Electronic Colloquium on Computational Complexity, Report No. 188 (2023)

Parameterized Inapproximability Hypothesis under ETH

Venkatesan Guruswami[*] Bingkai Lin[†] Xuandi Ren[‡] Yican Sun[§]
Kewen Wu[¶]

Abstract

The Parameterized Inapproximability Hypothesis (PIH) asserts that no fixed parameter tractable (FPT) algorithm can distinguish a satisfiable CSP instance, parameterized by the number of variables, from one where every assignment fails to satisfy an ε fraction of constraints for some absolute constant $\varepsilon > 0$. PIH plays the role of the PCP theorem in parameterized complexity. However, PIH has only been established under Gap-ETH, a very strong assumption with an inherent gap.

In this work, we prove PIH under the Exponential Time Hypothesis (ETH). This is the first proof of PIH from a gap-free assumption. Our proof is self-contained and elementary. We identify an ETH-hard CSP whose variables take vector values, and constraints are either linear or of a special parallel structure. Both kinds of constraints can be checked with constant soundness via a "parallel PCP of proximity" based on the Walsh-Hadamard code.

[*]Simons Institute for the Theory of Computing, and Departments of EECS and Mathematics, UC Berkeley. Email: venkatg@berkeley.edu. Research supported in part by NSF grants CCF-2228287 and CCF-2211972 and a Simons Investigator award.

[†]State Key Laboratory for Novel Software Technology, Nanjing University. Email: lin@nju.edu.cn

[‡]Department of EECS, UC Berkeley. Email: xuandi_ren@berkeley.edu. Supported in part by NSF grant CCF-2228287.

[§]School of Computer Science, Peking University. Email: sycpku@pku.edu.cn

[¶]Department of EECS, UC Berkeley. Email: shlw_kevin@hotmail.com. Supported by a Sloan Research Fellowship and NSF CAREER Award CCF-2145474.

ISSN 1433-8092

Contents

1 Introduction

A comprehensive understanding of NP-hard problems is an everlasting pursuit in the TCS community. Towards this goal, researchers have proposed many alternative hypotheses as strengthenings of the classic $\mathsf{P} \neq \mathsf{NP}$ assumption to obtain more fine-grained lower bounds for NP-hard problems, for example, Exponential Time Hypothesis (ETH) [IP01], Strong Exponential Time Hypothesis (SETH) [IP01, CIP09], Gap Exponential Time Hypothesis (Gap-ETH) [Din16].

Besides a richer family of hypotheses, *approximation* and *fixed parameter tractability (FPT)* are also two orthogonal approaches to cope with NP-hardness.

- In the approximation setting, we consider optimization problem, where input instances are associated with a cost function and the goal is to find a solution with cost function value close to the optimum.
- In the fixed parameter tractability (FPT) setting, every instance is attached with a parameter k indicating specific quantities (e.g., the optimum, the treewidth) of the instance. This setting treats k as a parameter much smaller than the instance size n, i.e., $1 \leq k \ll n$. Thus, the required runtime of the algorithm is relaxed from $n^{O(1)}$ to $f(k) \cdot n^{O(1)}$, for any computable function f. The class FPT is the set of parameterized problems that admit an algorithm within this running time.

  The seminal studies in this setting built up *parameterized complexity theory* [DF95a, DF95b, FG06]. In this theory, there are also alternative hypotheses as a strengthening of $\mathsf{P} \neq \mathsf{NP}$. For example, $\mathsf{W}[1] \neq \mathsf{FPT}$, which is equivalent to the statement that k-CLIQUE has no $f(k) \cdot n^{O(1)}$-time algorithm.

Recently, there has been an extensive study at the intersection of these two settings: the existence (or absence) of approximation algorithms that solve NP-hard problems in FPT time.

- On the algorithmic side, approximation algorithms with FPT runtime have been designed for various NP-complete problems. Examples include VERTEX-COLORING [DHK05, Mar08], MIN-k-CUT [GLL18b, GLL18a, KL20, LSS20], k-PATH-DELETION [Lee19], k-CLUSTERING [ABB+23], k-MEANS / k-MEDIANS [CGTS02, KMN+04, LS16, BPR+17, CGK+19, ANSW20], MAX k-HYPERGRAPH VERTEX COVER [SF17, Man19], FLOW TIME SCHEDULING [Wie18].
- In terms of computational hardness, the existence of such algorithms for certain NP-complete problems has also been ruled out under reasonable assumptions: k-SETCOVER [CCK+17, CL19, KLM19, Lin19, KN21, LRSW23a], k-SETINTERSECTION [Lin18, BKN21], k-STEINER ORIENTATION [Wło20], MAX-k-COVERAGE [Man20], k-SVP, k-MINDISTANCEPROBLEM and related problems [Man20, BBE+21, BCGR23].

  An exciting recent line of work [CCK+17, Lin21, LRSW22, KK22, CFLL23, LRSW23b] shows that approximating k-CLIQUE is not FPT under Gap-ETH, ETH, and $\mathsf{W}[1] \neq \mathsf{FPT}$.

We refer to the survey by Feldmann, Karthik, Lee, and Manurangsi [FKLM20] for a detailed discussion.

The Quest for Parameterized PCP-Type Theorems. Despite all the recent progress in the study of parameterized inapproximability, the reductions presented in these papers are often ad-hoc and tailored to the specific problems in question. Obtaining a unified and powerful machinery for parameterized inapproximability, therefore, becomes increasingly important.

A good candidate is to *establish a parameterized PCP-type theorem*. The PCP theorem [AS98, ALM+98, Din07], a cornerstone of modern complexity theory, gives a polynomial time reduction from an NP-hard problem like 3SAT to a gap version of 3SAT where the goal is to distinguish satisfiable instances from those for which every assignment fails to satisfy a γ fraction of clauses for some absolute constant $\gamma > 0$. This then serves as the starting point for a large body of inapproximability results for fundamental problems, including constraint satisfaction, graph theory, and optimization.

As discussed in [FKLM20], the current situation in the parameterized world is similar to that of the landscape of the traditional hardness of approximation *before* the celebrated PCP theorems was established. Given the similarity, the following folklore open problem has been recurring in the field of parameterized inapproximability:

Can we establish a PCP-type theorem in the parameterized complexity theory?

In light of its rising importance, Lokshtanov, Ramanujan, Saurabh, and Zehavi [LRSZ20] formalized and entitled the above question as *Parameterized Inapproximability Hypothesis (PIH)*. Here we present the following reformulation[1] of PIH due to [FKLM20]:

Hypothesis 1.1 (Parameterized Inapproximability Hypothesis)**.** *There is an absolute constant*[2] $\varepsilon > 0$*, such that no fixed parameter tractable algorithm which, takes as input a 2CSP* G *with* k *variables of size-*n *alphabets, can decide whether* G *is satisfiable or at least* ε *fraction of constraints must be violated.*

Similar to the PCP theorem, PIH, if true, serves as a shared beginning for results in parameterized hardness of approximation: k-CLIQUE, k-SETCOVER, k-EXACTCOVER [GRS23], SHORTEST VECTOR [BBE+21, BCGR23], DIRECT ODD CYCLE TRANSVERSAL [LRSZ20], DETERMINANT MAXIMIZATION and GRID TILING [Ohs22], Baby PIH [GRS23], k-MAXCOVER [KLM19], and more.

Prior to our work, PIH was only proved under the Gap-ETH assumption, the gap version of ETH. Since there is an inherent gap in Gap-ETH, the result can be obtained by a simple gap-preserving reduction (see, e.g., [FKLM20]). Indeed, it is often recognized that gap-preserving reductions are much easier than gap-producing reductions [FGL+96]. A more desirable result is, analogous to the PCP theorem, to create a gap from a gap-free assumption:

Can we prove PIH under an assumption without an inherent gap?

1.1 Our Results

We answer the above question in the affirmative by proving the *first* result to base PIH on a gap-free assumption. We consider the famous Exponential Time Hypothesis (ETH) [IP01], a fundamental gap-free hypothesis in the modern complexity theory and a weakening of the Gap-ETH assumption.

Hypothesis (Exponential Time Hypothesis (ETH), Informal)**.** *Solving* 3SAT *needs* $2^{\Omega(n)}$ *time.*

Our main theorem can be stated concisely as:

Theorem 1.2 (Main)**.** *ETH implies PIH.*

[1]The original statement of PIH in [LRSZ20] replaces the runtime bound by W[1]-hardness, and the reformulation by [FKLM20] suffices for applications.

[2]The exact constant here is not important. Starting from a constant $\varepsilon > 0$, one can boost it to $1 - \eta$ for any constant $\eta > 0$ by standard reductions.

In Theorem 3.1, we provide a quantitative version of Theorem 1.2 that presents an explicit runtime lower bound of $f(k) \cdot n^{\Omega\left(\sqrt{\log\log k}\right)}$ for the problem in Hypothesis 1.1 under ETH.

As a byproduct of the above quantitative bound, we have the following probabilistic checkable proof version of the main theorem (see Theorem 3.2 for the full version). This can be seen as a PCP theorem in the parameterized world where the proof length depends only on k (which is supposed to be a small growing parameter), but the alphabet size is the significantly growing parameter. The runtime of the PCP verifier is in FPT.

Theorem 1.3. *For any integer $k \geq 1$, 3SAT has a PCP verifier which can be constructed in time $f(k) \cdot |\Sigma|^{O(1)}$ for some computable function f, makes two queries on a proof with length $2^{2^{O(k^2)}}$ and alphabet size $|\Sigma| = 2^{O(n/k)}$, and has completeness 1 and soundness $1 - \frac{1}{9600}$.*

As mentioned, PIH serves as a unified starting point for many parameterized inapproximability results. Below, we highlight some new ETH-hardness of approximation for fundamental parameterized problems obtained by combining our result and existing reductions from PIH.

Application Highlight: k-ExactCover. k-EXACTCOVER (also known as k-UNIQUE SET COVER) is a variant of the famous k-SETCOVER problem. In the ρ-approximation version of this problem, denoted by $(k, \rho \cdot k)$-EXACTCOVER, we are given a universe U and a collection $\mathcal{S}$ of subsets of U. The goal is to distinguish the following two cases.

- There exists k *disjoint* subsets that cover the whole universe U, i.e., the union of the k subsets is exactly the whole universe U.
- Any $\rho \cdot k$ subsets of $\mathcal{S}$ cannot cover U.

Here, the parameter is the optimum k. Note that k-EXACTCOVER is an easier problem than k-SETCOVER due to the additional disjointness property, proving computational hardness even harder. On the positive side, this additional structure also makes $(k, \rho \cdot k)$-EXACTCOVER an excellent proxy for subsequent reductions. We refer interested readers to previous works for details [ABSS97, Man20].

For constant $\rho > 1$, the hardness of $(k, \rho \cdot k)$-EXACTCOVER was only proved under assumptions with inherent gaps [Man20, GRS23], imitating the reduction in the non-parameterized world [Fei98]. It was still a mystery whether we could derive the same result under a weaker and gap-free assumption. Combining its PIH hardness (see e.g., [GRS23][3]) with our main theorem (Theorem 1.2), we prove the first inapproximability for k-EXACTCOVER under a gap-free assumption.

Corollary 1.4. *Assuming ETH, for any absolute constant $\rho \geq 1$, no FPT algorithm can decide $(k, \rho \cdot k)$-EXACTCOVER.*

Application Highlight: Directed Odd Cycle Transversal. Given a directed graph D, its directed odd cycle transversal, denoted by DOCT(D), is the minimum set S of vertices such that deleting S from D results in a graph with no directed odd cycles. The ρ-approximating version of the directed odd cycle transversal problem, denoted by $(k, \rho \cdot k)$-DOCT, is to distinguish directed graphs D with DOCT$(D) \leq k$, from those with DOCT$(D) \geq \rho \cdot k$. The parameter of this problem is the optimum k. This problem is a generalization of several well studied problems including DIRECTED

[3][GRS23] proves the hardness of $(k, \rho \cdot k)$-EXACTCOVER under a weaker version of PIH, namely, Average Baby PIH with rectangular constraints.

