\alpha\subseteq\beta_U} E(\alpha,\beta)^2 &= \prod_{\mathbf{x}\in\beta_U}\left(\left(\frac{1}{2}((-1)^{s_{\mathbf{x}}} - (1-2\varepsilon)^{s_{\mathbf{x}}})\right)^2 + \left(\frac{1}{2}((-1)^{s_{\mathbf{x}}} + (1-2\varepsilon)^{s_{\mathbf{x}}})\right)^2\right)\\ &\le (1-\varepsilon)^{|\beta_U|}\end{aligned}$$

where the final inequality follows from the fact that each factor above has the form $a^2 + b^2$ where both $|a|$ and $|b|$ are bounded by $1 - \varepsilon$, and $|a| + |b| = 1$. Therefore each factor is bounded by $(1 - \varepsilon)^2 + \varepsilon^2$ which is at most $1 - \varepsilon$ whenever $\varepsilon \le 1/2$.

As in Claim 4.10 we split the sum depending on the size of β. First by Cauchy-Schwarz applied to the inner sum over $\alpha \subseteq \beta_U$ and then Plancherel we have

$$\begin{aligned}\sum_{\beta:|\beta|\ge C_{\varepsilon,\delta}} \hat{B}_W(\beta)^2 \sum_{\alpha\subseteq\beta_U} \hat{A}_U(\alpha)E(\alpha,\beta) &\le \sum_{\beta:|\beta|\ge C_{\varepsilon,\delta}} \hat{B}_W(\beta)^2 \left(\sum_{\alpha\subseteq\beta_U} \hat{A}_U(\alpha)^2\right)^{1/2} \left(\sum_{\alpha\subseteq\beta_U} E(\alpha,\beta)^2\right)^{1/2} \\ &\le \sum_{\beta:|\beta|\ge C_{\varepsilon,\delta}} \hat{B}_W(\beta)^2 (1-\varepsilon)^{\frac{|\beta_U|}{2}} \qquad (4.12)\end{aligned}$$

As in (4.7) it follows from our choice of $C_{\varepsilon,\delta}$ that, when averaging over U and W, (4.12) is bounded in absolute value by $\delta/8$, i.e.,

$$\left| \mathop{\mathbb{E}}_{U,W} \left[\sum_{\beta:|\beta|\ge C_{\varepsilon,\delta}} \hat{B}_W(\beta)^2 \sum_{\alpha\subseteq\beta_U} \hat{A}_U(\alpha)E(\alpha,\beta) \right] \right| \le \delta/8. \qquad (4.13)$$

Turning to the low-degree terms where $|\beta| \le C_{\varepsilon,\delta}$, we apply Claim 2.20 to conclude that except with probability $\frac{r}{t+r}|\beta| \le \delta/16$ over the choice of U, there are no collisions between any $\mathbf{y}, \mathbf{y}'$ in β under the map $\mathbf{y} \mapsto \mathbf{y}_U$. Letting $\mathbf{x} = \mathbf{y}_U$ we then have $s_{\mathbf{x}} = 1$. So restricting to such good choices of U we have $\frac{1}{2}((-1)^{s_{\mathbf{x}}} + (1-2\varepsilon)^{s_{\mathbf{x}}}) = -\varepsilon$. Since all other factors in the product in (4.11) are bounded in magnitude by 1 we have for such "good" U that $|E(\alpha,\beta)| \le \varepsilon^{|\beta_U \setminus \alpha|}$. Thus, in this case summing over all $\alpha \subsetneq \beta_U$ yields

$$\begin{aligned}\sum_{|\beta|\le C_{\varepsilon,\delta}} \hat{B}_W(\beta)^2 \sum_{\alpha\subsetneq\beta_U} \hat{A}_U(\alpha)E(\alpha,\beta) &\le \sum_{|\beta|\le C_{\varepsilon,\delta}} \hat{B}_W(\beta)^2 \left(\sum_{\alpha\subsetneq\beta_U} \hat{A}_U(\alpha)^2\right)^{1/2} \left(\sum_{\alpha\subsetneq\beta_U} E(\alpha,\beta)^2\right)^{1/2} \\ &\le \sum_{|\beta|\le C_{\varepsilon,\delta}} \hat{B}_W(\beta)^2 \left(\sum_{\alpha\subsetneq\beta_U} \varepsilon^{2|\beta_U\setminus\alpha|}\right)^{1/2} \\ &= \sum_{|\beta|\le C_{\varepsilon,\delta}} \hat{B}_W(\beta)^2 \left((1+\varepsilon^2)^{|\beta_U|} - 1\right)^{1/2} \\ &\le \sum_{|\beta|\le C_{\varepsilon,\delta}} \hat{B}_W(\beta)^2 2\varepsilon C_{\varepsilon,\delta}^{1/2},\end{aligned}$$

since $(1+a)^b \le 1 + 2ab$ whenever $0 \le ab \le \frac{1}{32}$. For the remaining at most $\delta/16$ fraction of choices of U where collisions occur, we simply use the previous bound $\sum_{\alpha\subseteq\beta_U} E(\alpha,\beta)^2 \le (1-\varepsilon)^{|\beta_U|} \le 1$. Thus averaging over U and W yields

$$\left| \mathop{\mathbb{E}}_{U,W} \left[\sum_{|\beta|\le C_{\varepsilon,\delta}} \sum_{\alpha\subsetneq\beta_U} \hat{A}_U(\alpha)\hat{B}_W(\beta)^2 E(\alpha,\beta) \right] \right| \le \mathop{\mathbb{E}}_{W} \left[\sum_{|\beta|\le C_{\varepsilon,\delta}} \hat{B}_W(\beta)^2 \left(2\varepsilon C_{\varepsilon,\delta}^{1/2} + \delta/16\right) \right] \le \frac{\delta}{8}. \qquad (4.14)$$

Finally, applying Cauchy-Schwarz and Plancherel to the sum where $\alpha = \beta_U$ yields

$$\left| \mathop{\mathbb{E}}_{U,W} \left[\sum_{|\beta|\le C_{\varepsilon,\delta}} \hat{A}_U(\beta_U)\hat{B}_W(\beta)^2 E(\alpha,\beta) \right] \right| \le \mathop{\mathbb{E}}_{U,W} \left[\sum_{|\beta|\le C_{\varepsilon,\delta}} \hat{A}_U(\beta_U)^2 \hat{B}_W(\beta)^2 (1-\varepsilon)^{2|\beta_U|} \right]^{1/2} \qquad (4.15)$$

Adding up (4.13), (4.14), and (4.15), we obtain (4.9) (via (4.10)). □

Now plugging the bounds of Claim 4.10 and Claim 4.11 into (4.4), we see that

$$\frac{\delta^2}{4} \le \mathop{\mathbb{E}}_{U,W}\left[\sum_{|\beta|\le C_{\varepsilon,\delta}} \hat{A}_U(\beta_U)^2\hat{B}_W(\beta)^2(1-\varepsilon)^{2|\beta_U|}\right]. \tag{4.16}$$

Note this is similar to the expression we obtained for the Max-3-Lin verifier in Lemma 4.5, with slightly different parameters. Using exactly the same strategy for the two provers with a similar analysis, we obtain a strategy with success probability $\frac{\delta^4}{64C_{\varepsilon,\delta}}$. □

4.3 Max-TSA

Thus far we have used Max-TSA with constant, but non-optimal, soundness as the starting point for all of our reductions. Now we also show that even obtaining any non-trivial approximation of Max-TSAis UFG-NP-hard.

Theorem 4.12. *For any $\varepsilon > 0$, Max-TSA is $(1, 1/2+\varepsilon)$-UFG-NP-hard.*

We now construct the following PCP verifier, which makes queries of the form $f_{\mathrm{TSA}}(x) = b$.

Definition 4.13. The TSA PCP verifier does the following:

(1) Pick a random pair of sets (U, W) sent to the two provers in the parallel game.
(2) Sample $f \in \mathcal{F}_U$ and $g_1, g_3, g_4 \in \mathcal{F}_W$ all independently and uniformly at random.
(3) Set $g_2(\mathbf{y}) = f(\mathbf{y}_U) + g_1(\mathbf{y}) + g_3(\mathbf{y}) \wedge g_4(\mathbf{y})$.
(4) Accept if and only if $A^{\boxplus}_U(f) + B^{\boxplus}_W(g_1) + B^{\boxplus}_W(g_2) + B_W(g_3) \wedge B_W(g_4) = 0$

A subtle yet crucial point is that the queries for g_3 and g_4 are done in the completely unfolded table B_W (not even folded over true). In this way, the negations introduced by the folding only ever appear on the queries for f, g_1, and g_2. Since these queries appear linearly in the test of the verifier, any negations added by the folding can be added up and moved to the right hand side of the test. Thus, the queries of the verifier are indeed all of the form $f_{\mathrm{TSA}}(\mathbf{x}) = 1$ or $f_{\mathrm{TSA}}(\mathbf{x}) = 0$ as desired. As usual, Fact 3.3 implies that the reduction given by this PCP verifier is factor graph-preserving. The fact that this PCP has completeness 1 is immediate and again the proof is near-identical to the proof of Lemma 4.4 so we omit it.

Lemma 4.14. *The PCP from Definition 4.13 has completeness* 1.

Next we prove soundness.

Lemma 4.15. *Let $\delta > 0$ and set $C_\delta = \log(\frac{2}{\delta})$. For $r > 0$ set $t = r\frac{4}{\delta^2}C_\delta$. If the PCP verifier from Definition 4.13 accepts with probability at least $\frac{1+\delta}{2}$, then there is a strategy in the original parallel game that makes the verifier accept with probability at least $\frac{\delta^4}{16C_\delta}$.*

Proof. To emphasize the difference from the folded tables, we write $C_W = B_W$ throughout this proof for the unfolded table B_W. If the verifier accepts with probability at least $\frac{1+\delta}{2}$, then

$$\begin{aligned}\delta &= \mathbb{E}\left[(-1)^{A^{\boxplus}_U(f)}(-1)^{B^{\boxplus}_W(g_1)}(-1)^{B^{\boxplus}_W(g_2)}(-1)^{C_W(g_3)\wedge C_W(g_4)}\right]\\ &= \mathop{\mathbb{E}}_{U,W}\left[\sum_{\alpha,\beta_1,\beta_2}\hat{A}^{\boxplus}_U(\alpha)\hat{B}^{\boxplus}_W(\beta_1)\hat{B}^{\boxplus}_W(\beta_2)\,\mathbb{E}\left[\chi_\alpha(f)\chi_{\beta_1}(g_1)\chi_{\beta_2}(g_2)(-1)^{C_W(g_3)\wedge C_W(g_4)}\right]\right]\end{aligned}$$

$$= \mathop{\mathbb{E}}_{U,W}\left[\sum_{\alpha,\beta_1,\beta_2} \hat{A}^{\boxplus}_U(\alpha)\hat{B}^{\boxplus}_W(\beta_1)\hat{B}^{\boxplus}_W(\beta_2)\,\mathbb{E}\left[\chi_\alpha(f)\chi_{\beta_1}(g_1)\chi_{\beta_2}(f+g_1+g_3\wedge g_4)(-1)^{C_W(g_3)\wedge C_W(g_4)}\right]\right]$$

$$= \mathop{\mathbb{E}}_{U,W}\left[\sum_{\beta} \hat{A}^{\boxplus}_U(\beta_{\oplus U})\hat{B}^{\boxplus}_W(\beta)^2 \mathop{\mathbb{E}}_{g_3,g_4}\left[\chi_\beta(g_3\wedge g_4)(-1)^{C_W(g_3)\wedge C_W(g_4)}\right]\right].$$

