Computer Science Review 59 (2026) 100834

Contents lists available at ScienceDirect

Computer Science Review

journal homepage: www.elsevier.com/locate/cosrev

Review Article

Parameterized inapproximability: From Clique to PIH

Yijia Chen [a], Bingkai Lin [b],*

[a] Shanghai Jiao Tong University, China
[b] Nanjing University, China

ARTICLE INFO

MSC:
primary 68Q17
68Q25
secondary 68R05
68W25

Keywords:
Parameterized Inapproximability Hypothesis
Clique problem
Constraint satisfaction problem

ABSTRACT

Parameterized approximation, first proposed by Mike Fellows, approaches NP-hard problems by allowing the running time of an approximation algorithm to be superpolynomial in the parameter of an problem instance yet still polynomial in the size of the instance itself. One of the main open questions in the area is whether we can approximate the parameterized clique problem within some nontrivial ratio. It is also conjectured by Fellows that no such algorithms exist. In this article, we explain some recent progress on this question.

Similarly to the classical polynomial time inapproximability of the clique problem, the constraint satisfaction problem, i.e., CSP, plays a key role in most of the known inapproximability results of the parameterized clique problem. As a matter of fact, the parameterized inapproximability hypothesis, i.e., PIH, concerning the binary CSP has been long believed as a viable path towards the inapproximability of the parameterized clique problem. Although it turns out that those recent results do not rely on PIH, the method discovered for the parameterized clique problem leads to a proof of a version of PIH under the exponential time hypothesis, which we will also explain in this article.

Contents

* Corresponding author.
E-mail address: lin@nju.edu.cn (B. Lin).

https://doi.org/10.1016/j.cosrev.2025.100834
Received 24 July 2024; Received in revised form 26 March 2025; Accepted 24 September 2025
Available online 4 October 2025

1. Introduction

Parameterized approximation combines two powerful approaches to dealing with NP-hard problems, i.e., approximation and parameterized algorithms. In the first method, we allow algorithms to compute suboptimal solutions, i.e., solutions hopefully not very far from the optimal ones. For example, there is a polynomial time approximation algorithm based on maximal matching which computes vertex covers of size at most 2 times of the minimum ones. This is known as an approximation algorithm for the minimum vertex cover problem of ratio 2. For the second approach, we identify some parameters from problem instances and look for exponential (or worse) time algorithms whose superpolynomial behavior is confined in these parameters. Such an algorithm is called a fixed-parameter algorithm (or fpt-algorithm),[1] pioneered by Downey and Fellows [1]. Again taking the vertex cover problem as an example, it is well known that we can decide on an input n-vertex graph G and a parameter $k \geq 1$ whether G contains a vertex cover of size at most k in time $2^{O(k)}n^{O(1)}$. In other words, the parameterized vertex cover problem is *fixed-parameter tractable*, or in FPT. Both methods have seen huge success in the last decades, with numerous deep and elegant algorithms having been discovered.

On the other hand, there are still many problems resisting attacks by both approaches. One famous example is the clique problem. In the context of approximation, it is known that, for any fixed small constant $\varepsilon > 0$, computing a clique of size n^{ε} in an n-vertex graph with a clique of size $n^{1-\varepsilon}$ is already NP-hard [2]. For parameterized complexity, k-Clique – the clique problem parameterized by the size of a clique is an archetypal W[1]-hard problem. Here, W[1] is one of the central parameterized intractable classes which plays a role in parameterized complexity analogously to NP in classical complexity. It is widely believed that FPT $\neq$ W[1], or equivalently that k-Clique is not fixed-parameter tractable. As a consequence of the failure of both approaches, parameterized approximation has been proposed to further relax both the running time and the optimality of the solution for an algorithm solving the clique problem. More precisely, we might aim at an fpt-algorithm for k-Clique that computes a clique of size $k/2$ in an input graph G provided that G has a clique of size k. However, Fellows already conjectured many years ago that such an approximation algorithm does not exist [3].

To prove the aforementioned NP-hardness of approximating the classical clique problem, one needs the very deep PCP theorem [4,5]. So it was postulated in [6] that a parameterized version of PCP is needed before we can show any inapproximability of the parameterized clique problem. As the proof of the classical PCP theorem is highly non-trivial, it is within everyone's expectation that the parameterized PCP theorem, or even its right formulation, is a difficult question. One natural attempt is to adapt the classical proof to the parameterized setting. In particular, under the exponential time hypothesis (or ETH for short), it has been long known (see for example [7]) that if there is a so-called *linear-size* PCP, then there is no fpt-approximation of the clique problem of constant approximation ratio. However, the status of the linear-size PCP is far from being settled, and it might not exist after all. To circumvent this obstacle, [8] starts with a further complexity-theoretic assumption known as Gap-ETH [9,10]. Recall that ETH states that there is no subexponential time algorithm solving the 3Sat problem. Gap-ETH strengthens ETH by excluding even the possibility of a subexponential time algorithm which can distinguish between satisfiable 3Sat instances and those instances where only a small fraction of clauses can be satisfied simultaneously. It is clear that Gap-ETH is tailored for proving inapproximability, as it assumes there is a gap between yes- and no-instances for free. [8] proves that, unless Gap-ETH fails, there is no fpt-approximation of the clique problem of ratio $\omega(k)$. In other words, the only possible fpt-approximation is to compute a clique of constant size, which Fellows coined as "hideous" [3] (i.e., Conjecture 2.2). As far as the lower bounds are concerned, the results in [8] are optimal. However, the assumption Gap-ETH, similar to the linear-size PCP,[2] might be too strong to be readily accepted, and proving Gap-ETH under ETH seems beyond our current reach.

It might come as a surprise that [11] proves that the constant approximation of the parameterized clique problem is W[1]-hard without establishing an explicit parameterized PCP theorem. Nevertheless, the proof follows the usual paradigm of the PCP-based inapproximability of the clique problem. Namely, we first reduce k-Clique to an algebraic problem known as k-VectorSum. Then we exploit the rich algebraic structure underlying k-VectorSum. That is, using Hadamard codes with their local testing and decoding, k-VectorSum is further reduced to the binary constraint satisfaction problem 2CSP. In fact, the 2CSP instances which we construct mimic a PCP-like proof-verification protocol of k-VectorSum. Finally, we turn the 2CSP instances back to k-Clique instances, where some constant gap between yes- and no-instances emerges. This result was quickly improved in [12], where it is shown that approximating k-Clique within ratio $k^{o(1)}$ is W[1]-hard. The proof follows the same route in [11], where the local testing of Hadamard codes is replaced by the list decoding of Hadmard codes. Afterwards, another surprise comes in [13], where the same $k^{o(1)}$ inapproximability result is established with a much simplified proof. Technically, it is still based on Hadamard codes, but no local testing or list decoding is required. Instead, [13] uses locally decoding of Hadamard codeword after encoding vertices using elements of Sidon sets [14–16]. As far as the W[1]-hardness is concerned, the ratio of $k^{o(1)}$ is the best inapproximability we can prove at the time this article is written. However, under ETH a better bound is established in [17], by replacing k-VectorSum by a more streamlined CSP problem (which is also underlying the proof in [13]) and by replacing Hadamard codes by some more efficient error-correcting codes.

It should not be a surprise that CSP plays a major role in almost all the proofs of the lower bounds discussed so far, given the prevalence of CSP in the study of polynomial time inapproximability. Among others, the classical PCP theorem can be equivalently phrased as the NP-hardness of distinguishing between satisfiable CSP instances and those instances far from being satisfiable. In the same vein, it was suggested [6] a long time ago that, before tackling k-Clique, one should first settle the constant inapproximability of the parameterized binary CSP. This is viewed as a potential parameterized PCP theorem and dubbed as the *parameterized inapproximability hypothesis*, or PIH for short. However, despite all the known W[1]-hardness for approximating k-Clique and the use of CSP in its proofs, it seems that we are still far from proving PIH assuming FPT $\neq$ W[1]. Nevertheless, [18] succeeds in proving PIH under ETH, and its proof uses the same paradigm in [17].

As the proofs of the inapproximability of k-Clique might still not be widely known, and there are important differences with those of the classical inapproximability, the goal of this article is to give some details of these proofs. In particular, we will focus the role played by CSP, and how it leads to a proof of PIH under ETH. Given the space limitation, we will not discuss further applications of the inapproximability of k-Clique and PIH. Instead we refer the reader to the survey [19] and the discussions in [18].

[1] All key notions will be defined precisely in Section 2.

[2] Actually, ETH plus the existence of the linear-sized PCP implies Gap-ETH.

Organization of the paper. We introduce all necessary notions and notations for parameterized approximation in Section 2. We explain the elementary proof of the $k^{o(1)}$ lower bound in [13] in Section 3. Then Section 4 gives a detailed account of the first proof [11] of the constant inapproximability of k-Clique and its subsequence work under ETH. In Section 5, we discuss the current status of PIH, in particular the proof of PIH under ETH. Finally we conclude in Section 6.

2. Preliminaries

We assume that the reader is familiar with basic graph theory, algebra, coding theory, and parameterized complexity. In the following we recall some key notions and also fix some notations. They are by no means complete, so the reader might consult some standard textbooks, e.g., [1,20–22].

$\mathbb{N}$ is the set of natural numbers starting from 0. For every $n \in \mathbb{N}$ we use $[n]$ to denote the subset $\{1, \dots, n\}$ of $\mathbb{N}$. Let S be a set and $k \in \mathbb{N}$. Then $\binom{S}{k}$ is the set of all k-element subsets of S.

Graphs

A graph $G = (V, E)$ consists of a finite nonempty set $V = V(G)$ of vertices and a set of edges $E = E(G) \subseteq \binom{V}{2}$. In particular, each edge $\{u, v\}$ is a set of two distinct vertices. It means that we only consider simple graphs, i.e., finite, undirected, and without self-loops. For a nonempty subset $S \subseteq V(G)$ the induced subgraph $G[S]$ of G has vertex set S and edge set $E(G) \cap \binom{S}{2}$. If $G[S]$ is a complete graph, i.e., $E(G) \cap \binom{S}{2} = \binom{S}{2}$, then S is a *clique* in G. If in addition $|S| = k$, then we refer to S as a k-clique. We use $\omega(G)$ to denote the size of a maximum clique in G. On the other extreme, if $G[S]$ contains no edge, then S is an *independent set*.

Finite fields and vector spaces

Let q be a prime power. Then $\mathbb{F}_q$ is the (unique) finite field with q elements. In case q is a prime, then $\mathbb{F}_q$ is a *prime field*. For every $d \geq 1$, $\mathbb{F}_q^d$ is the d-dimensional vector space over $\mathbb{F}_q$. We use boldface letters $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}, \dots$ to denote vectors in $\mathbb{F}_q^d$. For every $\boldsymbol{v} \in \mathbb{F}_q^d$ and $i \in [d]$, the ith entry of $\boldsymbol{v}$ is written $\boldsymbol{v}[i]$. Of course, $\boldsymbol{v} \in \mathbb{F}_q^d$ can be also viewed as a tuple of d elements in $\mathbb{F}_q$, and in that case we write $\bar{v} \in \mathbb{F}_q^d$ instead. The difference between $\boldsymbol{v}$ and $\bar{v}$ will play a role in some of our discussions.

Parameterized complexity

Let Σ be a fixed alphabet. A parameterized problem is a classical problem $Q \subseteq \Sigma^*$ augmented with a polynomial time computable *parameterization* $\kappa : \Sigma^* \to \mathbb{N}$, which maps every problem instance $x \in \Sigma^*$ to a small parameter $k := \kappa(x) \in \mathbb{N}$. One of the most important parameterized problems is k-Clique. Recall that in the classical k-Clique problem, we are given a graph G and $k \geq 1$, and the goal is to decide whether G contains a k-clique. Its parameterized version, which we still denote by k-Clique, has the natural parameterization $\kappa(G, k) = k$. We often describe a parameterized problem in the following exemplary form.

k-Clique	
Instance:	A graph G and $k \geq 1$.
Parameter:	k.
Problem:	Decide whether G has a clique of size k.

A parameterized problem (Q, κ) is said to be *fixed-parameter tractable* if there is an algorithm $\mathcal{A}$ which on any input $x \in \Sigma^*$ decides whether $x \in Q$ in time $f(\kappa(x))|x|^{O(1)}$, where $f : \mathbb{N} \to \mathbb{N}$ is a computable function. In other words, $\mathcal{A}$ is an fpt-*algorithm* for (Q, κ). The class of all fixed-parameter tractable problems is denoted by FPT.

It is widely believed that k-Clique is not fixed-parameter tractable, i.e., k-Clique $\notin$ FPT. Indeed, k-Clique is complete for the parameterized class W[1], and therefore k-Clique $\notin$ FPT is equivalent to FPT $\neq$ W[1]. Of course, this W[1]-completeness requires a notion of parameterized reduction. Let (Q_1, κ_1) and (Q_2, κ_2) be two parameterized problems. An fpt-*reduction* from (Q_1, κ_1) to (Q_2, κ_2) is an fpt-algorithm $\mathcal{R}$ such that for every instance x of Q_1 the algorithm $\mathcal{R}$ computes an instance $\mathcal{R}(x)$ of Q_2 satisfying:

- $x \in Q_1$ if and only if $\mathcal{R}(x) \in Q_2$,
- and $\kappa_2(\mathcal{R}(x)) \leq g(\kappa_1(x))$ for a computable function $g : \mathbb{N} \to \mathbb{N}$ independent of x.

Thereby, W[1] consists of all parameterized problems that can be reduced to k-Clique by an fpt-reduction. One of the evidences for FPT $\neq$ W[1] is that it is implied by the *exponential time hypothesis*, or ETH for short. Recall that 3Sat is the classical problem of deciding whether a propositional formula F in 3CNF is satisfiable. ETH states that there is no algorithm of running time $2^{o(n)}$ for 3Sat, where n is the number of variables in an input formula F.

2.1. Parameterized optimization and approximation

In classical complexity, k-Clique is intimately associated with the optimization problem Max-Clique:

Max-Clique	
Instance:	A graph G.
Solutions:	A clique S in G.
Cost:	$\lvert S \rvert$.
Goal:	max.

It is easy to see that k-Clique can be reduced to Max-Clique by a straightforward polynomial time *Turing* reduction. On the other hand, it is not obvious how to define a parameterized version of Max-Clique. In principle, we cannot take $\omega(G)$ as the parameterization of Max-Clique, since computing $\omega(G)$ is NP-hard. We refer the reader to [6] for a detailed discussion. But for our purposes, it suffices to say that Max-Clique is fixed-parameter tractable if there is an algorithm which on input graph G computes a maximum clique in time $f(\omega(G))|V(G)|^{O(1)}$ for a computable function $f : \mathbb{N} \to \mathbb{N}$.

In general we can only upper bound $\omega(G)$ by $|V(G)|$. However, there is a well-known variant of k-Clique in which we can bound $\omega(G)$ by the given parameter k, i.e., the *multi-colored* k-Clique *problem*:

Instance:	A graph G and $k \geq 1$ such that $V(G) = V_1 \dot\cup \cdots \dot\cup V_k$ where each V_i is an independent set in G.
Parameter:	k.
Problem:	Decides whether G has a k-clique.

The fpt-reduction from k-Clique to the above multi-colored version is trivial. For a given graph G and a parameter $k \geq 1$, we construct a new graph G' by

- making k disjoint copies $V(G)$, i.e., $V_i := \{(v, i) \mid v \in V(G)\}$, and set $V(G') := \dot\bigcup_{i \in [k]} V_i$,
- adding an edge between (u, i) and (v, j) in G' if $\{u, v\} \in V(G)$ and $i \neq j$.

Obviously G has a k-clique if and only if G' does. Moreover, each V_i is an independent set in G, and therefore $\omega(G') \leq k$. As a consequence, if Max-Clique is fixed-parameter tractable, then k-Clique is fixed-parameter tractable as well. This is one of many technical reasons that it is often convenient to assume that a k-Clique instance is an instance of the multi-colored k-Clique instance.

By the above discussions, under the assumption FPT $\neq$ W[1], Max-Clique is not fixed-parameter tractable. So it is asked whether we can *approximate* Max-Clique by an fpt-approximation algorithm.

Definition 2.1. Let $\rho : \mathbb{N} \to \mathbb{N}$. Then an algorithm $\mathcal{A}$ is an fpt-*approximation* of Max-Clique of ratio ρ if

- on an input graph G the algorithm $\mathcal{A}$ computes a clique in G of size at least
$$\frac{\omega(G)}{\rho(\omega(G))},$$
- and the running time of $\mathcal{A}$ can be bounded by $f(\omega(G))|V(G)|^{O(1)}$ for some computable function $f : \mathbb{N} \to \mathbb{N}$.

We also call $\mathcal{A}$ a ρ-*approximation*. If for some constant $c \geq 1$ we have $\rho(k) = c$ for all $k \in \mathbb{N}$, then $\mathcal{A}$ is a *constant approximation* of Max-Clique. Settling the existence of such a constant approximation is often the first step towards understanding of the approximability of Max-Clique.

The most trivial fpt-approximation of Max-Clique is to output a single vertex in any input graph, which archives a ratio $\rho(k) = k$. It can be easily improved to $\rho(k) = \lceil k/c \rceil$ for *any* fixed constant $c \geq 1$ by computing a clique of size c, if it exists. This is actually believed to be the best we can do.

Conjecture 2.2. *For every $\rho : \mathbb{N} \to \mathbb{N}$ such that $\rho(k) = o(k)$ there is no* fpt-*approximation of* Max-Clique *of ratio ρ.*

For the sake of argument, let us assume that $\mathcal{A}$ is an fpt-approximation of Max-Clique of ratio ρ. Consider a k-Clique instance (G, k). By the reduction from k-Clique to its multi-colored version, we can further assume $\omega(G) \leq k$. Then we feed $\mathcal{A}$ with G. By Definition 2.1 we conclude that if G has a k-clique, then $\mathcal{A}$ computes a clique in G of size at least $k/\rho(k)$. Moreover, $\mathcal{A}$ runs in time $f(k)|V(G)|^{O(1)}$ for a computable function $f : \mathbb{N} \to \mathbb{N}$. This motivates the introduction of approximation of k-Clique, despite the fact that k-Clique is a *decision* problem.

Definition 2.3. Let $\rho : \mathbb{N} \to \mathbb{N}$. Then an algorithm $\mathcal{A}$ is an fpt-*approximation* of k-Clique of ratio ρ if

- on an input graph G and $k \geq 1$ with $\omega(G) \geq k$ the algorithm $\mathcal{A}$ computes a clique in G of size at least
$$\frac{k}{\rho(k)},$$
- and $\mathcal{A}$ is an fpt-algorithm.

As a matter of fact, Definitions 2.1 and 2.3 are equivalent in the following sense.

Proposition 2.4. *For every $\rho : \mathbb{N} \to \mathbb{N}$,* Max-Clique *has an* fpt-*approximation of ratio ρ if and only if k-*Clique *has an* fpt-*approximation of ratio ρ.*

We have already given an informal proof (before Definition 2.3) of the direction from left to right in Proposition 2.4, and the proof of the other direction is left as an exercise to the reader. Given this equivalence, and that the presence of the additional parameter k often provides some technical convenience, we will mostly work with the fpt-approximation of k-Clique. In particular, Conjecture 2.2 is turned into the following equivalent form.

Conjecture 2.5. *For every $\rho : \mathbb{N} \to \mathbb{N}$ such that $\rho(k) = o(k)$ there is no* fpt-*approximation of k-*Clique *of ratio ρ.*

Clearly proving Conjecture 2.5 unconditionally will resolve some core complexity questions, e.g., FPT $\neq$ W[1] and hence also P $\neq$ NP. So our best hope is to prove it under the assumption FPT $\neq$ W[1]. In particular, we look for a *gap-creating* reduction for k-Clique.

Conjecture 2.6. *For every $\rho : \mathbb{N} \to \mathbb{N}$ such that $\rho(k) = o(k)$ there is an* fpt-*reduction $\mathcal{R}$ from k-*Clique *that on any input graph G and $k \geq 1$ computes a graph G' and $k' \geq 1$ with the following properties.*

Completeness. *If G has a k-clique, then G' has a k'-clique, i.e., $\omega(G) \geq k'$.*

Soundness. *If G has no k-clique, then G' has no clique of size $k'/\rho(k')$, i.e., $\omega(G') < k'/\rho(k')$.*

That is, there is a gap between k' and $k'/\rho(k')$ of yes- and no-instances that $\mathcal{R}$ creates.

Once we have such an fpt-reduction $\mathcal{R}$, it is W[1]-hard to *distinguish* between $\omega(G) \geq k$ and $\omega(G) < k/\rho(k)$ for any k-Clique instance (G, k). This is because any (hypothetical) fpt-algorithm $\mathcal{A}$ that can distinguish these gap instances can be composed with the reduction $\mathcal{R}$, which would give us an fpt-algorithm deciding the k-Clique problem. It is not hard to see that a ρ-approximation $\mathcal{A}$ of k-Clique can do exactly that. Therefore, Conjecture 2.6 implies Conjecture 2.5.