Feedback Vertex Set and Odd Cycle Transversal. For a brief history of this problem, we refer to the previous work [LRSZ20]. In their work, the authors prove the following hardness of $(k, \rho \cdot k)$-DOCT.

Theorem 1.5 ([LRSZ20])**.** *Assuming PIH, for some $\rho \in (1, 2)$, no FPT algorithm can decide $(k, \rho \cdot k)$-DOCT.*

Combining the theorem above with Theorem 1.2, we establish the first hardness of $(k, \rho \cdot k)$-DOCT under a gap-free assumption.

Corollary 1.6. *Assuming ETH, for some $\rho \in (1, 2)$, no FPT algorithm can decide $(k, \rho \cdot k)$-DOCT.*

1.2 Overview of Techniques

To prove our main theorem (Theorem 1.2), we present an efficient reduction from 3SAT formulas to parameterized CSPs of k variables with a constant gap.

To construct such a reduction, we follow the widely-used paradigm for proving PCP theorems [AS98, ALM+98, GOS20]. Via this approach, we first arithmetize 3SAT into an intermediate CSP (usually a constant-degree polynomial system in the literature) with k variables and alphabet Σ_1. Then, we decide on a locally testable and correctable code $C \colon \Sigma_1^k \to \Sigma_2^{k'}$ (e.g., the quadratic code [ALM+98], the Reed-Muller code [AS98], or the long code [GOS20]), and treat the proof π as an encoding of some assignment $\sigma \colon [k] \to \Sigma_1$ to the intermediate CSP (viewed as a vector in Σ_1^k). Leveraging the power of the local testability and correctability of C, we will check whether the input proof is (close to) the encoding of an assignment that satisfies the intermediate CSP.

Our Plan. To follow the outline above and also factor in the runtime and the parameter blowup, our plan is as follows:

1. First, we need to design an appropriate intermediate parameterized CSP problem, which has some runtime lower bound under ETH. In the parameterized setting, the number of variables is a small parameter k, while the alphabet $|\Sigma_1|$ holds the greatest order of magnitude.

2. Second, we need to construct an error correcting code C, which can be used to encode a solution of the intermediate CSP, and allows us to locally check its satisfiability. Here the efficiency of the code is also measured in a parameterized sense that the alphabet size is polynomial in $|\Sigma_1|$, and codeword length can be arbitrary in k but independent of $|\Sigma_1|$.

However, the plan above confronts the following basic obstacle. The constructions in proving the PCP theorems usually require the proof length $|\pi| = |\Sigma_1|^{\Omega(k)}$. On the other hand, as illustrated in Item 2 above, we must eliminate $|\Sigma_1|$ in the proof length to make sure that the reduction is FPT.

Vectorization. We bypass this obstacle by applying *vectorization*, an idea also used in [LRSW23b]. In detail, we enforce the alphabet Σ_1 to be a vector space $\mathbb{F}^d$, where $\mathbb{F}$ is a field of constant size. In this way, an assignment $\sigma \in \Sigma_1^k = (\mathbb{F}^d)^k$ can be viewed as d parallel sub-assignment in $\mathbb{F}^k$. Thus, if we have a good code $C \colon \mathbb{F}^k \to \mathbb{F}^{k'}$ that tests the validity of a sub-assignment, we can encode σ by separately encoding each sub-assignment and combining them as an element in $(\mathbb{F}^{k'})^d = (\mathbb{F}^d)^{k'}$. Since $|\mathbb{F}|$ is a constant, this makes k' dependent only on k but not on the whole alphabet $\Sigma_1 = \mathbb{F}^d$.

Guided by the vectorization idea, we aim to design an ETH-hard intermediate CSP problem where the alphabet is a vector space. Furthermore, to facilitate the construction of the error correcting code C in the second step, we also hope that there are appropriate restrictions on the constraints of this intermediate CSP problem. The constraints should be neither too restrictive (which loses the ETH-hardness) nor too complicated (which hinders an efficient testing procedure). Following these intuitions, we define the following *Vector-Valued CSPs* as our intermediate problem.

Vector-Valued CSPs. Vector-Valued CSPs (Definition 3.3) are CSPs with the following additional features:

- The parameter k is the number of variables.
- The alphabet is a vector space $\mathbb{F}^d$, where $\mathbb{F}$ is a finite field of characteristic two and constant size and d holds the greatest order of magnitude.
- Each constraint is either a (coordinate-wise) parallel constraint, or a linear constraint, where:
  - A parallel constraint is defined by a sub-constraint $\Pi^{sub} : \mathbb{F} \times \mathbb{F} \to \{0, 1\}$ and a subset of coordinates $Q \subseteq [d]$. It checks Π^{sub} for every coordinate in Q of the vector assignments.
  - A linear constraint is defined by a matrix M. It enforces that two vector assignments satisfy a linear equation specified by M.
- Each variable is related to at most one parallel constraint.

We emphasize that vector-valued CSPs will become fixed parameter tractable if all constraints are linear (resp., parallel). In detail, one can handle linear constraints by efficiently solving a system of linear equations, or handle parallel constraints by brute force enumeration individually for each coordinate. However, due to our reduction, one cannot solve vector-valued CSPs with both constraint types efficiently under ETH.

3SAT **to Vector-Valued CSPs.** In Theorem 3.4, we establish the ETH-hardness of vector-valued CSP instances by a series of standard transformations.

First, we partition the clauses and variables of a 3SAT formula respectively into k parts. Each of the $2k$ parts is then built as a CSP variable, which takes assignments of that part of clauses/variables. The alphabet is therefore a vector space.

Then, we impose constraints between clause parts and variable parts. Each constraint is a conjunction of clause validity and clause-variable consistency. These constraints ensure that the $2k$ partial assignments correspond to a global satisfying assignment to the original 3SAT formula.

However, the constraints above are neither parallel nor linear. To make them parallel, we first appropriately split constraints, then duplicate each variable into several copies and spread out its constraints. After this procedure, each variable is related to exactly one constraint, and each constraint is the same sub-constraint applied in a matching way on the d coordinates of the related vector-variables. We can thus permute the d coordinates of each variable accordingly and obtain the parallel constraint form we desire. In addition, we also need to check the (permuted) consistency between different duplicates. These checks can be done using permuted equality constraints, which are special forms of linear constraints.

Vector-Valued CSPs to Constant-Gap CSPs. In Theorem 3.5, we construct another FPT reduction from a vector-valued CSP to a general constant-gap parameterized CSP in three steps.

- First, we split the vector-valued CSP instance into two by partitioning the constraints into a linear part and a parallel part.
- Next, for each of the two sub-instances, we construct a randomized verifier to check whether all constraints in it are satisfied. The verifier takes as input a parallel encoding of a solution. It then flips random coins, makes a constant number of queries based on the randomness, and decides whether to accept the input proof or not based on the query result. The verifier will have a constant soundness and a constant proximity parameter. In the traditional complexity theory, such verifiers are also known as Probabilistic Checkable Proof of Proximity (PCPP) verifiers.

  In our proof, the verifier is designed separately for linear constraints and parallel constraints. The consistency of the two verifiers is guaranteed via a unified parallel Walsh-Hadamard encoding of the solution, shared by both verifiers.
- Finally, we obtain a constant-gap CSP instance by a standard reduction from probabilistic checkable proof verifiers to CSPs.

The proof is then completed by combining the two reductions above. The crux of our proof is the design of the PCPP verifiers. Below, we present high-level descriptions of this part.

PCPPs for Vector-Valued CSPs with Parallel Constraints. Fix a vector-valued CSP instant G with parallel constraints only. The key observation in designing PCPPs for G is that, though the parallel sub-constraints can be arbitrary, different coordinates of the vector-variables are independent. Let k be the number of variables in G and let d be the dimension of the vector-variables.

Following the observation above, we can split G into d sub-instances $G_1, \ldots, G_d$ with respect to the d coordinates. Each G_i is a CSP instance with k variables and alphabet $\mathbb{F}$. A vector-valued assignment σ satisfies G iff the sub-assignment of σ on the i-th coordinate satisfies G_i for each $i \in [d]$.

After splitting, the alphabet of each G_i is only $\mathbb{F}$. We can thus follow the classical construction [ALM+98, AB09] of PCPPs to construct a verifier A_i to efficiently and locally check the satisfiability of G_i. In addition, since every vector-variable is related to at most one parallel constraint in G, the number of distinct sub-instances among $G_1, \ldots, G_d$ depends only on k, not on d. This allows us to combine $A_1, \ldots, A_d$ into a single verifier A that works over the original alphabet $\mathbb{F}^d$ with blowup dependent only on k, not on d. See Section 5 for details.

PCPPs for Vector-Valued CSPs with Linear Constraints. To design a verifier for linear constraints, we leverage the power of the Walsh-Hadamard code to decode any linear combinations of the messages. Fix a vector-valued CSP instance G with linear constraints only. For each linear constraint $e = (u_e, v_e) \in E$, we further denote its form by $\mathbb{1}_{u_e = M_e v_e}$.

To test the conjunction of all linear constraints, it is natural to consider the linear combination of these constraints. In detail, we pick independently random $\lambda_1, \ldots, \lambda_{|E|} \in \mathbb{F}$, and test whether:

$$\sum_{e \in E} \lambda_e u_e = \sum_{e \in E} \lambda_e \cdot M_e v_e$$

By the random subsum principle [AB09], if any one of the linear constraints is violated, the equation above does not hold with high probability.

Following this idea, we introduce auxiliary variables $z_{v,e}$ for each variable v and constraint e, which is supposed to be $M_e v$. We set up the parallel version of the Walsh-Hadamard code over the assignments to the variables in G and the auxiliary variables $z_{v,e}$. In this way, we can decode both the LHS and RHS of the equation above by two queries on the Walsh-Hadamard code, and then check whether the equation holds.

We need extra testing procedures to ensure $z_{v,e}$ equals $M_e v$. This is again achieved by the random subsum principle. See Section 6 for details.

1.3 Discussions

Here we discuss related works and propose future directions.

Related Works. As mentioned above, prior to our work, PIH was only known to hold under Gap-ETH [CCK+17, DM18]. The techniques there do not apply here since their proofs rely on an inherent gap from the assumption, which ETH does not have.

Using a different approach, Lin, Ren, Sun, and Wang [LRSW22, LRSW23b] proposed to prove PIH via a strong lower bound for constant-gap k-CLIQUE. This is reminiscent of [BGS98], where the NP-hardness of constant-gap CLIQUE leads to a free-bit PCP. However, the construction in [BGS98] does not apply in the parameterized setting since the proof length will be too long. In addition, the framework of [LRSW23b] only designs a weaker variant of PCPP, which can only locally test the validity of a single constraint rather than the conjunction of all constraints. Moreover, the boosting from weak PCPPs to standard PCPPs seems to meet an information-theoretic barrier from locally decodable codes. In contrast, we successfully design PCPPs for special CSPs in this work, which is based on a key observation that CSPs remain ETH-hard even when the variables are vector-valued and the constraints are either parallel or linear.

Furthermore, a recent work by Guruswami, Ren, and Sandeep [GRS23] established a weaker version of PIH called Baby PIH, under $\mathsf{W}[1] \neq \mathsf{FPT}$. However, they also gave a counterexample to show that the basic direct product approach underlying their reduction is not strong enough to establish PIH.

Future Directions. We highlight some interesting open directions.

- Starting from PIH and by our work, many previous parameterized hardness of approximation results can now be based on ETH (see Corollary 1.4 and Corollary 1.6 as representatives). However, there are still many basic problems whose parameterized inapproximability remains unknown, e.g., MAX k-COVERAGE and k-BALANCED BICLIQUE [CCK+17, FKLM20].