Our goal is now to show that the inner expectation over g_3 and g_4 is bounded by a function tending to zero with $|\beta|$. First observe that

$$\mathop{\mathbb{E}}_{g_3,g_4}\left[\chi_\beta(g_3\wedge g_4)(-1)^{C_W(g_3)\wedge C_W(g_4)}\right]$$
$$= \frac{1}{4}\mathop{\mathbb{E}}_{g_3,g_4}\left[\chi_\beta(g_3\wedge g_4)(1-(-1)^{C_W(g_3)}-(-1)^{C_W(g_4)}+(-1)^{C_W(g_3)}(-1)^{C_W(g_4)})\right].$$

Observe that by independence of g_3 and g_4,

$$\mathop{\mathbb{E}}_{g_3,g_4}\left[\chi_\beta(g_3\wedge g_4)\right] = \mathop{\mathbb{E}}_{g_3}\left[\prod_{y\in\beta}\mathop{\mathbb{E}}_{g_4}\left[(-1)^{g_3(y)g_4(y)}\right]\right] = 2^{-|\beta|}$$

since the product of expectations over g_4 is zero unless g_3 is identically zero on β. Similarly,

$$\left|\mathop{\mathbb{E}}_{g_3,g_4}\left[\chi_\beta(g_3\wedge g_4)(-1)^{C_W(g_3)}\right]\right| = \left|\mathop{\mathbb{E}}_{g_3}\left[(-1)^{C_W(g_3)}\prod_{\mathrm{y}\in\beta}\mathop{\mathbb{E}}_{g_4}\left[(-1)^{g_3(\mathrm{y})g_4(\mathrm{y})}\right]\right]\right| \le 2^{-|\beta|}$$

with the same inequality holding for $\mathbb{E}\left[\chi_\beta(g_3\wedge g_4)(-1)^{C_W(g_4)}\right]$ by symmetry. Therefore we conclude that

$$\left|\mathop{\mathbb{E}}_{g_3,g_4}\left[\chi_\beta(g_3\wedge g_4)(-1)^{C_W(g_3)\wedge C_W(g_4)}\right]\right| \le \frac{3}{4}\cdot 2^{-|\beta|} + \frac{1}{4}\left|\mathop{\mathbb{E}}_{g_3,g_4}\left[\chi_\beta(g_3\wedge g_4)(-1)^{C_W(g_3)}(-1)^{C_W(g_4)}\right]\right|.$$

Next we take the Fourier expansion of C_W in the above expectation and get

$$\mathop{\mathbb{E}}_{g_3,g_4}\left[\chi_\beta(g_3\wedge g_4)(-1)^{C_W(g_3)}(-1)^{C_W(g_4)}\right] = \sum_{\gamma_1,\gamma_2}\hat{C}_W(\gamma_1)\hat{C}_W(\gamma_2)\mathop{\mathbb{E}}_{g_3,g_4}\left[\chi_\beta(g_3\wedge g_4)\chi_{\gamma_1}(g_3)\chi_{\gamma_2}(g_4)\right].$$

Any term above with γ_1, γ_2 not both being subsets of β has expectation zero. Furthermore, for any choice of g_4 where $g_4 \wedge \mathbf{1}_\beta \neq \mathbf{1}_{\gamma_1}$, taking the expectation over g_3 gives us zero, while if $g_4 \wedge \mathbf{1}_\beta = \mathbf{1}_{\gamma_1}$ the expectation over g_3 equals 1. In the latter case, which happens with probability $2^{-|\beta|}$ over the choice of g_4, we have $\chi_{\gamma_2}(g_4) = \chi_{\gamma_2}(\mathbf{1}_{\gamma_1})$ and thus we are left with

$$2^{-|\beta|}\sum_{\gamma_1,\gamma_2\subseteq\beta}\hat{C}_W(\gamma_1)\hat{C}_W(\gamma_2)\chi_{\gamma_2}(\mathbf{1}_{\gamma_1}). \tag{4.17}$$

Now observe that for any fixed γ_1,

$$\left|\sum_{\gamma_2\subseteq\beta}\hat{C}_W(\gamma_2)\chi_{\gamma_2}(\mathbf{1}_{\gamma_1})\right| \le 1,$$

because the sum equals the expected value of $(-1)^{C_W(g)}$ over all $g \in \mathcal{F}_W$ which agree with $\mathbf{1}_{\gamma_1}$ on β. Therefore we can bound (4.17) by

$$2^{-|\beta|}\sum_{\gamma_1\subseteq\beta}|\hat{C}_W(\gamma_1)| \le 2^{-|\beta|/2}\sum_{\gamma_1\subseteq\beta}\hat{C}_W(\gamma_1)^2 \le 2^{-|\beta|/2}.$$

So we conclude that

$$\left|\mathop{\mathbb{E}}_{g_3,g_4}\left[\chi_\beta(g_3 \wedge g_4)(-1)^{C_W(g_3)\wedge C_W(g_4)}\right]\right| \leq \frac{3}{4}\cdot 2^{-|\beta|} + \frac{1}{4}\cdot 2^{-|\beta|/2} < 2^{-|\beta|/2}.$$

Returning to our original formula for the acceptance probability we have by Cauchy-Schwartz and Plancherel

$$\delta^2 \leq \mathop{\mathbb{E}}_{U,W}\left[\left(\sum_\beta \left|\hat{A}_U^{\boxplus}(\beta_{\oplus U})\hat{B}_W^{\boxplus}(\beta)^2 2^{-|\beta|/2}\right|\right)^2\right] \leq \mathop{\mathbb{E}}_{U,W}\left[\sum_\beta \hat{A}_U^{\boxplus}(\beta_{\oplus U})^2\hat{B}_W^{\boxplus}(\beta)^2 2^{-|\beta|}\right]. \tag{4.18}$$

This is essentially the same as the bound (4.1) we obtained in Lemma 4.5 for Max-3-Lin, except with $2^{-|\beta|}$ in place of $(1-2\varepsilon)^{2|\beta|}$. We can now use exactly the same strategies for the provers as in that proof and succeed in the smooth parallel repeated game with probability at least $\frac{\delta^4}{16C_\delta}$. □

4.4 Max-Not-2 with perfect completeness

In this section we look at the predicate "Not-2" which is a predicate of arity 3 that accepts iff the input does not contain two bits that are one. As this is implied by the negation of the 3-Lin predicate, it follows from the proof of Theorem 4.2 that it is UFG-NP-hard to approximate Max-Not-2 within any constant greater than $\frac{5}{8}$. We are, however, interested in perfect completeness and following in the footsteps of [16] we establish the following theorem.

Theorem 4.16. *For any $\varepsilon > 0$, Max-Not-2 is $(1, 5/8+\varepsilon)$-UFG-NP-hard.*

It turns out that we only need minor modifications to the argument of [16]. Already that paper uses smooth label cover and we start by defining the underlying basic PCPs.

Definition 4.17. The NTW_δ PCP verifier does the following:

(1) Pick a random pair of sets (U, W) sent to the two provers in the parallel game.
(2) Pick a uniform random function $f \in \mathcal{F}_U$ and a uniform random $g_1 \in \mathcal{F}_W$.
(3) Define $g_2 \in \mathcal{F}_W$ by setting $g_2(\mathbf{y}) = 1 + g_1(\mathbf{y}) + f(\mathbf{y}_U)$ for all y.
(4) For each $\mathbf{x} \in \{0,1\}^U$ with probability δ pick one $\mathbf{y}$ such that $\mathbf{y}_U = \mathbf{x}$ and reassign $g_2(\mathbf{y})$ and $g_1(\mathbf{y})$ to the value $f(\mathbf{x})$.
(5) Accept if and only if not two of the bits $A_U^{\boxplus}(f)$, $B_W^{\boxplus}(g_1)$, and $B_W^{\boxplus}(g_2)$ are one.

The parameter d which, for any x, bounds the number of $\mathbf{y}$ such $\mathbf{y}_U = x$ is crucial in [16] and let us use it also here. Note that d depends on r but not on t. For each δ [16] defines two parameters $s_\delta = O(\log(1/\delta)/\log d)$, and $S_\delta = O(\log(1/\delta)d^3 2^{2d}\delta^{-2})$. The final PCP, $NTW_{\delta'}^k$ is now as follows:

Definition 4.18. The $NTW_{\delta'}^k$ PCP verifier does the following:

(1) Set $\delta_0 = \delta'$ and for $i = 1, \ldots k-1$ choose δ_i such that $s_{\delta_i} = S_{\delta_{i-1}}$.
(2) Pick a random $i \in [k]$ uniformly at random and run NTW_{δ_i}.

First note that, as s_δ tends to infinity when δ tends to 0, we do get a well defined sequence δ_i. We can also observe that $\log(1/\delta_k)$ is a constant that only depends on d, k and δ' and that it is bounded by a tower of exponentials of height around k.

The analysis of [16] is quite involved but the conclusion is that given any proof accepted by the verifier with probability $\frac{5}{8}+\varepsilon$ then, in our notation, it is possible to conclude that

$$\mathop{\mathbb{E}}_{U,W}\left[\sum_\beta \hat{A}_U(\beta_U)^2\hat{B}_W(\beta)^2(1-\gamma)^{|\beta|}\right] \geq \varepsilon^2. \tag{4.19}$$

The reasoning requires that k is sufficiently large compared to ε and δ' is sufficiently small compared to ε. The constant γ also depends on ε and δ', but also on the constant d (and hence on our constant r). A key fact is that γ does not depend on t. Following [16] we now extract a strategy for the basic two-prover game with $t = 0$.

First we find a bound $T(\delta', \varepsilon, r)$ such that if restrict the sum (4.19) to β of size at most $T(\delta', \varepsilon, r)$ then it remains at least $\varepsilon^2/2$. Now by choosing $t = t(\varepsilon, \delta', r)$ large enough we can assume two more properties of the elements in the sum while still keeping it large.

The first property, also used in [16], and there described as "β is shattered" is the property of Claim 2.20, namely, that any two elements in β project onto different elements. The second property is automatic in [16] because there conditioning is used instead of functional folding. In particular, we require that for any $j \in \beta$, this assignment satisfies all the $t + r$ constraints on W if and only if j_U satisfies all the constraints of U. Clearly, we can find t large enough so that (4.19) remains at least $\varepsilon^2/4$ when restricted to terms satisfying these two properties.

Finally we can find a set C of t constraints sent to both players such that sum (4.19) (of course with all the restrictions just mentioned), conditioned on these t constraints being the t constraints sent to both players, is at least $\varepsilon^2/4$. This follows by an averaging argument over the random choice of U and W.

Now we can define a strategy for the provers in the basic two-prover game with a given value of r and $t = 0$. Each prover adds the fixed constraint set C to its question and finds the corresponding tables A_U and B_W. Now the strategy is essentially the standard strategy.

- P_1 picks a set α with probability $\hat{A}_U(\alpha)^2$ and looks at all elements of α that satisfy all constraints on U. Of these it picks the element that is lexicographically first when considered only on the common variables of the fixed constraints C.
- P_2 picks a set β with probability $\hat{B}_W(\beta)^2$ and looks at all elements of β that satisfies all constraints on W. Of these it picks the element that is lexicographically first when considered only on the common variables.

This strategy succeeds with a probability $\varepsilon^2/4$ and in particular it is independent of r. As a final step we pick r large enough (and $t = t(\varepsilon, \delta', r)$ as discussed above) to make this strategy violate the soundness of the two-prover game.

5 PAIRWISE INDEPENDENCE AND HADAMARD PREDICATES

In this section we establish that the results of Chan [7] can be obtained with a universal factor graph. Chan showed that any predicate supporting a pairwise independent subgroup is approximation resistant. In fact, he even showed that such predicates satisfy a strong property called uselessness, introduced by Austrin and Håstad [3].