2.2. Constraint satisfaction problems

Besides k-Clique, another problem central to our exposition is the *constraint satisfaction problem*, or CSP for short. A CSP instance $\Gamma = (X, \Sigma, \Phi)$ consists of a variable set X, an *alphabet* Σ, and a set Φ of *constraints*. More precisely,

- X is a nonempty set of variables,
- $\Sigma = \bigcup_{x \in X} \Sigma_x$ contains for each variable $x \in X$ a subset Σ_x of values which can be assigned to x,
- and every $\varphi \in \Phi$ is a constraint specified by a tuple of variables $\bar{x} = x_1, \dots, x_\ell$ and a subset $C \subseteq \Sigma_{x_1} \times \cdots \times \Sigma_{x_\ell}$ for some $\ell \geq 1$. The number ℓ is the *arity* of $\varphi = (\bar{x}, C)$.

The size of Γ is defined by

$$|\Gamma| := |X| + |\Sigma| + \bigcup_{(\bar{x}, C) \in \Phi} |C|.$$

A solution of Γ is an assignment $\sigma : X \to \Sigma$ such that $\sigma(x) \in \Sigma_x$ for every $x \in X$. A constraint $\varphi \in \Phi$ is satisfied by σ if

$$\big(\sigma(x_1), \dots, \sigma(x_\ell)\big) \in C,$$

where $\varphi = \big(x_1 \dots x_\ell, C\big)$. Then, σ satisfies Γ if every constraint in Γ is satisfied by σ. The constraint satisfaction problem is to decide whether a given CSP instance Γ is satisfiable.

The maximum arity of all the constraints in Γ is the arity of Γ. We are mostly interested in *binary* CSP instances Γ, that is, every constraint in Γ involves exactly two distinct variables. For instance, k-Clique can be reformulates as a binary CSP problem. Let G be a graph and $k \geq 1$. Then we introduce a set X of k variables, i.e., $X := \{x_1, \dots, x_k\}$, and $\Sigma := \Sigma_{x_1} := \cdots := \Sigma_{x_k} := V(G)$. For every $1 \leq i < i' \leq k$ we have a constraint $\varphi_{ii'} = \big(x_i x_{i'}, C_{ii'}\big)$ where

$$C_{ii'} = \big\{(u, v) \bigm| \{u, v\} \in E(G)\big\},$$

thereby $\Phi := \big\{\varphi_{ii'} \bigm| 1 \leq i < i' \leq k\big\}$. Then it is easy to see that G has a k-clique if and only if

$$\Gamma_{G,k} := \big(X, \Sigma, \Phi\big) \tag{1}$$

is satisfiable. Therefore, the *binary constraint satisfaction problem*, abbreviated as 2CSP, subsumes k-Clique.

The most natural way to parameterize a CSP instance $\Gamma = (X, \Sigma, \Phi)$ is by taking the number $|X|$ of variables as the parameter. However, the number $|\Phi|$ of constraints cannot in general be bounded by $|X|$, which sometimes leads to technical issues, e.g., the gap amplification in Lemma 2.8 (see 7 on Page 32). So we define the *parameterized constraint satisfaction problem* as

Instance:	A CSP instance $\Gamma = (X, \Sigma, \Phi)$.
Parameter:	$\lvert X\rvert + \lvert\Phi\rvert$.
Problem:	Decide whether Γ is satisfiable.

Nevertheless, for all the natural CSP instances discussed in this paper, we can bound $|\Phi|$ by a function of $|X|$. For example, for the CSP instance $\Gamma_{G,k}$ in (1), we have $|X| = k$ and $|\Phi| = \binom{k}{2} \leq |X|^2$.

In the following we will mostly work with binary CSP instances, and the resulting *parameterized binary constraint satisfaction problem* is still denoted by 2CSP. Clearly, the mapping $(G, k) \mapsto \Gamma_{G,k}$ is an fpt-reduction from k-Clique to 2CSP. Hence, 2CSP is hard for W[1].

In the context of optimization, it is natural to look for an assignment maximizing the number of satisfied constraints. This leads to the optimization problem:

Max-2CSP	
Instance:	A 2CSP instance $\Gamma = (X, \Sigma, \Phi)$.
Solutions:	An assignment $\sigma : X \to \Sigma$.
Cost:	The number of constraints in Γ that are satisfied by σ.
Goal:	max.

Max-2CSP is said to be fixed-parameter tractable if it can be solved by an fpt-algorithm, i.e., in time $f(|X| + |\Phi|)|\Gamma|^{O(1)}$ for any given binary CSP instance $\Gamma = (X, \Sigma, \Phi)$, where $f : \mathbb{N} \to \mathbb{N}$ is a computable function. Similar to Max-Clique, Max-2CSP is fixed-parameter tractable if and only if 2CSP is fixed-parameter tractable. Then, the *parameterized inapproximability hypothesis*, i.e., PIH, basically states that Max-2CSP is *constant inapproximable.*

Conjecture 2.7 (*PIH*). *There is a constant $1 > \varepsilon > 0$ such that it is* W[1]*-hard to distinguish between*

- *satisfiable* 2CSP *instances,*
- *and* 2CSP *instances $\Gamma = (X, \Sigma, \Phi)$ where any assignment cannot satisfy $\lceil \varepsilon|\Phi| \rceil$ many constraints in Γ, i.e., no assignment can satisfy ε-fraction of the constraints.*

As discussed in Conjecture 2.6, this amounts to show that there is an fpt-*reduction $\mathcal{R}$ from k-*Clique *to* 2CSP *with the following properties. Let G be a graph and $k \geq 1$.*

Completeness. *If G has a k-clique, then $\mathcal{R}(G)$ is satisfiable.*

Soundness. *If G has no k-clique, then no assignment can satisfy ε-fraction of the constraints in $\mathcal{R}(G)$.*

It is frequently mentioned, e.g., [19,23] , that the constant ε in PIH does not matter. That is, for some given constant $0 < \varepsilon < 1$, we use PIH$_\varepsilon$ to denote Conjecture 2.7 restricted to ε. Then:

Lemma 2.8. *If* PIH$_\varepsilon$ *holds for some $0 < \varepsilon < 1$, then* PIH$_\varepsilon$ *is true for all $0 < \varepsilon < 1$.*

Lemma 2.8 follows from the parallel repetition theorem [24]. As we are not aware of a complete proof in the literature, we include one in Appendix A.

Now we are ready to show that PIH implies that k-Clique is constant inapproximable. At the core of the proof is the classical FGLSS-reduction [25].

Theorem 2.9. *Assume* PIH. *Then any constant approximation of k-*Clique *is* W[1]*-hard.*

Proof. As we have assumed that PIH holds, i.e., Conjecture 2.7, there is an fpt-reduction $\mathcal{R}$ from k-Clique to 2CSP. Particularly, for any k-Clique instance (G, k), let $\Gamma = (X, \Sigma, \Phi) := \mathcal{R}(G)$ be the 2CSP instance $\mathcal{R}$ construct from (G, k). Therefore,

(P1) if G has a k-clique, then Γ is satisfiable;

(P2) otherwise, no assignment can satisfy ε-fraction of the constraints in Γ, where $\varepsilon < 1$ will be determined later.

Let $k' := |\Phi|$, and since $\mathcal{R}$ is an fpt-reduction, k' can be bounded in terms of k.

Now we construct a second graph H as follows. First for every constraint $\varphi = (x_1 x_2, C)$ we let

$$V_\varphi := \left\{\theta \;\middle|\; \theta \text{ is an assignment on } x_1, x_2 \text{ such that } (\theta(x_1), \theta(x_2)) \in C\right\}.$$

and then $V(H) := \bigcup_{\varphi \in \Phi} V_\varphi$. That is, every vertex in $V(H)$ corresponds to a partial assignment which satisfies a constraint. Now for the edge set $E(H)$, let $\theta \in V_\varphi$ and $\theta' \in V_{\varphi'}$ for some $\varphi, \varphi' \in \Phi$ with $\varphi \neq \varphi'$. Then there is an edge between θ and θ' in H if and only if they assign the same value to any shared variable.

It is straightforward to verify that for every $\ell \geq 1$

$$\text{there is an assignment satisfying } \ell \text{ constraints in } \Gamma \iff H \text{ has a clique of size } \ell.$$

So by the above **(P1)** and **(P2)** we conclude

- if G has a k-clique, then H has a clique of size $|\Phi| = k'$;
- otherwise, $\omega(H) < \varepsilon|\Phi| = \varepsilon k'$.

By Lemma 2.8 for every constant $c \geq 1$, we can choose $\varepsilon = 1/c$. As discussed for Conjecture 2.6 we conclude that the constant approximation of k-Clique with ratio c is W[1]-hard. $\square$

Remark 2.10. In fact, we can use the standard graph product [26] to amplify the gap for the clique instances, thus bypassing Lemma 2.8. ⊣

2.3. Parallelized Hadamard codes

Fix a finite field $\mathbb{F}$. A vector $\boldsymbol{v} \in \mathbb{F}^t$ for some $t \geq 1$ is mostly considered as a column vector, written as

$$\boldsymbol{v} = \left(\boldsymbol{v}[1], \ldots, \boldsymbol{v}[t]\right)^\mathsf{T}.$$

On the other hand, for $k, d \geq 1$ we might view any $\boldsymbol{a} = \left(\boldsymbol{a}[1], \ldots, \boldsymbol{a}[k]\right) \in (\mathbb{F}^d)^k$ as a $d \times k$ matrix over $\mathbb{F}$. Thus, $\boldsymbol{a}[i]$ is a d-dimensional column vector in $\mathbb{F}^d$ for every $i \in [k]$, and

$$\boldsymbol{a}_j := \left(\boldsymbol{a}[1][j], \ldots, \boldsymbol{a}[k][j]\right). \tag{2}$$

is a row vector in $\mathbb{F}^k$ for every $j \in [d]$. See Fig. 1(a) for an example.

Definition 2.11. Let $\mathcal{H}_k^d : (\mathbb{F}^d)^k \to \left(\mathbb{F}^k \to \mathbb{F}^d\right)$ be defined a follows. For every $\boldsymbol{a} \in (\mathbb{F}^d)^k$ the function

$$\mathcal{H}_k^d(\boldsymbol{a}) : \mathbb{F}^k \to \mathbb{F}^d$$

maps every $\bar{r} = (r_1, \ldots, r_k)^\mathsf{T} \in \mathbb{F}^k$ to

$$\mathcal{H}_k^d(\boldsymbol{a})[\bar{r}] := r_1\boldsymbol{a}[1] + \cdots + r_k\boldsymbol{a}[k] \in \mathbb{F}^d. \tag{3}$$

Here, we write $\bar{r}$ instead of $\boldsymbol{r}$ to emphasize that it is used as a tuple of coefficients for linear combinations of vectors in $\mathbb{F}^d$.

Remark 2.12. $\mathcal{H}_k^d$ is known in the literature as *interleaved Hadamard code* and *linear transformation code* [27,28], also it is termed *parallel Walsh–Hadamard code* in [18].

Recall that a *Hadamard code* can be identified with a function

$$\mathcal{H}_k : \mathbb{F}^k \to \left(\mathbb{F}^k \to \mathbb{F}\right)$$

such that for every $\boldsymbol{\alpha} \in \mathbb{F}^k$ and $\bar{r} = (r_1, \ldots, r_k) \in \mathbb{F}^k$ we have

$$\mathcal{H}_k(\boldsymbol{\alpha})(\bar{r}) = r_1\boldsymbol{\alpha}[1] + \cdots + r_k\boldsymbol{\alpha}[k] \in \mathbb{F}. \tag{4}$$

So it is easy to see (cf. Fig. 1(b)) that

$$\mathcal{H}_k^d(\boldsymbol{a})(\bar{r}) = \begin{pmatrix} \mathcal{H}_k(\boldsymbol{a}_1)(\bar{r}) \\ \vdots \\ \mathcal{H}_k(\boldsymbol{a}_d)(\bar{r}) \end{pmatrix} \in \mathbb{F}^d,$$

where each $\boldsymbol{a}_j$ is defined in (2). Hence, we call $\mathcal{H}_k^d$ the *d-parallelized Hadamard Code*. Recall that the original view of $\mathcal{H}_k^d(\boldsymbol{a})$ is the set of all the linear combinations (3) of the column vectors of $\boldsymbol{a}$. ⊣

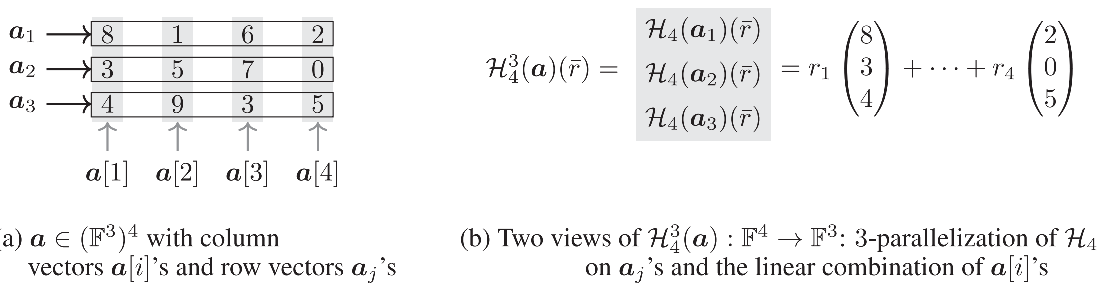

a_1
a_2
a_3
8 1 6 2
3 5 7 0
4 9 3 5
$a[1]$ $a[2]$ $a[3]$ $a[4]$
$\mathcal{H}_4^3(a)(\bar r) = \mathcal{H}_4(a_1)(\bar r)\ \mathcal{H}_4(a_2)(\bar r)\ \mathcal{H}_4(a_3)(\bar r) = r_1 (8,3,4) + \cdots + r_4 (2,0,5)$
(a) $a \in (\mathbb{F}^3)^4$ with column vectors $a[i]$'s and row vectors a_j's
(b) Two views of $\mathcal{H}_4^3(a) : \mathbb{F}^4 \to \mathbb{F}^3$: 3-parallelization of $\mathcal{H}_4$ on a_j's and the linear combination of $a[i]$'s

Fig. 1. Parallelization of Hadamard codes.

The same as Hadamard codes, each $\mathcal{H}_k^d(\boldsymbol{a})$ is a linear function as described below.

Definition 2.13. Let $k, d \geq 1$ and $f : \mathbb{F}^k \to \mathbb{F}^d$.

(i) f is a *(group) homomorphism* if for all $\bar r, \bar r' \in \mathbb{F}^k$ we have

$$f(\bar r + \bar r') = f(\bar r) + f(\bar r').$$

(ii) If, in addition

$$f(r\bar r) = r f(\bar r)$$

holds for all $r \in \mathbb{F}$ and $\bar r \in \mathbb{F}^k$, then f is a *linear* function.

Lemma 2.14. *$f : \mathbb{F}^k \to \mathbb{F}^d$ is linear if and only if there is an $\boldsymbol{a} = (\boldsymbol{a}[1], \dots, \boldsymbol{a}[k]) \in (\mathbb{F}^d)^k$ such that*

$$f(\bar r) = r_1 \boldsymbol{a}[1] + \cdots + r_k \boldsymbol{a}[k]$$

for every $\bar r = (r_1, \dots, r_k) \in \mathbb{F}^k$. By (3) this means that

$$f = \mathcal{H}_k^d(\boldsymbol{a}).$$

In other words, $\mathcal{H}_k^d$ contains exactly all the linear functions from $\mathbb{F}^k$ to $\mathbb{F}^d$.

In Definition 2.13 we introduce linear functions via homomorphisms mainly because the linearity test (Proposition 2.16) we will use is in fact a group homomorphism test [29]. However for a prime field $\mathbb{F}$, group homomorphisms coincide with linear functions.

Lemma 2.15. *Assume that $\mathbb{F}$ is a prime field, i.e., $\mathbb{F} = \mathbb{F}_q$ with a prime q. Then any homomorphism $f : \mathbb{F}^k \to \mathbb{F}^d$ is a linear function.*

For our purpose, we need the following instantiation of Theorem 2.3 in [29].

Proposition 2.16 (*Linearity Test*)**.** *Assume that $\mathbb{F}$ is a prime field and $f : \mathbb{F}^k \to \mathbb{F}^d$. If*

$$\Pr_{\bar r, \bar r' \in \mathbb{F}^k}\left[f(\bar r) + f(\bar r') = f(\bar r + \bar r')\right] \geq 1 - \delta/2$$

where $\delta < 1/3$. Then there is an $\boldsymbol{a} \in (\mathbb{F}^d)^k$ such that

$$\Pr_{\bar r \in \mathbb{F}^k}\left[f(\bar r) = \mathcal{H}_k^d(\boldsymbol{a})(\bar r)\right] \geq 1 - \delta.$$

That is, f is $(1-\delta)$-close to the linear function $\mathcal{H}_k^d(\boldsymbol{a})$.

3. A simple combinatorial proof for the super constant inapproximability of clique

The first W[1]-hardness proof of the constant inapproximability of the parameterized clique problem was found in [11]. The reduction of [11] first transforms the k-Clique problem into the *parameterized vector sum problem* and then applies the parallelized vector encoding. This *vectorization* approach has inspired subsequent research [12,13,17,30] to establish parameterized inapproximability for k-Clique. By replacing the Hadamard code with Reed–Muller code based on quadratic polynomials, [30] improved the lower bounds of constant approximating algorithms for k-Clique to $n^{\Omega(\log k)}$ under ETH. By replacing the Hadamard code in [11] with list decoding Hadamard code, [12] shows the $k^{o(1)}$-ratio inapproximability of k-Clique under FPT $\neq$ W[1].

Surprisingly, a much simpler proof of the same $k^{o(1)}$-ratio inapproximability of k-Clique has been found in [13]. Technically it still uses Hadamard codes, yet without invoking the linearity test. This is more in the spirit of the *Network Coding*, and we refer the reader to [13] for a detailed discussion. In this section, we describe the proof in [13], since it is the simplest at present. The vectorization of k-Clique in [13] is done by the so-called *linear Sidon sets*.

Definition 3.1. Let $\mathbb{F}$ be a finite field and $d \geq 1$. A subset $S \subseteq \mathbb{F}^d$ is an *Linear Sidon set* if for all $r, r' \in \mathbb{F}^* (= \mathbb{F} \setminus \{0\})$ and $\boldsymbol{u}, \boldsymbol{u}', \boldsymbol{v}, \boldsymbol{v}' \in S$ with $\boldsymbol{u} \neq \boldsymbol{u}'$ and $\boldsymbol{v} \neq \boldsymbol{v}'$ we have

$$r\boldsymbol{u} + r'\boldsymbol{u}' = r\boldsymbol{v} + r'\boldsymbol{v}' \quad \Longrightarrow \quad \{\boldsymbol{u}, \boldsymbol{u}'\} = \{\boldsymbol{v}, \boldsymbol{v}'\}.$$

Linear Sidon sets generalize the Sidon sets which are studied in additive combinatorics [14–16]. A straightforward greedy algorithm yields:

Lemma 3.2. *Let $\mathbb{F} = \mathbb{F}_q$ be a finite field and $n \geq 1$. We set $d := \left\lceil \frac{3\log n}{\log q} + 3 \right\rceil$. Thus $n \leq q^{(d-3)/3}$. Then we can construct a linear Sidon set $S \subseteq \mathbb{F}^d$ with $|S| = n$ and $|\mathbb{F}^d| \leq q^4 \cdot n^3$ in time polynomial in $n + q$.*

Let (G, k) be an instance of k-Clique with $n := |V(G)|$. Moreover, let $\mathbb{F} := \mathbb{F}_q$ be a finite field whose q will be determined later. By Lemma 3.2 we can assume without loss of generality that

$$V(G) \subseteq \mathbb{F}^d \quad \text{for } d = \Theta(\log n / \log q)$$

is a linear Sidon set. Our goal is to turn (G, k) into a gap instance of k-Clique. To have a clear comparison with the construction in Section 4, we divide the reduction in [13] into two steps with an intermediate CSP instance.

3.1. From k-Clique to CSP

As usual we can assume that (G, k) is an instance of the *multi-colored* clique problem, i.e.,

$$V = V_1 \dot\cup \cdots \dot\cup V_k$$

with each $G[V_i]$ being an independent set. Hence any clique in G might pick at most one vertex in each V_i. Now for the desired CSP instance, we have variable set

$$X := \left\{ x_{\bar r} \;\middle|\; \bar r = (r_1, \dots, r_k) \in \mathbb{F}^k \right\}, \tag{5}$$

where each variable is supposed to take the value

$$x_{\bar r} := \mathcal{H}_k^d(\boldsymbol{v}_1, \dots, \boldsymbol{v}_k)(\bar r) = r_1 \boldsymbol{v}_1 + \cdots + r_k \boldsymbol{v}_k \in \mathbb{F}^d. \tag{6}$$

for a purported clique on vertices $\boldsymbol{v}_1 \in V_1, \dots, \boldsymbol{v}_k \in V_k$. Here, to lessen the notation burden, we use $x_{\bar r}$ (and others) both for a variable and for the value $\sigma(x_{\bar r})$ of this variable under some given assignment σ. It will cause no confusion, given the clear context. To check whether the assignment really gives us a clique in G, we have following tests as constraints.