  Can we discover more ETH-based parameterized inapproximability results? Since there is already a gap in PIH, we expect that reducing PIH to other parameterized problems would be easier than reducing directly from gap-free 3SAT.

- We have presented a gap-producing reduction from ETH to PIH. It is natural to ask whether we can prove PIH under the minimal hypothesis $\mathsf{W}[1] \neq \mathsf{FPT}$. Our paper constructs an FPT reduction from vector-valued CSPs to gap CSPs.

  We remark that our vector-valued CSP instances are closely related to an $\mathsf{M}[1]$-complete problem MINI-3SAT [Fel03] where $\mathsf{M}[1]$ is an intermediate complexity class between FPT and $\mathsf{W}[1]$. Thus, unless $\mathsf{M}[1] = \mathsf{W}[1]$, our proof may not be directly generalized to prove PIH under $\mathsf{W}[1] \neq \mathsf{FPT}$. We refer interested readers to [CG07] for a detailed discussion of these complexity classes and hierarchies.

Paper Organization. In Section 2, we define necessary notation and introduce useful tools from the literature. Then, the paper is organized in a modular manner. First, in Section 3, we present the proof of our main result with the proofs of technical lemmas deferred to later sections. Then, in Section 4, we show how to obtain a vector-valued CSP instance with desired structures from 3SAT as needed in Section 3. Next, in Section 5, we design the probabilistic verifier for parallel constraints in the CSP instance, another building block needed in Section 3. Finally, in Section 6, we give the probabilistic verifier for linear constraints in the CSP instance, the last missing piece of Section 3.

2 Preliminaries

For a positive integer n, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. We use log to denote the logarithm with base 2. For an event $\mathcal{E}$, we use $\mathbb{1}_{\mathcal{E}}$ as the indicator function, which equals 1 if $\mathcal{E}$ happens and equals 0 if otherwise. For disjoint sets S and T, we use $S \dot\cup T$ to denote their union while emphasizing $S \cap T = \emptyset$. For a prime power $q = p^t$ where p is a prime and $t \geq 1$ is an integer, we use $\mathbb{F}_q$ to denote the finite field of order p^t and characteristic p.

We use superscript $\top$ to denote vector / matrix transpose. For two vectors $u, v \in \mathbb{F}^d$, we use $\langle u, v \rangle$ to denote their inner product which equals $u^\top v$ (or $v^\top u$). For two matrices $A, B \in \mathbb{F}^{d \times d}$, we use $\langle A, B \rangle = \sum_{i,j \in [d]} A_{i,j} B_{j,i}$ to denote their inner product.

Throughout the paper, we use $O(\cdot), \Theta(\cdot), \Omega(\cdot)$ to hide absolute constants that do not depend on any other parameter.

2.1 (Parameterized) Constraint Satisfaction Problems

CSP. In this paper, we only focus on constraint satisfaction problems (CSPs) of arity two. Formally, a CSP instance G is a quadruple $(V, E, \Sigma, \{\Pi_e\}_{e \in E})$, where:

- V is for the set of variables.
- E is for the set of constraints. Each constraint $e = \{u_e, v_e\} \in E$ has arity 2 and is related to two distinct variables $u_e, v_e \in V$.

  The *constraint graph* is the undirected graph on vertices V and edges E. Note that we allow multiple constraints between a same pair of variables and thus the constraint graph may have parallel edges.
- Σ is for the alphabet of each variable in V. For convenience, we sometimes have different alphabets for different variables and we will view them as a subset of a grand alphabet Σ with some natural embedding.
- $\{\Pi_e\}_{e \in E}$ is the set of constraint validity functions. Given a constraint $e \in E$, the validity function $\Pi_e(\cdot, \cdot) \colon \Sigma \times \Sigma \to \{0, 1\}$ checks whether the constraint e between u_e and v_e is satisfied.

We use $|G| = (|V| + |E|) \cdot |\Sigma|$ to denote the *size* of a CSP instance G.

Assignment and Satisfiability Value. An *assignment* is a function $\sigma \colon V \to \Sigma$ that assigns each variable a value in the alphabet. The *satisfiability value* for an assignment σ, denoted by $\mathsf{val}(G, \sigma)$, is the fraction of constraints satisfied by σ, i.e., $\mathsf{val}(G, \sigma) = \frac{1}{|E|} \sum_{e \in E} \Pi_e(\sigma(u_e), \sigma(v_e))$. The satisfiability value for G, denoted by $\mathsf{val}(G)$, is the maximum satisfiability value among all assignments, i.e., $\mathsf{val}(G) = \max_{\sigma\colon V\to\Sigma} \mathsf{val}(G,\sigma)$. We say that an assignment σ is a *solution* to a CSP instance G if $\mathsf{val}(G,\sigma) = 1$, and G is *satisfiable* iff G has a solution.

When the context is clear, we omt σ in the description of a constraint, i.e., $\Pi_e(u_e, v_e)$ stands for $\Pi(\sigma(u_e), \sigma(v_e))$.

Parameterization and Fixed Parameter Tractability. For an instance G, the *parameterization* refers to attaching the parameter $k := |V|$ (the size of the variable set) to G and treating the input as a (G, k) pair. We think of k as a growing parameter that is much smaller than the instance size $n := |G|$. A promise problem $L_{\mathsf{yes}} \cup L_{\mathsf{no}}$ is *fixed parameter tractable (FPT)* if it has an algorithm which, for every instance G, decides whether $G \in L_{\mathsf{yes}}$ or $G \in L_{\mathsf{no}}$ in $f(k) \cdot n^{O(1)}$ time for some computable function f.

FPT Reduction. An *FPT reduction* from $L_{\mathsf{yes}} \cup L_{\mathsf{no}}$ to $L'_{\mathsf{yes}} \cup L'_{\mathsf{no}}$ is an algorithm $\mathcal{A}$ which, on every input $G = (V, E, \Sigma, \{\Pi_e\}_{e\in E})$ outputs another instance $G' = (V', E', \Sigma', \{\Pi'_e\}_{e\in E'})$ such that:

- COMPLETENESS. If $G \in L_{\mathsf{yes}}$, then $G' \in L'_{\mathsf{yes}}$.
- SOUNDNESS. If $G \in L_{\mathsf{no}}$, then $G' \in L'_{\mathsf{no}}$.
- FPT. There exist universal computable functions f and g such that $|V'| \le g(|V|)$ and the runtime of $\mathcal{A}$ is bounded by $f(|V|) \cdot |G|^{O(1)}$.

ε-Gap k-CSP. We mainly focus on the gap version of the parameterized CSP problem. Formally, an ε-GAP k-CSP problem needs to decide whether a given CSP instance $(G, |V|)$ with $|V| = k$ satisfies $\mathsf{val}(G) = 1$ or $\mathsf{val}(G) < 1 - \varepsilon$. The exact version is equivalent to 0-GAP k-CSP.

Parameterized Inapproximability Hypothesis (PIH). *Parameterized Inapproximability Hypothesis (PIH)*, first[4] formulated by Lokshtanov, Ramanujan, Saurabh, and Zehavi [LRSZ20], is a central conjecture in the parameterized complexity theory, which, if true, serves as a parameterized counterpart of the celebrated PCP theorem. Below, we present a slight reformulation of PIH, asserting fixed parameter intractability (rather than $W[1]$-hardness specifically) of gap CSP.

Hypothesis 2.1 (PIH). *For an absolute constant $0 < \varepsilon < 1$, no FPT algorithm can decide ε-GAP k-CSP.*

Exponential Time Hypothesis (ETH). *Exponential Time Hypothesis (ETH)*, first proposed by Impagliazzo and Paturi [IP01], is a famous strengthening of the $\mathsf{P} \neq \mathsf{NP}$ hypothesis and provides a foundation for fine-grained understandings in the modern complexity theory.

Definition 2.2 (3SAT). A 3CNF formula φ on n Boolean variables is a conjunction of m clauses, where each clause is a disjunction of three literals and each literal is a variable or its negation. The goal of the 3SAT problem is to decide whether φ is satisfiable or not.

The original ETH is stated in the general 3SAT problem. In this paper, for convenience, we use the following variant due to the sparsification lemma [IPZ01] and Tovy's reduction [Tov84], which gives 3SAT additional structure.

Hypothesis 2.3 (ETH). *No algorithm can decide 3SAT within runtime $2^{o(n)}$, where additionally each variable is contained in at most four clauses and each clause contains exactly three distinct variables.*[5]

[4]As noted in [LRSZ20], prior to their work, this hypothesis was already informally stated by quite a few researchers as a natural formulation of the PCP theorem in parameterized complexity.

[5]We say a variable x is contained in a clause C if the literal x or $\neg x$ appears in C.

2.2 Parallel Walsh-Hadamard Code

As mentioned in Subsection 1.2, the key step to bypass the obstacle in previous constructions is vectorization and parallel encoding of an error correcting code. In this paper, we only consider the parallelization of the famous *Walsh-Hadamard code*, a classic error correcting code that is locally testable and correctable. First, we recall standard notions in coding theory.

Given two words (aka strings) $x, y \in \Sigma^K$ and same length K, their *relative distance* $\Delta(x, y)$ is the fraction of coordinates that they differ, i.e., $\Delta(x,y) = \frac{1}{K}|\{i \in [K] \colon x_i \neq y_i\}|$. We say $x \in \Sigma^K$ is δ-far (resp., δ-close) from a set of words $S \subseteq \Sigma^K$ if $\Delta(x, S) := \min_{y \in S} \Delta(x, y) \geq \delta$ (resp., $\leq \delta$).

Definition 2.4 (Error Correcting Codes (ECCs))**.** An error correcting code is the image of the encoding map $C \colon \Sigma_1^k \to \Sigma_2^K$ with message length k, codeword length K. We say that the ECC has a relative distance δ if $\Delta(C(x), C(y)) \geq \delta$ holds for any distinct $x, y \in \Sigma_1^k$. We use Im(C) to denote the codewords of C.

Definition 2.5 (Parallel Walsh-Hadamard Code)**.** Let $\mathbb{F}$ be a finite field and $(a_1, a_2, \ldots, a_k) \in (\mathbb{F}^d)^k$ be a tuple of k vectors in $\mathbb{F}^d$. We view it as a matrix $A = (a_1, a_2, \ldots, a_k) \in \mathbb{F}^{d \times k}$ where the i-th column is the vector a_i.

The parallel Walsh-Hadamard encoding $\mathtt{PWH}(A)$ of A is a codeword indexed by $\mathbb{F}^k$ where each entry is a vector in $\mathbb{F}^d$. Alternatively, $\mathtt{PWH}(A)$ is a function mapping $\mathbb{F}^k$ to $\mathbb{F}^d$ that enumerates linear combinations of the column vectors of A. Formally, for each $b \in \mathbb{F}^k$, we have $\mathtt{PWH}(A)[b] = Ab$.

We remark that the parallel Walsh-Hadamard code is also known as interleaved Hadamard code and linear transformation code [GGR11, DGKS08].

In the notation of Definition 2.4, $\mathtt{PWH}$ has $\Sigma_1 = \mathbb{F}^d$, $\Sigma_2 = \mathbb{F}^d$, and $K = |\mathbb{F}|^k$. Note that when $d = 1$, the parallel Walsh-Hadamard code coincides with the standard Walsh-Hadamard code. It is clear that $\mathtt{PWH}$ has the relative distance $\delta = 1 - \frac{1}{|\mathbb{F}|}$, which is at least $\frac{1}{2}$ since $|\mathbb{F}| \geq 2$ holds always.

Local Testability and Correctability. Fix a word $w \in (\mathbb{F}^d)^{\mathbb{F}^k}$ and treat it as a map from $\mathbb{F}^k$ to $\mathbb{F}^d$. To test whether w is close to a codeword of $\mathtt{PWH}$, we perform the famous *BLR test* [BLR93], which samples uniformly random $a, b \in \mathbb{F}^k$ and accept if $w[a] + w[b] = w[a + b]$ by three queries to w. The following theorem establishes the soundness of this test.