Definition 5.1. The predicate $f : \{0,1\}^k \longrightarrow \{0,1\}$ is *useless for a set of functions* $G = \{g : \{0,1\}^k \longrightarrow \mathbb{R}\}$ if for every $\varepsilon > 0$, the following promise decision problem is NP-hard. Given a Max-CSP($f^{\pm}$) instance $I = (X, C)$, distinguish between

(1) (Yes) I is $(1-\varepsilon)$-satisfiable.
(2) (No) for every $g \in G$ and assignment $\mathbf{a} : X \longrightarrow \{0,1\}$ it holds that

$$\left| \mathop{\mathbb{E}}_{(S,\mathbf{b})\sim C}[g(\mathbf{a}_S + \mathbf{b})] - \mathop{\mathbb{E}}_{\mathbf{u}\sim\{0,1\}^k}[g(\mathbf{u})] \right| \le \varepsilon.$$

If G is the set of all functions on k bits then we say that f is *universally useless* (or simply *useless*, if there is no risk of ambiguity). If the uselessness is established by a factor-graph preserving reduction from any problem that is UFG-NP-hard then we say that f is *UFG-useless* (for G).

Note that approximation resistance is the property that f is useless for the set of functions $G = \{f\}$ consisting only of f itself. Also observe that (universal) uselessness is equivalent to being useless for the set G consisting of all 2^k parity functions on k bits (since any $g : \{0,1\}^k \to \{0,1\}$ is a linear combination over the reals of such functions). From this observation it is not hard to see that the proofs of approximation resistance for Max-3-Sat and Max-3-Lin in Section 4 in fact also establish universal uselessness. E.g. in the reduction of Max-3-Sat, we have in the soundness case:

(1) Claim 4.11 establishes that the parity of all three variables is within $\delta/4 + t$ of being unbiased, where t is a quantity that is small unless we are able to decode the Long Codes to a good strategy of the parallel repeated game.
(2) Claim 4.10 establishes that the parity of the two variables read from the $B^{\boxplus}_W$ table is within $\delta/4$ of being unbiased.
(3) The observations following (4.3) establish that the remaining five non-constant parities are completely unbiased.

The case of Max-3-Lin is easier, and as it also is a special case of the main topic of this section, namely Theorem 5.3 below, we omit the details.

The class of predicates for which Chan's results hold are those supporting a pairwise independent subgroup.

Definition 5.2. A predicate $f : \{0,1\}^k \to \{0,1\}$ *supports a pairwise independent subgroup* if there exists a subgroup $H \subseteq f^{-1}(1)$ where the group operation is coordinate-wise addition modulo 2, and a distribution μ supported on H such that for all pairs $i, j \in [k]$, the joint probability distribution of x_i, x_j for $x \sim \mu$ is equal to the uniform distribution on $\{0,1\}^2$.

The main result of Chan, that we establish with a universal factor graph, can be stated as follows.

Theorem 5.3 ([7], Theorem 1.1, with a universal factor graph). *Let $f : \{0,1\}^k \to \{0,1\}$ be any predicate that supports a pairwise independent subgroup of $\{0,1\}^k$ (under the group operation of addition mod 2). Then f is UFG-useless.*

As the Hadamard predicates Had_k support a pairwise independent subgroup, an immediate and often used corollary is the following.

Corollary 5.4. *For any $\varepsilon > 0$, Max-Had_k is $(1-\varepsilon, (k+1)2^{-k} + \varepsilon)$-UFG-NP-hard.*

5.1 Analytic Notation, Influences, and Noise

For the purposes of proving Chan's result with universal factor graphs, it turns out to be more notationally convenient to use the $\{-1,1\}$ domain for Fourier analysis. Therefore, just for this section, we make the following notational changes. If I is a set of coordinates $\mathcal{F}_I$ denotes the set of functions $f : \{0,1\}^I \to \{-1,1\}$, and we use $\mathcal{F}_n$ to denote $\mathcal{F}_{[n]}$ as before. All of the long code tables A will be functions $A : \mathcal{F}_n \to \{-1,1\}$. Finally, for $\alpha \subseteq \{0,1\}^n$ the Fourier character $\chi_\alpha : \mathcal{F}_n \to \{-1,1\}$ will be given by

$$\chi_\alpha(f) = \prod_{\mathbf{x} \in \alpha} f(\mathbf{x}).$$

With this notation set up, we additionally need to define influences and the noise operator.

Definition 5.5. For a function $A : \mathcal{F}_n \to \mathbb{R}$ and set $\mathcal{B} \subseteq \{0,1\}^n$ of coordinates, the *influence* of $\mathcal{B}$ is

$$\text{Inf}_{\mathcal{B}}(A) = \sum_{\alpha \cap \mathcal{B} \neq \emptyset} \hat{A}(\alpha)^2.$$

For a single coordinate $\mathbf{x} \in \{0,1\}^n$ we write $\text{Inf}_{\mathbf{x}}(A)$ for $\text{Inf}_{\{\mathbf{x}\}}(A)$.

Fact 5.6. $\mathrm{Inf}_{\mathcal{B}}(A) \le \sum_{\mathbf{x}\in\mathcal{B}} \mathrm{Inf}_{\mathbf{x}}(A)$

Definition 5.7. For a noise rate $0 \le \eta \le 1$ the noise operator $T_{1-\eta}$ maps functions $A : \mathcal{F}_n \to \mathbb{R}$, to noisy functions $T_{1-\eta}A : \mathcal{F}_n \to \mathbb{R}$ defined by

$$T_{1-\eta}A(f) = \mathop{\mathbb{E}}_{\tilde{f}\sim_{1-\eta} f}[A(\tilde{f})],$$

where $\tilde{f} \sim_{1-\eta} f$ indicates that $\tilde{f}(\mathbf{x})$ is chosen as $f(\mathbf{x})$ with probability $1-\eta$, and as a uniformly random bit with probability η, independently for each $\mathbf{x} \in \{0,1\}^n$.

Fact 5.8. *For every $\eta > 0$ and every table $A : \mathcal{F}_n \to \mathbb{R}$,*

$$\sum_{\mathbf{x}\in\{0,1\}^n} \mathrm{Inf}_{\mathbf{x}}(T_{1-\eta}A) \le 1/\eta.$$

5.2 Overview

As with other results, we very much follow in the footsteps of the original proof. Given that Chan's proof is rather long we do not repeat the entire argument here. We only recall some crucial details and describe how to modify them in our setting. The main difference is, not surprisingly, that [7] uses conditioning and this leads both to a simpler proof and the possibility to use simpler notation. To keep notation here simpler, we only present the arguments for the concrete case of Hadamard predicates, but they generalize easily.

The high level view of Chan's proof is what can be expected. He starts with an instance of label cover with very good soundness. To get better numerical dependencies Chan uses a different starting point, but let us here assume that the starting point is the r-fold parallel repetition game described in Section 2.4. He then produces a PCP whose acceptance condition is given by Had_k and proves that whenever there is a PCP proof where some function g exceeds its expectation on the answers to a random query, this can be used to derive successful strategies in the two-prover game.

It turns out to be difficult to directly define a PCP where every g has small expectation. An easier task is to define a PCP where all characters $\psi : \{-1,1\}^k \to \{-1,1\}$ that are *j-relevant* for some fixed $j \in [k]$ have small expectation. In the Boolean setting, each character is simply a product $\psi(b_1,\ldots,b_k) = \prod_{i\in S} b_i$ for some $S \subseteq [k]$, and ψ is j-relevant if $j \in S$.

We have the following theorem.

Theorem 5.9 ([7], Theorem 5.4, with a universal factor graph). *For every $j \in [k]$, Had_k is UFG-useless for the set of all j-relevant characters $\psi : \{-1,1\}^k \to \{-1,1\}$.*

The proof of this theorem is the main technical part of Chan's work, and it is also the part that needs modifications in our setting with functional folding in lieu of conditioning. We describe these modifications and the proof in Section 5.3 below.

When we have Theorem 5.9, we can combine it with the very powerful construction, discovered by Chan, of taking the direct sum of instances.

Definition 5.10. Given two Max-CSP($f^{\pm}$) instances $I = (X, C)$ and $I' = (X', C')$, their *direct sum* is defined as $I \oplus I' = (X \times X', C \oplus C')$. For each constraint $f(\mathbf{x}_S + \mathbf{b}) = 1$ in C and each constraint $f(\mathbf{x}'_{S'} + \mathbf{b}') = 1$ in C', we have the constraint $f(\mathbf{x}''_{S\circledcirc S'} + \mathbf{b} + \mathbf{b}') = 1$ in $C \oplus C'$, where for two tuples $\mathbf{u} = (u_1,\ldots,u_k)$ and $\mathbf{v} = (v_1,\ldots,v_k)$ we write $\mathbf{u} \circledcirc \mathbf{v}$ to denote coordinate-wise concatenation of $\mathbf{u}$ and $\mathbf{v}$, i.e., the tuple of pairs $((u_1,v_1),\ldots,(u_k,v_k))$.

As shown by Chan ([7], Lemma 5.3), taking direct sum preserves uselessness for characters (if either I or I' satisfies the "No" case of Definition 5.1 with respect to some character ψ then $I \oplus I'$ does as well). Thus, since the characters form an orthonormal basis for all functions $g : \{-1,1\}^k \to \mathbb{R}$, taking the direct sum of the k instances arising from Theorem 5.9 and making the following observation we obtain Theorem 5.3.

Observation 5.11. The factor graph of $I \oplus I'$ depends only on the factor graphs of I and I', and not on the negation patterns in I and I'.

5.3 Protocol For a Single Coordinate

In this section we sketch Chan's proof of Theorem 5.9 and the modifications needed to make it hold with a universal factor graph.

Throughout this section, fix the value of the index $j \in [k]$, and let $J = [k] \setminus \{j\}$, i.e., all elements except j. Let $\eta > 0$ be a small parameter to be chosen later.

Given a Max-TSA instance I we construct a new set of variables X' as in the reduction template Definition 3.5, and construct the following PCP verifier.

(1) Pick a random pair of sets (U, W) sent to the two provers in the parallel game.
(2) Pick a uniformly random function $f \in \mathcal{F}_U$.
(3) For $i \in J$ and $\mathbf{y} \in \{0,1\}^W$ choose $g_i(\mathbf{y})$ uniformly at random subject to condition that the string $(g_i(\mathbf{y}))_{i\in J}$ with $f(\mathbf{y}_U)$ inserted in position j satisfies Had_k.
(4) Let $\tilde{f}$ and $\tilde{g}_i$ be η-noisy perturbations of f and g_i.
(5) Accept if and only if $(B^{\boxplus}_W(\tilde{g}_i))_{i\in J}$ with $A^{\boxplus}_U(\tilde{f})$ inserted in the jth position satisfies Had_k.

Remark 5.12. Let us briefly compare the notation used here to the notation used in Chan's protocol. The following table shows the notation used for the main objects.

	j'th coordinate					i'th coordinate for $i \neq j$				
Chan's notation	f_j	$\tilde{f}_j$	g_j	$\mathbf{z}^{(j)}$	$\mathbf{z}^{(j)}_t$	f_i	$\tilde{f}_i$	g_i	$\mathbf{z}^{(i)}$	$\mathbf{z}^{(i)}_s$
Our notation	A_U	$A^{\boxplus}_U$	$T_{1-\eta}A^{\boxplus}_U$	f	$f(\mathbf{x})$	B_W	$B^{\boxplus}_W$	$T_{1-\eta}B^{\boxplus}_W$	g_i	$g_i(\mathbf{y})$

Note in particular that while the g_i's in Chan's protocol are the purported long codes with η-noise applied, the g_i's in our protocol are the inputs to the purported long codes on the W side.

The completeness analysis of the above protocol is easy and is not affected by the modifications we have made, so let us turn to the soundness analysis.