Vertex Test. For every $i \in [k]$, $\bar{r} \in \mathbb{F}^k$, and $r \in \mathbb{F}^*$ test whether

$$x_{\bar{r}+re_i} - x_{\bar{r}} \in rV_i, \tag{7}$$

where $rV_i := \{ra \mid a \in V_i\}$. Here, a more accurate formulation (cf. Section 2.2) of this binary constraint is

$\varphi_{i,\bar{r},r} := (x_{\bar{r}+re_i} x_{\bar{r}}, C)$, where $C = \{ (c_1, c_2) \mid c_1, c_2 \in \mathbb{F}^d$ with $c_1 - c_2 \in rV_i \}$.

This is certainly less succinct than (7), and perhaps less intuitive too. So we will present all the constraints in the form of (7).

Edge Test. For every $\bar{r} = (r_1, \dots, r_k) \in \mathbb{F}^k$, $1 \le i < i' \le k$, and $r, r' \in \mathbb{F}^*$ test whether

$$x_{\bar{r}+re_i+r'e_{i'}} - x_{\bar{r}} = r\boldsymbol{v} + r\boldsymbol{v}' \tag{8}$$

for some edge $\{\boldsymbol{v}, \boldsymbol{v}'\} \in E(G)$ with $\boldsymbol{v} \in V_i$ and $\boldsymbol{v}' \in V_{i'}$. Here is a key observation. By Definition 3.1,

if such $\boldsymbol{v}$ and $\boldsymbol{v}'$ exist for (8), then they must be unique. (9)

Now we turn to the analysis of the above CSP instance. Let us call it Γ.

Definition 3.3. For an assignment of the variable set X in (5) for Γ, we say a subset $X' \subseteq X$ is *satisfied* if every constraint which only contains variables in X' is satisfied by this assignment.

Lemma 3.4 (*Completeness*). *If G has a k-clique, then there is an assignment for Γ that satisfies X, i.e., all constraints in Γ are satisfied.*

Proof. Let $\boldsymbol{v}_1 \in V_1, \dots, \boldsymbol{v}_k \in V_k$ form a k-clique in G. Then we set each variable $x_{\bar{r}}$ as in (6). It is routine to verify that the resulting assignment passes all the Vertex Tests and the Edge Tests. □

Lemma 3.5 (*Soundness*). *Assume that G has no k-clique and $X' \subseteq X$ is satisfied by an assignment for Γ, then*

$$|X'| \le kq^{k-1}.$$

Proof. We define

$$R := \{\bar{r} \in \mathbb{F}^k \mid x_{\bar{r}} \in X'\}$$

and observe that $|R| = |X'|$. Our goal is to show

$$\left|\mathbb{F}^k \setminus R\right| \ge \frac{(q-1)|R|}{k}, \tag{10}$$

which easily implies $|R| \le kq^{k-1}$. To that end, we use a *double-counting* argument.

Claim. *For every $\bar{r} \in R$ we argue that there is an $i \in [k]$ such that for all $r \in \mathbb{F}^*$ we have*

$$\bar{r} + re_i \notin R.$$

Proof of the claim. Otherwise, for every $i \in [k]$ there is an $r_i \in \mathbb{F}^*$ with $\bar{r} + r_i e_i \in R$, i.e., $x_{\bar{r}+r_ie_i} \in X'$. Then we define

$$\boldsymbol{v}_i := r_i^{-1}(x_{\bar{r}+r_ie_i} - x_{\bar{r}}).$$

Since $x_{\bar{r}+r_ie_i}, x_{\bar{r}} \in X'$, and X' is satisfied by our assignment, we conclude that $x_{\bar{r}+r_ie_i}$ and $x_{\bar{r}} \in X'$ must pass the Vertex Test, hence $\boldsymbol{v}_i \in V_i$. Furthermore, for every $1 \le i < i' \le k$ the variables $x_{\bar{r}+r_ie_i}$ and $x_{\bar{r}+r_{i'}e_{i'}}$ pass the Edge Test. Hence, for

$$x_{\bar{r}+r_ie_i} - x_{\bar{r}+r_{i'}e_{i'}} = (x_{\bar{r}+r_ie_i} - x_{\bar{r}}) - (x_{\bar{r}+r_{i'}e_{i'}} - x_{\bar{x}}) = r_i\boldsymbol{v}_i - r_{i'}\boldsymbol{v}_{i'} = r_i\boldsymbol{v}_i + (-r_{i'})\boldsymbol{v}_{i'}$$

we have $\{\boldsymbol{v}_i, \boldsymbol{v}_{i'}\} \in E(G)$ by our observation (9). This implies that $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_k\}$ is a k-clique in G, contradicting our assumption. ⊣

Now for every $\bar{r} \in R$ we fix an $i_{\bar{r}} \in [k]$ as in the above claim and define

$$T_{\bar{r}} := \{\bar{r} + re_{i_{\bar{r}}} \mid r \in \mathbb{F}^*\}.$$

So $T_{\bar{r}} \cap R = \emptyset$, and we might say that $\bar{r}$ kills all the indices in $T_{\bar{r}}$ from the $i_{\bar{r}}$th direction. As $T_{\bar{r}} \subseteq \mathbb{F}^k \setminus R$, to establish (10) it suffices to prove

$$\left|\bigcup_{\bar{r}\in R} T_{\bar{r}}\right| \ge \frac{(q-1)|R|}{k} \tag{11}$$

Clearly each $T_{\bar{r}}$ has size $|T_{\bar{r}}| = q - 1$. On the other hand, consider $\bar{s} \in T_{\bar{r}} \cap T_{\bar{r}'}$ for two distinct $\bar{r}, \bar{r}' \in R$. In other words, $\bar{s}$ is killed both by $\bar{r}$ and $\bar{r}'$ from the $i_{\bar{r}}$th and $i_{\bar{r}'}$th direction respectively. Then $i_{\bar{r}} \ne i_{\bar{r}'}$, otherwise $\bar{r}'$ would have been killed by $\bar{r}$ from the $i_{\bar{r}}$th direction, and cannot be in R. Therefore, each $\bar{s} \in \bigcup_{\bar{r}\in R} T_{\bar{r}}$ can appear in at most k different $T_{\bar{r}}$'s, Hence (11) follows. □

3.2. From CSP *to* k-Clique

Vertex Set. For every $\bar{r} \in \mathbb{F}^k$ we have a set

$$V_{\bar{r}} := \mathbb{F}^d.$$

A vertex $\boldsymbol{v} \in V_{\bar{r}}$ represents an assignment of the variable $x_{\bar{r}}$ by $\boldsymbol{v}$. Then we take

$$V(G') := \bigcup_{\bar{r}\in\mathbb{F}^k} V_{\bar{r}}.$$

Edge Set. We add an edge between two vertices in $V(G')$, if

- they do not assign different values to a same variable, which means that each $V_{\bar{r}}$ is an independent set in G',
- and the assignment does not fail any Vertex Test or Edge Test. For example, consider $\bar{r} \in \mathbb{F}^k$, $i \in [k]$, $r \in \mathbb{F}^*$, $\boldsymbol{u} \in V_{\bar{r}}$ and $\boldsymbol{v} \in V_{\bar{r}+re_i}$. Then there is an edge between $\boldsymbol{u}$ and $\boldsymbol{v}$ if and only if $\boldsymbol{u} - \boldsymbol{v} \in rV_i$. That is, the assignment $x_{\bar{r}} := \boldsymbol{u}$ and $x_{\bar{r}+re_i} := \boldsymbol{v}$ passes the Vertex Test (7).

It is routine to verify that:

Lemma 3.6. *For any $k' \ge 1$ the graph G' has a clique of size k' if and only if there is an assignment of X for Γ which satisfies a subset $X' \subseteq X$ of size $|X'| = k'$.*

Equipped with Lemmas 3.4, 3.5, and 3.6, we have the desired gap-creating self-reduction of k-Clique.

Proposition 3.7. *There is an* fpt*-reduction which computes from any graph G, $k \ge 1$, and a prime power q which only depends on k a graph G' such that*

(i) *if G has a k-clique, then G' has a clique of size q^k;*

(ii) *if G has no k-clique, then a maximum clique in G' has size at most kq^{k-1}.*

The above reduction creates a gap of q^k vs. kq^{k-1} between yes- and no-instances of k-Clique. With a bit more technical work [13], we can slightly increase the gap to q^k vs. q^{k-1}. Then by choosing q as any sufficiently large function of k we conclude:

Theorem 3.8 (*[12,13]*). *The k-Clique problem has no* fpt*-approximation with ratio $k^{o(1)}$, unless* FPT = W[1].

4. Inapproximability of clique under ETH

In this section, we describe the reductions in [11] and its subsequent work [17] that gave a general framework to prove the parameterized inapproximability of k-Clique under ETH. In the process of proving the ETH-based inapproximability of k-Clique, [17] proposed a *vectorization* of k-variable constraint satisfaction problems. This idea inspired [18] to introduce VecCSP, which plays a key role in the proof of parameterized inapproximability hypotheses under ETH.

To begin with, the reduction in [11] starts from the *parameterized vector sum problem*:

k-VectorSum	
Instance:	A finite field $\mathbb{F}$, $d, k \geq 1$, $V_1, \ldots, V_k \subseteq \mathbb{F}^d$, and $\boldsymbol{t} \in \mathbb{F}^d$.
Parameter:	k.
Problem:	Decide whether there are $\boldsymbol{v}_1 \in V_1, \ldots, \boldsymbol{v}_k \in V_k$ with $\sum_{i\in[k]} \boldsymbol{v}_i = \boldsymbol{t}$.

Proposition 4.1 ([11,31]). *k-VectorSum is W[1]-hard. More precisely, for any fixed finite field $\mathbb{F}$ there is an fpt-reduction from k-Clique to k-VectorSum which on an input n-vertex graph G and $k \geq 1$ produces an instance $(\mathbb{F}, d, k', V_1, \ldots, V_{k'}, \boldsymbol{t})$ of k-VectorSum of the form*

$$\left(\mathbb{F}, \underbrace{\Theta(k^2 \log n)}_{d}, \underbrace{k + \binom{k}{2}}_{k'}, V_1, \ldots, V_{k+\binom{k}{2}}, \boldsymbol{t}\right), \tag{12}$$

where each V_i has size $n^{O(1)}$.

We often refer to an instance of k-VectorSum as $(V_1, \ldots, V_k, \boldsymbol{t})$, ignoring $\mathbb{F}$, d, and k which are implicit in $(V_1, \ldots, V_k, \boldsymbol{t})$. In essence, k-VectorSum can be viewed as an algebraization of k-Clique, which enables us to apply a wealth of algebraic tools in order to create and amplify the desired gap between yes- and no-instances of k-Clique.

From now on, we assume that $\mathbb{F}$ is a prime field for which k-VectorSum remains W[1]-hard. The reduction from k-VectorSum to k-Clique is divided into two steps via an appropriate intermediate CSP instance.

4.1. From k-VectorSum to CSP

For a k-VectorSum instance $(V_1, \ldots, V_k, \boldsymbol{t})$ with $V_1, \ldots, V_k \subseteq \mathbb{F}^d$ and $\boldsymbol{t} \in \mathbb{F}^d$, we create a CSP instance as follows. The variable set is

$$X := \left\{ x_{\bar{r}} \;\middle|\; \bar{r} = (r_1, \ldots, r_k) \in \mathbb{F}^k \right\},$$

where each variable is supposed to be set as

$$x_{\bar{r}} := \mathcal{H}_k^d(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)(\bar{r}) = r_1\boldsymbol{v}_1 + \cdots + r_k\boldsymbol{v}_k \in \mathbb{F}^d \tag{13}$$

for some purported solution $\boldsymbol{v}_1 \in V_1, \ldots, \boldsymbol{v}_k \in V_k$. Thereby the constraints are designed to test whether the values of these variables are coming from a yes instance of k-VectorSum.

Codeword Test. For every $\bar{r}, \bar{r}' \in \mathbb{F}^k$ test whether

$$x_{\bar{r}} + x_{\bar{r}'} = x_{\bar{r}+\bar{r}'}. \tag{14}$$

Similar to (7), the exact definition of the constraint (14) is

$$\varphi_{\bar{r},\bar{r}'} := \left(x_{\bar{r}}x_{\bar{r}'}x_{\bar{r}+\bar{r}'}, C\right), \quad \text{where } C = \left\{ (\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3) \;\middle|\; \boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3 \in \mathbb{F}^d \text{ with } \boldsymbol{c}_1 + \boldsymbol{c}_2 = \boldsymbol{c}_3 \right\}.$$

Membership Test. For every $i \in [k]$, $\bar{r} \in \mathbb{F}^k$, and $r \in \mathbb{F}$ test whether

$$x_{\bar{r}+re_i} - x_{\bar{r}} \in rV_i, \tag{15}$$

where $rV_i := \left\{ r\boldsymbol{a} \;\middle|\; \boldsymbol{a} \in V_i \right\}$.

Sum Test. For every $\bar{r} = (r_1, \ldots, r_k) \in \mathbb{F}^k$, and $r \in \mathbb{F}$ test whether

$$x_{\bar{r}+r\mathbf{1}} - x_{\bar{r}} = r\boldsymbol{t}. \tag{16}$$

Note $\bar{r} + r\mathbf{1} = \left(r_1 + r, \ldots, r_k + r\right)$.

In some sense, the Membership Test corresponds to the Vertex Test in Section 3.1 and the Sum Test to the Edge Test.

For every $\bar{r} = (r_1, \ldots, r_k) \in \mathbb{F}^k$ and $i \in [k]$ we define a set

$$S_{\bar{r},i} := \left\{ (r_1, \ldots, r_i + r, \ldots, r_k) \;\middle|\; r \in \mathbb{F} \right\},$$

which contains all the indices of $x_{\bar{r}+re_i}$ queried in the Membership Tests (15). Similarly

$$S_{\bar{r}} := \left\{ (r_1 + r, \ldots, r_k + r) \;\middle|\; r \in \mathbb{F} \right\},$$

is the set of all the indices of $x_{\bar{r}+r\mathbf{1}}$ queried in the Sum Tests (16). The next technical lemma is key to our analysis for gap of the clique instances we will construct from the above CSP instances.

Lemma 4.2.

(i) *For every $\bar{r} \in \mathbb{F}^k$ and $i \in [k]$ we have $|S_{\bar{r}}| = |S_{\bar{r},i}| = |\mathbb{F}|$.*

(ii) *Let $\bar{r}, \bar{r}' \in \mathbb{F}^k$. Then either $S_{\bar{r}} = S_{\bar{r}'}$ or $S_{\bar{r}} \cap S_{\bar{r}'} = \emptyset$. So (i) implies that $\mathbb{F}^k$ can be partitioned into $|\mathbb{F}|^{k-1}$ pairwise disjoint $S_{\bar{r}}$'s.*

(iii) *Let $i \in [k]$. Then for every $\bar{r}, \bar{r}' \in \mathbb{F}^k$ either $S_{\bar{r},i} = S_{\bar{r}',i}$ or $S_{\bar{r},i} \cap S_{\bar{r}',i} = \emptyset$. So (i) implies that $\mathbb{F}^k$ can be partitioned into $|\mathbb{F}|^{k-1}$ pairwise disjoint $S_{\bar{r},i}$'s.*

Lemma 4.2 follows easily from the definitions of $S_{\bar{r},i}$'s and $S_{\bar{r}}$'s.

4.2. From CSP to k-Clique – the modified FGLSS-reduction

Now we construct a graph G' from a CSP instance constructed in Section 4.1.

Vertex Set. For every $\bar{r}, \bar{r}' \in \mathbb{F}^k$ let

$$V_{\bar{r},\bar{r}'} := \left\{ (\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}) \in (\mathbb{F}^d)^3 \;\middle|\; \boldsymbol{u} + \boldsymbol{v} = \boldsymbol{w} \right\}. \tag{17}$$

Each vertex in $V_{\bar{r},\bar{r}'}$ is understood as an assignment of the variables $x_{\bar{r}}$, $x_{\bar{r}'}$, and $x_{\bar{r}+\bar{r}'}$ which passes the Codeword Test.[3]

Then

$$V(G') := \bigcup_{\bar{r},\bar{r}' \in \mathbb{F}^k} V_{\bar{r},\bar{r}'}.$$

Edge Set. We add an edge between two vertices in $V(G')$, if they do not assign different values to a same variable, and if the assignment does not fail any Membership Test or Sum Test. For example, each $V_{\bar{r},\bar{r}'}$ is an independent set in G'. For another example, assume $(\boldsymbol{u}_1, \boldsymbol{v}_1, \boldsymbol{w}_1)$ is in some $V_{\bar{r},\bar{r}'}$ and $(\boldsymbol{u}_2, \boldsymbol{v}_2, \boldsymbol{w}_2) \in V_{\bar{r}+re_i,\bar{r}'}$, then they are adjacent in G' if and only if $\boldsymbol{u}_2 - \boldsymbol{u}_1 \in rV_i$ according to (15).

Note that, all the Codeword Tests of ternary constraints has been taken care of by our definition (17) of $V_{\bar{r},\bar{r}'}$. Therefore, they do not need to be captured by our edge construction.

Remark 4.3. The original FGLSS-reduction [25] (see also the proof of Theorem 2.9) constructs a graph G from a given CSP instance in such a way that every vertex in G corresponds to a partial satisfying assignment of the variables in a constraint, and edges are added between vertices that assign the same value to shared variables. Therefore, all the constraints are already satisfied by the choices of vertices.

For the above modified FGLSS-reduction, the constraints underlying the Membership Test and the Sum Test are checked by edges in the graph. ⊣

[3] In the original reduction in [11], $V(G')$ also contains subsets $U_{\bar{r},i} := \mathbb{F}^d$ for every $i \in [|\mathbb{F}^k|]$ and $\bar{r} \in \mathbb{F}^k$. Here we adopt the approach of [12].

4.3. The correctness of the reduction

For the completeness, say we have a yes-instance of k-VectorSum, i.e., for some $\boldsymbol{v}_1 \in V_1, \dots, \boldsymbol{v}_k \in V_k$ it holds that $\boldsymbol{v}_1 + \cdots + \boldsymbol{v}_k = \boldsymbol{t}$. To construct a clique in G', let

$$f := \mathcal{H}_k^d(\boldsymbol{v}_1, \dots, \boldsymbol{v}_k),$$

i.e., f is the codeword of $(\boldsymbol{v}_1, \dots, \boldsymbol{v}_k) \in (\mathbb{F}^d)^k$ in the d-parallelized Hadamard code. Then

- for every $\bar{r}, \bar{r}' \in \mathbb{F}^k$ we pick the vertex $\big(f(\bar{r}), f(\bar{r}'), f(\bar{r}+\bar{r}')\big) \in V_{\bar{r},\bar{r}'}$.

It is straightforward to verify that this gives us a clique of size $|\mathbb{F}|^{2k}$ in G'.

Now we turn to the soundness. In particular, for a no-instance $(V_1, \dots, V_k, \boldsymbol{t})$ of k-VectorSum we show the graph G' contains no clique of size

$$(1-\varepsilon)|\mathbb{F}|^{2k}$$

for a small constant $\varepsilon > 0$. Towards a contradiction, let X be a clique in G' with

$$|X| \geq (1-\varepsilon)|\mathbb{F}|^{2k}. \tag{18}$$

We define

$$J_X := \left\{(\bar{r}, \bar{r}') \in (\mathbb{F}^k)^2 \;\middle|\; X \cap V_{\bar{r},\bar{r}'} \neq \emptyset\right\}.$$

$$I_X := \{\bar{a} \in \mathbb{F}^k \mid \text{there exists } (\bar{r}, \bar{r}') \in J_X, \text{ such that } \bar{a} \in \{\bar{r}, \bar{r}', \bar{r}+\bar{r}'\}\}.$$

In addition, we define a function $f : \mathbb{F}^k \to \mathbb{F}^d$ where

$$f(\bar{r}) = \begin{cases} \boldsymbol{u} & \text{if } X \cap V_{\bar{r},\bar{r}'} = \{(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{u}+\boldsymbol{v})\} \text{ for some } \bar{r}' \in \mathbb{F}^k \\ \boldsymbol{v} & \text{if } X \cap V_{\bar{r}',\bar{r}} = \{(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{u}+\boldsymbol{v})\} \text{ for some } \bar{r}' \in \mathbb{F}^k \\ \boldsymbol{u}+\boldsymbol{v} & \text{if } X \cap V_{\bar{r}-\bar{r}',\bar{r}'} = \{(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{u}+\boldsymbol{v})\} \text{ for some } \bar{r}' \in \mathbb{F}^k \\ \boldsymbol{0} & \text{otherwise} \end{cases}$$

for every $\bar{r} \in \mathbb{F}^k$. Note that by the definition of edge set, the vertices from a clique X will not assign different vectors to the same index $\bar{r}$. Thus, the function f is well-defined. In fact, f is viewed as an assignment to the instance of our CSP. In particular, we apply all the Codeword Test (14), the Membership Test (15), and the Sum Test (16) to the assignment $x_{\bar{r}} := f(\bar{r})$ for every $\bar{r} \in \mathbb{F}^k$.