Theorem 2.6. *If* $\mathbf{Pr}_{a,b \in \mathbb{F}^k}[w[a] + w[b] = w[a + b]] \geq 1 - \varepsilon$, *then* $\Delta(x, \mathrm{Im}(\mathtt{PWH})) \leq 6\varepsilon$.

Proof. If $\varepsilon < 1/6$, we apply the bound from [Gol16, Theorem 3] and obtain $\Delta(x, \mathrm{Im}(\mathtt{PWH})) \leq 2\varepsilon \leq 6\varepsilon$. Otherwise $\varepsilon \geq 1/6$ and we naturally have $\Delta(x, \mathrm{Im}(\mathtt{PWH})) \leq 1 \leq 6\varepsilon$. □

Assume w is η-close to an actual codeword w^* of $\mathtt{PWH}$. To obtain the value of $w^*[x]$ for some $x \in \mathbb{F}^k$, we can draw a uniform $a \in \mathbb{F}^k$ and compute $w[x + a] - w[a]$ by two queries. The following fact concerns the soundness of this procedure.

Fact 2.7. *If w is η-close to some $w^* \in \mathrm{Im}(\mathtt{PWH})$, then* $\mathbf{Pr}_{a \in \mathbb{F}^k}[w[x + a] - w[a] = w^*[x]] \geq 1 - 2\eta$.

2.3 Probabilistic Checkable Proofs with Proximity

Probabilistic Checkable Proofs of Proximity (PCPP, also known as assignment testers) [BGH+06, DR06] are essential gadgets when proving the PCP theorem [AS98, Din07, AB09]. There, the gadget is used to verify whether a set of Boolean variables is close to a solution of a formula given by a circuit.

In this paper, we reformulate PCPP under the parameterized regime. Our reformulation is compatible with the parallel encoding. To conveniently combine different PCPPs, we specialize PCPPs into their PWH-based constructions.[6] Formally, we define the following *parallel probabilistic checkable proofs with proximity (PPCPP)*.

Definition 2.8 ($(q,\delta,\varepsilon,f,g)$-PPCPPs)**.** Let f and g be two computable functions. Given a finite field $\mathbb{F}$ and a CSP instance $G = (V,E,\Sigma,\{\Pi_e\}_{e\in E})$ where $\Sigma = \mathbb{F}^d$. Its $(q,\delta,\varepsilon,f,g)$-PPCPP is a randomized verifier A with the following workflow: Recall that $k = |V|$ is the parameter of the CSP instance G.

- A takes as input two blocks of proofs $\pi_1 \circ \pi_2$ with alphabet $\mathbb{F}^d$, where:
  - π_1 has length $|\mathbb{F}|^k$ with entries indexed by vectors in $\mathbb{F}^k$, which is supposed to be the parallel Walsh-Hadamard encoding of some assignment to V.
  - π_2 has length at most $f(k)$. It is an auxiliary proof enabling an efficient verification procedure.
- A chooses a uniform $r \in [R_A]$, where R_A is at most $g(k)$, queries at most q positions in $\pi_1 \circ \pi_2$ based on r, and decides to accept or reject the proof after getting the query result.
- The list of queries made by A can be generated in time at most $h(k)\cdot|G|^{O(1)}$ for some computable function h.

The verifier A has the following properties.

- COMPLETENESS. For every solution σ of G, there exists a π_2 such that $\mathbf{Pr}[A \text{ accepts } \mathtt{PWH}(\sigma)\circ \pi_2] = 1$, where we treat an assignment $\sigma\colon V \to \mathbb{F}^d$ as a vector in $(\mathbb{F}^d)^{|V|}$.
- SOUNDNESS. If $\mathbf{Pr}[A \text{ accepts } \pi_1\circ\pi_2] \geq 1-\varepsilon$, there exists some solution σ of G such that $\Delta(\pi_1, \mathtt{PWH}(\sigma)) \leq \delta$.

Intuitively, PPCPPs check whether π_1 is close to the Walsh-Hadamard encoding of some solution of G. Like the traditional PCPP, parallel PCPPs are also tightly connected with CSPs. The following standard reduction establishes the connection.

Definition 2.9 (Reduction from PPCPPs to CSPs)**.** Given a $(q,\delta,\varepsilon,f,g)$-PPCPP verifier A for a CSP $G = (V,E,\Sigma,\{\Pi_e\}_{e\in E})$ with $\Sigma = \mathbb{F}^d$, we define a CSP instance $G' = (V',E',\Sigma',\{\Pi'_e\}_{e\in E'})$, where $V' = V'_1 \dot\cup V'_2 \dot\cup V'_3$ and $\Sigma' = (\mathbb{F}^d)^q$, by the following steps:

- First, for $i = 1,2$, we treat each position of π_i as a single variable in V'_i with alphabet $\mathbb{F}^d$. Note that $|V'_1| = |\mathbb{F}|^k$ and $|V'_2| \leq f(k)$.
- Then, for each randomness $r \in [R_A]$, let S_r be the set of query positions over $\pi_1\circ\pi_2$ under randomness r; and we add a supernode z_r to V'_3 whose alphabet is $(\mathbb{F}^d)^{|S_r|}$, i.e., all possible configurations of the query result. Note that $|V'_3| \leq g(k)$.
- Finally, we add constraints between z_r and every query position $i \in S_r$. The constraint checks whether z_r is an accepting configuration, and the assignment of the position i is consistent with the assignment of z_r.

[6]The choice of encoding is typically abstracted out in standard definitions of PCPPs.

By construction, we can see that the completeness and soundness are preserved up to a factor of q under this reduction, where the loss comes from the construction where we split q queries into q consistency checks. In addition, since $|\pi_1 \circ \pi_2| \le |\mathbb{F}|^k + f(k)$, $R_A \le g(k)$, and the list of queries made by A can be generated in time $h(k) \cdot |G|^{O(1)}$, the reduction from G to G' is a FPT reduction.

Fact 2.10. *The reduction described in Definition 2.9 is an FPT reduction. Recall that $k = |V|$ is the parameter of G and $\Sigma = \mathbb{F}^d$ is the alphabet of G. We have the following properties for G':*

- Alphabet. *The alphabet of G' is $\Sigma' = \mathbb{F}^{d \cdot q}$.*
- Parameter Blowup. *The parameter of G' is $|V'| \le |\mathbb{F}|^k + f(k) + g(k)$.*
- Completeness. *For every solution σ of G, there exists a solution σ' of G' assigning $\mathtt{PWH}(\sigma)$ to V'_1.*
- Soundness. *For any assignment σ' satisfying $1 - \frac{\varepsilon}{q}$ fraction of the constraints in G', there exists a solution σ of G such that $\Delta(\sigma'(V'_1), \mathtt{PWH}(\sigma)) \le \delta$.*

3 Proof of The Main Theorem

In this section, we prove the following quantitative version of our main theorem (Theorem 1.2). To depict a clear picture, we will treat some technical constructions as black-boxes and relegate their proofs in subsequent sections.

Theorem 3.1. *Assume ETH is true. No algorithm can decide $\frac{1}{9600}$-Gap k-CSP within runtime $f(k) \cdot n^{o(\sqrt{\log\log k})}$ for any computable function f.*

As a byproduct of the quantitative analysis, we also have the following PCP-style theorem, which can be viewed as a parameterized PCP theorem.

Theorem 3.2. *For any integer $k \ge 1$, 3SAT has a PCP verifier which*

- *can be constructed in time $f(k) \cdot |\Sigma|^{O(1)}$ for some computable function f,*
- *makes two queries on a proof of length $2^{2^{O(k^2)}}$ and alphabet size $|\Sigma| = 2^{O(n/k)}$,*
- *has perfect completeness and soundness $1 - \frac{1}{9600}$.*

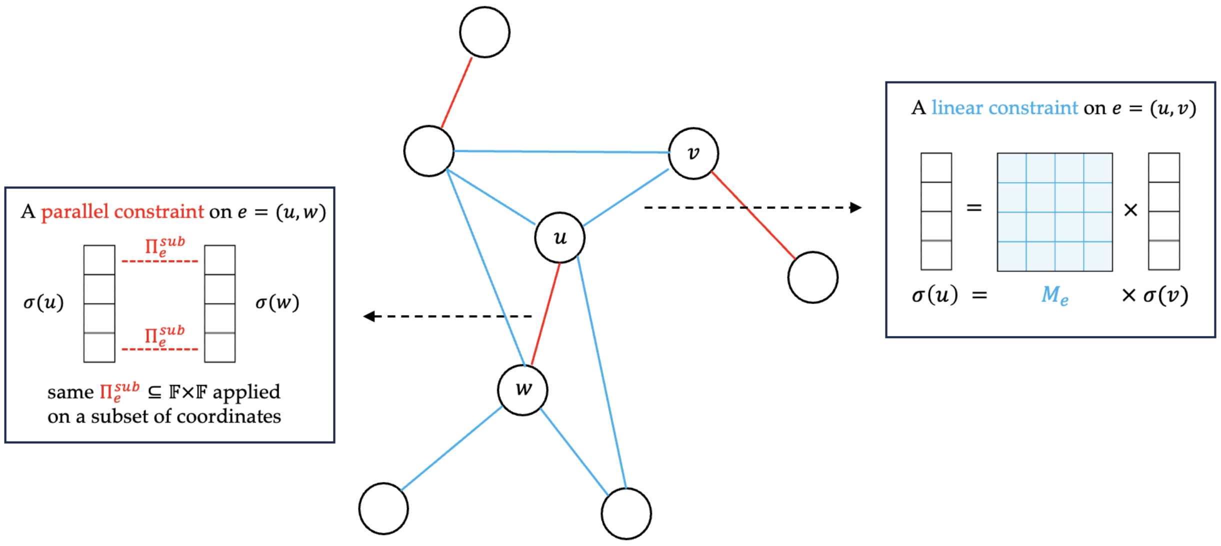
Our proof relies on an intermediate structured CSP, termed *Vector-Valued CSPs (VecCSP for short).*

Definition 3.3 (Vector-Valued CSP)**.** A CSP instance $G = (V, E, \Sigma, \{\Pi_e\}_{e \in E})$ is a VecCSP if the following additional properties hold.

- $\Sigma = \mathbb{F}^d$ is a d-dimensional vector space over a finite field $\mathbb{F}$ with characteristic 2.
- For each constraint $e = \{u, v\} \in E$ where $u = (u_1, u_2, \ldots, u_d)$ and $v = (v_1, v_2, \ldots, v_d)$, the constraint validity function Π_e is classified as one of the following forms in order[7]:
  - Linear. There exists a matrix[8] $M_e \in \mathbb{F}^{d \times d}$ such that
$$\Pi_e(u, v) = \mathbb{1}_{u = M_e v}.$$

[7]A constraint can be both linear and parallel (e.g., equality constraint). In this case, we classify it as linear instead of parallel, consistent with the order defined here.

[8]In the instance reduced from 3SAT, M_e is always a permutation matrix.



A parallel constraint on $e = (u, w)$
Π_e^{sub}
$\sigma(u)$
$\sigma(w)$
same $\Pi_e^{sub} \subseteq \mathbb{F}\times\mathbb{F}$ applied on a subset of coordinates
u
v
w
A linear constraint on $e = (u, v)$
$\sigma(u) = M_e \times \sigma(v)$

Figure 1: An example of a vector-valued CSP.

- PARALLEL. There exists a sub-constraint $\Pi_e^{sub} : \mathbb{F} \times \mathbb{F} \to \{0, 1\}$ and a subset of coordinates $Q_e \subseteq [d]$ such that Π_e checks Π_e^{sub} for every coordinate in Q_e, i.e.,

$$\Pi_e(u, v) = \bigwedge_{i \in Q_e} \Pi_e^{sub}(u_i, v_i).$$

- Each variable is related to at most one parallel constraint.