Fix some j-relevant character ψ and suppose that the expectation of ψ over the answers to the provers deviates from its expectation (0) by at least ε, i.e.,

$$\left| \mathop{\mathbb{E}}_{(U,W)} \left[\mathop{\mathbb{E}}_{\tilde{f}, \{\tilde{g}_i\}_{i\in J}} [\psi(A^{\boxplus}_U(\tilde{f}), \{B^{\boxplus}_W(\tilde{g}_i)\}_{i\in J})] \right] \right| > \varepsilon.$$

By Markov's inequality, it holds that for at least an $\varepsilon/2$ fraction of all query pairs (U, W), the inner expectation is at least $\varepsilon/2$ in absolute value. Fix one such "good" pair (U, W) and to simplify notation let $A(f) = T_{1-\eta}A^{\boxplus}_U$ and $B(g) = T_{1-\eta}B^{\boxplus}_W$. Thus we have

$$\left| \mathop{\mathbb{E}}_{f, \{g_i\}_{i\in J}} [\psi(A(f), \{B(g_i)\}_{i\in J})] \right| > \varepsilon/2. \tag{5.1}$$

For $\mathbf{x} \in \{0,1\}^U$, define $\mathcal{B}(\mathbf{x})$ to be the set of $\mathbf{y} \in \{0,1\}^W$ such that $\mathbf{y}_U = \mathbf{x}$ (in Chan's notation, this is the "block" $B(t)$). The key quantity to study is

$$\sum_{\mathbf{x}} \mathrm{Inf}_{\mathbf{x}}(A)\, \mathrm{Inf}_{\mathcal{B}(\mathbf{x})}(B), \tag{5.2}$$

which measures the presence of common influences between the noised tables A and B. Using an invariance-style proof, it is shown that when (5.1) holds then (5.2) must also be large. Concretely we have the following theorem.

Theorem 5.13 ([7], Theorem 6.7). *In the notation above, let $\mathcal{Z} \subseteq \{0,1\}^U$ be any set of assignments such that*

$$\sum_{\mathbf{x} \notin \mathcal{Z}} \mathrm{Inf}_{\mathbf{x}}(A)\,\mathrm{Inf}_{\mathcal{B}(\mathbf{x})}(B) \leq \tau,$$

and define $A^{\mathcal{Z}} : \mathcal{F}_U \longrightarrow [0,1]$ to be the part of A depending only on $\mathcal{Z}$.[3] *Then*

$$\left| \mathop{\mathbb{E}}_{f, \{g_i\}_{i \in J}} [\psi(A(f), \{B(g_i)\}_{i \in J})] \right| \leq \left| \mathop{\mathbb{E}}_{f, \{g_i\}_{i \in J}} [\psi(A^{\mathcal{Z}}(f), \{B(g_i)\}_{i \in J})] \right| + \delta(k, \eta, \tau)$$

where for every fixed k and η, $\delta(k,\eta,\tau)$ tends to 0 as τ tends to 0.

This is a theorem purely about analysis of Boolean functions and its proof relies only on the pairwise independence of the underlying CSP and not on the structure of the parallel repeated game, and as such it applies without modification in our setting. Chan only states the theorem for $\mathcal{Z} = \emptyset$ but inspection of the proof, which is based on rerandomizing one coordinate at a time, reveals that it holds for any $\mathcal{Z}$.

In Chan's original proof, the case $\mathcal{Z} = \emptyset$ is all that is needed, since combined with (5.1) it lets us conclude that the tables A and B have shared influential coordinates, which can then immediately be used in a standard way to define strategies that are accepted with constant probability in the parallel repeated game.

In our setting it is not á priori clear that this yields a good strategy, since the functional folding might not guarantee that any influential coordinate of A or B actually satisfies the constraints. However, as we shall now see, it turns out that this is indeed the case, so the same strategy does work also in our setting[4].

Since A and B are tables with noise applied, it follows from Fact 5.8 and Fact 5.6 that if we let $\mathcal{Z}$ be the set of assignments $\mathbf{x}$ such that both

$$\mathrm{Inf}_{\mathbf{x}}(A) \geq \tau\eta/2 \qquad \text{and} \qquad \mathrm{Inf}_{\mathcal{B}(\mathbf{x})}(B) \geq \tau\eta/2 \tag{5.3}$$

then

$$\sum_{\mathbf{x} \notin \mathcal{Z}} \mathrm{Inf}_{\mathbf{x}}(A)\,\mathrm{Inf}_{\mathcal{B}(\mathbf{x})}(B) \leq \tau.$$

Choosing τ small enough so that $\delta(k,\eta,\tau) < \varepsilon/2$, it follows from Theorem 5.13 and (5.1) that

$$\left| \mathop{\mathbb{E}}_{f, \{g_i\}_{i \in J}} [\psi(A^{\mathcal{Z}}(f), \{B(g_i)\}_{i \in J})] \right| > 0 \tag{5.4}$$

We now use the functional folding, and have the following observation.

Claim 5.14. If no $\mathbf{x} \in \mathcal{Z}$ satisfies all equations $h_i(\mathbf{x}) = b_i$ for $i = 1, \ldots, t$, then $A^{\mathcal{Z}}$ is identically 0.

Proof. By Lemma 3.4 each non-zero Fourier coefficient of A contains an element that satisfies all the constraints. If we rerandomize all these values then the expectation is zero. □

Since ψ is a j-relevant character, the left hand side of (5.4) would be 0 if $A^{\mathcal{Z}}$ was identically 0, so it follows that there must be some $\mathbf{x}^* \in \mathcal{Z}$ which satisfies all equations on U.

We now define the strategies for the provers in the repeated game in a standard way: we choose answer $\mathbf{x}$ for U with probability proportional to $\mathrm{Inf}_{\mathbf{x}}(A)$ and, independently, analogously for W. By Fact 5.8, $\mathbf{x}$ and $\mathbf{y}$ are chosen with probabilities at least $\eta\,\mathrm{Inf}_{\mathbf{x}}(A)$ and $\eta\,\mathrm{Inf}_{\mathbf{y}}(B)$, so the probability

[3] Equivalently, $A^{\mathcal{Z}}(f)$ is the expectation of $A(\overline{f})$ on a copy of f where all coordinates outside $\mathcal{Z}$ have been rerandomized.

[4] Of course, there is no reason why the provers would ever output a value that does not satisfy the relevant constraints but it is slightly easier to analyze this variant of their strategies.

that the good assignment $\mathbf{x}^*$ is chosen and is consistent with the answer of the other prover is at least

$$\eta \operatorname{Inf}_{\mathbf{x}^*}(A) \sum_{\mathbf{y} \in \mathcal{B}(\mathbf{x}^*)} \eta \operatorname{Inf}_{\mathbf{y}}(B) \geq \eta^2 \operatorname{Inf}_{\mathbf{x}^*}(A) \operatorname{Inf}_{\mathcal{B}(\mathbf{x}^*)}(B) \geq \eta^4 \tau^2/4.$$

Aggregating this over the $\varepsilon/2$ fraction of good pairs (U, W) of queries in the repeated game, we conclude that in expectation a fraction $\varepsilon\eta^4\tau^2/8$ of all query pairs are assigned answers that are consistent and where all constraints on U are satisfied. The only remaining issue is to establish that the answers of the other prover often satisfy the additional constraints on W.

Fix any assignment to the variables in W which does not satisfy all constraints. By Claim 2.20 the probability, over the choice of U, that it projects to an assignment that satisfies all constraints on this smaller set is bounded by $r/(r+t)$. It follows that if we choose $t \geq 16r\varepsilon^{-1}\eta^{-4}\tau^{-2}$ then the total expected fraction of query pairs where the assignments are consistent and satisfy the constraints on U but not those on W is bounded by $\frac{r}{r+t} \leq \varepsilon\eta^4\tau^2/16$. We conclude that in expectation the influence-based random strategy wins the repeated game with probability at least $\varepsilon\eta^4\tau^2/16$. This concludes the description of the modifications of the proof Chan and our proof of Theorem 5.9.

6 Max-K-CSP WITH PERFECT COMPLETENESS

In this section we establish that the results of Huang [17] can be obtained with a universal factor graph, showing that it is hard to approximate the Max-K-CSP problem within a factor $K^{cK^{1/3}}/2^K$ for some constant c, even on satisfiable instances. More concretely, Huang, building upon the work of Chan discussed in Section 5, showed that for infinitely many K there exists a K-ary predicate $P : \{0,1\}^K \to \{0,1\}$ with $|P^{-1}(1)| = 2^{O(K^{1/3}\log K)}$ accepting assignments such that the Max-CSP($P^{\pm}$) is approximation resistant on satisfiable instances, i.e., it is NP-hard to distinguish satisfiable instances from those where with value $|P^{-1}(1)|/2^K$.

In this section we show how his proof carries over to the UFG setting. In order to do this, we need to somewhat generalize the reduction framework used in the previous results. In particular, rather than employing the standard label cover problem (derived from parallel repetition of the basic two-prover game), we need a multilayered label cover (derived from parallel repetition of a k-prover game).

As these results build upon the hardness based on pairwise independence described in Section 5, we reuse the notational conventions and concepts introduced there.

6.1 The Predicate

Fix a parameter k. Let $\mathcal{S}_i = \binom{[k]}{i}$ be the family of all size-i subsets of $[k]$.

The base predicate we are interested in is $P_0 : \{-1,1\}^{\mathcal{S}_1 \cup \mathcal{S}_3} \to \{-1,1\}$ which accepts an input $(x_S)_{S \in \mathcal{S}_1 \cup \mathcal{S}_3}$ if and only if

$$x_{ijk} = x_i \cdot x_j \cdot x_k$$

for all $\{i,j,k\} \in \mathcal{S}_3$. The arity of P_0 is $K := k + \binom{k}{3} = \Theta(k^3)$, and P_0 has exactly 2^k satisfying assignments.

The actual predicate that we prove is hard with perfect completeness is the predicate $P : \{-1,1\}^{\mathcal{S}_1 \cup \mathcal{S}_3} \to \{-1,1\}$ which accepts an input $\mathbf{x}$ if and only if $\mathbf{x}$ is within Hamming distance at most k of a satisfying assignment to P_0. Note that the number of satisfying assignments to P is at most $\binom{k+\binom{k}{3}}{k} \cdot 2^k = 2^{O(k\log k)}$.

To that end we also consider the predicate P_1, accepting the set of inputs that are at Hamming distance 1 away from P_0. The structure of the proof is that we construct a PCP with perfect completeness for P_1 and soundness against a certain family of characters. This "certain family" is such that we can cover all characters using k such families and thus taking the direct sum of the k resulting PCPs we get a new PCP which has perfect completeness for P, and soundness against all characters.

6.2 The Parallel Repetition

For this result we need a k-layered smooth label cover. For parameters $0 = r_1 \leq r_2 \leq \ldots \leq r_k$ and t we have the following k-prover verifier, starting from a MaxTSA instance (or any other equational MaxCSP which is hard with perfect completeness).

(1) The verifier picks $r_k + t$ random equations with scopes $S_1, \ldots, S_{r_k+t}$ uniformly at random.
(2) For $j = 1, \ldots, r_k$, choose a variable $x_{i_j} \in \mathbf{x}_{S_{t+j}}$ uniformly at random.
(3) For $i = 1, \ldots, k$, send the list of variables $(\mathbf{x}_{S_j})_{j=1}^{t+r_i}$ and $(x_{i_j})_{r_i+1}^{r_k}$ in random order to the ith prover.
(4) Receive from each prover an assignment to the variables sent to that prover. Check that
   - for each prover i and each $j = 1, \ldots t + r_i$, the assignment $\mathbf{a}_{S_j}$ given by P_1 to $\mathbf{x}_{S_j}$ satisfies the corresponding equation, and
   - for each variable, all provers providing a value for that variable gave the same value.

The soundness of this protocol is that if *any pair* (i, j) of provers provide consistent and acceptable answers with non-negligible (as a function of the parameters t and $\mathbf{r}$) then we can find a good assignment to the underlying MaxTSA instance.

We also have smoothness between all pairs of layers analogous to Claim 2.20.