For each $(\bar{r}, \bar{r}') \in J_X$ we consider the linearity test (i.e. the Codeword Test (14)) on variables $x_{\bar{r}}$, $x_{\bar{r}'}$, and $x_{\bar{r}+\bar{r}'}$. Then by our definition of G' and f, we conclude

$$f(\bar{r}) + f(\bar{r}') = f(\bar{r}+\bar{r}'), \quad \text{i.e., } x_{\bar{r}} + x_{\bar{r}'} = x_{\bar{r}+\bar{r}'}.$$

That is, we pass the Codeword Test on $x_{\bar{r}}$, $x_{\bar{r}'}$, and $x_{\bar{r}+\bar{r}'}$. On the other hand, since $|X| \geq (1-\varepsilon)|\mathbb{F}|^{2k}$,

$$|J_X| \geq (1-\varepsilon)|\mathbb{F}|^{2k}.$$

Then Proposition 2.16 implies that there are $\boldsymbol{v}_1, \dots, \boldsymbol{v}_k \in \mathbb{F}^d$ with

$$\Pr_{\bar{r} \in \mathbb{F}^k}\left[f(\bar{r}) = \mathcal{H}_k^d(\boldsymbol{v}_1, \dots, \boldsymbol{v}_k)(\bar{r})\right] \geq 1 - 2\varepsilon.$$

That is, let $I := \left\{\bar{r} \in \mathbb{F}^k \;\middle|\; f(\bar{r}) = \mathcal{H}_k^d(\boldsymbol{v}_1, \dots, \boldsymbol{v}_k)(\bar{r})\right\}$, we have

$$|I| \geq (1-2\varepsilon)|\mathbb{F}|^k. \tag{19}$$

Since $(1-\varepsilon)|\mathbb{F}^{2k}| \leq |J_X| \leq |I_X|^2$, we have $|I_X| \geq \sqrt{1-\varepsilon}|\mathbb{F}^k| \geq (1-\varepsilon)|\mathbb{F}^k|$ for $0 \leq \varepsilon \leq 1$. Thus $|I \cap I_X| \geq (1-3\varepsilon)|\mathbb{F}^k|$. We conclude that the set

$$R := I \cap I_X = \left\{\bar{r} \in I_X \;\middle|\; f(\bar{r}) = \mathcal{H}_k^d(\boldsymbol{v}_1, \dots, \boldsymbol{v}_k)(\bar{r})\right\}$$

has size

$$|R| \geq (1-3\varepsilon)|\mathbb{F}|^k > |\mathbb{F}|^{k-1},$$

when $\mathbb{F}$ is chosen sufficiently large. Now we invoke Lemma 4.2. Let $i \in [k]$. By the Pigeonhole Principle there is an $\bar{r} \in \mathbb{F}^k$ and $S_{\bar{r},i}$ such that

$$|R \cap S_{\bar{r},i}| \geq 2.$$

So there exist $r, r' \in \mathbb{F}$ with $\bar{r} \neq \bar{r}'$ such that

$$\bar{r} + r\boldsymbol{e}_i, \bar{r} + r'\boldsymbol{e}_i \in R.$$

We deduce

$$\begin{aligned} f(\bar{r} + r\boldsymbol{e}_i) - f(\bar{r} + r'\boldsymbol{e}_i) &= \mathcal{H}_k^d(\boldsymbol{v}_1, \dots, \boldsymbol{v}_k)(\bar{r} + r\boldsymbol{e}_i) - \mathcal{H}_k^d(\boldsymbol{v}_1, \dots, \boldsymbol{v}_k)(\bar{r} + r'\boldsymbol{e}_i) \\ &= \mathcal{H}_k^d(\boldsymbol{v}_1, \dots, \boldsymbol{v}_k)\big((r-r')\boldsymbol{e}_i\big) = (r-r')\boldsymbol{v}_i. \end{aligned}$$

Since, $\bar{r} + r\boldsymbol{e}_i, \bar{r} + r'\boldsymbol{e}_i \in R \subseteq I_X$, we must have passed the Membership Test $\boldsymbol{v}_i \in V_i$. For a similar reason, using the Sum Test, we conclude $\boldsymbol{v}_1 + \cdots + \boldsymbol{v}_k = \boldsymbol{t}$. This contradicts our assumption that $(V_1, \dots, V_k, \boldsymbol{t})$ is a no-instance of k-VectorSum.

This proves the correctness of our reduction. However, we observe that the size of the graph G' is

$$|\mathbb{F}|^{O(k+d)}.$$

As shown (12) in Proposition 4.1 we need to set the dimension $d := \Theta(k^2 \log n)$ for $\mathbb{F}^d$. Therefore the size of G' becomes $n^{\Theta(k^2)}$ which is not allowed by an fpt-reduction.

4.4. Dimension reduction

In order to reduce the dimension d, [11] uses two more techniques:

- For a given k-VectorSum instance $(\mathbb{F}, d, k, V_1, \dots, V_k, \boldsymbol{t})$ we map vectors $\boldsymbol{v}$ in each $V_i \subseteq \mathbb{F}^d$ to a vector in $(\mathbb{F}^h)^\ell$ with $h = \Theta(k^2)$ and $\ell = \Theta(k + \log n)$ by multiplying $\boldsymbol{v}$ with a sequence of ℓ *random matrices* (see (27)). The resulting instance is in essence still a k-VectorSum instance (see (28)).
- We replace the parallelized Hadamard code $\mathcal{H}_k^d$ with the parallelized *high-order* Hadamard code $\boldsymbol{\mathcal{H}}_{h,k}^\ell$ as in Definition B.4.

Then the previous construction and analysis can be carried over, and we get the desired constant gap instances for k-Clique by an fpt-reduction. We include the detail in Appendix B.

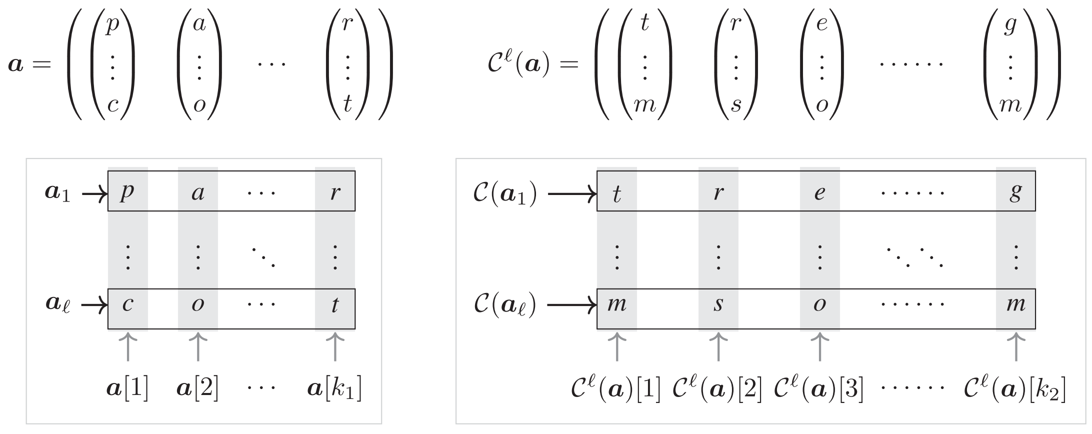
Instead of doing dimension reduction after the encoding, the authors of [17] observed that doing the dimension reduction before applying the parallel encoding can simply the proof. They define a class of CSP's called *vector 2CSP*.

Definition 4.4 (*Vector 2CSP*). A vector 2CSP instance (X, Σ, Φ) is a special 2CSP instance, where:

- The variable set is $X = \{x_1, \dots, x_k\}$.
- $\Sigma = \mathbb{F}^d$, i.e., The alphabet consists of all d-dimensional vectors over a field $\mathbb{F}$.
- $\boldsymbol{\Phi} = \{\varphi_i\}_{i=1}^k \bigcup \{\varphi_{i,j}\}_{1\leq i<j\leq k}$ consists of $k + \binom{k}{2}$ constraints, where:
  - For each $i \in [k]$, the constraint φ_i concerns a single variable x_i and is of the form $x_i \in S_i$, where S_i is a prescribed subset of $\mathbb{F}^d$.
  - For each $1 \leq i < j \leq k$, the constraint $\varphi_{i,j}$ concerns two variables x_i, x_j and is of the form $x_i + x_j \in S_{i,j}$, where $S_{i,j}$ is also a prescribed subset of $\mathbb{F}^d$.

It is not hard to apply the Hadamard code to vector 2CSP's and obtain gap instances of k-Clique. It remains to show that the vector 2CSP parameterized by the number of variables is W[1]-hard when $d = \Theta(\log n)$. In [17], the authors give an fpt-reduction from k-Clique to a k-variable vector 2CSP problem with $d = 5 \cdot \frac{\log n}{\log |\mathbb{F}|}$. The reduction is based on the following fact:

- Let $f : V \to \mathbb{F}^d$ be a random function from V to a d-dimension vector space with $d = C \log n / \log |\mathbb{F}|$. If C is large enough, then with high probability, $f(v) \neq f(v')$ and $f(v) + f(u) \neq f(v') + f(u')$ for $v \neq v'$ and $(v, u) \neq (v', u')$.



$\boldsymbol{a} = \left(\begin{pmatrix} p \\ \vdots \\ c\end{pmatrix} \begin{pmatrix} a \\ \vdots \\ o\end{pmatrix} \cdots \begin{pmatrix} r \\ \vdots \\ t\end{pmatrix}\right)$
$\mathcal{C}^\ell(\boldsymbol{a}) = \left(\begin{pmatrix} t \\ \vdots \\ m\end{pmatrix} \begin{pmatrix} r \\ \vdots \\ s\end{pmatrix} \begin{pmatrix} e \\ \vdots \\ o\end{pmatrix} \cdots\cdots \begin{pmatrix} g \\ \vdots \\ m\end{pmatrix}\right)$
$\boldsymbol{a}_1$
$\boldsymbol{a}_\ell$
$\boldsymbol{a}[1]$ $\boldsymbol{a}[2]$ $\cdots$ $\boldsymbol{a}[k_1]$
$\mathcal{C}(\boldsymbol{a}_1)$
$\mathcal{C}(\boldsymbol{a}_\ell)$
$\mathcal{C}^\ell(\boldsymbol{a})[1]$ $\mathcal{C}^\ell(\boldsymbol{a})[2]$ $\mathcal{C}^\ell(\boldsymbol{a})[3]$ $\cdots\cdots$ $\mathcal{C}^\ell(\boldsymbol{a})[k_2]$

(a) $\boldsymbol{a} = (\boldsymbol{a}[1], \ldots, \boldsymbol{a}[k_1]) \in (\Sigma_1^\ell)^{k_1}$ with $\boldsymbol{a}_1 = parameter$ and $\boldsymbol{a}_\ell = complexit$

(b) $\mathcal{C}(parameter) = treewidthmodelchecking$ and $\mathcal{C}(complexit) = msodefcourcelletheorem$

Fig. 2. Parallelization of arbitrary codes.

Note that the simple proof of [13] is also based on a similar property of the Sidon sets.

The benefit of defining vector 2CSP is that it provides a framework for proving inapproximability of k-Clique. To show better inapproximability results for k-Clique, it suffices to design better codes.

4.5. Going beyond the $k^{o(1)}$ lower bound under ETH

In the framework of [11] the main obstacle to lifting the constant gap to super constant is that the linearity test in Proposition 2.16 (and Proposition B.5) only works when the acceptance regime is close to 1 (i.e., $1-\delta/2$ in Proposition 2.16). This is independent of the choice of $\mathbb{F}$ and inherent for the local testability of Hadamard codes. To break this barrier, the authors of [12] use prime fields $\mathbb{F} = \mathbb{F}_q$ where q increases with k, and replace the linearity test of Hadamard codes by the list decoding of Hadamard codes over $\mathbb{F}_q$. Thereby they are able to show the inapproximability of ratio $k^{o(1)}$ with a more involved construction and a more intricate soundness analysis.

Assuming FPT $\neq$ W[1], the best lower bound so far for the parameterized approximation of the k-Clique problem is Theorem 3.8. However, better lower bounds are known under ETH [17], which we explain now. As we have seen in Section 4 that the parallelization of Hadamard Codes plays a central role in creating the desired gap for k-Clique. So it is natural to apply this idea to other codes with better error-correcting properties.

Definition 4.5. Let Σ_1 and Σ_2 be two finite alphabets, $k_1, k_2 \geq 1$, and $C : \Sigma_1^{k_1} \to \Sigma_2^{k_2}$. That is, C encodes every string of length k_1 over Σ_1 by a string of length k_2 over Σ_2. Then for every $\ell \geq 1$ the *ℓ-parallelization of C* is the function

$$C^\ell : (\Sigma_1^\ell)^{k_1} \to (\Sigma_2^\ell)^{k_2}$$

defined by

$$C^\ell(\boldsymbol{a}) = \left(\begin{pmatrix} C(\boldsymbol{a}_1)[1] \\ \vdots \\ C(\boldsymbol{a}_\ell)[1]\end{pmatrix} \begin{pmatrix} C(\boldsymbol{a}_1)[2] \\ \vdots \\ C(\boldsymbol{a}_\ell)[2]\end{pmatrix} \cdots \begin{pmatrix} C(\boldsymbol{a}_1)[k_2] \\ \vdots \\ C(\boldsymbol{a}_\ell)[k_2]\end{pmatrix}\right)$$

for every $\boldsymbol{a} \in (\Sigma_1^\ell)^k$, which is depicted in Fig. 2. Note that, the same as in (2), for every $j \in [\ell]$ we let

$$\boldsymbol{a}_j := (\boldsymbol{a}[1][j], \ldots, \boldsymbol{a}[k][j]) \in \Sigma_1^\ell.$$

It is observed in [17] that the key error-correcting properties we need for the proofs of [11] are the so-called parallel local testability and decodability.

Definition 4.6 (*Parallel Locally Testable Codes*)**.** Let $C : (\Sigma_1)^{k_1} \to (\Sigma_2)^{k_2}$, $p \geq 1$, and $\delta, \varepsilon > 0$. Then the code C is *(p, δ, ε)-parallel locally testable* if there is a probabilistic algorithm $\mathbb{T}$ which on input $\ell \geq 1$ and $\boldsymbol{w} \in (\Sigma_2^\ell)^{k_2}$ tests whether $\boldsymbol{w}$ is close to some codeword in $C^\ell((\Sigma_1^\ell)^{k_1}) \subseteq (\Sigma_2^\ell)^{k_2}$ with the following properties.

Non-adaptiveness. The tester $\mathbb{T}$ randomly chooses p positions in $[k_2]$, and the choices only depend on k_2 but neither ℓ nor w. Then $\mathbb{T}$ queries these p positions on $\boldsymbol{w}$ and decides deterministically whether accepts $\boldsymbol{w}$ or not in polynomial time. In particular, this means $\mathbb{T}$ is *non-adaptive*.

Perfect completeness. If $\boldsymbol{w}$ is a codeword, i.e., there is an $\boldsymbol{a} \in (\Sigma_1^\ell)^{k_1}$ with $\boldsymbol{w} = C^\ell(\boldsymbol{a})$, then $\mathbb{T}$ accepts $\boldsymbol{w}$ with probability 1.

Local testability. If $\mathbb{T}$ accepts $\boldsymbol{w}$ with probability at least ε, then there is an $\boldsymbol{a} \in (\Sigma_1^\ell)^{k_1}$ such that

$$\Pr_{i \in [k_2]}[\boldsymbol{w}[i] = C^\ell(\boldsymbol{a})[i]] \geq 1 - \delta.$$

Remark 4.7. Propositions 2.16 and B.5 show that Hadamard codes $\mathcal{H}_k$ and high-order Hadamard codes $\boldsymbol{\mathcal{H}}_{h,k}$ are $(3, \delta, 1 - \delta/2)$-parallel locally testable for any $\delta < 1/3$. ⊣

Definition 4.8 (*Parallel Smoothly Locally Decodable Codes*)**.** Let $C : (\Sigma_1)^{k_1} \to (\Sigma_2)^{k_2}$, $p \geq 1$, and $\delta, \varepsilon > 0$. Moreover, let $g : \Sigma_1^{k_1} \to A$ be a function for some set A. We say that C is *(p, δ, ε)-parallel smoothly locally decodable with respect to* g if there is a probabilistic algorithm $\mathbb{D}_g$ with input $\ell \geq 1$ and $\boldsymbol{w} \in (\Sigma_2^\ell)^{k_2}$ whose goal is to compute

$$g^\ell(\boldsymbol{a}) \quad \text{if } \boldsymbol{w} = C^\ell(\boldsymbol{a}[1], \ldots, \boldsymbol{a}[k_1]) \text{ for some } \boldsymbol{a} \in (\Sigma_1^\ell)^{k_1}.$$

Here $g^\ell : (\Sigma_1^\ell)^{k_1} \to A^\ell$ is the *ℓ-parallelization* of g defined in the same way as Definition 4.5. In other words, $\mathbb{D}_g$ first decodes the codeword $\boldsymbol{w}$ to $\boldsymbol{a}$, and then computes $g^\ell(\boldsymbol{a})$. Similarly to Definition 4.6, $\mathbb{T}_g$ satisfies some additional properties.

Non-adaptiveness. The decoder $\mathbb{D}_g$ randomly chooses p positions in $[k_2]$, and the choices only depend on k_2 but neither ℓ nor w. Then $\mathbb{D}_g$ queries these p positions on $\boldsymbol{w}$ and in deterministically polynomial time outputs a value in A^ℓ.

Smoothness. Each position in $[k_2]$ is queried with equal probability p/k_2.[4]

[4] Suppose $\{x_1, \ldots, x_p\} \in \binom{[k_2]}{p}$ is a random set of positions queried by $\mathbb{D}_g$. For every $i \in [k_2]$ and $j \in [p]$, we assume that the probability of "$x_j = i$" is $1/k_2$. Note that for $j, j' \in [p]$ with $j \neq j'$, the events "$x_j = i$" and "$x_{j'} = i$" are disjoint, which implies that $\Pr[i \in \{x_1, \ldots, x_p\}] = p/k_2$.