We refer to Figure 1 as an illustration of VecCSP.

Our reduction is accomplished by combining two separate sub-reductions. First, in Subsection 3.1, we provide a reduction from 3SAT to VecCSPs. Second, in Subsection 3.2, we provide another reduction from VecCSPs to parameterized CSPs of constant gap. Finally, in Subsection 3.3, we show how to combine the two reductions above to prove Theorem 3.1 and Theorem 3.2.

3.1 Reduction I: From 3SAT to Vector-Valued CSPs

In this step, we reduce 3SAT to VecCSPs. By Hypothesis 2.3, we may assume 3SAT has some additional structure.

Theorem 3.4 (Proved in Section 4)**.** *There is a reduction algorithm such that the following holds. For any positive integer ℓ and given as input a SAT formula φ of n variables and m clauses, where each variable is contained in at most four clauses and each clause contains exactly three distinct variables, the reduction algorithm produces a VecCSP instance $G = (V, E, \Sigma, \{\Pi_e\}_{e \in E})$ where:*

(S1) VARIABLES AND CONSTRAINTS. $|V| = 48\ell^2$ *and* $|E| = 72\ell^2$.

(S2) RUNTIME. *The reduction runs in time* $\ell^{O(1)} \cdot 2^{O(n/\ell)}$.

(S3) ALPHABET. $\Sigma = \mathbb{F}_8^d$ *where* $d = \max\{\lceil m/\ell \rceil, \lceil n/\ell \rceil\}$.

(S4) COMPLETENESS AND SOUNDNESS. *G is satisfiable iff φ is satisfiable.*

3.2 Reduction II: From Vector-Valued CSPs to Gap CSPs

Now we present our gap-producing reduction from VecCSPs to instances of ε-GAP k-CSP.

Theorem 3.5. *Fix an absolute constant $\varepsilon^* = \frac{1}{9600}$. There is a reduction algorithm such that the following holds. Given as input a VecCSP instance $G = (V, E, \Sigma = \mathbb{F}^d, \{\Pi_e\}_{e \in E})$ where*

- *$k = |V|$ is the parameter of G,*
- *$|\mathbb{F}| = 2^t \le h(k)$ for some computable function h,*
- *$|E| \le m(k)$[9] for some computable function m such that $m(k) \ge 1$,*

the reduction algorithm produces a CSP instance $G^* = (V^*, E^*, \Sigma^* = \mathbb{F}^{4d}, \{\Pi_e^*\}_{e \in E^*})$ where:

- FPT REDUCTION. *The reduction from G to G^* is an FPT reduction.*
- PARAMETER BLOWUP. *The parameter of G^* is $|V^*| \le 2^{2^{k \cdot m(k) \cdot |\mathbb{F}|^{O(1)}}}$.*
- COMPLETENESS. *If G is satisfiable, then G^* is satisfiable.*
- SOUNDNESS. *If G is not satisfiable, then $\mathsf{val}(G^*) < 1 - \varepsilon^*$.*

Below, we present our reduction and proof for Theorem 3.5. Fix a VecCSP instance $G = (V, E, \Sigma, \{\Pi_e\}_{e \in E})$ satisfying the conditions in Theorem 3.5. Our reduction is achieved in three steps.

3.2.1 Step a: Instance Splitting

Recall that G has two kinds of constraints: linear and parallel constraints. In this step, we partition the constraint set E into two parts $E_L \dot\cup E_P$, where E_L and E_P consist of all linear and parallel constraints of E, and define $G_L = (V, E_L, \Sigma, \{\Pi_e\}_{e \in E_L})$ and $G_P = (V, E_P, \Sigma, \{\Pi_e\}_{e \in E_P})$ as the sub-CSP instance where the constraint set is E_L and E_P, respectively. Note that G_L and G_P are still VecCSPs with the same parameter $k = |V|$. Furthermore, we have the simple observation as follows.

Fact 3.6. *For every assignment σ over V, σ is a solution of G if and only if it is the solution of both G_L and G_P.*

3.2.2 Step b: Designing Parallel PCPPs for Sub-Instances

In this step, we construct PPCPP verifiers A_L and A_P in FPT time to test whether all constraints in G_L and G_P are satisfied, respectively. We first handle parallel constraints and obtain A_P.

Proposition 3.7 (PPCPP for Parallel Constraints. Proved in Section 5)**.** *Let h be a computable function. Let G be a VecCSP instance with k variables where (1) the alphabet is $\mathbb{F}^d$ and $|\mathbb{F}| = 2^t \le h(k)$, and (2) all constraints are parallel constraints. Then for every $\varepsilon \in (0, \frac{1}{800})$, there is a $(4, 48\varepsilon, \varepsilon, f(k) = 2^{2^{k \cdot |\mathbb{F}|^{O(1)}}}, g(k) = 2^{2^{k \cdot |\mathbb{F}|^{O(1)}}})$-PPCPP verifier for G, where $f(k)$ is the length of the auxiliary proof, and $g(k)$ is the number of random choices.*

[9]Note that we allow multiple constraints between a same pair of variables. Hence in general $|E|$ may not be bounded by a function of k.

Recall that the alphabet of G is $\mathbb{F}^d$ where $|\mathbb{F}| = 2^t$, and G_P consists of parallel constraints of G only. Thus, by plugging $\varepsilon = \frac{1}{1200}$ into the proposition above, we can obtain a $(q_P = 4, \delta_P = \frac{1}{25}, \varepsilon_P = \frac{1}{1200}, f_P(k) = 2^{2^k \cdot |\mathbb{F}|^{O(1)}}, g_P(k) = 2^{2^k \cdot |\mathbb{F}|^{O(1)}})$-PPCPP verifier A_P for G_P. Now, we turn to linear constraints and obtain A_L.

Proposition 3.8 (PPCPP for Linear Constraints. Proved in Section 6)**.** *Let h and m be two computable functions. Let G be a VecCSP instance with k variables where (1) the alphabet is $\mathbb{F}^d$ and $|\mathbb{F}| \le h(k)$, (2) all constraints are linear constraints, and (3) there are at most $m(k)$ constraints. Then for every $\varepsilon \in (0, \frac{1}{400})$, there is a $(4, 24\varepsilon, \varepsilon, f(k) = |\mathbb{F}|^{k \cdot m(k)}, g(k) = |\mathbb{F}|^{8k \cdot m(k)})$-PPCPP verifier for G.*

By plugging $\varepsilon = \frac{1}{600}$ into the proposition above, we can derive a $(q_L = 4, \delta_L = \frac{1}{25}, \varepsilon_L = \frac{1}{600}, f_L(k) = |\mathbb{F}|^{k \cdot m(k)}, g_L(k) = |\mathbb{F}|^{8k \cdot m(k)})$-PPCPP verifier A_L for G_L.

Now, we combine A_L and A_P into a single PPCPP A for the general VecCSP G from Theorem 3.5. In step c, we will convert A into a CSP instance with an inherent gap, completing the proof of Theorem 3.5.

Here A executes A_L and A_P as in a black-box way where A takes $\pi_1 \circ \pi_L \circ \pi_P$ as a proof and with equal probability, A invokes A_L with proof $\pi_1 \circ \pi_L$ or invokes A_P with proof $\pi_1 \circ \pi_P$.

Intuitively, π_1 serves as a unified encoding of a solution of G via the parallel Walsh-Hadamard code $\mathtt{PWH}$, and π_L and π_P are auxiliary proofs to convince A_L and A_P respectively. The following proposition shows that A is a PPCPP that efficiently checks all the constraints in G.

Proposition 3.9 (Combined PCPP)**.** *Given a VecCSP instance G satisfying the preconditions in Theorem 3.5, the verifier A described above is a $(q = 4, \delta = \frac{1}{25}, \varepsilon = \frac{1}{2400}, f(k) = 2^{2^k \cdot m(k) \cdot |\mathbb{F}|^{O(1)}}, g(k) = 2^{2^k \cdot m(k) \cdot |\mathbb{F}|^{O(1)}})$-PPCPP verifier for G.*

Proof. Since A invokes either A_L or A_P, the number of queries is the maximum of $q_L = 4$ and $q_P = 4$. The length of the auxiliary proof is

$$|\pi_L \circ \pi_P| \le f_L(k) + f_P(k) = |\mathbb{F}|^{k \cdot m(k)} + 2^{2^k \cdot |\mathbb{F}|^{O(1)}} \le 2^{2^k \cdot m(k) \cdot |\mathbb{F}|^{O(1)}} = f(k).$$

For alignment, we pad the randomness of A_P and A_L to ensure that they have same amount

$$R \le g_P(k) g_L(k) = |\mathbb{F}|^{8k \cdot m(k)} \cdot 2^{2^k \cdot |\mathbb{F}|^{O(1)}} \le 2^{2^k \cdot m(k) \cdot |\mathbb{F}|^{O(1)}}$$

of uniform choices. Thus the total number of uniform choices of A is $2R \le g(k)$. Finally, we analyze the completeness and soundness.

Completeness. Let σ be a solution of G. We set $\pi_1 = \mathtt{PWH}(\sigma)$. By Fact 3.6, σ is also a solution of G_L and G_P. As a result, by the definition of PPCPP (Definition 2.8), there exists π_L and π_P such that A_L and A_P always accept $\pi_1 \circ \pi_L$ and $\pi_1 \circ \pi_P$ respectively. Thus, A always accepts $\pi_1 \circ \pi_L \circ \pi_P$.

Soundness. Assume A accepts $\pi_1 \circ \pi_L \circ \pi_P$ with probability at least $1 - \varepsilon$. By construction, A_L accepts $\pi_1 \circ \pi_L$ with probability at least $1 - 2\varepsilon \ge 1 - \varepsilon_L$. Thus by the definition of PPCPP (Definition 2.8), there exists a solution σ_L of G_L such that $\Delta(\pi_1, \mathtt{PWH}(\sigma_L)) \le \delta_L \le \delta$. Similarly for A_P, there exists a solution σ_P of G_P such that $\Delta(\pi_1, \mathtt{PWH}(\sigma_P)) \le \delta$. Thus

$$\Delta(\mathtt{PWH}(\sigma_P), \mathtt{PWH}(\sigma_L)) \le 2\delta < \frac{1}{2}.$$

Recall from Subsection 2.2 that the relative distance of $\mathtt{PWH}$ is at least $\frac{1}{2}$. Hence we must have that $\sigma_L = \sigma_P$, which means $\sigma_L = \sigma_P$ is a solution of G such that $\Delta(\pi_1, \mathtt{PWH}(\sigma_L)) \le \delta$. □

3.2.3 Step c: Reducing Parallel PCPPs to Gap CSPs

Finally, we complete the proof of Theorem 3.5 by converting the verifier A into a constant-gap parameterized CSP G^*.

Proof of Theorem 3.5. Set $G^* = (V^*, E^*, \Sigma^*, \{\Pi_e^*\}_{e \in E^*})$ to be the CSP instance obtained by applying the reduction in Definition 2.9 on the verifier A. Then the claimed runtime of the reduction, as well as the alphabet size, completeness and soundness of G^*, follow immediately from combining Proposition 3.9 and Fact 2.10. Here we simply note the parameter blowup:

$$|V^*| \le |\mathbb{F}|^k + f(k) + g(k) = 2^{2^{k \cdot m(k) \cdot |\mathbb{F}|^{O(1)}}}. \quad \square$$

3.3 Putting Everything Together

In this part, we combine Theorem 3.4 and Theorem 3.5 to prove Theorem 3.1 and Theorem 3.2.