As in our standard setup used previously, we will do a reduction where the answers from the provers are long-coded and functionally folded. We shall denote by U_i the set of variables sent to prover P_i (note that $U_1 \subseteq U_2 \subseteq \ldots \subseteq U_k$), and by $A_{U_i} : \mathcal{F}_{U_i} \to \{-1, 1\}$ the corresponding purported long code which we take to be functionally folded over the equations induced by U_i.

6.3 The Long Code Test

For sets of variables $U_1 \subseteq \ldots \subseteq U_k$ sent to the k provers, we define the distribution $\mathcal{T} := \mathcal{T}(U_1, \ldots, U_k)$ over vectors of functions $(f_S)_{S \in \mathcal{S}_1 \cup \mathcal{S}_3}$ with $f_S \in \mathcal{F}_{U_{\max S}}$ as follows.

(1) For $i \in [k]$ pick $f_i^0 \in \mathcal{F}_{U_i}$ uniformly at random.
(2) For $\{i, j, k\} \in \mathcal{S}_3$, let $f_{ijk}^0 \in \mathcal{F}_{U_{\max(ijk)}}$ be defined by $f_{ijk}^0(\mathbf{x}) = f_i^0(\mathbf{x}_{U_i}) \cdot f_j^0(\mathbf{x}_{U_j}) \cdot f_k^0(\mathbf{x}_{U_k})$.
(3) Add "noise", by defining $\eta_S \in \mathcal{F}_{U_{\max S}}$ as follows. For each $\mathbf{x} \in \{0, 1\}^{U_1}$, pick one $S \in \mathcal{S}_1 \cup \mathcal{S}_3$ uniformly at random, and for all $\mathbf{y}$ with $\mathbf{y}_{U_1} = \mathbf{x}$, set $\eta_S(\mathbf{y})$ to a uniformly random bit. All other values of η_S are 1.
(4) For all S let $f_S = f_S^0 \odot \eta_S$ (coordinatewise multiplication).

The verifier acts by picking a random $U_1, \ldots, U_k$, sampling (f_S) according to $\mathcal{T}(U_1, \ldots, U_k)$ and querying $A_{U_{\max S}}(f_S)$ for all S, obtaining bits b_S.

The completeness is clear.

LEMMA 6.1. *If we start with a satisfiable MaxCSP instance then there is an assignment to the A_U's such that the $(b_S)_{S \in \mathcal{S}_1 \cup \mathcal{S}_3}$ satisfies P_1 with probability 1 over the randomness of the verifier.*

To state the soundness we need the following definition. Let ψ be some non-constant character over the variables of the predicate and let $\mathcal{S} \subseteq \mathcal{S}_1 \cup \mathcal{S}_3$ be the set of variables it depends on. For $j, \ell \in [k]$ we say that $\mathcal{S}$ (and ψ) is (j, ℓ)-*odd* if

$$|\{\, S \in \mathcal{S} \mid j \in S \wedge \max(S) = \ell \}|$$

is odd.

The key soundness lemma is (somewhat informally) as follows.

Lemma 6.2. *If the character ψ is (j,ℓ)-odd for some j,ℓ and the expected value of ψ on (b_S) is non-negligible then there is a strategy in the parallel repeated game.*

It was further shown in [17] that by taking all k cyclic permutations of the base set $[k]$, every character ψ becomes (j,ℓ)-odd for some j,ℓ and at least one of these permutations, so our final construction is to take the direct sum of those resulting PCPs. The details of this are unaffected in the UFG setting so we focus on the above lemma.

6.4 Soundness Analysis

Throughout this section, fix the character ψ and the set $\mathcal{S}$ of coordinates it depends on, and fix $j,\ell \in [k]$ to be the values such that $\mathcal{S}$ is (j,ℓ)-odd.

The first step is to add a bit more noise. In particular we define the distribution $\mathcal{T}' := \mathcal{T}'(U_1,\ldots,U_k)$ as the distribution obtained by sampling (f_S) according to $\mathcal{T}$ and then applying independent γ-noise to all bits. As shown by Huang ([17] Lemma 4.7), the expectation of ψ w.r.t. $\mathcal{T}$ differs from the expectation of ψ w.r.t. $\mathcal{T}'$ only by an error that can be made arbitrarily small as a function of γ. This bound relies on smoothness. Thus from now on we may think of the long code tables as being noised and having decaying tails.

As with Chan's result the idea of the soundness is based on rerandomizing some long code inputs to decouple the test distribution, and to show that this does not change the expectation by much unless there is noticeable shared influences. We will do the argument slightly differently from how we did it in Section 5. There we chose $\mathcal{Z}$ to be the set of all assignments that were influential in both long codes and then argued that this must contain a satisfying assignment.

In the present setting it seems a bit more convenient to do it the other way around. Concretely, let $\mathcal{Z} \subseteq \{0,1\}^{U_k}$ be the set of all assignments that satisfy all equations on U_1 (note, not U_k!). Note that $\mathcal{Z}_{U_i}$ contains the set of all assignments that satisfy all equations on U_i. We consider what happens when we rerandomize these coordinates of the f_S's. Informally, there are three things to establish:

(1) That if no consistent pair of assignments in $\mathcal{Z}_i$ have high influence in their corresponding tables then the rerandomization essentially does not change the expectation of ψ.
(2) That if we rerandomize these coordinates then the expectation of ψ on the (partially) decoupled distribution is 0.
(3) If some consistent pair of assignments have high influence then we can define a good strategy for the provers.

To simplify notation here let us fix a choice of $U_1,\ldots,U_k$ and write A_i instead of A_{U_i} and similarly $\mathcal{Z}_i$ instead of $\mathcal{Z}_{U_i}$.

Let $\mathcal{L}_i = \{S \in \mathcal{S} \mid \max(S) = i\}$. We can then write the expectation of ψ as

$$E := E(A_1,\ldots,A_k) = \mathop{\mathbb{E}}_{(f_S)\sim\mathcal{T}'}\left[\prod_{S\in\mathcal{S}} A_{\max S}(f_S)\right] = \mathop{\mathbb{E}}_{(f_S)\sim\mathcal{T}'}\left[\prod_{i=1}^{k}\prod_{S\in\mathcal{L}_i} A_i(f_S)\right].$$

We can write the rerandomized expectation as

$$E^{\mathcal{Z}} := \mathop{\mathbb{E}}_{(f_S)\sim\mathcal{T}'}\left[\prod_{i=1}^{k}\mathop{\mathbb{E}}_{(f'_S)\sim\mathcal{T}'}\left[\prod_{S\in\mathcal{L}_i} A_i(f'_S|_{\mathcal{Z}_i}, f_S|_{\overline{\mathcal{Z}_i}})\right]\right].$$

Here, the notation $A_i(f'_S|_{\mathcal{Z}_i}, f_S|_{\overline{\mathcal{Z}_i}})$ means the value $A_i(g)$, where $g : \{0,1\}^{U_i} \to \{-1,1\}$ is the function defined by

$$g(\mathbf{x}) = \begin{cases} f'_S(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{Z}_i \\ f_S(\mathbf{x}) & \text{otherwise} \end{cases}$$

Item 1 above is then formalized as follows.

Theorem 6.3. *For every choice of $\mathcal{Z} \subseteq \{0,1\}^{U_k}$ the following holds. If for all $1 \leq i < i' \leq k$ it holds that*

$$\sum_{\substack{\mathbf{x} \in \{0,1\}^{U_i} \\ \mathbf{y} \in \{0,1\}^{U_{i'}} \\ \mathbf{y}_{U_1} = \mathbf{x}_{U_1} \in \mathcal{Z}_{U_1}}} \mathrm{Inf}_{\mathbf{x}}(T_{1-\gamma} A_i)\, \mathrm{Inf}_{\mathbf{y}}(T_{1-\gamma} A_{i'}) \leq \varepsilon$$

then $|E - E^{\mathcal{Z}}| \leq \delta(\varepsilon, k, \gamma)$ where δ tends to 0 with ε.

[17] proves this for $\mathcal{Z} = \{0,1\}^{U_1}$ (i.e., all coordinates). In the next section we show that the general case can be reduced to that special case.

For item 2, we wish to establish the following, which is where functional folding comes to our aid.

Lemma 6.4. *If $\mathcal{Z}_\ell \subseteq \{0,1\}^{U_\ell}$ contains all assignments that satisfy all equations on U_ℓ then $E^{\mathcal{Z}} = 0$.*

Proof. Recall that

$$E^{\mathcal{Z}} = \mathop{\mathbb{E}}_{(f_S) \sim \mathcal{T}'} \left[\prod_{i=1}^{k} \mathop{\mathbb{E}}_{(f'_S) \sim \mathcal{T}'} \left[\prod_{S \in \mathcal{L}_i} A_i(f'_S|_{\mathcal{Z}_i}, f_S|_{\overline{\mathcal{Z}_i}}) \right] \right].$$

For any fixed choice of (f_S) in the outer expectation, let B_ℓ denote the restriction of A_ℓ under the partial assignment $f_S|_{\overline{\mathcal{Z}_i}}$, i.e., $B_\ell(f'_S) = A_\ell(f'_S|_{\mathcal{Z}_i}, f_S|_{\overline{\mathcal{Z}_i}})$ (but note that this is a table that depends only on the coordinates $\mathcal{Z}_\ell$). By the functional folding, we know that every non-zero Fourier coefficient α of A_ℓ contains an odd number of assignments from $\mathcal{Z}_\ell$. This implies that B_ℓ is an odd function.

Now consider the factor in the above expectation at $i = \ell$. By the definition of B_ℓ this equals

$$\mathop{\mathbb{E}}_{(f'_S) \sim \mathcal{T}'} \left[\prod_{S \in \mathcal{L}_\ell} B_\ell(f'_S) \right]. \tag{6.1}$$

By the (j, ℓ)-odd property, we know that an odd number of $S \in \mathcal{L}_\ell$ contain j. This, combined with the fact that B_ℓ is odd, implies that if in the sampling procedure for $\mathcal{T}$ we negate f_j^0 then $\prod_{S \in \mathcal{L}_\ell} B_\ell(f'_S)$ changes sign. Thus we conclude that (6.1) equals 0 and by extension $E^{\mathcal{Z}}$ does as well. □

There then only remains the third item on our list of tasks above, namely to use these properties to decode the long codes to good strategies in the k-prover game. This part works essentially the same way as in Section 5 so let us only outline the steps.

Recall that we let $\mathcal{Z} \subseteq \{0,1\}^{U_k}$ be the set of all assignments that satisfy all equations on U_1, and that for each $1 \leq i \leq k$, $\mathcal{Z}_{U_i}$ contains all assignments that satisfy all equations on U_i. By Theorem 6.3 applied to $\mathcal{Z}$ and Lemma 6.4, whenever our ψ has non-negligible expectation there must exist $i < i'$ such that

$$\sum_{\substack{\mathbf{x} \in \{0,1\}^{U_i} \\ \mathbf{y} \in \{0,1\}^{U_{i'}} \\ \mathbf{y}_{U_1} = \mathbf{x}_{U_1} \in \mathcal{Z}_{U_1}}} \mathrm{Inf}_{\mathbf{x}}(T_{1-\gamma} A_i)\, \mathrm{Inf}_{\mathbf{y}}(T_{1-\gamma} A_{i'}) \geq \varepsilon$$

Furthermore, by a Markov argument we may assume that in aggregate over a random choice of questions $(U_1, \ldots, U_k)$ to the provers some pair of layers i, i' satisfy this inequality a γ fraction of the time for some positive constant γ (depending on ε, k and the expectation of ψ).

Thus, just like at the end of Section 5, if we assign strategies for the provers based on influences (i.e., we choose answer $\mathbf{x}$ for U_i with probability proportional to $\mathrm{Inf}_{\mathbf{x}}(T_{1-\gamma}A_{U_i})$) we conclude that the answers from provers i and i' have a positive constant probability of being consistent and satisfying all equations on U_1. Repeating the smoothness argument from the end of Section 5 we then conclude that with sufficient smoothness the answers furthermore have a positive constant probability of satisfying all equations on $U_{i'}$ (and when the answers on $U_{i'}$ are consistent with those on U_i this implies that all equations on U_i are satisfied as well).