Perfect completeness. If $\boldsymbol{w} = C^{\ell}(\boldsymbol{a})$ for some $\boldsymbol{a} \in \left(\Sigma_1^{\ell}\right)^{k_1}$, then

$$\Pr\left[\mathbb{D}_g(\boldsymbol{w}) = g(\boldsymbol{a})\right] = 1.$$

Local decodability. If there is an $\boldsymbol{a} \in \left(\Sigma_1^{\ell}\right)^{k_1}$ with

$$\Pr_{i\in[k_2]}\left[\boldsymbol{w}[i] = C^{\ell}(\boldsymbol{a})[i]\right] \geq 1-\delta,$$

then

$$\Pr\left[\mathbb{D}_g(\boldsymbol{w}) = g(\boldsymbol{a})\right] \geq \varepsilon.$$

Remark 4.9. Consider Hadamard code $\mathcal{H}_k : \mathbb{F}^k \to (\mathbb{F}^k \to \mathbb{F})$, which can be recast as a code in the form of Definitions 4.6 and 4.8 with $k_1 \leftarrow k$, $k_2 \leftarrow |\mathbb{F}|^k$, $\Sigma_1 \leftarrow \mathbb{F}$ and $\Sigma_2 \leftarrow \mathbb{F}$

$$\underbrace{\mathcal{H}_k}_{C} : \underbrace{\mathbb{F}^k}_{\Sigma_1^{k_1}} \to \underbrace{\mathbb{F}^{|\mathbb{F}|^k}}_{\Sigma_2^{k_2}}.$$

Let $sum : \mathbb{F}^k \to \mathbb{F}$ be the sum function, i.e., for every $\boldsymbol{\alpha} \in \mathbb{F}^k$ we have

$$sum(\boldsymbol{\alpha}) = \boldsymbol{\alpha}[1] + \cdots + \boldsymbol{\alpha}[k].$$

A desired decoder $\mathbb{T}_{sum}$, on input $\ell \geq 1$ and $\boldsymbol{w} \in (\mathbb{F}^{\ell})^{|\mathbb{F}|^k}$ which can be understood as a *function* $\boldsymbol{w} : \mathbb{F}^k \to \mathbb{F}^{\ell}$,

- chooses $\bar{r} = (\boldsymbol{r}_1, \dots, \boldsymbol{r}_k) \in \mathbb{F}^k$ and $r \in \mathbb{F}$ both uniformly at random,
- queries two positions $\boldsymbol{w}(\bar{r})$ and $\boldsymbol{w}(\bar{r} + r\mathbf{1}) = \boldsymbol{w}(r_1 + r, \dots, r_k + r)$,
- and outputs

$$r^{-1}\left(\boldsymbol{w}(\bar{r} + r\mathbf{1}) - \boldsymbol{w}(\bar{r})\right).$$

The reader might notice that $\mathbb{T}_{sum}$ captures the Sum Test in Section 4.1. Assume that there is an $\boldsymbol{a} = (\boldsymbol{a}[1], \dots, \boldsymbol{a}[k]) \in \left(\mathbb{F}^{\ell}\right)^k$ such that

$$\Pr_{\bar{r}\in\mathbb{F}^k}\left[\boldsymbol{w}(\bar{r}) = \mathcal{H}_k^{\ell}(\boldsymbol{a})(\bar{r})\right] \geq 1-\delta$$

for some $\delta < 1/2$. Then a simple union bound yields

$$\Pr_{\bar{r}\in\mathbb{F}^k, r\in\mathbb{F}}\left[\boldsymbol{w}(\bar{r}) = \mathcal{H}_k^{\ell}(\boldsymbol{a})(\bar{r}) \text{ and } \boldsymbol{w}(\bar{r}+r\mathbf{1}) = \mathcal{H}_k^{\ell}(\boldsymbol{a})(\bar{r}+r\mathbf{1})\right] \geq 1-2\delta.$$

Observe that

$$\mathcal{H}_k^{\ell}(\boldsymbol{a})(\bar{r}+r\mathbf{1}) - \mathcal{H}_k^{\ell}(\boldsymbol{a})(\bar{r}) = r\boldsymbol{a}[1] + \cdots + r\boldsymbol{a}[k] = r\,sum^{\ell}(\boldsymbol{a}),$$

so

$$\Pr_{\bar{r}\in\mathbb{F}^k, r\in\mathbb{F}}\left[r^{-1}\left(\boldsymbol{w}(\bar{r}+r\mathbf{1}) - \boldsymbol{w}(\bar{r})\right) = sum^{\ell}(\boldsymbol{a})\right] \geq 1-2\delta.$$

This shows that Hadamard code $\mathcal{H}_k$ is $(2, \delta, 1-2\delta)$-parallel locally decodable with respect to *sum*. ⊣

4.5.1. The reduction

Given an instance (G, k) of k-Clique with $n := |V(G)|$, we first turns it into an instance of the vector 2CSP problem as follows:

- the variable set is $\{x_1, \dots, x_k\}$,
- for each $i \in [k]$ we have a constraint $x_i \in S_i$ where $S_i \subseteq \mathbb{F}_q^{\ell}$ with $\ell = \Theta(\log n / \log q)$,
- for all $1 \leq i < i' \leq k$ we have a constraint $x_i + x_i' \in S_{i,i'} \subseteq \mathbb{F}_q^{\ell}$.

Next, we construct a k-Clique instance from the above CSP instance, again by a modified version of the FGLSS-reduction. To that end, we need a code $C : \Sigma_1^{k_1} \to \Sigma_2^{k_2}$ with $\Sigma_1 = \mathbb{F}_q$ and $k_1 = k$ which is

- $(p, \delta, \varepsilon_T)$-parallel locally testable by a testing algorithm $\mathbb{T}$
- and $(2, 5\delta, \varepsilon_D)$-parallel locally decodable with respect to all functions $proj_i$ and $sum_{i,i'}$ for $1 \leq i < i' \leq k$. We use $\mathbb{D}_i$ and $\mathbb{D}_{i,i'}$ to denote the corresponding decoding algorithms.

Here the constants satisfy $q \geq 2$, $0 < \varepsilon_D, \varepsilon_T < 1$, and $0 < \delta < 1/12$. And for every $\boldsymbol{\alpha} \in \mathbb{F}_q^k$

$$proj_i(\boldsymbol{a}) = \boldsymbol{a}[i] \quad \text{and} \quad sum_{i,i'}(\bar{a}) = \boldsymbol{a}[i] + \boldsymbol{a}[i'].$$

Then using the ℓ-parallelization C^{ℓ} of C we construct a graph G' as follows.

Vertex Set. We encode the behavior of the tester $\mathbb{T}$ by $V(G')$. In particular, let[5]

$$I := \left\{ (i_1, \dots, i_p) \;\middle|\; \mathbb{T} \text{ queries positions } i_1, \dots, i_p \in [k_2] \text{ under some internal randomness of } \mathbb{T} \right\}.$$

Clearly $|I| \leq k_2^p$. Then for every $\bar{i} = (i_1, \dots, i_p) \in I$ we have

$$V_{\bar{i}} := \left\{ (w_1, \dots, w_p) \in (\Sigma_2^{\ell})^p \;\middle|\; \text{if } \mathbb{T} \text{ reads positions } i_1, \dots, i_p \text{ of some } \boldsymbol{w} \in \left(\Sigma_2^{\ell}\right)^{k_2} \text{ with } \boldsymbol{w}[i_j] = w_j \text{ for } j \in [p], \text{ then it accepts} \right\}.$$

Then

$$V(G') := \bigcup_{\bar{i}\in I} V_{\bar{i}}.$$

Edge Set. Two vertices in $V(G')$ are *not* adjacent in G', if they assign *different* values to a same variable, or the assignment causes $\mathbb{D}_i$ to decode a value $x_i \notin S_i$, or the assignment causes $\mathbb{D}_{i,i'}$ to decode a sum $x_i + x_{i'} \notin S_{i,i'}$.

Theorem 4.10 ([17]). *Assume $\delta k_2 > 1$ and $\varepsilon_T |I| \geq 2$.*

(Completeness) *If G has a clique of size k, then G' has a clique of size $|I|$.*

(Soundness) *If $\delta_T < 1/12$ and G has no clique of size k, then G' has no clique of size $\varepsilon_T |I|$.*

Moreover,

$$|V(G')| \leq |\Sigma_2|^{\ell p} |I|,$$

and G' can be constructed in time $|V(G')|^{O(1)}$.

Similar to Remark 4.9 we can prove that Hadamard code $\mathcal{H}_k$ is $(2, \delta, 1-2\delta)$-parallel locally decodable with respect to all functions $proj_i$'s and $sum_{i,i'}$'s. So combined with Remark 4.7, we can plug in C with $\mathcal{H}_k : (\mathbb{F}_q)^k \to (\mathbb{F}_q)^{|\mathbb{F}_q|^k}$ in Theorem 4.10, and set

$$p := 3, \quad \delta := 1/13, \quad \varepsilon_T := 1 - \delta/2, \quad \varepsilon_D := 1 - 2\delta.$$

Then $k_2 = q^k$ and

$$|I| = q^{qk}.$$

So we create a gap of q^{qk} vs. $25q^{qk}/26$ for the k-Clique problem. By further choosing $q := 2$, we observe that under ETH there is no constant approximation of k-Clique with running time

$$f(k)n^{o(\log k)},$$

where recall $n = |V(G)|$. Otherwise, we apply such an fpt-approximation to the instance

$$\left(G', 2^{2k}\right)$$

constructed above, thereby decide whether the original graph G has a k-clique. Overall, this can be done in time

$$2^{O(k)}n^{O(1)} + f(2^{2k})n^{o(\log 2^{2k})} = 2^{O(k)}n^{O(1)} + f(2^{2k})n^{o(k)},$$

where $f : \mathbb{N} \to \mathbb{N}$ is a computable function and where the first term $2^{O(k)}n^{O(1)}$ counts the running time of our reduction. This is well known to contradict ETH [32].

[5] It could happen that for different internal randomness $\mathbb{T}$ queries the same positions $(i_1, \dots, i_p)$. So it is more precise to define I as the set of all internal randomness of $\mathbb{T}$. But this will further complicate our notations.

4.5.2. Better lower bounds by better codes under ETH

Theorem 4.11 (*[17]*). *Let $\mathbb{F} = \mathbb{F}_q$ be a prime field, $m, r, k \geq 1$, and $d := 2r + 1$. Assume*

$$q > \max\{3d, 9216\} \quad \text{and} \quad k \leq \binom{m+d}{d} \leq q^m$$

then there is a code

$$C : \mathbb{F}^k \to \left(\mathbb{F}^{(d+1)\times\binom{2m+r}{r}}\right)^{|\mathbb{F}|^{4m}}$$

that is

- $(11, \delta, 1 - 2\delta/13)$*-parallel locally testable for every* $0.035 < \delta < 0.06$,
- $(2, \delta, 1 - 2\delta)$*-parallel smoothly locally decodable with respect to all* $proj_i$*'s and* $sum_{i,i'}$*'s for every* $0 < \delta < 1/2$.

Moreover, C can be constructed from q, r, k.

Now applying the above C to Theorem 4.10 we can reduce every instance (G, k) of k-Clique to an instance (G', k') with

$$k' = k^{\Theta(\log\log k)}.$$

Corollary 4.12. *There is no constant approximation of k-Clique with running time*

$$f(k)n^{k^{\iota/\log\log k}} \quad \text{for some } \iota > 0,$$

unless ETH *fails.*

Using a standard amplification argument by *disperser* [8,30] we obtain:

Theorem 4.13 (*[17]*). *Assume* ETH. *Then there is no* fpt-*approximation of k-Clique with ratio*

$$\frac{k}{(\log k)^{\iota \log\log\log k}} \quad \text{for some constant } \iota > 0.$$

5. Towards the parameterized inapproximability hypothesis

5.1. A wishful thinking approach

Recall that the parameterized inapproximability hypothesis (PIH) states that it is W[1]-hard to distinguish between satisfiable binary CSP instances and unsatisfiable ones such that every assignment can satisfy at most ϵ-fraction of the constraints for some/all small constant $\epsilon < 1$. Here, the number of variables is the parameter. To prove PIH, we need to reduce a W[1]-hard problem to CSP which creates a constant gap between yes- and no-instances. There are a number of reductions discussed in the previous sections, i.e., Sections 4.1, 3.1, and 4.5.1, which reduce W[1]-hard problems to CSP. Unfortunately, none of them creates a constant gap in the resulting CSP instances.

Take the reduction in Section 4.5.1 as an example, and let (G, k) be an instance of the multi-colored k-Clique problem with $V(G) = V_1 \dot\cup \cdots \dot\cup V_k \subseteq \mathbb{F}^d$. Let us further assume that G has no clique of size k, but there are $\boldsymbol{v}_1 \in V_1, \dots, \boldsymbol{v}_k \in V_k$ which are pairwise adjacent except for $\boldsymbol{v}_1$ and $\boldsymbol{v}_k$. In other words, $\boldsymbol{v}_1, \dots, \boldsymbol{v}_k$ induce a k-clique *except for one edge* $\{\boldsymbol{v}_1, \boldsymbol{v}_k\} \notin E(G)$. Then for every $\bar{r} = (r_1, \dots, r_k) \in \mathbb{F}^k$ we set

$$x_{\bar{r}} := r_1\boldsymbol{v}_1 + \cdots + r_k\boldsymbol{v}_k.$$

Clearly this assignment satisfies all the Vertex Tests (7). And for the Edge Tests, it only fails those (8) with $\bar{r} \in \mathbb{F}^k$, $i = 1$, $i' = k$, and $r, r' \in \mathbb{F}^*$. So the fraction of satisfied constraints is

$$1 - \frac{|\mathbb{F}|^k(|\mathbb{F}| - 1)^2}{|\mathbb{F}|^k k(|\mathbb{F}| - 1) + |\mathbb{F}|^k\binom{k}{2}(|\mathbb{F}| - 1)^2} = 1 - \frac{|\mathbb{F}| - 1}{k + \binom{k}{2}(|\mathbb{F}| - 1)},$$

which tends to 1 when k approaches infinity.

Given that the W[1]-hardness of the constant approximation of k-Clique has already been established, it is tempting to apply the above reduction directly to gap instances of k-Clique, i.e., either G has a k-clique or no clique of size k/c for a large constant $c \geq 1$ independent of k. Nevertheless, in the second case the reduction might produce a CSP instance where we can still satisfy $(1 - o(1))$-fraction of the constraints. This phenomenon can be better understood in terms of the *multi-colored densest k-subgraph problem.*

k-DensestSubgraph	
Instance:	A graph $G = (V, E)$ and $k \geq 1$ with $V = V_1 \dot\cup \cdots \dot\cup V_k$.
Parameter:	k.
Problem:	Find $v_1 \in V_1, \dots, v_k \in V_k$ such that $G\big[\{v_1, \dots, v_k\}\big]$ contains maximum number of edges.

Every instance (G, k) of k-DensestSubgraph can be reduced straightforwardly to a CSP instance Γ. That is, for every V_i we introduce a variable x_i with alphabet $\Sigma_i := V_i$; and for every $1 \leq i < i' \leq k$ there is a constraint checking whether $\{x_i, x_{i'}\} \in E(G)$. Then, for every $m \geq 1$, the graph G has a k-vertex induced subgraph with m edges if and only if there is an assignment satisfying m constraints in Γ. Hence, the reduction is gap-preserving. On the other hand, there is a trivial reduction from the multi-colored k-Clique problem to the k-DensestSubgraph problem as well, i.e., a multi-colored graph G has a k-clique if and only if G has a k-vertex induced subgraph with $\binom{k}{2}$ edges. If this reduction were also gap-preserving, then we could have composed it with the previous reduction from k-DensestSubgraph to Max-2CSP to obtain a gap-preserving reduction from k-Clique to Max-2CSP, thereby proving PIH. Unfortunately, this is not the case. By famous Turán's theorem, a k-vertex graph without a clique of size k/c can have

$$\left(1 - \frac{c}{k} + o(1)\right)\frac{k^2}{2} = (1 - o(1))\binom{k}{2}$$

edges. That is, even if a graph G contains no clique of size k/c, it might still have a k-vertex graph with nearly $\binom{k}{2}$ edges. As a matter of fact, this remains true even in case we allow $c = o(k)$. Therefore, the inapproximability of ratio $k^{o(1)}$ described in Section 3.1 does not help either.

To circumvent the above difficulty, [30] takes a further detour via the *k-biclique problem*

k-Biclique	
Instance:	A graph G and $k \geq 1$.
Parameter:	k.
Problem:	Decide whether G contains $K_{k,k}$ as a subgraph, i.e., a *biclique of size* k.

where one can invoke the following result from extremal graph theory.

Theorem 5.1 (*Kővári-Sós-Turán [33]*). *Let H be a k-vertex graph and $r \geq 2$. If H does not contain $K_{r,r}$ as a subgraph, then the number of edges in H is upper bounded by*

$$\frac{1}{2}(r - 1)^{1/r}k^{2-1/r} + \frac{1}{2}(r - 1)k.$$

An immediate consequence of Theorem 5.1 is that, for any $\epsilon < 1$ there is a constant $c \in \mathbb{N}$ such that if a graph G contains no biclique of size $c \cdot \log k$, then any k-vertex induced subgraph in G contains at most ϵk^2 edges.

Corollary 5.2. *If it is* W[1]*-hard to distinguish between a graph containing a biclique of size k and no biclique of size $c \cdot \log k$, then it is* W[1]*-hard to distinguish between a graph containing a k-vertex subgraph with $k^2/4$ edges and no k-vertex subgraph with $k^2/5$ edges. Consequently,* PIH *holds.*

Unfortunately, at the moment it is still open whether the fpt-approximation of k-Biclique even within a *constant* ratio is W[1]-hard. However, again using dispersers (cf. the proof of Theorem 4.13), [17] proves:

Theorem 5.3. *If there is no constant approximation of k-Clique in time $f(k)n^{\omega(\sqrt{k})}$ for any computable function $f : \mathbb{N} \to \mathbb{N}$, then there is no* fpt-*approximation of* Max-2CSP *for some constant ratio.*

As a consequence, we consider the following version of PIH.

Conjecture 5.4 (PIH^t). *There is a constant $1 > \varepsilon > 0$ such that there is no* fpt-*algorithm which distinguishes between*

- *satisfiable* 2CSP *instances,*
- *and* 2CSP *instances $\Gamma = (X, \Sigma, \Phi)$ where any assignment cannot satisfy $\lceil \varepsilon|\Phi| \rceil$ many constraints in Γ, i.e., no assignment can satisfy ε-fraction of the constraints.*

Remark 5.5.

(i) Assume FPT $\neq$ W[1]. Clearly, PIH implies PIH^t, and all the time lower bounds derived from PIH can be derived from PIH^t.

(ii) By the same proof of Lemma 2.8, the constant ε in Conjecture 5.4 is immaterial. ⊣

As W[1]-hardness usually can only rule out fpt-algorithms, i.e., algorithm of running time $f(k)n^{O(1)}$ instead of, say $f(k)n^{\omega(\sqrt{k})}$, to establish the premise in Theorem 5.3 we likely need some complexity assumption stronger than FPT $\neq$ W[1], e.g., ETH. It turns out that under ETH we can prove PIH^t directly without going through k-DensestSubgraph.

5.2. *Proving* PIH^t *under* ETH

Theorem 5.6 (*[18]*). ETH *implies* PIH^t.

Recall that ETH states that 3Sat cannot be solved in time $2^{o(n)}$, where n is the number of variables in an input 3CNF-formula. To show that ETH implies FPT $\neq$ W[1], we construct from any 3CNF-formula F with n variables and an additional parameter $k \geq 1$ a graph G_F with $2^{O(n/k)}$ vertices such that

$$F \text{ is satisfiable} \iff G_F \text{ has a } k\text{-clique}.$$

Then if k-Clique can be solved in time $f(k)n^{O(1)}$, by setting the above $k := f^{-1}(n)$, we can decide 3Sat in time $2^{O(n/f^{-1}(n))} = 2^{o(n)}$, contradicting ETH. The first step of the proof of Theorem 5.6 in [18] is similar, where k-Clique is replaced by a more intricate CSP problem. It is called *vector-valued* CSP, or VecCSP, in [18] whose instances Γ are of the following form.

- The set of variables is $\{x_1, \dots, x_k\}$ with k being the parameter. Each variable has an alphabet $\Sigma_i \subseteq \mathbb{F}^d$ where $\mathbb{F}$ is a finite field of characteristic 2 and $d \geq 1$.
- Each constraint e is associated with two variables x_i and $x_{i'}$ for some $1 \leq i < i' \leq k$. Hence, Γ is a binary CSP. And there are two types of constraints.

  Linear constraints. e checks whether

  $$x_i = M_e x_{i'}$$

  for a $d \times d$ matrix M_e over $\mathbb{F}$.

  Parallel constraints. There is a subset $\Pi_e \subseteq \mathbb{F} \times \mathbb{F}$ such that e checks whether

  $$\left(x_i[j], x_{i'}[j]\right) \in \Pi_e \text{ for every } j \in [d].$$

  Recall that $x_i \in \mathbb{F}^d$ is a d-dimensional vector over $\mathbb{F}$, so $x_i[j] \in \mathbb{F}$ is its jth coordinate.

Remark 5.7. In [18], the authors defined parallel constraints which check if a sub-constraints Π_e holds for a subset of coordinate $P_e \subseteq [d]$ between vectors. By duplicating variables and introducing matrices that can check the consistency between subsets of coordinates [34], we can assume that all parallel constraints have $P_e = [d]$.

Lemma 5.8. *There is an algorithm $\mathcal{R}$ which on a* 3CNF *formula F*[6] *with n variables and an additional $k \geq 1$ computes a* VecCSP *instance Γ_F with $48k^2$ variables, $O(k^2)$ constraints, and alphabet $\Sigma = \mathbb{F}_8^d$ for $d = O(n/k)$ such that F is satisfiable if and only if Γ_F is satisfiable. Moreover, the running time of $\mathcal{R}$ can be bounded by*

$$k^{O(1)}2^{O(n/k)}.$$

If the above Γ_F's were gap instances, and if we can approximate the binary CSP problem in time $f(k)|\Gamma|^{O(1)}$ for any instance Γ with k variables, then by setting $k := \sqrt{f^{-1}(n)}/48$ we can decide the satisfiability of the formulas F's in time

$$k^{O(1)}2^{O(n/k)} + f(48k^2)|\Gamma_F|^{O(1)} = f^{-1}(n)^{O(1)}2^{O(n/\sqrt{f^{-1}(n)})}$$
$$+ nf^{-1}(n)^{O(1)}2^{O(n/\sqrt{f^{-1}(n)})} = 2^{o(n)}.$$

This would have already established Theorem 5.6. Hence we need to provide a further reduction for VecCSP to create a constant gap between yes- and no-instances. [18] achieves this goal by extending the classical *Probabilistic Checkable Proof of Proximity (*PCPP*)* to parallelized error-correcting codes (cf. Definition 4.5).