Proof of Theorem 3.1. Assuming ETH (Hypothesis 2.3), there is no $2^{o(n)}$ algorithm for deciding 3SAT formula φ where each variable is contained in at most four clauses and each clause contains exactly three distinct variables. For any such a formula φ, we show how to reduce φ to a parameterized CSP instance G^* with a constant inherent gap.

Let ℓ be a parameter to be chosen later. We first invoke Theorem 3.4 and obtain a VecCSP instance G in $\ell^{O(1)} \cdot 2^{O(n/\ell)}$ time where:

- There are $k = 48\ell^2$ variables and their alphabet is $\mathbb{F}_8^d$ with $d = O(n/\ell)$.
- G is satisfiable iff φ is satisfiable.

Note that the size of G is $|G| = \ell^{O(1)} \cdot 2^{O(n/\ell)}$.

Then we apply Theorem 3.5 on G with parameters $h(k) = 8$ and $m(k) = 2k$. After this reduction, we obtain a CSP instance $G^* = (V^*, E^*, \Sigma^*, \{\Pi_e^*\}_{e \in E^*})$ such that:

- The runtime and instance size $N := |G^*|$ of the reduction are bounded by $r_1(k) \cdot |G|^{O(1)} = r_2(\ell) \cdot 2^{O(n/\ell)}$ for some computable functions r_1, r_2.
- The parameter of G^* is $K = |V^*| \le 2^{2^{O(k)}} = 2^{2^{O(\ell^2)}}$.
- If G is satisfiable, then G^* is satisfiable.
- If G is not satisfiable, then $\mathsf{val}(G^*) < 1 - \frac{1}{9600}$.

In the paramterized complexity theory, we treat the parameter $K = |G^*|$ of G^* as a superconstant that is much smaller than the size $N = |G^*|$ of G^*. This means that the initial parameter ℓ is also a superconstant that is much smaller than n. Therefore the total reduction time $\ell^{O(1)} \cdot 2^{O(n/\ell)} + r_2(\ell) \cdot 2^{O(n/\ell)}$ is still $2^{o(n)}$.

Since the size of G^* is $N \le r_2(\ell) \cdot 2^{O(n/\ell)}$, ETH (Hypothesis 2.3) rules out any algorithm with runtime $(N/r_2(\ell))^{o(\ell)}$ to decide $(\frac{1}{9600})$-GAP K-CSP. Since $K \le 2^{2^{O(\ell^2)}}$ and by the same parameterized complexity theoretic perspective, this encompasses algorithms with runtime $f(K) \cdot N^{o(\sqrt{\log\log K})}$ for any computable function f. $\square$

The above quantitative analysis readily gives the PCP-style statement (Theorem 3.2).

Proof of Theorem 3.2. Given a 3SAT formula of size n, we apply the sparsification lemma [IPZ01] and Tovy's reduction [Tov84] to obtain a 3SAT instance φ on n Boolean variables where each variable is contained in at most four clauses and each clauses contains exactly three distinct variables. In addition, this reduction runs in $n^{O(1)}$ time and preserves the satisfiability of the original 3SAT formula.

Let $\ell \geq 1$ be a parameter. Then we apply the analysis of Theorem 3.1 on φ to obtain a CSP instance $G^* = (V^*, E^*, \Sigma^*, \{\Pi_e^*\}_{e \in E^*})$ with $|V^*| = K$ variables and alphabet size $|\Sigma^*| \leq 2^{O(n/\ell)}$ in time $r_2(\ell) \cdot 2^{O(n/\ell)}$, where $K = 2^{2^{O(\ell^2)}}$ and r_2 is some computable function. In addition, if φ is satisfiable, then $\mathsf{val}(G^*) = 1$; otherwise $\mathsf{val}(G^*) < 1 - \frac{1}{9600}$.

Now a PCP verifier takes a proof $\pi \colon V^* \to \Sigma^*$, viewed as a string of length K and alphabet Σ^*, and picks a uniform random constraint $e \in E^*$ to check. Note that the runtime of this verifier is bounded by the size $|G^*| \leq r_2(\ell) \cdot 2^{O(n/\ell)}$ of G^*. If the original 3SAT formula is satisfiable, then φ is satisfiable and thus there exists a proof that always passes the check. Otherwise, by the soundness guarantee of G^*, any proof will violate at least $\frac{1}{9600}$ fraction of the constraints in E^*, which implies the soundness gap of the PCP verifier. Setting $\ell = k$ and $f(k) = r_2(k)$ completes the proof of Theorem 3.2. □

4 From 3SAT to Vector-Valued CSP

This section is devoted to the proof of Theorem 3.4 which shows how to obtain a VecCSP from a 3SAT instance.

Before going to the details, we mark the high level picture of the reduction as follows:

1. First, we divide the clauses and variables of the 3SAT instance φ into k parts, and build a vector-valued variable (in the following, we will denote them as "vertices" in order to distinguish them from the variables in φ) for each part of clauses and each part of variables. Then we apply tests for checking the consistency between clauses and variables.

2. Next, we appropriately duplicate each vertex into several copies. Then we split the constraints and spread them out to different copies of vertices, such that
   - each vertex is related to at most one constraint;
   - the sub-constraints inside each constraint form a matching on the $2d$ coordinates of the two endpoints.

3. Finally, given the properties above, we can rearrange the d coordinates of each vertex according to its only constraint, to make the sub-constraints parallel. Furthermore, we add a cycle of constraints on the copies of each vertex, forcing them to take the same value (before rearranging the coordinates). Such constraints are then permuted equalities, which are in turn special linear constraints.

We refer to Figure 2 for an informal illustration of the above process.

Theorem (Theorem 3.4 Restated)**.** *There is a reduction algorithm such that the following holds. For any positive integer ℓ and given as input a SAT formula φ of n variables and m clauses, where each variable is contained in at most four clauses and each clause contains exactly three distinct variables, the reduction algorithm produces a VecCSP instance $G = (V, E, \Sigma, \{\Pi_e\}_{e \in E})$ in $\ell^{O(1)} \cdot 2^{O(n/\ell)}$ time, where:*

(S1) Variables and Constraints. $|V| = 48\ell^2$ *and* $|E| = 72\ell^2$.

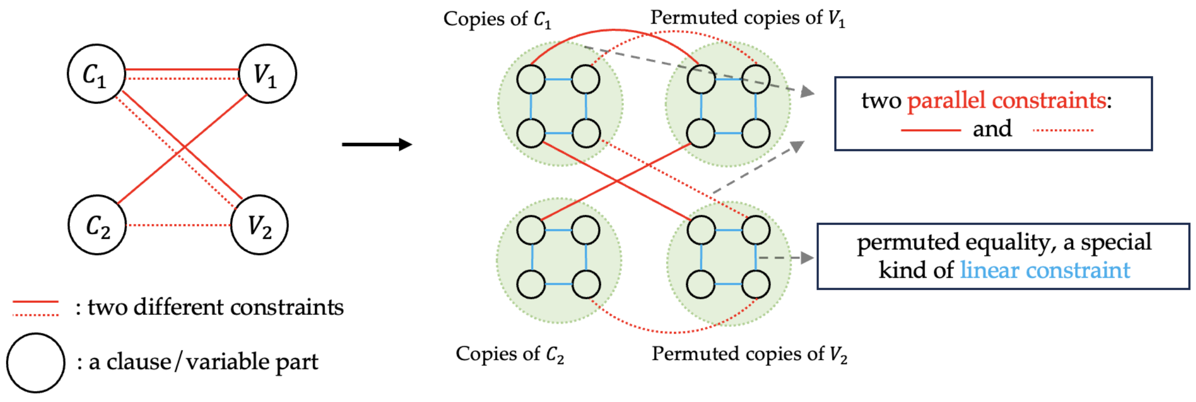


Figure 2: An example showing the vertex duplicating and constraint splitting steps.

(S2) RUNTIME. *The reduction runs in time* $\ell^{O(1)} \cdot 2^{O(n/\ell)}$.

(S3) ALPHABET. $\Sigma = \mathbb{F}_8^d$ *where* $d = \max\{\lceil m/\ell \rceil, \lceil n/\ell \rceil\}$.

(S4) COMPLETENESS AND SOUNDNESS. *G is satisfiable iff φ is satisfiable.*

Fix φ and ℓ from Theorem 3.4. For each variable, we fix an arbitrary order of its (at most four) appearances in clauses. The order is used to construct parallel constraints.

We partition m clauses of φ into $\mathcal{C}_1, \ldots, \mathcal{C}_\ell$ where each $\mathcal{C}_i$ contains at most $\lceil m/\ell \rceil$ clauses. Similarly we partition n variables of φ into $\mathcal{V}_1, \ldots, \mathcal{V}_\ell$ where each $\mathcal{V}_i$ contains at most $\lceil n/\ell \rceil$ variables. For each clause $C \in \mathcal{C}_1 \dot\cup \cdots \dot\cup \mathcal{C}_\ell$, we identify $\mathbb{F}_8 = \{0,1\}^3$ as the set of partial assignments to the clause C, where $(0,0,0) \in \mathbb{F}_8$ is the only unsatisfying assignment of C. For each variable $x \in \mathcal{V}_1 \dot\cup \cdots \dot\cup \mathcal{V}_\ell$, we also treat its assignment $\in \{0,1\}$ as an element of $\mathbb{F}_8$.

Formally, given a clause $C = y_{i_1} \vee y_{i_2} \vee y_{i_3}$, where each literal y_{i_j} equals variable x_{i_j} or its negation $\neg x_{i_j}$, every $\tau \in \mathbb{F}_8$, viewed as an element in $\{0,1\}^3$, corresponds to a unique assignment by setting $y_{i_j} = \tau(j)$ for $j \in [3]$, which in turn assigns the value of x_{i_j}.

Now we define six tests as sub-constraints to be used later. For $j \in [3]$ and $b \in \{0,1\}$, we define $\Pi_{j,b} \colon \mathbb{F}_8 \times \mathbb{F}_8 \to \{0,1\}$ by

$$\Pi_{j,b}(\tau, c) = \mathbb{1}_{c \in \{0,1\}} \cdot \mathbb{1}_{\tau \neq (0,0,0)} \cdot \mathbb{1}_{\tau(j) = c \oplus b},$$

Intuitively, these constraints checks that: (1) the variable assignment is binary, (2) the clause assignment is satisfying, and (3) the clause assignment and variable assignment are consistent.

Vertices and Alphabets. We first define the vertices and the alphabet of G. In detail, for each $p \in [\ell], q \in [\ell], j \in [3], s \in [4], b \in \{0,1\}$, we put into V a vertex $z_{p,q,j,s,b}$ with alphabet $\mathbb{F}_8^{|\mathcal{C}_p|}$, and a vertex $w_{p,q,j,s,b}$ with alphabet $\mathbb{F}_8^{|\mathcal{V}_q|}$. Intuitively, each vector entry of $z_{p,q,j,s,b}$ corresponds to the assignment of a clause $\in \mathcal{C}_p$, and each vector entry of $w_{p,q,j,s,b}$ corresponds to the assignment of a variable $\in \mathcal{V}_q$. Thus, we index entries of $z_{p,q,j,s,b}$ by clauses in $\mathcal{C}_p$ and entries of $w_{p,q,j,s,b}$ by variables in $\mathcal{V}_q$. Since $d = \max\{\lceil m/\ell \rceil, \lceil n/\ell \rceil\} = \max\{|\mathcal{C}_p|, |\mathcal{V}_q|\}$, some entries may be left unused.

At a high level, the vertices $z_{p,q,j,s,b}$ and $w_{p,q,j,s,b}$ are duplicates of assignments to $\mathcal{C}_p$ and $\mathcal{V}_q$ respectively. They will be used to check the test $\Pi_{j,b}$ between clauses in $\mathcal{C}_p$ and variables in $\mathcal{V}_q$ for the s-th appearance. Note that since we assume that every variable in φ is contained in at most four clauses, we can safely restrict the range of s to be $[4]$.