6.5 Invariance Under Partial Rerandomization

It remains to establish Theorem 6.3, a more general form of equations (4.23)-(4.24) in [17]. Instead of the tedious labor of going over the invariance proof used and checking that all steps work out when only rerandomizing some variables instead of all, let us do this in a somewhat more black-box way.

We are interested in bounding $|E - E^{\mathcal{Z}}|$. For $S \in \mathcal{L}_i$ define $B_{f_S} : \mathcal{F}_{U_i} \to \{-1, 1\}$ depending only on $\mathcal{Z}_i$ by $B_{f_S}(f) = A_i(f|_{\mathcal{Z}_i}, f_S|_{\overline{\mathcal{Z}_i}})$. Then

$$\begin{aligned}
|E - E^{\mathcal{Z}}| &= \left| \mathop{\mathbb{E}}_{\mathcal{T}'} \left[\prod_{i=1}^{k} \prod_{S \in \mathcal{L}_i} A_i(f_S) \right] - \mathop{\mathbb{E}}_{(f_S) \sim \mathcal{T}'} \left[\prod_{i=1}^{k} \mathop{\mathbb{E}}_{(f'_S) \sim \mathcal{T}'} \left[\prod_{S \in \mathcal{L}_i} A_i(f'_S|_{\mathcal{Z}_i}, f_S|_{\overline{\mathcal{Z}_i}}) \right] \right] \right| \\
&\leq \mathop{\mathbb{E}}_{(f_S) \sim \mathcal{T}'} \left[\left| \prod_{i=1}^{k} \prod_{S \in \mathcal{L}_i} A_i(f_S) - \prod_{i=1}^{k} \mathop{\mathbb{E}}_{(f'_S) \sim \mathcal{T}'} \left[\prod_{S \in \mathcal{L}_i} A_i(f'_S|_{\mathcal{Z}_i}, f_S|_{\overline{\mathcal{Z}_i}}) \right] \right| \right] \\
&= \mathop{\mathbb{E}}_{(f_S) \sim \mathcal{T}'} \left[\left| \mathop{\mathbb{E}}_{(f'_S) \sim \mathcal{T}'} \left[\prod_{i=1}^{k} \prod_{S \in \mathcal{L}_i} A_i(f'_S|_{\mathcal{Z}_i}, f_S|_{\overline{\mathcal{Z}_i}}) \right] - \prod_{i=1}^{k} \mathop{\mathbb{E}}_{(f'_S) \sim \mathcal{T}'} \left[\prod_{S \in \mathcal{L}_i} A_i(f'_S|_{\mathcal{Z}_i}, f_S|_{\overline{\mathcal{Z}_i}}) \right] \right| \right] \\
&= \mathop{\mathbb{E}}_{(f_S) \sim \mathcal{T}'} \left[\left| \mathop{\mathbb{E}}_{(f'_S) \sim \mathcal{T}'} \left[\prod_{i=1}^{k} \prod_{S \in \mathcal{L}_i} B_{f_S}(f'_S) \right] - \prod_{i=1}^{k} \mathop{\mathbb{E}}_{(f'_S) \sim \mathcal{T}'} \left[\prod_{S \in \mathcal{L}_i} B_{f_S}(f'_S) \right] \right| \right] \\
&= \mathop{\mathbb{E}}_{(f_S) \sim \mathcal{T}'} \left[\left| E(\{B_{f_S}\}) - E^{\mathcal{Z}}(\{B_{f_S}\}) \right| \right]
\end{aligned}$$

where the equality between the second and third lines relied on the property of $\mathcal{Z}_i$ that it does not "split" any block of the input space $\{0,1\}^{U_i}$ (i.e., if $x \in \mathcal{Z}_i$ then all x' with $x'_{U_1} = x_{U_1}$ are also in $\mathcal{Z}_i$).

Note that since B_{f_S} only depends on the variables in $\mathcal{Z}_i$, $E^{\mathcal{Z}}(\{B_{f_S}\})$ is the same as when we rerandomize all coordinates and this lets us apply the bounds of [17] (equations (4.23)-(4.24)). There is still a slight difference in that we now have a function B_{f_S} for each subset S, whereas before all functions belonging to layer i had the same function A_i. However it is easy to see that this property is not really used in [17].

What remains is to bound the influences of B_{f_S} and we have the following bound.

Lemma 6.5. *For all $S \in \mathcal{L}_i$ and $S' \in \mathcal{L}_{i'}$ and all $x \in \{0,1\}^{U_i}$ and $y \in \{0,1\}^{U_{i'}}$ we have*

$$\mathop{\mathbb{E}}_{(f_S) \sim \mathcal{T}} \left[\mathrm{Inf}_x(T_{1-\gamma}B_{f_S}) \, \mathrm{Inf}_y(T_{1-\gamma}B_{f_{S'}}) \right] \leq c(\gamma) \, \mathrm{Inf}_x(T_{1-\gamma}A_i) \, \mathrm{Inf}_y(T_{1-\gamma}A_{i'})$$

where $c(\gamma)$ is a universal constant depending only on γ.

This is a relatively straightforward application of hypercontractivity but let us do the proof.

Proof. Let us write $C_S = T_{1-\gamma}B_{f_S}$. We have $\mathrm{Inf}_x(C_S) = \sum_{\alpha \ni x} \hat{C}_S(\alpha)^2$, so the LHS we seek to bound is the sum over all $\alpha \ni x$, $\beta \ni y$ of

$$\mathop{\mathbb{E}}_{(f_S)\sim\mathcal{T}}\left[\hat{C}_S(\alpha)^2\hat{C}_{S'}(\beta)^2\right] \leq \sqrt{\mathop{\mathbb{E}}_{(f_S)\sim\mathcal{T}}\left[\hat{C}_S(\alpha)^4\right]\mathop{\mathbb{E}}_{(f_S)\sim\mathcal{T}}\left[\hat{C}_{S'}(\beta)^4\right]} \tag{6.2}$$

We also have

$$\hat{C}_S(\alpha) = \sum_{\beta\subseteq\overline{\mathcal{Z}_i}} \widehat{T_{1-\gamma}A}_i(\alpha\cup\beta)\chi_\beta(f_S),$$

which is a polynomial in the (f_S) with exponentially decaying tails and hence we can bound its fourth moment using hypercontractivity by

$$\mathop{\mathbb{E}}_{(f_S)\sim\mathcal{T}}\left[\hat{C}_S(\alpha)^4\right] \leq c(\gamma)\mathop{\mathbb{E}}_{(f_S)\sim\mathcal{T}}\left[\hat{C}_S(\alpha)^2\right]^2.$$

Plugging this into (6.2) we see that

$$\mathop{\mathbb{E}}_{(f_S)\sim\mathcal{T}}\left[\hat{C}_S(\alpha)^2\hat{C}_{S'}(\beta)^2\right] \leq c(\gamma)\mathop{\mathbb{E}}_{(f_S)\sim\mathcal{T}}\left[\hat{C}_S(\alpha)^2\right]\mathop{\mathbb{E}}_{(f_S)\sim\mathcal{T}}\left[\hat{C}_{S'}(\beta)^2\right]$$

Summing over all $\alpha \ni i$, $\beta \ni i'$, we obtain the desired bound. □

From Lemma 6.5, Markov's inequality, and a union bound over all k^6 pairs (S, S'), it follows that with probability at least $1 - \sqrt{\varepsilon}c(\gamma)k^6$ over the choice of (f_S), it holds that

$$\sum_{\substack{x\in\{0,1\}^{U_i}\\ y\in\{0,1\}^{U_{i'}}\\ y_{U_1}=x_{U_1}}} \mathrm{Inf}_x(T_{1-\gamma}B_{f_S})\,\mathrm{Inf}_y(T_{1-\gamma}B_{f_{S'}}) \leq \sqrt{\varepsilon}c(\gamma)$$

for all S, S'. For such choices of (f_S), applying the special case of Theorem 6.3 proved by Huang [17] implies that $|E(\{B_{f_S}\}) - E^{\mathcal{Z}}(\{B_{f_S}\})| \leq \delta(\sqrt{\varepsilon}c(\gamma), k, \gamma)$, which tends to 0 with ε. For the remaining bad choices of (f_S), $|E(\{B_{f_S}\}) - E^{\mathcal{Z}}(\{B_{f_S}\})| \leq 2$ and hence these contribute at most $2\sqrt{\varepsilon}c(\gamma)k^6$ which also tends to 0 with ε.

7 PROMISE CSPS

Functional folding can also be used to obtain hardness results for *promise CSPs* (PCSPs) with universal factor graphs. The search version of a promise CSP is a constraint satisfaction problem in which the instance is promised to have a satisfying assignment to every constraint, but the goal is to output an assignment which need only satisfy a weakened version of each constraint. The decision version, which we consider here, is to decide between the case where all the stronger "promised" constraints are satisfied, and the case where not even the weakened version can be satisfied. A prototypical example of a PCSP is the "$(2+\varepsilon)$-Sat" problem, in which we are given a $(2k+1)$-CNF formula ϕ and the objective is to distinguish the case where there is an assignment satisfying at least k literals in every clause of ϕ, from the case where ϕ is unsatisfiable. Here the stronger "promise" constraint is that k literals of each clause can be satisfied, and the weakened version is that at least one literal of each clause needs to be satisfied.

One can view 3-Sat as the problem of deciding whether at least 1-in-3 of the literals in each clause can be satisfied, and 2-Sat as deciding whether at least 1-in-2 of the literals in each clause can be satisfied. Thus the reason for the name "$(2+\varepsilon)$-Sat" is that, as k grows the goal is to decide whether $\frac{1}{2+\varepsilon}$ fraction of the clauses can be satisfied, or the formula is unsatisfiable.

We now recall the pertinent definitions.

Definition 7.1. A *PCSP language* is a pair (Γ, Λ) of two indexed constraint languages $\Gamma = \{f_1, \dots, f_t\}$ and $\Lambda = \{g_1, \dots, g_t\}$ such that f_i and g_i have the same arity and $f_i(\mathbf{x}) \le g_i(\mathbf{x})$ for all i and x.

A PCSP language has *free negations*, if for every k-ary constraint pair (f_i, g_i) and every $\mathbf{b} \in \{0,1\}^k$, the constraint pair $(f_i^{\mathbf{b}}, g_i^{\mathbf{b}})$ is also in the language, where $f^{\mathbf{b}}(\mathbf{x}) = f(\mathbf{x} + \mathbf{b})$.

An instance I of the PCSP(Γ, Λ) problem is a pair (X, C) where X is a set of variables and C a set of constraints. Each constraint $c \in C$ is a pair (i, S), for a constraint type $i \in [t]$ and scope S. We write I_Γ for the induced CSP(Γ) instance where each constraint (i, S) is replaced by (f_i, S) and I_Λ for the induced CSP(Λ) instance where (i, S) is replaced by (g_i, S).

PCSP(Γ, Λ) is the promise decision problem where given an instance I the objective is to distinguish whether I_Γ is satisfiable or I_Λ is unsatisfiable.

Definition 2.12 of UFG-NP-hardness extends naturally to PCSP problems. To state the hardness result for PCSPs, we also need the notion of polymorphisms, defined next.

Definition 7.2. A *polymorphism* of a PCSP language (Γ, Λ) is a function $p : \Sigma^\ell \to \Sigma$ such that, for every pair of constraint types $(f_i, g_i) \in (\Gamma, \Lambda)$ and all $x_1, \dots, x_n \in f_i^{-1}(1)$ (where k is the arity of f_i and g_i) it holds that

$$(p(x_{1,1}, \dots, x_{\ell,1}), p(x_{1,2}, \dots, x_{\ell,2}), \dots, p(x_{1,k}, \dots, x_{\ell,k})) \in g_i^{-1}(1).$$

A polymorphism p is *folded* if $p(\neg\mathbf{x}) = \neg p(\mathbf{x})$ for all $\mathbf{x} \in \{0,1\}^\ell$.