Remark 5.9. It is of great interest to ask if there exists an algorithm which on input a graph G and a positive integer k, outputs an equivalent instance of VecCSP with $k' = g(k)$-variables, constant $|\mathbb{F}|$ and vector dimension $d = O(\log |G|)$ in $f(k)|G|^{O(1)}$-time for some computable functions $g, f : \mathbb{N} \to \mathbb{N}$. The existence of such a algorithm together with the gap-creating reduction for VecCSP in [18] would prove FPT $\neq$ W[1] implies PIH^t. Observe that to encode an n-vertex graph G, one need $\binom{n}{2}$ bits in general. On the other hand, any VecCSP with k' variables and dimension d could be encoded using at most $O(\binom{k'}{2}d^2\log|\mathbb{F}|)$ bits, which is less than $\binom{n}{2}$ in our setting. Unless the instances of k-Clique problem can be compressed, such an algorithm is unlikely to exist.

5.2.1. Parallel probabilistic checkable proof of proximity (PPCPP) for VecCSP

Definition 5.10 (*[18]*). Given $p \geq 1$, $1 > \delta, \varepsilon > 0$, and two computable functions $f, g : \mathbb{N} \to \mathbb{N}$, a $(p, \delta, \varepsilon, f, g)$-PPCPP *verifier* is a probabilistic algorithm $\mathbb{V}$ with the following properties.

Input and Proof. $\mathbb{V}$ has three inputs:

1. A VecCSP instance Γ with a set of k variables $x_1, \dots, x_k$, a set C of constraints, and an alphabet $\Sigma = \mathbb{F}^d$ for a finite field $\mathbb{F}$ and $d \geq 1$.
2. $\pi_1 \in (\Sigma)^{|\mathbb{F}|^k}$, viewed as a function $\pi_1 : \mathbb{F}^k \to \mathbb{F}^d$, is supposed to be $\mathcal{H}_k^d(\boldsymbol{v}_1, \dots, \boldsymbol{v}_k)$ for some $\boldsymbol{v}_1, \dots, \boldsymbol{v}_k \in \mathbb{F}^d$ (cf. Definition 2.11). Thereby, π_1 "assigns" $x_i := \boldsymbol{v}_i$ for every $i \in [k]$.
3. $\pi_2 \in \Sigma^{\leq f(k)}$, i.e. $\pi_2 \in \Sigma^*$ with $|\pi_2| \leq f(k)$, which helps $\mathbb{V}$ to check if π_1 encodes a satisfying assignment of Γ.

Intuitively, $\mathbb{V}$ verifies whether $\pi_1 \circ \pi_2$, the concatenation of π_1 and π_2, "proves" that Γ is a yes-instance. We call the length of the concatenation of π_1 and π_2 the proof length of the PPCPP.

[6] In fact, [18] requires that F satisfies some further conditions, which can be achieved by the Sparsification Lemma [35] and Tovy's reduction [36].

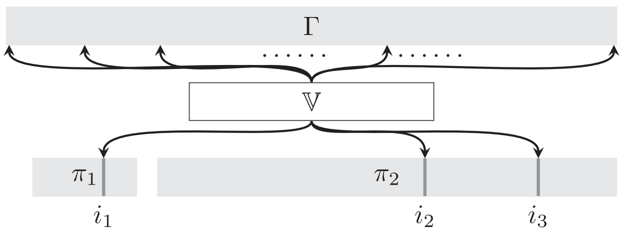


Fig. 3. $\mathbb{V}$ reads Γ with k variables, picks a random number $r \in [R_k]$, and then chooses three positions i_1, i_2, i_3 in $\pi_1 \circ \pi_2$. After reading these positions, $\mathbb{V}$ accepts or rejects.

Non-adaptiveness. $\mathbb{V}$ first computes a number $R_k \leq g(k)$ from k and picks an $r \in [R_k]$ uniformly at random. And then from r it chooses at most p positions in $\pi_1 \circ \pi_2$ deterministically. Hence, these choices are non-adaptive (i.e., only depend on Γ and r). And they can be computed in time $h(k)|\Gamma|^{O(1)}$ for a computable function $h : \mathbb{N} \to \mathbb{N}$. Finally $\mathbb{V}$ queries these positions in $\pi_1 \circ \pi_2$ and then decides in time $|\Gamma|^{O(1)}$ whether accepts or not.

Completeness. If Γ is satisfied by an assignment $x_1 := \boldsymbol{v}_1, \ldots, x_k := \boldsymbol{v}_k$, then for $\pi_1 := \mathcal{H}_k^d(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)$ there is an $\pi_2 \in \Sigma^{\leq f(k)}$ such that $\mathbb{V}$ accepts Γ and $\pi_1 \circ \pi_2$ with probability 1.

Soundness. If $\Pr[\mathbb{V} \text{ accepts } \Gamma \text{ and } \pi_1 \circ \pi_2] \geq \varepsilon$, then there is an assignment $x_1 := \boldsymbol{v}_1, \ldots, x_k := \boldsymbol{v}_k$ which satisfies Γ and

$$\Pr_{\bar{r} \in \mathbb{F}^k}\left[\pi_1[\bar{r}] = \mathcal{H}_k^d(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)[\bar{r}]\right] \geq 1 - \delta.$$

In particular, if $\pi_1 \circ \pi_2$ passes most tests, then Γ is satisfiable and π_1 is close to the Hadamard code of an satisfying assignment of Γ.

See Fig. 3 for an illustration.

Remark 5.11. In some cases, e.g. Lemmas 5.13 and 5.15, the inputs of a PPCPP verifier $\mathbb{V}$ are restricted to a subset K of VecCSP instances. One can easily modify Definition 5.10 to accommodate such a restriction, and we call $\mathbb{V}$ *a* PPCPP *verifier for* K. ⊣

Once having a $(p, \delta, \varepsilon, f, g)$-PPCPP verifier $\mathbb{V}$, we can compute from any given VecCSP instance $\Gamma = (X, \Sigma, \Phi)$ a CSP instance Γ' as follows.

Variables and alphabets.
- For every position j in $\pi_1 \circ \pi_2$ we introduce a variable x_i^π whose alphabet is $\Sigma = \mathbb{F}^d$. Note that $1 \leq i \leq |\mathbb{F}|^k + f(k)$.
- For every $r \in [R_k]$ we introduce a variable $x_r^{\mathbb{V}}$. Moreover, let $\bar{i}_r := (i_1, \ldots, i_{p'})$ such that on input Γ the verifier $\mathbb{V}$ reads p' positions $i_1, \ldots, i_{p'}$ of $\pi_1 \circ \pi_2$ when its internal randomness is r. Then the alphabet for $x_r^{\mathbb{V}}$ is $\Sigma^{p'} = (\mathbb{F}^d)^{p'}$. Recall $p' \leq p$.

Constraints. For every $r \in [R_k]$ with $\bar{i}_r = (i_1, \ldots, i_{p'})$ and every $j \in [p']$ we have a constraint on two variables $x_{\bar{r}}^{\mathbb{V}}$ and $x_{i_j}^\pi$ checking whether

- under the internal randomness r the verifier $\mathbb{V}$ accepts Γ and some/all $\pi_1 \circ \pi_2$ with $\pi_1 \circ \pi_2[i_{j'}] = x_{\bar{r}}^{\mathbb{V}}[j']$ for every $j' \in [p']$,
- and $x_{\bar{r}}^{\mathbb{V}}[j] = x_{i_j}^\pi$, i.e., the assignment of $x_{\bar{r}}^{\mathbb{V}}$ is consistent with that of $x_{i_j}^\pi$.

Lemma 5.12. *Γ' has the following properties.*

- *It has at most*

  $$|\mathbb{F}|^k + f(k) + |R_k| \leq |\mathbb{F}|^k + f(k) + g(k)$$

  variables.
- *If Γ is satisfiable, then so is Γ'.*
- *If Γ is not satisfiable, then for every assignment of Γ' the fraction of constraints in Γ' it satisfies is less than*

  $$\varepsilon + (1 - \varepsilon)\frac{p-1}{p} = \frac{p - 1 + \varepsilon}{p}.$$

  As $\varepsilon < 1$, this is a constant strictly smaller than 1.

Lemma 5.12 shows that $\Gamma \mapsto \Gamma'$ is an fpt-reduction from VecCSP that creates binary CSP instances with constant gap between yes- and no-instances. Composed with the reduction in Lemma 5.8, this will establish Theorem 5.6. Nevertheless, Lemma 5.12 is conditioned on the existence of a $(p, \delta, \varepsilon, f, g)$-PPCPP verifier for VecCSP. So the main technical hurdle that [18] overcomes is the design of such a verifier.

As a matter of fact, the PPCPP-verifier in [18] is composed of two verifiers, which deal with linear constraints and parallel constraints separately. To that end, for every VecCSP instance $\Gamma = (X, \Sigma, \Phi)$ we first split it into two sub-instances

$$\Gamma_L = (X, \Sigma, C_L) \quad \text{and} \quad \Gamma_P = (X, \Sigma, C_P),$$

where C_L consists of all the *linear* constraints in C, and C_P of the remaining ones, i.e., all *parallel* constraints.

Lemma 5.13 (PPCPP *for parallel constraints*)**.** *Let $h : \mathbb{N} \to \mathbb{N}$ be a computable function and $0 < \iota < 1/800$. Then for*

$$p = 4, \quad \delta = 48\iota, \quad \varepsilon = 1 - \iota, \quad f(k) = 2^{k^2 \cdot h(k)^{O(1)}}, \text{ and } \quad g(k) = 2^{k^2 \cdot h(k)^{O(1)}}$$

there is a $(p, \delta, \varepsilon, f, g)$-PPCPP verifier for the set of all VecCSP instances $G = (X, \Sigma, \Phi)$ (cf. Remark 5.11) such that

- *$\Sigma = \mathbb{F}^d$ for a finite field $\mathbb{F}$ of characteristic 2 with $|\mathbb{F}| \leq h(|X|)$ and $d \geq 1$,*
- *and C only contains parallel constraints, or equivalently $C = C_P$.*

Remark 5.14. The construction of [18] first transforms a k-variable CSP with parallel constraints into a c-variable CSP with parallel quadratic equation constraints, where $c \leq k \cdot |\mathbb{F}|^{O(1)}$. The PPCPP is obtained by parallel applying the classical PCP of proximity [37] for the quadratic equation problem. Note that for a c-variable quadratic equation system, the PCP described in [37] has proof length $2^c + 2^{c^2} \leq 2^{k^2 \cdot |\mathbb{F}|^{O(1)}}$.

Lemma 5.15 (PPCPP *for Linear Constraints*)**.** *Let $h, m : \mathbb{N} \to \mathbb{N}$ be two computable functions and $0 < \iota < 1/400$. Then for*

$$p = 4, \quad \delta = 24\iota, \quad \varepsilon = 1 - \iota, \quad f(k) = h(k)^{k \cdot m(k)}, \text{ and } \quad g(k) = h(k)^{8k \cdot m(k)}$$

there is a $(p, \delta, \varepsilon, f, g)$-PPCPP verifier for the set of all VecCSP instances $G = (X, \Sigma, \Phi)$ such that

- *$\Sigma = \mathbb{F}^d$ for a finite field $\mathbb{F}$ of characteristic 2 with $|\mathbb{F}| \leq h(|X|)$ and $d \geq 1$,*
- *C only contains linear constraints, or equivalently $C = C_L$, and $|C| \leq m(|X|)$.*

See Fig. 4 for an illustration.

Proposition 5.16 (*Combined* PPCPP *for* VecCSP)**.** *Let $h, m : \mathbb{N} \to \mathbb{N}$ be two computable functions. Then for*

$$p = 4, \quad \delta = 1/25, \quad \varepsilon = 1 - 1/2400, \quad f(k) = h(k)^{k \cdot m(k)} + 2^{k^2 \cdot h(k)^{O(1)}}, \text{ and}$$
$$g(k) = h(k)^{8k \cdot m(k)} \cdot 2^{k^2 \cdot h(k)^{O(1)}}$$

there is a $(p, \delta, \varepsilon, f, g)$-PPCPP verifier for the set of all VecCSP instances $G = (X, \Sigma, \Phi)$ such that

- *$\Sigma = \mathbb{F}^d$ for a finite field $\mathbb{F}$ of characteristic 2 with $|\mathbb{F}| \leq h(|X|)$ and $d \geq 1$,*
- *and $|C| \leq m(|X|)$.*

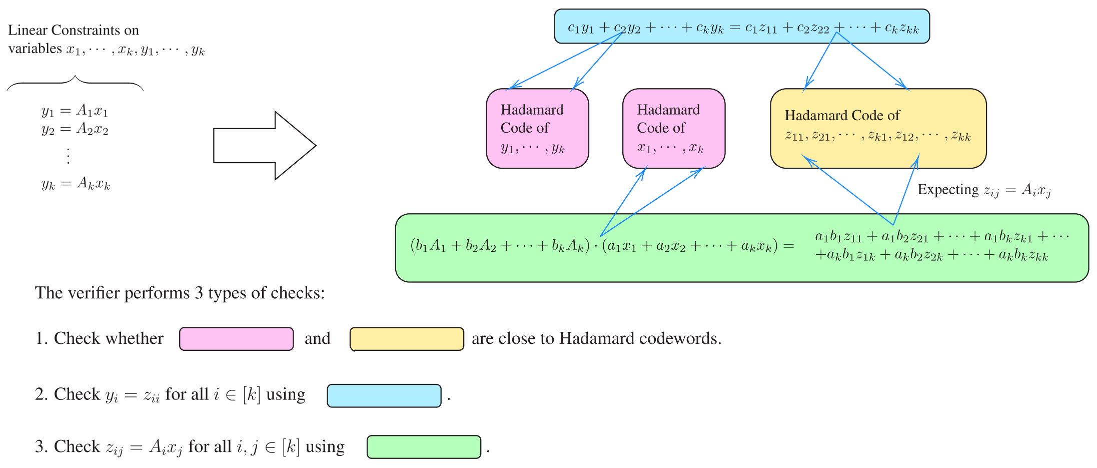

Linear Constraints on variables $x_1, \cdots, x_k, y_1, \cdots, y_k$
$y_1 = A_1 x_1$
$y_2 = A_2 x_2$
$\vdots$
$y_k = A_k x_k$
$c_1y_1 + c_2y_2 + \cdots + c_ky_k = c_1z_{11} + c_2z_{22} + \cdots + c_kz_{kk}$
Hadamard Code of $y_1, \cdots, y_k$
Hadamard Code of $x_1, \cdots, x_k$
Hadamard Code of $z_{11}, z_{21}, \cdots, z_{k1}, z_{12}, \cdots, z_{kk}$
Expecting $z_{ij} = A_i x_j$
$(b_1A_1 + b_2A_2 + \cdots + b_kA_k) \cdot (a_1x_1 + a_2x_2 + \cdots + a_kx_k) =$ $a_1b_1z_{11} + a_1b_2z_{21} + \cdots + a_1b_kz_{k1} + \cdots$ $+ a_kb_1z_{1k} + a_kb_2z_{2k} + \cdots + a_kb_kz_{kk}$
The verifier performs 3 types of checks:
1. Check whether 🟪 and 🟨 are close to Hadamard codewords.
2. Check $y_i = z_{ii}$ for all $i \in [k]$ using 🟦.
3. Check $z_{ij} = A_i x_j$ for all $i, j \in [k]$ using 🟩.

Fig. 4. A PPCPP for a toy example of VecCSP with $2k$ variables and $m(k) = k$ linear constraints. The proof length of this PPCPP is $2|\mathbb{F}|^k + |\mathbb{F}|^{k \cdot m(k)}$.

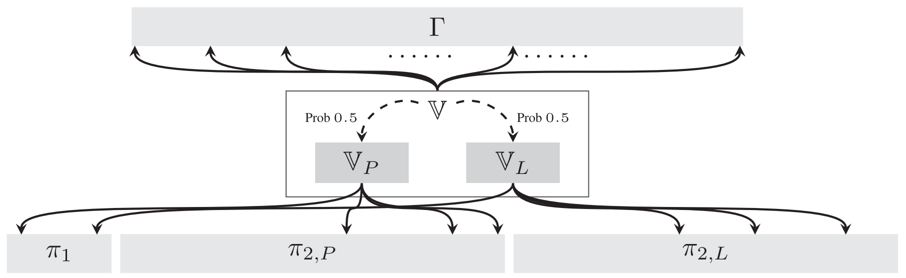


Fig. 5. Our PPCPP verifier for VecCSP.

Proof. By Lemma 5.13 with $\iota = 1/1200$ we have a

$$\left(4, 1/25, 1 - 1/1200, f_P(k) = 2^{k^2 \cdot h(k)^{O(1)}}, g_P(k) = 2^{k^2 \cdot h(k)^{O(1)}}\right)\text{-PPCPP}$$

verifier $\mathbb{V}_P$

for VecCSP instances with only parallel constraints. Similarly, By Lemma 5.15 with $\iota = 1/600$ we have a

$$\left(4, 1/25, 1 - 1/600, f_L(k) = h(k)^{k \cdot m(k)}, g_L(k) = h(k)^{8k \cdot m(k)}\right)\text{-PPCPP}$$

verifier $\mathbb{V}_L$

for VecCSP instances with only linear constraints. Then the desired PPCPP verifier $\mathbb{V}$ (shown in Fig. 5) has three inputs

(1) a VecCSP instance $\Gamma = (X, \Sigma, \Phi)$ with both parallel and linear constraints,
(2) $\pi_1 \in (\Sigma)^{|\mathbb{F}|^k}$ for $X = \{x_1, \ldots, x_k\}$,
(3) and $\pi_2 = \pi_{2,P} \circ \pi_{2,L}$.

It uses $\mathbb{V}_P$ and $\mathbb{V}_L$ as sub-verifiers on inputs $(\Gamma_P, \pi_1, \pi_{2,P})$ and $(\Gamma_L, \pi_1, \pi_{2,L})$, respectively. That is, $\mathbb{V}$ executes $\mathbb{V}_P(\Gamma_P, \pi_1, \pi_{2,P})$ with probability $1/2$, and $\mathbb{V}_L(\Gamma, \pi_1, \pi_{2,L})$ with probability $1/2$. Then we can verify that the following conditions hold.

- $\mathbb{V}$ queries $p = 4$ positions in the combined proof $\pi_1 \circ \pi_2 = \pi_1 \circ \left(\pi_{2,P} \circ \pi_{2,L}\right)$,
- $|\pi_2| \leq f_P(k) + f_L(k) \leq h(k)^{k \cdot m(k)} + 2^{k^2 \cdot h(k)^{O(1)}} = f(k)$,
- The (uniform) randomness of $\mathbb{V}$ can be chosen in such a way that satisfies $R_k \leq g_P(k) g_L(k) \leq h(k)^{8km(k)} \cdot 2^{k^2 \cdot h(k)^{O(1)}} = g(k)$.

The completeness of $\mathbb{V}$ is straightforward. For the soundness, let us assume that $\mathbb{V}$ accepts a proof $\pi_1 \circ \left(\pi_{2,P} \circ \pi_{2,L}\right)$ with probability at least $\varepsilon = 1 - 1/2400$. It implies that $\mathbb{V}_L$ accepts $(\Gamma_L, \pi_1, \pi_{2,L})$ with probability at least $1 - 1/1200 = \varepsilon_L$. Hence, there is an assignment $x_1 := \boldsymbol{v}_1, \ldots, x_k := \boldsymbol{v}_k$ which satisfies Γ_L and

$$\Pr_{\bar{r} \in \mathbb{F}^k}\left[\pi_1[\bar{r}] = \mathcal{H}_k^d(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)[\bar{r}]\right] \geq 1 - \delta = 24/25. \tag{20}$$

Similarly, $\mathbb{V}_P$ accepts $(\Gamma_P, \pi_1, \pi_{2,P})$ with probability at least $1 - 1/1200 > \varepsilon_R$. So there is an assignment $x_1 := \boldsymbol{u}_1, \ldots, x_k := \boldsymbol{u}_k$ which satisfies Γ_P and

$$\Pr_{\bar{r} \in \mathbb{F}^k}\left[\pi_1[\bar{r}] = \mathcal{H}_k^d(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k)[\bar{r}]\right] \geq 1 - \delta = 24/25. \tag{21}$$

By (20) and (21) it is easy to see that $\boldsymbol{v}_i = \boldsymbol{u}_i$ for every $i \in [k]$. Therefore, the assignment $x_1 := \boldsymbol{v}_1, \ldots, x_k := \boldsymbol{v}_k$ satisfies both Γ_L and Γ_P, i.e., it satisfies the original VecCSP instance Γ. □

Now combining Lemma 5.8, Proposition 5.16, and Lemma 5.12, we can establish Theorem 5.6.