Now, we can verify that Item **(S1)** holds since in total we have $\ell \cdot \ell \cdot 3 \cdot 2 \cdot 4 \cdot 2 = 48\ell^2$ vertices in V and each vertex is related to one parallel constraint and two linear constraints. In addition, Item **(S3)** holds since all variables have alphabet $\mathbb{F}_8^d$.

Constraints. Below, we describe the constraints in the VecCSP G. At the beginning, we add parallel constraint between $z_{p,q,j,s,b}$ and $w_{p,q,j,s,b}$. For simplicity, in this paragraph, we use ζ to denote a choice of $p \in [\ell], q \in [\ell], j \in [3], s \in [4]$. Below, we enumerate every $\zeta \in [\ell] \times [\ell] \times [3] \times [4]$ and $b \in \{0,1\}$. We first define

$$T_{\zeta,0} = \left\{(C,x) \in \mathcal{C}_p \times \mathcal{V}_q \colon \text{the } s\text{-th appearance of variable } x \text{ is the } j\text{-th literal in clause } C \text{ as } x\right\}$$

and

$$T_{\zeta,1} = \left\{(C,x) \in \mathcal{C}_p \times \mathcal{V}_q \colon \text{the } s\text{-th appearance of variable } x \text{ is the } j\text{-th literal in clause } C \text{ as } \neg x\right\}.$$

Then, for every $(C,x) \in T_{\zeta,b}$, we put a sub-constraint $\Pi_{j,b}$ (j is encapsulated in $\zeta = (p,q,j,s)$) between the C-th entry of $z_{\zeta,b}$ and the x-th entry of $w_{\zeta,b}$, which checks whether the assignment of literals in C is consistent with the assignment of x. Observe that between entries of $z_{\zeta,b}$ and $w_{\zeta,b}$, we only put the sub-constraint $\Pi_{j,b}$. In addition, $T_{\zeta,b}$ forms a (not necessarily perfect) matching over $\mathcal{C}_p \times \mathcal{V}_q \subseteq [d] \times [d]$ as any two distinct $(C,x), (C',x') \in T_{\zeta,b}$ satisfy $C \neq C'$ and $x \neq x'$. Thus, we can rearrange entries of $w_{\zeta,b}$ so that the sub-constraints between $z_{\zeta,b}$ and $w_{\zeta,b}$ is parallel. We use $\kappa_{\zeta,b} \colon [d] \to [d]$ to denote the permutation applied in the rearrangement, i.e., $\kappa_{\zeta,b}(C) = x$ for all $(C,x) \in T_{\zeta,b}$. Specifically, $w_{\zeta,b}$ is rearranged in such a way that its new C-th entry takes the value of its old $\kappa_{\zeta,b}(C)$-th entry.

Finally, we remark that each variable $w_{\zeta,b}$ only need to be rearranged once according to $\kappa_{\zeta,b}$. Thus, the constraint between $z_{\zeta,b}$ and $w_{\zeta,b}$ is well-defined. From the construction above, we obtain a parallel constraint $e := \left\{z_{\zeta,b}, w_{\zeta,b}\right\} \in E$ with the sub-constraint $\Pi_e^{sub} = \Pi_{j,b}$ and $Q_e = \left\{C \colon (C,x) \in T_{\zeta,b}\right\}$.

After adding constraints for "clause-variable" consistency, we need to further establish consistency check to ensure $z_{p,\cdot,\cdot,\cdot,\cdot}$ corresponds to the same assignment over $\mathcal{C}_p$. Similarly, we also need a consistency check to ensure that $w_{\cdot,q,\cdot,\cdot,\cdot}$ corresponds to the same assignment for $\mathcal{V}_q$. Thus, we need constraints as follows.

- For each $p \in [\ell]$, we connect $\{z_{p,q,j,s,b} : q \in [\ell], j \in [3], s \in [4], b \in \{0,1\}\}$ in the constraint graph G by an arbitrary cycle (denoted by the cycle of $\mathcal{C}_p$), where every constraint in this cycle is a linear constraint. In detail, for every two vertices $\widehat{z} = z_{p,q,j,s,b}$ and $\widetilde{z} = z_{p,q',j',b',s'}$ connected in the cycle of $\mathcal{C}_p$, we impose the linear constraint that $\mathbb{1}_{\widehat{z}=\widetilde{z}}$.
- Similarly, for each $q \in [\ell]$, we also connect $\{w_{p,q,j,s,b} : p \in [\ell], j \in [3], s \in [4], b \in \{0,1\}\}$ in the constraint graph G by an arbitrary cycle (denoted by the cycle of $\mathcal{V}_q$). Also, every constraint in this cycle is a linear constraint. Note that we have rearragned $w_{p,q,j,s,b}$ by the permutation $\kappa_{p,q,j,s,b}$ to ensure the constraint between z and w is parallel. Thus, we add the permutated equality between two connected vertices $\widehat{w}$ and $\widetilde{w}$ in the cycle. In detail, we impose the linear constraint that $\mathbb{1}_{\widehat{w}=M_{\widehat{w},\widetilde{w}}\widetilde{w}}$ where $M_{\widehat{w},\widetilde{w}} \in \{0,1\}^{d\times d}$ is the permutation matrix of the permutation $\kappa_{\widehat{w}} \circ \kappa_{\widetilde{w}}^{-1}$.

To see that the instance is indeed a VecCSP, we observe that every $z_{p,q,j,s,b}$ and $w_{p,q,j,s,b}$ is related to at most one parallel constraint.

Since every variable of ϕ is contained in at most four clauses and every clause contains three distinct variables, we have $m \le 4n/3$. Hence $|\Sigma| = 8^d = 2^{O(n/\ell)}$ and the construction of G above can be done in time $\ell^{O(1)} \cdot 2^{O(n/\ell)}$, showing Item **(S2)**.

Finally, we establish the completeness and soundness to verify Item **(S4)**, which almost writes itself given the construction above.

Completeness. Assume φ is satisfiable by an assignment σ to the variables. This implies an assignment τ to the literals in clauses. Then for each entry C of $z_{p,q,j,s,b}$, we assign it as $\tau(C)$, which is among $\{0,1\}^3 \setminus \{(0,0,0)\}$ as σ is a satisfying assignment. For each entry x of $w_{p,q,j,s,b}$, we assign it as $\sigma(x)$.

It is easy to see that the linear constraints in G are all satisfied, since those are simply checking equality of the assignments. For parallel constraints, we observe that each sub-constraint $\Pi_{j,b}$ between entry C of $z_{p,q,j,s,b}$ and entry x of $w_{p,q,j,s,b}$ checks whether x is assigned to be consistent with its appearance in C, where b indicates if x appears as literal x or $\neg x$, and j represents the location of this literal in C. Since our assignments of these vertices are based on the assignment σ, it naturally passes all the tests, which finishes the proof of completeness.

Soundness. Assume G is satisfiable. By the linear constraints among z's in G, we obtain an assignment τ to the clauses of φ that satisfies each clause, and by the linear constraints among w's in G, we obtain an assignment σ to each variable. Now it remains to show that τ is consistent with σ, implying that σ really corresponds to a solution of φ.

Assume towards contradiction that the value τ assigns variable $x \in \mathcal{V}_q$ in clause $C \in \mathcal{C}_p$ is different from $\sigma(x)$. Assume C is the s-th appearance of variable x and x is at its location $j \in [3]$. In addition, let $b \in \{0,1\}$ indicate whether x appears as literal $\neg x$ in C. Then we have a sub-constraint $\Pi_{j,b}$ between the entry C of $z_{p,q,j,b,s}$ and the entry x (before rearrangement) of $w_{p,q,j,b,s}$. This is a contradiction since the test $\Pi_{j,b}$ will force $\tau(C)$ to assign variable x the same value as $\sigma(x)$, which completes the proof of soundness.

5 Parallel PCPPs for Vector-Valued CSPs with Parallel Constraints

This section is devoted to proving Proposition 3.7, which is restated as follows.

Proposition (Proposition 3.7 Restated)**.** *Let h be a computable function. Let G be a VecCSP instance with k variables where (1) the alphabet is $\mathbb{F}^d$ and $|\mathbb{F}| = 2^t \le h(k)$, and (2) all constraints are parallel constraints. Then for every $\varepsilon \in (0, \frac{1}{800})$, there is a $(4, 48\varepsilon, \varepsilon, f(k) = 2^{2^{k}\cdot|\mathbb{F}|^{O(1)}}, g(k) = 2^{2^{k}\cdot|\mathbb{F}|^{O(1)}})$-PPCPP verifier for G.*

The construction of PPCPP in this section is a generalization of an assignment tester used in the proof of the classic exponential length PCP showing result $\mathsf{NP} \subseteq \mathsf{PCP}[\mathsf{poly}(n), O(1)]$ [ALM+98]. There, we first convert a Boolean circuit to a quadratic equation system (known as the QUADEQ problem) such that they share the same satisfiability, then we use the Walsh-Hadamard code to encode an assignment to the quadratic equation system and certify its satisfiability.

Our analysis relies on the famous random subsum principle (see e.g., [AB09]), demonstrated as follows.

Lemma 5.1 (Random Subsum Principle)**.** *Given a finite field $\mathbb{F}$ and distinct $M_1, M_2 \in \mathbb{F}^{\ell\times\ell'}$ with $\ell, \ell' \ge 1$, then $\mathbf{Pr}_{x\in\mathbb{F}^{\ell}}\left[x^{\top}M_1 \ne x^{\top}M_2\right] \ge 1 - \frac{1}{|\mathbb{F}|}$.*

5.1 An Exposition of the QUADEQ Problem

In the following, we first define the QUADEQ problem, and then introduce a PCP verifier for it. While this problem and the construction are standard in the literature (see, e.g., [AB09]), we choose to give a brief exposition here for referencing purpose in our actual construction.

Definition 5.2 (QUADEQ)**.** An instance Γ of the QUADEQ problem consists of q quadratic equations on c binary variables, written concisely as $D_1, \ldots, D_q \in \mathbb{F}_2^{c \times c}$ and $b_1, \ldots, b_q \in \mathbb{F}_2$. The goal of the QUADEQ problem is to decide whether there exists a solution $u \in \mathbb{F}_2^c$ such that $u^\top D_i u = b_i$ holds for all $i \in [q]$.

The benefit of using the QUADEQ problem is that any Boolean circuit satisfiability problem can be efficiently reduced to QUADEQ by introducing dummy variables.

Fact 5.3 (Folklore, see e.g., [AB09])**.** *Any Boolean circuit*[10] *$\mathcal{C}$ with c gates (including input gates) can be converted into a* QUADEQ *instance Γ of c variables and $q = O(c)$ equations in $c^{O(1)}$ time such that $\mathcal{C}$ is satisfiable iff Γ is satisfiable.*

Moreover, there is a one-to-one correspondence between entries of the assignment of Γ and gates of $\mathcal{C}$ such that the following holds. Let $u \in \mathbb{F}_2^c$ be an assignment of Γ. Then u is a solution of Γ iff $\mathcal{C}$ represents a valid computation and outputs 1 *after assigning values of entries of u to gates of $\mathcal{C}$ by the above correspondence.*

The QUADEQ problem also admits an efficient randomized verifier. Given instance Γ, the verifier will make at most four queries on a proof π and decide whether to accept or reject. If Γ is satisfiable, then there is a proof that it always accepts; otherwise, it rejects every proof with a constant probability.

For completeness and simplicity, we recall the standard (non-parallel) Walsh-Hadamard code over $\mathbb{F}_2$, which enjoys the same local testability (Theorem 2.6) and correctability (Fact 2.7) as $\mathtt{PWH}$.

Definition 5.4 (Walsh-Hadamard Code over $\mathbb{F}_2$)**.** The Walsh-Hadamard encoding $\mathtt{WH}_2(a)$ of $a \in \mathbb{F}_2^n$ enumerates linear combinations of entries of a. Formally, $\mathtt{WH}_2(a)[b] = a^\top b$ for each $b \in \mathbb{F}_2^n$.