In short, a polymorphism is any operation that, when applied coordinate-wise to a set of assignments satisfying f_i, outputs an assignment satisfying g_i. One natural example is in the case where $f : \mathbb{Z}_4^4 \to \{0,1\}$ outputs one if the sum of the inputs is zero mod 4, and $g : \mathbb{Z}_4^4 \to \{0,1\}$ outputs one if the sum of the inputs is zero mod 2. In this case, the operation $p : \mathbb{Z}_4^2 \to \mathbb{Z}_4$ which sums up its inputs mod 2 is a polymorphism of the PCSP language $(\{f\}, \{g\})$.

Our main hardness result of PCSPs having universal factor graphs is the following.

Theorem 7.3. *Let (Γ, Λ) be a finite PCSP language with free negations, and suppose that there exists a universal constant $C = C(\Gamma, \Lambda) < \infty$ such that every folded polymorphism of (Γ, Λ) is a C-junta. Then* PCSP(Γ, Λ) *is UFG-NP-hard.*

This theorem is approximately Theorem 4.7 of [2]. That theorem was simplified and generalized by Brakensiek and Guruswami [5] to PCSP languages where the polymorphisms are only required to be *C-fixing*, a weaker condition than being a C-junta where it is only required that setting all the C coordinates to 0 fixes the value of the function. Another difference is that the result of Brakensiek and Guruswami does not require the PCSP language to have free negations, which is something we require in order to be able to apply functional folding. The proof below follows the simplified proof of Brakensiek and Guruswami very closely but there is one step where we need the stronger condition of being a C-junta instead of just C-fixing—see Footnote 5 for further details.

Using the fact that the polymorphisms of $(2+\varepsilon)$-Sat are juntas [2], we have the following immediate corollary.

Corollary 7.4. *$(2+\varepsilon)$-Sat is UFG-NP-hard.*

Proof of Theorem 7.3. Given a Max-TSA instance I we construct a new set of variables X' as in the reduction template Definition 3.5.

For every query U (resp. W to P_2) in the repeated game, we add constraints on $A_U^{\boxplus}$ (resp. $B_W^{\boxplus}$) forcing it to be a polymorphism of (Γ, Λ). In particular, for every k-ary constraint pair $(f, g) \in (\Gamma, \Lambda)$, and for every sequence of functions $h_1, \dots, h_k \in \mathcal{F}_U$ such that $f(h_1(\mathbf{x}), \dots, h_k(\mathbf{x})) = 1$ for all $\mathbf{x}$, we add the constraint $g(A_U^{\boxplus}(h_1), \dots, A_U^{\boxplus}(h_k)) = 1$. An equivalent set of constraints is added for

$B^{\boxplus}_W$. Clearly, this set of constraints does not depend on the Max-TSA instance I, except via the functional folding of $A^{\boxplus}_U$, and thus this part of the construction is factor-graph preserving.

Furthermore, for every pair of queries $U \subseteq W$ sent to the two provers in the parallel game, and all functions $f \in \mathcal{F}_U$, we identify the values of $A^{\boxplus}_U(f)$ and $B^{\boxplus}_W(f)$ (where we think of $f \in \mathcal{F}_U$ as a function $f \in \mathcal{F}_W$ that only depends on the coordinates in U, in the obvious way). This simply means that whenever we would have accessed $A^{\boxplus}_U(f)$, we instead access $B^{\boxplus}_W(f)$. It is clear that this construction is factor graph-preserving.

Claim 7.5 (Completeness). If I is satisfiable then $R(I)_\Gamma$ is satisfiable.

The proof of completeness is immediate from the definitions and we omit it.

Claim 7.6 (Soundness). If $R(I)_\Lambda$ is satisfiable and $t \geq C^2 r$ then the repeated game is $\frac{1}{2C}$-satisfiable.

Proof. Given a satisfying assignment (consisting of supposed long codes) to $R(I)_\Lambda$, let α^U (resp. β^W) be the set of up to C coordinates that $A^{\boxplus}_U$ (resp. $B^{\boxplus}_W$) depends on.

The key observation, that we now proceed to establish, is that for a pair of queries $U \subseteq W$ sent to the two provers such that $|\beta^W_U| = |\beta^W|$ (which, by the choice $t \geq C^2 r$ and Claim 2.20, are at least $1/2$ of all query pairs), we must have $\beta^W_U \subseteq \alpha^U$. Indeed, suppose for contradiction that $\mathbf{x}^* \in \beta^W$ but $\mathbf{x}^*_U \notin \alpha^U$. Let $g \in \mathcal{F}_W$ be a function such that $B^{\boxplus}_W(g) \neq B^{\boxplus}_W(g + \mathbf{1}_{\mathbf{x}^*})$ and $g(\mathbf{x}) = 1$ for all $\mathbf{x} \notin \beta^W$. Because $|\beta^W_U| = |\beta^W|$ we can also view g as a function $f \in \mathcal{F}_U$ (defined by $f(\mathbf{x}) = 1$ if $\mathbf{y} \notin \beta^W_U$ and otherwise $f(\mathbf{x}) = g(\mathbf{y})$ where $\mathbf{y}$ is the unique $\mathbf{y} \in \beta^W$ such that $\mathbf{y}_U = \mathbf{x}$). Then using that $\mathbf{x}^*_U \notin \alpha^U$ and the identification of values in $A^{\boxplus}_U$ and $B^{\boxplus}_W$ we have the contradiction

$$B^{\boxplus}_W(g + \mathbf{1}_{\mathbf{x}^*}) = A^{\boxplus}_U(f + \mathbf{1}_{\mathbf{x}^*_U}) = A^{\boxplus}_U(f) = B^{\boxplus}_W(g) \neq B^{\boxplus}_W(g + \mathbf{1}_{\mathbf{x}^*})$$

and the key observation follows.[5]

By Lemma 3.4, at least one $\mathbf{x} \in \beta^W$ satisfies all constraints in W, and the strategy for P_2 in the repeated game is to use an arbitrary such $\mathbf{x}$. The strategy for P_1 is to select a random assignment $\mathbf{x} \in \alpha^U$. Since at least half the query pairs satisfy $|\beta^W_U| = |\alpha^U|$ and $|\alpha^U| \leq C$ for all U, this strategy is accepted with probability at least $\frac{1}{2C}$. □

Combining the completeness and soundness claims, the theorem follows. □

8 MISCELLANEOUS EXTENSIONS

In this section we discuss various further extensions to our results.

8.1 More Hardness Results by Gadgets

One major method for deriving new hardness results is by a method usually referred to as "gadget reductions". In such a reduction from Max-CSP(Γ_1) to Max-CSP(Γ_2) one takes one constraint in the source problem and produces one or several constraints in the target problem. These new constraints contain the variables from the original problem and some new variables which are unique to the constraint processed. A general theory for constructing optimal gadgets was introduced by Trevisan et al. [24].

To make such a reduction factor graph-preserving one simply needs to ensure that the factor graph of constant size obtained from a single constraint does not the depend on which constraint

[5] This argument is where we use that the polymorphisms are C-juntas as opposed to just C-fixing. In the previous PCSP hardness results it was sufficient to establish that $\beta^W_U \cap \alpha^U \neq \emptyset$, but this is not sufficient for us. In our setting the functional folding only guarantees that at least one $\mathbf{x} \in \beta^W$ satisfies the equations on W (and this property is easy to establish also for tables that are C-fixing rather than C-juntas), and we need to make sure that this specific $\mathbf{x}$ projects to an element of α^U. For this reason, just having non-empty intersection between β^W_U and α^U is not sufficient.

from the family Γ_1 was used. This is a simple property to test and turns out to be true for most reductions. In particular, one favorite starting point of such a reduction is Max-3-Lin and if we allow negations of variables then Γ_1 is the single predicate parity. This implies that as soon as we reduce to another Max-CSP that allows negations, the reduction is automatically factor graph-preserving. Let us state a couple of immediate corollaries to this observation and some reductions described in [15] and constructed based on the methods of [24].

Corollary 8.1. *For any $\varepsilon > 0$, Max-2-Lin is $(\frac{3}{4} - \varepsilon, \frac{11}{16} + \varepsilon)$-UFG-NP-hard.*

The reduction takes a single equation of the form $x + y + z = 0$ and produces 16 equations each containing two variables from the set $\{x, y, z\}$ joint with 5 new variables. If the equation is satisfied then we can set the new variables to satisfy 12 equations while if it is not satisfied it is only possible to satisfy 10 equations. For 2-Sat we have the following corollary.

Corollary 8.2. *For any $\varepsilon > 0$ Max-2-Sat is $(\frac{11}{12} - \varepsilon, \frac{21}{24} + \varepsilon)$-UFG-NP-hard.*

Here the reduction takes one equation and produces 12 clauses of size two of which 11 can be satisfied if the equation is satisfied while if it is not, the optimum is 10.

It might be instructive to see what happens to the gadget reduction from Max-3-Lin to Max-Cut in [24]. Here each variable in the original problem corresponds to a node in the resulting Max-Cut instance. Each equation containing x, y and z produces a set of new variables which are connected in one of two ways depending on whether the right hand size of the equation is 0 or 1. This implies that the reduction is not factor graph-preserving. Of course, this is not very surprising since when Max-Cut is viewed as a Max-CSP the predicate family is the single predicate of non-equality. This implies that the factor graph uniquely defines the instance and thus it is not an interesting problem in the current context.

8.2 Larger Domains

Several of the hardness results that we reproved with universal factor graphs are known to apply also to CSPs with larger domains (e.g., the Max-3-Lin mod q problem). While we have chosen to focus on Boolean CSPs throughout the paper to keep the notation as simple as possible and focus on the core ideas, these results for larger domains can also be obtained in the universal factor graph setting.

In particular, the approximation resistance of Max-3-Lin (Theorem 4.2) and Max-TSA (Theorem 4.12), and the uselessness of predicates supporting pairwise independent subgroups (Theorem 5.3) can be generalized to arbitrary domains. Let us briefly describe how. Here we do not go into depth and assume to a greater extent than in other parts of the paper that the reader is familiar with the corresponding results in the standard settings, and the Fourier transform over $\mathbb{Z}_q$.

For a domain of size q, the long code-based reductions from the parallel repeated game are modified in the exact same way as one modifies the standard hardness reductions for these problems, by working with q-ary long code tables $A_U : \mathbb{Z}_q^U \to \mathbb{Z}_q$ and $B_W : \mathbb{Z}_q^W \to \mathbb{Z}_q$ and doing Fourier analysis over $\mathbb{Z}_q^n$ instead of over $\{0, 1\}^n$. However, we can not start from a parallel repetition of the Max-TSA problem, but instead need to start over some system of equations over $\mathbb{Z}_q$. This is because the functional folding involves taking quotients of the domain $\mathbb{Z}_q^U$ over the constraint equations of our CSP instance and hence those constraint equations also need to be over $\mathbb{Z}_q$ rather than $\mathbb{Z}_2$.

Fortunately, there are several easy ways to overcome this obstacle to obtain a starting point that can be used and we now sketch one. Starting with a Max-3-Sat instance I, we construct the following system of equations over $\mathbb{Z}_q$. For each clause $x^a \vee y^b \vee z^c$, where x, y, and z are variables, and $a, b, c \in \{-1, 1\}$ indicate whether a variable appears positively or negatively, add three new variables X, Y, Z (separately for each clause of I) and four equations

$$o(X, Y, Z) = 1 \qquad x \cdot X = a \qquad y \cdot Y = b \qquad z \cdot Z = c$$

over $\mathbb{Z}_q$, where the function $o(X, Y, Z)$ is 1 if and only if at least one of X, Y, and Z equals -1. It is easy to see that if the best assignment to I falsifies a δ fraction of clauses then the best assignment to the system of equations falsifies a $\delta/4$ fraction of equations.