Remark 5.17. Some further improvement of Theorem 5.6 is announced recently in [34], where a tighter time lower bound for constant parameterized approximation of Max-2CSP is established under ETH. ⊣

5.3. *Baby* PIH *and its average version*

Although being a big step forward, Theorem 5.6 still has an important difference with the original PIH. Namely, it is based on ETH instead of FPT $\neq$ W[1]. As discussed in [18], if we can show M[1] = W[1], another important open question put forward by Fellows [38], then PIH follows from the proof of Theorem 5.6. However, at this moment it seems to be far from our reach, and there are even reasons to believe that M[1] $\neq$ W[1] [39].

Inspired by [40], another approach to PIH is proposed in [23] via some *list* version of PIH. Let $\Gamma = (X, \Sigma, \Phi)$ be a binary CSP instance. An *r-list assignment* has a list $L(x)$ of at most r values in Σ for each variable $x \in X$. A constraint between variables x and y is said to be satisfied by the r-list assignment if there are $u \in L(x)$ and $v \in L(y)$ such that $x := u, y := v$ satisfies all the constraints between x and y.

Conjecture 5.18 (Baby PIH)**.** *For every $r \geq 1$ it is* W[1]*-hard to distinguish between binary* CSP *instances $\Gamma = (X, \Sigma, \Phi)$ on k variables (i.e., $|X| = k$) such that:*

Completeness. *Γ is satisfiable.*

Soundness. *No r-list assignment satisfies all the constraints in Γ.*

Lemma 5.19. PIH *implies* Baby PIH.

Proof. Let $r \geq 1$ and we set $\varepsilon := 1/r^2$. By Lemma 2.8 we can assume PIH states that it is W[1]-hard to distinguish between binary CSP instances Γ such that:

- either Γ is satisfiable
- or every assignment *cannot* satisfy ε fraction of the constraints in Γ.

Now let Γ be such an instance. We argue in the second case that no r-list assignment satisfy Γ. Otherwise, let $x \mapsto L(x)$ for all $x \in X$ be such an assignment. Then for every variable x we choose a value in $u \in L(x)$ uniformly at random and set $x := u$. Since each constraint has two variables, the expected fraction of satisfied constraints by this random assignment is at least $1/r^2 = \varepsilon$. Hence, there is a single assignment that satisfies ε fraction of the constraints in Γ, contradicting our assumption. So this is a reduction from PIH to Baby PIH. □

Therefore, a necessary condition for PIH is Baby PIH, which is actually established in [23] by a completely combinatorial proof resembling the proof of the Baby PCP theorem in [40].

Theorem 5.20. Baby PIH *holds.*

In fact, [23] suggests a further conjecture between PIH and Baby PIH, an average version of Baby PIH. Again consider an assignment for a CSP instance $\Gamma = (X, \Sigma, \Phi)$ which associates every variable $x \in X$ a list $L(x)$ of values in Σ. Then it is an *r-average list assignment* if the average size of $L(x)$ is at most r, or equivalently

$$\sum_{x \in X} |L(x)| \leq r|X|.$$

Conjecture 5.21 (Average Baby PIH)**.** *For every $r \geq 1$ it is* W[1]*-hard to distinguish between binary* CSP *instances $\Gamma = (X, \Sigma, \Phi)$ on k variables (i.e., $|X| = k$) such that:*

Completeness. *Γ is satisfiable.*

Soundness. *No r-average-list assignment satisfies all the constraints in Γ.*

We can prove that PIH implies Average Baby PIH by a similar argument as in the proof of Lemma 5.19. Moreover, using the W[1]-hardness of the constant approximation of k-Clique, we can show Average Baby PIH holds for a restricted range of r-average-list assignments.

Proposition 5.22. *Assume $1 \leq r < 2$. Then it is* W[1]*-hard to distinguish between binary* CSP *instances Γ that are either satisfiable or cannot be satisfied by any r-average-list assignment.*

Proof. Let $\varepsilon := 2 - r$. Recall that it is W[1]-hard to distinguish between a graph G with a k-clique or no clique of size εk. Let (G, k) be such an instance. We define a binary CSP instance $\Gamma = (X, \Sigma, \Phi)$ with variable set $X = \{x_1, \ldots, x_k\}$ and alphabet $\Sigma = V(G)$. Moreover, for every $1 \leq i < i' \leq k$ we have a constraint on variables x_i and $x_{i'}$ checking whether $\{x_i, x_{i'}\} \in E(G)$.

It is obvious that if G contains a k-clique, then Γ is satisfiable. Otherwise, let $x \mapsto L(x)$ for all $x \in X$ be an r-average-list assignment. Then we define

$$X_1 := \left\{x \;\middle|\; |L(x)| = 1\right\} \quad \text{and} \quad X_2 := X \setminus X_1 = \left\{x \;\middle|\; |L(x)| \geq 2\right\}.$$

It is easy to see that

$$\bigcup_{x \in X_1} L(x)$$

is a clique in G of size $|X_1|$. So by our assumption, $|X_1| < \varepsilon k$ and hence $|X_2| > (1 - \varepsilon)k$. It follows that

$$\sum_{x \in X} |L(x)| = \sum_{x \in X_1} |L(x)| + \sum_{x \in X_2} |L(x)| \geq |X_1| + 2|X_2|$$
$$= |X| + |X_2| > k + (1 - \varepsilon)k = (2 - \varepsilon)k = rk.$$

In other words, Γ cannot be satisfied by an r-average-list assignment. □

6. Conlusions

In this article, we discussed the recent progress on the parameterized inapproximability of k-Clique and also the parameterized inapproximability hypothesis PIH. We gave an overview of the existing proofs and techniques, highlighting the role played by the constraint satisfaction problem CSP. Some of them can be traced back to the classical polynomial time inapproximability of k-Clique and CSP based on the celebrated PCP theorem. But some others are rather elementary, taking the advantage of the stronger complexity assumptions, i.e., FPT $\neq$ W[1] and ETH which we can use.

Of course, the parameterized inapproximability of k-Clique is still far from being completely settled. Among others, under FPT $\neq$ W[1], we only have $k^{o(1)}$ lower bound at the moment, which can be improved to some extent under ETH. As we mentioned, Fellows conjectured that k-Clique is "hideous", i.e., it is W[1]-hard to approximate within ratio $o(k)$.

The status of PIH is even less satisfactory. The only hardness we have is under ETH. Moreover, observe that PIH merely states that Max-2CSP is constant inapproximable. For upper bound, the standard polynomial time approximation of CSP using a random assignment only achieves a ratio of $O(k)$. We conjecture that, similarly to k-Clique, this is the best we can do.

Conjecture 6.1. *Let $\rho : \mathbb{N} \to \mathbb{N}$ be a function with $\rho(k) = o(k)$. Then it is* W[1]*-hard to distinguish between*

- *satisfiable* 2CSP *instances,*
- *and* 2CSP *instances $\Gamma = (X, \Sigma, \Phi)$ where any assignment cannot satisfy $\lfloor |\Phi| / \rho(|\Phi|) \rfloor$ many constraints in Γ.*

It has been shown in [41] that assuming Gap-ETH, no FPT algorithm can distinguish between satisfiable k-variable 2CSP instances and those having an assignment that satisfies $\left\lfloor 2^{(\log k)^{1/2+\varepsilon}} / k \right\rfloor$-fraction of constraints for any constant $\varepsilon > 0$. Using the same proof of Theorem 2.9, we can prove that Conjecture 6.1 implies Conjecture 2.2.

Funding

Yijia Chen is supported by the National Natural Science Foundation of China (Project 62372291).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Proof of Lemma 2.8

We give a proof of Lemma 2.8, which is repeated below for the reader's convenience.

Lemma A.1. *If* PIH_ϵ *holds for some* $0 < \epsilon < 1$, *then* PIH_ϵ *is true for all* $0 < \epsilon < 1$.

To aid discussions, for every 2CSP instance $\Gamma = (X, \Sigma, \Phi)$ we define

$$\mathrm{val}(\Gamma) = \max_\sigma \frac{\left|\{\varphi \in \Phi \mid \varphi \text{ is satisfied by } \sigma\}\right|}{|\Phi|},$$

i.e., the maximum fraction of satisfied constraints in Γ by any assignment.

To simplify presentation, we show $\mathrm{PIH}_{2/3}$ implies PIH_ϵ for any $0 < \epsilon < 1$, which can be easily generalized to any constant other than $2/3$. Therefore, it suffices to prove the following lemma.

Lemma A.2. *There is an* fpt*-reduction* $\mathcal{R}$ *which on input any* 2CSP *instance* $\Gamma = (X, \Sigma, \Phi)$, *output a* 2CSP *instance* $\mathcal{R}(\Gamma)$ *of size* $(|X| + |\Phi|)^\ell + (2|\Phi|)^\ell$ *in* $|\Gamma|^{O(\ell)}$*-time such that*

Yes case. *if* $\mathrm{val}(\Gamma) = 1$, *i.e.,* Γ *is satisfiable, then* $\mathrm{val}(\mathcal{R}(\Gamma)) = 1$,

No case. *if* $\mathrm{val}(\Gamma) < 2/3$, *then* $\mathrm{val}(\mathcal{R}(\Gamma)) < \epsilon$.

Proof. As a first step, we define an intermediate 2CSP instance $\Gamma' = (X', \Sigma', \Phi')$ by the well-known "variable-clause" construction. In particular, the variable set[7] is

$$X' := \left\{y_x \,\middle|\, x \in X\right\} \cup \left\{y_\varphi \,\middle|\, \varphi \in \Phi\right\}.$$

To define Σ', we let $\Sigma'_{y_x} := \Sigma_x$ for every $x \in X$, and $\Sigma'_{y_\varphi} := C$ for every $\varphi = (x_1x_2, C) \in \Phi$. Then set

$$\Sigma' := \bigcup_{x\in X} \Sigma'_{y_x} \cup \bigcup_{\varphi\in\Phi} \Sigma'_{y_\varphi}.$$

The constraint set Φ' consists of all "consistency" constraints, i.e., for every $\varphi = (x_1x_2, C) \in \Phi$ we have two binary constraints $(y_{x_1}y_\varphi, C'_1)$ and $(y_{x_2}y_\varphi, C'_2)$, where

$$C'_1 := \left\{(a_1, (a_1, a_2)) \,\middle|\, (a_1, a_2) \in C\right\} \quad \text{and}$$
$$C'_2 := \left\{(a_2, (a_1, a_2)) \,\middle|\, (a_1, a_2) \in C\right\}. \tag{22}$$

This implies that

$$|\Phi'| = 2|\Phi|.$$

Claim 1. *If* Γ *is satisfiable, so is* Γ'. *Conversely, if* $\mathrm{val}(\Gamma) < 2/3$, *then* $\mathrm{val}(\Gamma') < (1 + 2/3)/2 = 5/6 < 1$.

proof the claim. The first part of the claim is trivial. For the second part, let $\sigma' : X' \to \Sigma'$ be an assignment for Γ'. Clearly σ' induces an assignment $\sigma : X \to \Sigma$ by

$$\sigma(x) := \sigma'(y_x)$$

for every $x \in X$. Let $\varphi = (x_1x_2, C) \in \Phi$ such that

$$(\sigma(x_1), \sigma(x_2)) \notin C,$$

i.e., σ does not satisfy φ. It follows that at least one of the two constraints $(y_{x_1}y_\varphi, C'_1)$ and $(y_{x_2}y_\varphi, C'_2)$ in Γ' are not satisfied. Hence

$$\left|\{\varphi' \in \Phi' \mid \sigma' \text{ does not satisfy } \varphi'\}\right| \geq \left|\{\varphi \in \Phi \mid \sigma \text{ does not satisfy } \varphi\}\right|.$$

By $|\Phi'| = 2|\Phi|$ we obtain

$$\frac{\left|\{\varphi' \in \Phi' \mid \varphi' \text{ is not satisfied by } \sigma'\}\right|}{|\Phi'|}$$
$$\geq \frac{\left|\{\varphi \in \Phi \mid \varphi \text{ is not satisfied by } \sigma\}\right|}{2|\Phi|} \geq \frac{1 - \mathrm{val}(\Gamma)}{2} > \frac{1 - 2/3}{2}.$$

It further implies

$$\mathrm{val}(\Gamma') < 1 - \frac{1 - 2/3}{2} = \frac{1 + 2/3}{2} = \frac{5}{6} < 1. \tag{23}$$

This finishes the proof of the claim. $\dashv$

The second step is by a parallel repetition theorem due to [24]. We first observe that the previous $\Gamma' = (X', \Sigma', \Phi')$ is essentially an instance of the *label cover* problem (which we formulate as a 2CSP) with the so-called *projection* property:

- $X' = X'_L \,\dot\cup\, X'_R$ with the *left* variable set $X'_L := \{y_x \mid x \in X\}$ and the *right* variable set $X'_R := \{y_\varphi \mid \varphi \in \Phi\}$.
- Every binary constraint $\varphi' = (y_1y_2, C') \in \Phi'$ is between a left variable and a right variable, i.e., either $\varphi' = (y_{x_1}y_\varphi, C'_1)$ or $\varphi' = (y_{x_2}y_\varphi, C'_2)$ for some $\varphi = (x_1x_2, C) \in \Phi$. Moreover, it has the projection property in the sense that φ' is associated with a function

  $$proj_{\varphi'} : \Sigma'_{y_2} \to \Sigma'_{y_1}$$

  such that φ' is satisfied by an assignment $\sigma' : X' \to \Sigma'$ if and only if $proj_{\varphi'}(\sigma'(y_2)) = \sigma'(y_1)$. That is, the value of the left variable is completely determined by the right variable in a satisfying assignment. This is certainly true for $\varphi' = (y_{x_1}y_\varphi, C'_1)$, since we can take $proj_{\varphi'}(a_1, a_2) = a_1$ (cf. (22)). Similarly, it holds for $\varphi' = (y_{x_2}y_\varphi, C'_1)$ as well.

Now for some constant $\ell > 1$ yet to be determined, we define the ℓ*-repetition* of Γ' as the 2CSP instance $\Gamma'' = (X'', \Sigma'', \Pi'')$ with:

- $X'' := X''_L \,\dot\cup\, X''_R$ where

  $$X''_L := \left\{z_{y_1\dots y_\ell} \,\middle|\, y_1, \dots, y_\ell \in X'_L\right\} \quad \text{and}$$
  $$X''_R := \left\{z_{y_1\dots y_\ell} \,\middle|\, y_1, \dots, y_\ell \in X'_R\right\}.$$

- For every $z = z_{y_1\dots y_\ell} \in X''$ we set

  $$\Sigma'_z := \Sigma_{y_1} \times \cdots \times \Sigma_{y_\ell}.$$

  And then let $\Sigma' := \bigcup_{z\in X''} \Sigma''_z$.
- Let $\varphi'_1, \dots, \varphi'_\ell \in \Phi'$ with $\varphi'_i = (y_{i1}y_{i2}, C'_i)$ for every $i \in [\ell]$. We introduce a constraint $\varphi'' = (z_1z_2, C'') \in \Phi''$ with $z_1 = z_{y_{11}\dots y_{\ell 1}}$, $z_2 = z_{y_{12}\dots y_{\ell 2}}$, and

  $$C'' := \left\{\left((a_1, \dots, a_\ell), (b_1, \dots, b_\ell)\right) \,\middle|\, (a_i, b_i) \in C'_i \text{ for every } i \in [\ell]\right\}$$

By [24, Theorem4][8] we conclude that

- if $\mathrm{val}(\Gamma') = 1$, then $\mathrm{val}(\Gamma'') = 1$,
- if $\mathrm{val}(\Gamma') < 5/6 = 1 - 1/6$ in (23), then

  $$\mathrm{val}(\Gamma'') < \left(1 - \frac{1/6}{2}\right)^{\alpha(1/6)\ell} = (1 - 1/12)^{\alpha\ell/6},$$

where $\alpha > 0$ is a constant independent of $\ell \geq 1$ (and also the constant $2/3$ we chose in the beginning for $\mathrm{PIH}_{2/3}$). Now we choose a sufficiently large $\ell \geq 1$ such that $(1 - 1/12)^{\alpha\ell/6} \leq \epsilon$. This gives us immediately:

[7] This is where we need to take $|X| + |\Phi|$ as the parameter of $\Gamma = (X, \Sigma, \Phi)$. Otherwise, $\Gamma \mapsto \Gamma'$ is not an fpt-reduction, since we cannot bound $|X'|$ in terms of $|X|$.

[8] Theorem 4 in [24] is actually stated for the so-called *projection games*. But as discussed in [24], this implies the same result for the label cover problem with projection property, in particular the 2CSP instances Γ' and Γ'' as we defined.

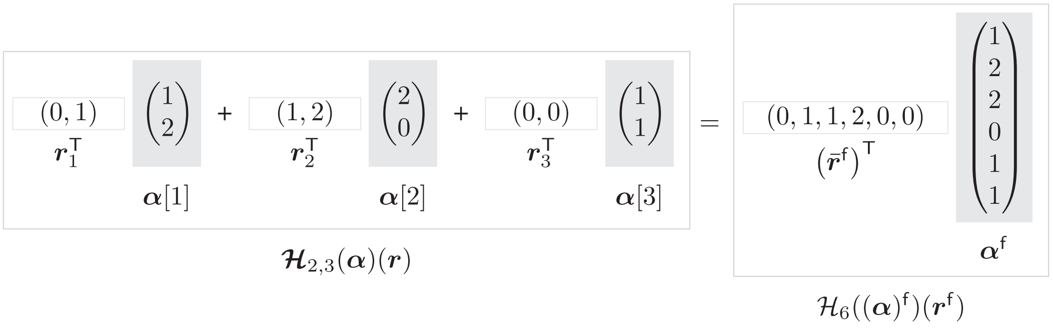


Fig. 6. High-order Hadamard codes and their flattening.

Claim 2. *If* $\mathrm{val}(\Gamma') = 1$, *then* $\mathrm{val}(\Gamma'') = 1$. *Conversely, if* $\mathrm{val}(\Gamma') < 5/6$, *then* $\mathrm{val}(\Gamma'') < \varepsilon$.

*Let $\mathcal{R}$ be algorithm that computes Γ'' from Γ. Then Claims 1 and 2 imply that the aforementioned **Yes case** and **No case** hold. Furthermore, we observe that the running time of $\mathcal{R}$ can be bounded by*

$$\underbrace{|\Gamma|^{O(1)}}_{\text{constructing } \Gamma'} + \underbrace{|\Gamma'|^{O(\ell)}}_{\text{constructing } \Gamma''} = |\Gamma|^{O(\ell)}.$$

As ℓ only depends on ε which is a constant, $\mathcal{R}$ runs in polynomial time. Finally

$$|X''| + |\Phi''| \le |X'|^\ell + |\Phi'|^\ell \le (|X| + |\Phi|)^\ell + (2|\Phi|)^\ell,$$

meaning that the parameter of Γ'' can be bounded in terms of the parameter of Γ. Hence $\mathcal{R}$ is an fpt*-reduction.* □

Appendix B. High-order Hadamard codes, random matrices, and dimension reduction

Definition B.1 (*High-order Hadamard Codes (HHC)*)**.** For every $k, h \ge 1$ we define

$$\boldsymbol{\mathcal{H}}_{h,k} : (\mathbb{F}^h)^k \to \left((\mathbb{F}^h)^k \to \mathbb{F}\right)$$

by setting for every $\boldsymbol{\alpha} \in (\mathbb{F}^h)^k$ and $\bar{\boldsymbol{r}} = (\boldsymbol{r}_1, \dots, \boldsymbol{r}_k) \in (\mathbb{F}^h)^k$

$$\boldsymbol{\mathcal{H}}_{h,k}(\boldsymbol{\alpha})(\bar{\boldsymbol{r}}) = \boldsymbol{r}_1^\mathsf{T}\boldsymbol{\alpha}[1] + \cdots + \boldsymbol{r}_k^\mathsf{T}\boldsymbol{\alpha}[k] \in \mathbb{F}. \tag{24}$$

We might view (24) as a high-order version of (4). As a matter of fact, $\boldsymbol{\mathcal{H}}_{1,k}$ is precisely the Hadamard code $\mathcal{H}_k$. And for $h \ge 2$ we can *flatten* $\boldsymbol{\mathcal{H}}_{h,k}$ to $\mathcal{H}_{hk}$ in the following sense.

Definition B.2. Let $d_1, d_2 \ge 1$. For every $\boldsymbol{\alpha} \in (\mathbb{F}^{d_1})^{d_2}$ we define the *flattening* of $\boldsymbol{\alpha}$ as a vector $\boldsymbol{\alpha}^\mathsf{f} \in \mathbb{F}^{d_1 d_2}$ such that for every $i \in [d_2]$ and $j \in [d_1]$

$$\boldsymbol{\alpha}^\mathsf{f}\left[(i-1)d_1 + j\right] = \boldsymbol{\alpha}[i][j].$$

For example, for $\mathbb{F} = \mathbb{F}_3$, $d_1 = 2$, and $d_2 = 3$

$$\left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\2\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}\right)^\mathsf{f} = (0,1,1,2,0,0)^\mathsf{T}.$$

Clearly every vector in $\mathbb{F}^{d_1 d_2}$ is the flattening of a unique vector in $(\mathbb{F}^{d_1})^{d_2}$.

Lemma B.3. *For every $\boldsymbol{\alpha} \in (\mathbb{F}^h)^k$ and $\bar{\boldsymbol{r}} = (\boldsymbol{r}_1, \dots, \boldsymbol{r}_k) \in (\mathbb{F}^h)^k$*

$$\boldsymbol{\mathcal{H}}_{h,k}(\boldsymbol{\alpha})(\bar{\boldsymbol{r}}) = \mathcal{H}_{hk}\left(\boldsymbol{\alpha}^\mathsf{f}\right)\left(\bar{\boldsymbol{r}}^\mathsf{f}\right).$$

Fig. 6 gives an illustration.

Definition B.4 (*Parallelized HHC*)**.** For every $\ell \ge 1$ we define $\boldsymbol{\mathcal{H}}_{h,k}^\ell$: $\left((\mathbb{F}^h)^\ell\right)^k \to \left((\mathbb{F}^h)^k \to \mathbb{F}^\ell\right)$ as follows. Let $\boldsymbol{a} \in ((\mathbb{F}^h)^\ell)^k$. For every $j \in [\ell]$ we set

$$\boldsymbol{a}_j = \left(\boldsymbol{a}[1][j], \dots, \boldsymbol{a}[k][j]\right) \in (\mathbb{F}^h)^k.$$

Then

$$\boldsymbol{\mathcal{H}}_{h,k}^\ell(\boldsymbol{a}) : (\mathbb{F}^h)^k \to \mathbb{F}^\ell$$

maps every $\bar{\boldsymbol{r}} = (\boldsymbol{r}_1, \dots, \boldsymbol{r}_k) \in (\mathbb{F}^h)^k$ to

$$\boldsymbol{\mathcal{H}}_{h,k}^\ell(\boldsymbol{a})(\bar{\boldsymbol{r}}) = \begin{pmatrix} \boldsymbol{\mathcal{H}}_{h,k}(\boldsymbol{a}_1)(\bar{\boldsymbol{r}}) \\ \vdots \\ \boldsymbol{\mathcal{H}}_{h,k}(\boldsymbol{a}_d)(\bar{\boldsymbol{r}}) \end{pmatrix} \in \mathbb{F}^\ell.$$

It is easy to verify that

$$\boldsymbol{\mathcal{H}}_{h,k}^\ell(\boldsymbol{a})(\bar{\boldsymbol{r}}) = \left(\boldsymbol{r}_1^\mathsf{T}\boldsymbol{a}[1] + \cdots + \boldsymbol{r}_k^\mathsf{T}\boldsymbol{a}[k]\right)^\mathsf{T},$$

as illustrated by Fig. 7. This is a high-order analog of (3).

By Lemma B.3 every ith row of $\boldsymbol{\mathcal{H}}_{h,k}^\ell(\boldsymbol{a})(\bar{\boldsymbol{r}})$ is exactly $\mathcal{H}_{hk}(\boldsymbol{a}_i^\mathsf{f})(\bar{\boldsymbol{r}}^\mathsf{f})$. Therefore, Proposition 2.16 implies:

Proposition B.5 (*Linearity Test of HHC*)**.** *Assume that $\mathbb{F}$ is a prime field and $f : (\mathbb{F}^h)^k \to \mathbb{F}^\ell$. If*

$$\Pr_{\bar{\boldsymbol{r}}, \bar{\boldsymbol{r}}' \in (\mathbb{F}^h)^k}\left[f(\bar{\boldsymbol{r}}) + f(\bar{\boldsymbol{r}}') = f(\bar{\boldsymbol{r}} + \bar{\boldsymbol{r}}')\right] \ge 1 - \delta/2$$

where $\delta < 1/3$. Then there is an $\boldsymbol{a} \in \left((\mathbb{F}^h)^\ell\right)^k$ such that

$$\Pr_{\bar{\boldsymbol{r}} \in (\mathbb{F}^h)^k}\left[f(\bar{\boldsymbol{r}}) = \boldsymbol{\mathcal{H}}_{h,k}^\ell(\boldsymbol{a})(\bar{\boldsymbol{r}})\right] \ge 1 - \delta.$$

That is, f is $(1-\delta)$-close to the function $\boldsymbol{\mathcal{H}}_{h,k}^\ell(\boldsymbol{a})$.

$\boldsymbol{\mathcal{H}}_{h,k}^\ell$ will play the role of $\mathcal{H}_k^d$ (as in Section 4.1) in the construction of our new CSP with

$$h = \Theta(k^2) \quad \text{and} \quad \ell = \Theta(k^2 + \log n).$$

Furthermore, the variable set is

$$X := \left\{x_{\bar{\boldsymbol{r}}} \;\middle|\; \bar{\boldsymbol{r}} = (\boldsymbol{r}_1, \dots, \boldsymbol{r}_k) \in (\mathbb{F}^h)^k\right\}, \tag{25}$$

where each variable is assigned to a vector in $\mathbb{F}^\ell$. More precisely, let $\boldsymbol{v}_1 \in V_1, \dots, \boldsymbol{v}_k \in V_k$ be a solution of the k-VectorSum problem. Then we take

$$x_{\bar{\boldsymbol{r}}} = \boldsymbol{\mathcal{H}}_{k,h}^\ell(\boldsymbol{p}_1, \dots, \boldsymbol{p}_k)(\bar{\boldsymbol{r}}) \in \mathbb{F}^\ell, \tag{26}$$

where $\boldsymbol{p}_1, \dots, \boldsymbol{p}_k \in (\mathbb{F}^h)^\ell$ are the "projection" of $\boldsymbol{v}_1, \dots, \boldsymbol{v}_k \in \mathbb{F}^d$ defined below. Thereby, we reduce the domain of variables from $\mathbb{F}^d = \mathbb{F}^{\Theta(k^2 \log n)}$ in (13) to $\mathbb{F}^\ell = \mathbb{F}^{\Theta(k^2 + \log n)}$ in (26). The latter clearly can be afforded by an fpt-reduction.

To obtain $\boldsymbol{p}_i$'s, we first pick ℓ random $h \times d$ matrices over $\mathbb{F}$

$$\bar{A} = A_1, \dots, A_\ell$$

uniformly and independently, and then define a function $g_{\bar{A}} : \mathbb{F}^d \to (\mathbb{F}^h)^\ell$ by

$$g_{\bar{A}}(\boldsymbol{u}) = \left(A_1\boldsymbol{u}, \dots, A_\ell\boldsymbol{u}\right) \in (\mathbb{F}^h)^\ell \tag{27}$$

for every $\boldsymbol{u} \in \mathbb{F}^d$. Let us emphasize that $g_{\bar{A}}(\boldsymbol{u}) \in (\mathbb{F}^h)^k$ can be viewed as an $h \times k$ matrix over $\mathbb{F}$.

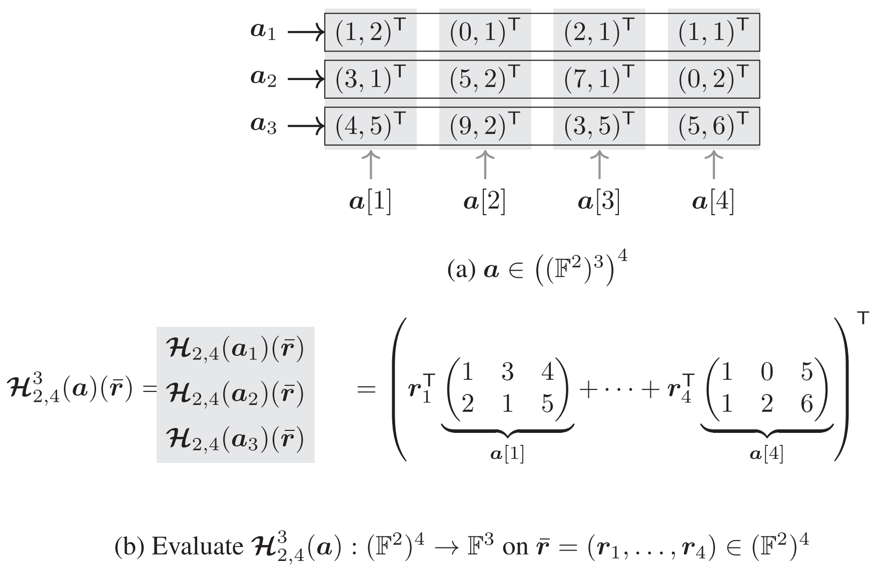

a_1
a_2
a_3
$(1,2)^\top$ $(0,1)^\top$ $(2,1)^\top$ $(1,1)^\top$
$(3,1)^\top$ $(5,2)^\top$ $(7,1)^\top$ $(0,2)^\top$
$(4,5)^\top$ $(9,2)^\top$ $(3,5)^\top$ $(5,6)^\top$
$a[1]$ $a[2]$ $a[3]$ $a[4]$
(a) $a \in ((\mathbb{F}^2)^3)^4$
$\mathcal{H}^3_{2,4}(a)(\bar{r}) = \begin{pmatrix}\mathcal{H}_{2,4}(a_1)(\bar{r}) \\ \mathcal{H}_{2,4}(a_2)(\bar{r}) \\ \mathcal{H}_{2,4}(a_3)(\bar{r})\end{pmatrix} = \left(r_1^\top \underbrace{\begin{pmatrix}1&3&4\\2&1&5\end{pmatrix}}_{a[1]} + \cdots + r_4^\top \underbrace{\begin{pmatrix}1&0&5\\1&2&6\end{pmatrix}}_{a[4]}\right)^\top$
(b) Evaluate $\mathcal{H}^3_{2,4}(a) : (\mathbb{F}^2)^4 \to \mathbb{F}^3$ on $\bar{r} = (r_1, \ldots, r_4) \in (\mathbb{F}^2)^4$

Fig. 7. Parallelization of high-order Hadamard codes.

Lemma B.6.

(i) *$g_{\bar{A}}$ is linear for any $\bar{A}$, i.e., $g_{\bar{A}}(u+v) = g_{\bar{A}}(u) + g_{\bar{A}}(v)$ and $g_{\bar{A}}(ru) = rg_{\bar{A}}(u)$ for all $u, v \in \mathbb{F}^d$ and $r \in \mathbb{F}$.*

(ii) *With high probability (over the choices of $\bar{A}$)*

(a) the function $g_{\bar{A}}$ is injective,

(b) for all distinct $u, v \in \bigcup V_i$ and all $r \in \mathbb{F}^h$

$$r^\top g_{\bar{A}}(u) \neq r^\top g_{\bar{A}}(v),$$

(c) for all pairwise distinct $u, v, w \in \bigcup V_i$ and all nonzero $r, r' \in \mathbb{F}^h$

$$r^\top g_{\bar{A}}(u+w) \neq (r')^\top g_{\bar{A}}(v+w).$$

Now assume that both (i) and (ii) in Lemma B.6 hold for some fixed $\bar{A}$. Recall that $v_1 \in V_1, \ldots, v_k \in V_k$, Then for every $t \in \mathbb{F}^d$

$$v_1 + \cdots + v_k = t \iff g_{\bar{A}}(v_1) + \cdots + g_{\bar{A}}(v_k) = g_{\bar{A}}(t) \tag{28}$$

by Lemma B.6(i) and (ii) (a). Finally we let

$$p_1 := g_{\bar{A}}(v_1), \quad \ldots, \quad p_k := g_{\bar{A}}(v_k), \text{ and } \quad q = g_{\bar{A}}(t).$$

So (28) implies that $v_1 + \cdots + v_k = t$ if and only if $p_1 + \cdots + p_k = q$. Over the new variable set X in (25) we again have constraints for Codeword Tests, Membership Tests, and Sum Tests similarly as described in Section 4.1. E.g., for every $(r_1, \ldots, r_k) \in (\mathbb{F}^h)^k$ and every $r \in \mathbb{F}^h$ we test whether

$$x_{r_1+r,\ldots,r_k+r} - x_{r_1,\ldots,r_k} = r^\top q.$$

To turn the new CSP to a graph G' we use basically the same construction in Section 4.2. Among others, we have a subset of vertices

$$V_{\bar{r},\bar{r}'} := \left\{ (u, v, w) \in ((\mathbb{F}^h)^d)^3 \;\middle|\; u + v = w \right\}$$

for every $\bar{r}, \bar{r}' \in (\mathbb{F}^h)^k$. The correctness of our reduction can be analyzed as in Section 4.3.

Remark B.7.

(i) It is not hard to derandomize Lemma B.6 (ii) [11], i.e., we can construct some $\bar{A}$ satisfying (a)–(c) in deterministic fpt time.

(ii) The above reduction only proves the W[1]-hardness of approximating k-Clique for some *specific* constant ratio, i.e., $1 - \varepsilon$ for a small constant $\varepsilon > 0$. Then using the standard graph product, the inapproximation ratio can be amplified to *any* constant, e.g., 2^{1000}. ⊣

Data availability

No data was used for the research described in the article.

References

[1] Rodney G. Downey, Michael R. Fellows, Parameterized Complexity, Springer-Verlag, 1999.

[2] David Zuckerman, Linear degree extractors and the inapproximability of max clique and chromatic number, in: Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing, 2006, pp. 681–690.

[3] Michael R. Fellows, Open problems, 2005, Dagstuhl Seminar 05301, Exact Algorithms and Fixed-Parameter Tractability.

[4] Sanjeev Arora, Shmuel Safra, Probabilistic checking of proofs: A new characterization of NP, J. ACM 45 (1) (1998) 70–122.

[5] Irit Dinur, The PCP theorem by gap amplification, J. ACM 54 (3) (2007) 12.

[6] Yijia Chen, Martin Grohe, Magdalena Grüber, On parameterized approximability, in: International Workshop on Parameterized and Exact Computation, Springer, 2006, pp. 109–120.

[7] Edouard Bonnet, Bruno Escoffier, Eun Jung Kim, Vangelis Th Paschos, On subexponential and FPT-time inapproximability, Algorithmica 71 (3) (2015) 541–565.

[8] Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, Luca Trevisan, From gap-ETH to FPT-inapproximability: Clique, dominating set, and more, in: Chris Umans (Ed.), 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017, IEEE Computer Society, 2017, pp. 743–754.

[9] Irit Dinur, Mildly exponential reduction from gap-3SAT to polynomial-gap label-cover, in: Electronic Colloquium on Computational Complexity ECCC; Research Reports, Surveys and Books in Computational Complexity, 2016.

[10] Pasin Manurangsi, Prasad Raghavendra, A birthday repetition theorem and complexity of approximating dense CSPs, 2016, arXiv preprint arXiv:1607.02986.

[11] Bingkai Lin, Constant approximating k-clique is W[1]-hard, in: Samir Khuller, Virginia Vassilevska Williams (Eds.), STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021, ACM, 2021, pp. 1749–1756.

[12] C.S. Karthik, Subhash Khot, Almost polynomial factor inapproximability for parameterized k-clique, in: Shachar Lovett (Ed.), 37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA, in: LIPIcs, vol. 234, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, pp. 6:1–6:21.

[13] Yijia Chen, Yi Feng, Bundit Laekhanukit, Yanlin Liu, Simple combinatorial construction of the $k^{o(1)}$-lower bound for approximating the parameterized k-clique, 2023, CoRR abs/2304.07516.

[14] P. Erdös, P. Turán, On a problem of sidon in additive number theory, and on some related problems, J. Lond. Math. Soc. s1-16 (4) (1941) 212–215.

[15] Imre Z. Ruzsa, Solving a linear equation in a set of integers I, Acta Arith. 65 (3) (1993) 259–282.

[16] Imre Z. Ruzsa, Solving a linear equation in a set of integers II, Acta Arith. 72 (4) (1995) 385–397.

[17] Bingkai Lin, Xuandi Ren, Yican Sun, Xiuhan Wang, Improved hardness of approximating k-clique under ETH, FOCS (2023).
[18] Venkatesan Guruswami, Bingkai Lin, Xuandi Ren, Yican Sun, Kewen Wu, Parameterized inapproximability hypothesis under exponential time hypothesis, in: Proceedings of the 56th Annual ACM Symposium on Theory of Computing, 2024, pp. 24–35.
[19] Andreas Emil Feldmann, C.S. Karthik, Euiwoong Lee, Pasin Manurangsi, A survey on approximation in parameterized complexity: Hardness and algorithms, Algorithms 13 (6) (2020) 146.
[20] Venkatesan Guruswami, Atri Rudra, Madhu Sudan, Essential coding theory, 2022, Draft Available At http://www.cse.buffalo.edu/atri/courses/coding-theory/book.
[21] Jörg Flum, Martin Grohe, Parameterized Complexity Theory, Springer, 2006.
[22] Stasys Jukna, Extremal Combinatorics: with Applications in Computer Science, Springer Science & Business Media, 2011, pp. 148–151.
[23] Venkatesan Guruswami, Xuandi Ren, Sai Sandeep, Baby PIH: Parameterized inapproximability of min CSP, 2023, arXiv preprint arXiv:2310.16344.
[24] Anup Rao, Parallel repetition in projection games and a concentration bound, SIAM J. Comput. 40 (6) (2011) 1871–1891.
[25] Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, Mario Szegedy, Interactive proofs and the hardness of approximating cliques, J. ACM 43 (2) (1996) 268–292.
[26] Pavol Hell, Jaroslav Nesetril, Graphs and Homomorphisms, vol. 28, OUP Oxford, 2004.
[27] Parikshit Gopalan, Venkatesan Guruswami, Prasad Raghavendra, List decoding tensor products and interleaved codes, SIAM J. Comput. 40 (5) (2011) 1432–1462.
[28] Irit Dinur, Elena Grigorescu, Swastik Kopparty, Madhu Sudan, Decodability of group homomorphisms beyond the johnson bound, in: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, 2008, pp. 275–284.
[29] Oded Goldreich, Introduction to Property Testing, Cambridge University Press, 2017.
[30] Bingkai Lin, Xuandi Ren, Yican Sun, Xiuhan Wang, On lower bounds of approximating parameterized k-clique, in: 49th International Colloquium on Automata, Languages, and Programming (ICALP 2022), Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.
[31] Amir Abboud, Kevin Lewi, Ryan Williams, Losing weight by gaining edges, in: European Symposium on Algorithms, Springer, 2014, pp. 1–12.
[32] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, Ge Xia, Strong computational lower bounds via parameterized complexity, J. Comput. System Sci. 72 (8) (2006) 1346–1367.
[33] Tamás Kővári, Vera T. Sós, Paul Turán, On a problem of K. Zarankiewicz, Colloq. Math. 3 (1954) 50–57.
[34] Venkatesan Guruswami, Bingkai Lin, Xuandi Ren, Yican Sun, Kewen Wu, Almost optimal time lower bound for approximating parameterized clique, CSP, and more, under ETH, 2024, CoRR abs/2404.08870.
[35] Russell Impagliazzo, Ramamohan Paturi, Francis Zane, Which problems have strongly exponential complexity? J. Comput. System Sci. 63 (4) (2001) 512–530.
[36] C. Tovey, A simplified NP-complete satisfiability problem, Discrete Appl. Math. 8 (1984) 85–89.
[37] Sanjeev Arora, Boaz Barak, Computational Complexity - A Modern Approach, Cambridge University Press, 2009.
[38] Michael R. Fellows, Blow-ups, win/win's, and crown rules: Some new directions in FPT, in: Graph-Theoretic Concepts in Computer Science: 29th International Workshop, WG 2003. Elspeet, the Netherlands, June 19-21, 2003. Revised Papers 29, Springer, 2003, pp. 1–12.
[39] Yijia Chen, Martin Grohe, An isomorphism between subexponential and parameterized complexity theory, SIAM J. Comput. 37 (4) (2007) 1228–1258.
[40] Libor Barto, Marcin Kozik, Combinatorial gap theorem and reductions between promise CSPs, in: Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, SIAM, 2022, pp. 1204–1220.
[41] Irit Dinur, Pasin Manurangsi, ETH-hardness of approximating 2-CSPs and directed steiner network, in: Anna R. Karlin (Ed.), 9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA, in: LIPIcs, vol. 94, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, pp. 36:1–36:20.

Yijia Chen is a professor of Computer Science at Shanghai Jiao Tong University. He mainly works in logic in computer science, computational complexity, and algorithmic graph theory, with a particular emphasis on the interaction between finite model theory and parameterized complexity.

Bingkai Lin is a professor of Computer Science at Nanjing University. He mainly works in parameterized complexity, approximation algorithms, and algorithmic graph theory.