Furthermore, the left hand sides of the equations depend only on the factor graph of I, and the negations of I only appear as right hand sides of equations. Thus this is a factor graph-preserving reduction establishing $(1, 1 - \delta)$-UFG-NP-hardness for an equational[6] Max-CSP over $\mathbb{Z}_q$. So analogously to the Boolean case, we apply smooth parallel repetition, introduce q-ary long codes and apply functional folding to these. We then have the following analogue over Lemma 3.4 which says that any non-zero Fourier coefficient of a functionally folded table must depend on an assignment satisfying all the constraints.

Lemma 8.3. *Let A be a supposed q-ary long code, $\{h_i(\mathbf{x}) = b_i\}_{i=1}^r$ be a set of equational constraints over $\mathbb{Z}_q$, $\mathbf{b} = (b_1, \dots, b_r) \in \{0, 1\}^r$, and $H = \{h_1, \dots, h_r\}$. Let $A_{H,b}$ be A folded over all functions on H with respect to $\mathbf{b}$. If $\hat{A}_{H,b}(\alpha) \neq 0$ then the sum of $\alpha(\mathbf{x})$ over the assignments $\mathbf{x}$ that satisfy all r equations equals* $1 \bmod q$.

Proof. Let $h(\mathbf{x}) = (h_1(\mathbf{x}), \dots, h_r(\mathbf{x}))$. Recall that

$$\hat{A}_{H,b}(\alpha) = \mathop{\mathbb{E}}_{f}\left[\omega^{A_{H,b}(f) - \langle \alpha, f\rangle}\right],$$

where $\omega = e^{2\pi i/q}$ is a complex q'th root of unity and $\langle \alpha, f\rangle = \sum_{\mathbf{x}} \alpha(\mathbf{x}) f(\mathbf{x})$ is the inner product of the functions α and f.

By the folding we have for every $c \in \mathbb{Z}_q$ that

$$A_{H,b}(f + c \cdot \mathbf{1}_b(h)) = A_{H,b}(f) + c \cdot \mathbf{1}_b(\mathbf{b}) = A_{H,b}(f) + c.$$

Let z be the sum of $\alpha(\mathbf{x})$ over all $\mathbf{x}$ satisfying all r equations. Then

$$\omega^{\langle \alpha, f + c\mathbf{1}_b(h)\rangle} = \omega^{\langle \alpha, f\rangle + c\sum_{\mathbf{1}_b(h(\mathbf{x}))=1} \alpha(\mathbf{x})} = \omega^{\langle \alpha, f\rangle + cz}.$$

Since the distribution over $f + c\mathbf{1}_b(h)$ over randomly chosen f and c is the same as the distribution over f, it follows that

$$\hat{A}_{H,b}(\alpha) = \mathop{\mathbb{E}}_{f,c}\left[\omega^{A_{H,b}(f) + c - \langle \alpha, f\rangle - cz}\right] = \hat{A}_{H,b}(\alpha) \cdot \mathop{\mathbb{E}}_{c}\left[\omega^{c(1-z)}\right].$$

The expectation over c is 0 unless $z = 1$, so the claim follows. □

With this key property of functional folding established, it is a straightforward but tedious task to go over the existing hardness of approximation proofs for these results and adapt them to the universal factor graph setting in exactly the same way as done for the Boolean case in the preceding sections.

[6] For an appropriate generalization of equational CSPs to the setting where we have two different types of equations instead of just one as in Definition 2.4.

9 CONCLUDING REMARKS AND OPEN QUESTIONS

We have established that many of the current best inapproximability results for various Max-CSPs and PCSPs can be made to hold with universal factor graphs, meaning that the hardness of the problems stem from the variable negations and not from the constraint-variable incidence structure.

Given these new hardness results one can wonder whether there are any natural situations where preprocessing helps. As discussed in the introduction, Max-3-Lin in the universal factor graph setting corresponds to having a fixed linear code and the input is only the vector to which one wants to find a close point. In a similar situation, where one is given a fixed integer lattice and asked to find close points to input vectors, preprocessing seems to help [8, 21], but we do not know of a corresponding result for the problem on codes.

The only natural example we are aware of where preprocessing seems to help in the CSP setting is the example [11] pointed out in [10]. Here a graph structure in the the factor graph can be used to efficiently refute random instances of 3-Sat with $n^{1.4}$ clauses. In addition to this one can come up with contrived examples, for instance by taking a Max-CSP consisting of only two predicates, one being very sparse and hard to approximate and one being very dense. Then without universal factor graphs this problem is very hard to approximate due to the sparse predicate, but in the universal factor graph setting an algorithm can precompute the optimal solution to the instance when all constraints use the sparse predicate, and then either use this assignment or a random assignment to get a better approximation ratio.

The landscape of CSPs would certainly be more interesting if preprocessing was helpful in more general situations, and as we mention below there are some natural problems where we currently do not know whether preprocessing helps or not.

Let us mention some interesting avenues for potential future work.

(1) From an efficiency point of view our reductions leave something to be desired. The main source of this is our need to use smooth parallel repetition, which incurs a large polynomial blow-up with the degree depending on ε. E.g. starting from an n-variable Max-TSA instance that is $(1, 1-\delta)$-UFG-NP-hard our reduction produces a $(1, \frac{7}{8})$-UFG-NP-hard instance of Max-3-Sat on $n^{\Omega(1/\varepsilon^5+1/\delta^3)}$ variables. As a consequence our results do not rule out approximating Max-3-Lin with factor graph preprocessing within a factor $1/2+\varepsilon$ in time $\exp(n^{\varepsilon})$, whereas approximating Max-3-Lin *without* factor graph preprocessing to within $1/2+o(1)$ does not even have $\exp(n^{1-o(1)})$ time algorithms assuming ETH [22].

(2) All our hardness results are based on functional folding, which inherently introduces negations, either through negated literals or through linearity as in Max-3-Lin and Max-TSA. As such, our methods can not be used to prove hardness of problems where folding over true is not possible, such as the Max-3-Sat problem without mixed clauses, which is known to be NP-hard to approximate within $7/8+\varepsilon$ (even on satisfiable instances) [14]. Does this and similar problems remain equally hard to approximate with factor graph preprocessing?

(3) It would be interesting to obtain universal factor graphs for problems whose hardness is based on the Unique Games Conjecture (UGC), such as approximating Max-2-Lin to within a factor 0.879 [20] or Max-2-Sat to within a factor 0.941 [1]. Any such result would probably have to be based on some strengthened version of the UGC, but it is very unclear to us even what a suitable formulation of such a strengthened UGC could be that would allow us to perform the reduction to Max-2-Lin in a factor graph-preserving way.

ACKNOWLEDGMENTS

Research supported by the Approximability and Proof Complexity project funded by the Knut and Alice Wallenberg Foundation.

REFERENCES

[1] Per Austrin. 2007. Balanced max 2-sat might not be the hardest. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007.* 189–197. https://doi.org/10.1145/1250790.1250818

[2] Per Austrin, Venkatesan Guruswami, and Johan Håstad. 2017. $(2+\varepsilon)$-Sat Is NP-hard. *SIAM J. Comput.* 46, 5 (2017), 1554–1573. https://doi.org/10.1137/15M1006507

[3] Per Austrin and Johan Håstad. 2013. On the usefulness of predicates. *TOCT* 5, 1 (2013), 1:1–1:24. https://doi.org/10.1145/2462896.2462897

[4] M. Bellare, O. Goldreich, and M. Sudan. 1998. Free Bits, PCPs and Non-Approximability—Towards tight Results. *SIAM J. Comput.* 27 (1998), 804–915.

[5] Joshua Brakensiek and Venkatesan Guruswami. 2018. Promise Constraint Satisfaction: Structure Theory and a Symmetric Boolean Dichotomy. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018.* 1782–1801. https://doi.org/10.1137/1.9781611975031.117

[6] Jakub Bulín, Andrei A. Krokhin, and Jakub Oprsal. 2019. Algebraic approach to promise constraint satisfaction. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.* 602–613. https://doi.org/10.1145/3313276.3316300

[7] Siu On Chan. 2016. Approximation resistance from pairwise independent subgroups. *J. ACM* 63 (2016), 1–32.

[8] Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. 2014. On the Closest Vector Problem with a Distance Guarantee. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014.* 98–109. https://doi.org/10.1109/CCC.2014.18

[9] Víctor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. 2002. Constraint Satisfaction, Bounded Treewidth, and Finite-Variable Logics. In *Principles and Practice of Constraint Programming - CP 2002*, Pascal Van Hentenryck (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 310–326.

[10] Uriel Feige and Shlomo Jozeph. 2012. Universal Factor Graphs. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 7391)*, Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer (Eds.). Springer, 339–350. https://doi.org/10.1007/978-3-642-31594-7_29

[11] Uriel Feige, Jeong Han Kim, and Eran Ofek. 2006. Witnesses for Non-satisfiability of Dense Random 3CNF Formulas. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS '06).* IEEE Computer Society, Washington, DC, USA, 497–508. https://doi.org/10.1109/FOCS.2006.78

[12] Oded Goldreich. 2011. *Candidate One-Way Functions Based on Expander Graphs.* Springer Berlin Heidelberg, Berlin, Heidelberg, 76–87. https://doi.org/10.1007/978-3-642-22670-0_10

[13] Martin Grohe. 2007. The Complexity of Homomorphism and Constraint Satisfaction Problems Seen from the Other Side. *J. ACM* 54, 1, Article 1 (March 2007), 24 pages. https://doi.org/10.1145/1206035.1206036

[14] Venkatesan Guruswami and Subhash Khot. 2005. Hardness of Max 3SAT with No Mixed Clauses. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA.* 154–162. https://doi.org/10.1109/CCC.2005.10

[15] Johan Håstad. 2001. Some Optimal Inapproximability Results. *J. ACM* 48, 4 (July 2001), 798–859. https://doi.org/10.1145/502090.502098

[16] Johan Håstad. 2014. On the NP-hardness of Max-Not-2. *SIAM J. Comput.* 43 (2014), 179–193.

[17] Sangxia Huang. 2014. Approximation Resistance on Satisfiable Instances for Sparse Predicates. *Theory of Computing* 10, 14 (2014), 359–388. https://doi.org/10.4086/toc.2014.v010a014

[18] Shlomo Jozeph. 2014. Universal Factor Graphs for Every NP-Hard Boolean CSP. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 28)*, Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore (Eds.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 274–283. https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2014.274

[19] Subhash Khot. 2002. Hardness Results for Coloring 3 -Colorable 3 -Uniform Hypergraphs. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings.* 23–32. https://doi.org/10.1109/SFCS.2002.1181879

[20] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. 2007. Optimal Inapproximability Results for MAX-CUT and Other 2-Variable CSPs? *SIAM J. Comput.* 37, 1 (2007), 319–357. https://doi.org/10.1137/S0097539705447372

[21] J. C. Lagarias, H. W. Lenstra, and C. P. Schnorr. 1990. Korkin-Zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica* 10, 4 (01 Dec 1990), 333–348. https://doi.org/10.1007/BF02128669

[22] Dana Moshkovitz and Ran Raz. 2008. Two-query PCP with Subconstant Error. *J. ACM* 57, 5 (2008), 29:1–29:29. https://doi.org/10.1145/1754399.1754402

[23] R. Raz. 1998. A parallel repetition theorem. *SIAM J. on Computing* 27 (1998), 763–803.

[24] L. Trevisan, G. Sorkin, M. Sudan, and D. Williamson. 2000. Gadgets, approximation and linear programming. *SIAM J. Comput.* 29 (2000), 2074–2097.