The proof π for Γ consists of a length-2^c binary string π_1 and a length-2^{c^2} binary string π_2. Here, π_1 is supposed to be the Walsh-Hadamard encoding $\mathtt{WH}_2(u)$ of a solution $u \in \mathbb{F}_2^c$, and π_2 is supposed to be the Walsh-Hadamard encoding $\mathtt{WH}_2(w)$ of $w = uu^\top \in \mathbb{F}_2^{c \times c}$ where we view matrix w as a length-c^2 vector. Then, the verifier checks one of the following tests with equal probability.

1. LINEARITY TEST. Perform BLR test (recall from Subsection 2.2) on π_1 or π_2 with equal probability and three queries.

   By Theorem 2.6, if the test passes with high probability, then π_1 and π_2 are close to $\mathtt{WH}_2(u)$ and $\mathtt{WH}_2(w)$ of some $u \in \mathbb{F}_2^c$ and $w \in \mathbb{F}_2^{c \times c}$ respectively. By the local correctability (Fact 2.7), we can assume that we have access to $\mathtt{WH}_2(u), \mathtt{WH}_2(w)$ via π_1, π_2.

2. TENSOR TEST. Test whether the w equals to $uu^\top$. This is achieved by generating uniformly random vectors $r, r' \in \mathbb{F}_2^c$ and making four queries, where two queries are used to obtain $\mathtt{WH}_2(u)[r]$ and $\mathtt{WH}_2(u)[r']$, and the other two queries are used to locally correct $\mathtt{WH}_2(w)[r'r^\top]$ via Fact 2.7. The test accepts if $\mathtt{WH}_2(w)[r'r^\top] = r^\top w r'$ equals $\mathtt{WH}_2(u)[r] \cdot \mathtt{WH}_2(u)[r'] = r^\top(uu^\top)r'$.

   By applying the random subsum principle (Lemma 5.1) twice, we know that if the test passes with high probability, then $w = uu^\top$. Now, we can assume that $w = uu^\top$.

[10] A Boolean circuit consists of input gates, as well as AND, OR, NOT gates with fan-in (at most) two and fan-out unbounded. Here, we focus on Boolean circuits with a single output gate.

3. CONSTRAINT TEST. Check whether u is a solution of Γ, i.e., whether $u^\top D_i u = b_i$ holds for every $i \in [q]$. This is achieved by generating a uniform $H \subseteq [q]$ and making two queries to locally correct $\mathtt{WH}_2(w)\left[\sum_{i\in H} D_i\right]$ via Fact 2.7. The test accepts if $\mathtt{WH}_2(w)\left[\sum_{i\in H} D_i\right] = \sum_{i\in H} u^\top D_i u$ equals $\sum_{i\in H} b_i$.

   By Lemma 5.1, if the test passes with high probability, then u is indeed a solution.

5.2 From Parallel Constraints to Parallel QUADEQ

We will generalize the above verifier to the parallel setting to prove Proposition 3.7. To this end, we first need to convert the parallel constraints into the QUADEQ form. Here, we will have *parallel* QUADEQ since the alphabet of VecCSP is a vector space of d coordinates.

Recall that we are given a VecCSP instance G from Proposition 3.7 with k variables and alphabet $\mathbb{F}^d$, where $|\mathbb{F}| = 2^t$ and all constraints are parallel constraints. We use $V = \{x_1, \ldots, x_k\}$ to denote the variables in G, and use $E = \{e_1, \ldots, e_m\}$ to denote the constraints in G. Recall the definition of VecCSP (Definition 3.3). We know that each variable is related with at most one parallel constraint, which implies $m \le k/2$. By rearranging, we assume without loss of generality that e_ℓ connects $x_{2\ell-1}$ and $x_{2\ell}$ for each $\ell \in [m]$. We also recall that a parallel constraint e_ℓ checks a specific sub-constraint $\Pi_\ell \colon \mathbb{F} \times \mathbb{F} \to \{0,1\}$ on all coordinates in $Q_\ell \subseteq [d]$ simultaneously between $x_{2\ell-1}$ and $x_{2\ell}$.

We will need the following additional notations:

- Let $\chi \colon \mathbb{F} \to \mathbb{F}_2^t$ be a one-to-one map that flattens elements in $\mathbb{F}$ into t bits. The map χ preserves the addition operator, i.e., i.e, $\chi(a) + \chi(b) = \chi(a+b)$.
- For each sub-constraint $\Pi_\ell \colon \mathbb{F} \times \mathbb{F} \to \{0,1\}$, we define $\overline{\Pi}_\ell \colon \mathbb{F}_2^t \times \mathbb{F}_2^t \to \{0,1\}$ by setting $\overline{\Pi}_\ell(a,b) = \Pi_e(\chi^{-1}(a), \chi^{-1}(b))$ for all $a, b \in \mathbb{F}_2^t$. In other words, we map sub-constraints with field inputs to sub-constraints with binary bits as input.

  Note that we can represent each $\overline{\Pi}_\ell$ as a Boolean circuit of size $2^{O(t)}$ in time $2^{O(t)}$.
- For each coordinate $j \in [d]$, we define $\kappa(j) = \{\ell \in [m] \colon j \in Q_\ell\}$ as the set of sub-constraints applied on the j-th coordinate.
- For each $S \subseteq [m]$, we build a Boolean circuit $\mathcal{C}_S$ to compute the conjunction of the sub-constraints $\overline{\Pi}_\ell$ for $\ell \in S$.

  Formally, $\mathcal{C}_S$ is the Boolean function mapping $\mathbb{F}_2^{k \cdot t}$ to $\{0,1\}$ such that

$$\mathcal{C}_S(y_1, \ldots, y_k) = \bigwedge_{\ell \in S} \overline{\Pi}_\ell(y_{2\ell-1}, y_{2\ell}) = \bigwedge_{\ell \in S} \Pi_\ell\left(\chi^{-1}(y_{2\ell-1}), \chi^{-1}(y_{2\ell})\right),$$

  where each $y_i \in \mathbb{F}_2^t$ is the binary representation of a coordinate of the original x_i via additive isomorphism χ.

  By adding dummy gates, we assume each $\mathcal{C}_S$ has exactly $c = k \cdot 2^{O(t)} = k \cdot |\mathbb{F}|^{O(1)}$ gates, since the circuit representation of each $\overline{\Pi}_\ell$ has size $2^{O(t)}$. In addition, by rearranging indices, we assume the first $k \cdot t$ gates are input gates corresponding to $(y_1, \ldots, y_k)$. The construction of each $\mathcal{C}_S$ can also be done in time $k \cdot |\mathbb{F}|^{O(1)}$.

We remark that the reason why we convert $\mathbb{F}$ into binary bits is that QUADEQ can only handle binary circuits and sticking with $\mathbb{F}$ will require equation systems of higher degree to preserve the satisfiability, which complicates the analysis.

Now the satisfiability of the VecCSP instance G is equivalent to the satisfiability of the Boolean circuits $\mathcal{C}_S$'s. This is formalized in Claim 5.5. For convenience, for each assignment $\sigma\colon V \to \mathbb{F}^d$ of G and each coordinate $j \in [d]$, we define $\sigma^j\colon V \to \mathbb{F}$ as the sub-assignment of σ on the j-th coordinate of all variables in V. Note that σ and σ^j can be equivalently viewed as vectors in $(\mathbb{F}^d)^k$ and $\mathbb{F}^k$ respectively.

Claim 5.5. Let $\sigma\colon V \to \mathbb{F}^d$ be an assignment of V. Then σ is a solution of G iff $\mathcal{C}_{\kappa(j)}(y_1^j, \ldots, y_k^j) = 1$ holds for every $j \in [d]$, where each $y_i^j = \chi(\sigma^j(x_i))$ is the binary representation of $\sigma^j(x_i)$.

At this point, we appeal to the QUADEQ problem to further encode the satisfiability of each $\mathcal{C}_S$ as the satisfiability of a quadratic equation system. For each $S \subseteq [m]$, we construct a QUADEQ instance Γ_S by Fact 5.3, which consists of matrices $D_{S,1}, \ldots, D_{S,q} \in \mathbb{F}_2^{c\times c}$ and bits $b_{S,1}, \ldots, b_{S,q} \in \mathbb{F}_2$ with $q = O(c)$. Note that we assume each Γ_S shares the same quantity q by padding dummy quadratic equations like $D \equiv 0^{c\times c}, b \equiv 0$. In addition, by the "moreover" part of Fact 5.3, we assume the first $k \cdot t$ bits in an assignment $u \in \mathbb{F}_2^c$ correspond to the input gates of the circuit $\mathcal{C}_S$; and the rest corresponds to values of other gates in $\mathcal{C}_S$.

Then Claim 5.6 establishes the conversion from G to a parallel QUADEQ instance.

Claim 5.6. Let σ be an assignment of V. Recall that σ^j is the sub-assignment of σ on the j-th coordinate. Let $(D_{S,1}, \ldots, D_{S,q}, b_{S,1}, \ldots, b_{S,q})$ be the QUADEQ instance Γ_S for $\mathcal{C}_S$.

Then σ is a solution of G iff $(u^j)^\top D_{\kappa(j),i} u^j = b_{\kappa(j),i}$ holds for all $j \in [d]$ and $i \in [q]$, where each $u^j \in \mathbb{F}_2^c$ is some vector with the first $k \cdot t$ bits equal to σ^j. In addition, we have $c = k \cdot |\mathbb{F}|^{O(1)}$ and $q = k \cdot |\mathbb{F}|^{O(1)}$ here.

Moreover, u^j represents the values of gates in $\mathcal{C}_{\kappa(j)}$ given as input the first $k \cdot t$ bits of u^j.

We remark that the computation so far is very efficient and runs in FPT time since $|\mathbb{F}| \le h(k)$.

5.3 Designing Parallel PCPPs for Parallel QUADEQ

In light of Claim 5.6, we now aim to generalize the PCP verifier of QUADEQ to the parallel setting to verify the computation on d coordinates simultaneously. The key observation is that, there are only $2^m = 2^{O(k)}$ many different QUADEQ instances in Claim 5.6 since $m \le k/2$. Thus, by tensoring up the proofs for different instances, we can access different positions in different proofs at the same time while still in FPT time.

Recall that for every $j \in [d]$, the set $\kappa(j) \subseteq [m]$ is the set of sub-constraints applied on the j-th coordinate. We abuse the notation to view each $S \subseteq [m]$ as an integer in $[2^m]$ by some natural bijection. For each $S \in [2^m]$, we recall that $(D_{S,1}, \ldots, D_{S,q}, b_{S,1}, \ldots, b_{S,q})$ is the QUADEQ instance Γ_S (see Claim 5.6) reduced from circuit $\mathcal{C}_S$ (see Claim 5.5). We also recall that $\sigma^j(x_i) \in \mathbb{F}$ is the j-th entry of $\sigma(x_i) \in \mathbb{F}^d$. For clarity, we use $\mathtt{PWH}_2$ to denote the parallel Walsh-Hadamard encoding with field $\mathbb{F}_2$ and reserve $\mathtt{PWH}$ for the parallel Walsh-Hadamard encoding with field $\mathbb{F}$.

The verifier A is defined as follows.

Input of A. The verifier A takes as input $\pi_1 \circ \pi_2$, where:

- π_1 has length $|\mathbb{F}|^k$ and alphabet $\mathbb{F}^d$.

  It is supposed to be $\mathtt{PWH}(\sigma)$ for an assignment σ to the variables of G.

- π_2 consists of two parts: a $2^{2^m \cdot c}$-length string τ_1 with alphabet $\mathbb{F}_2^d$ and a $2^{2^m \cdot c^2}$-length string τ_2 with alphabet $\mathbb{F}_2^d$.

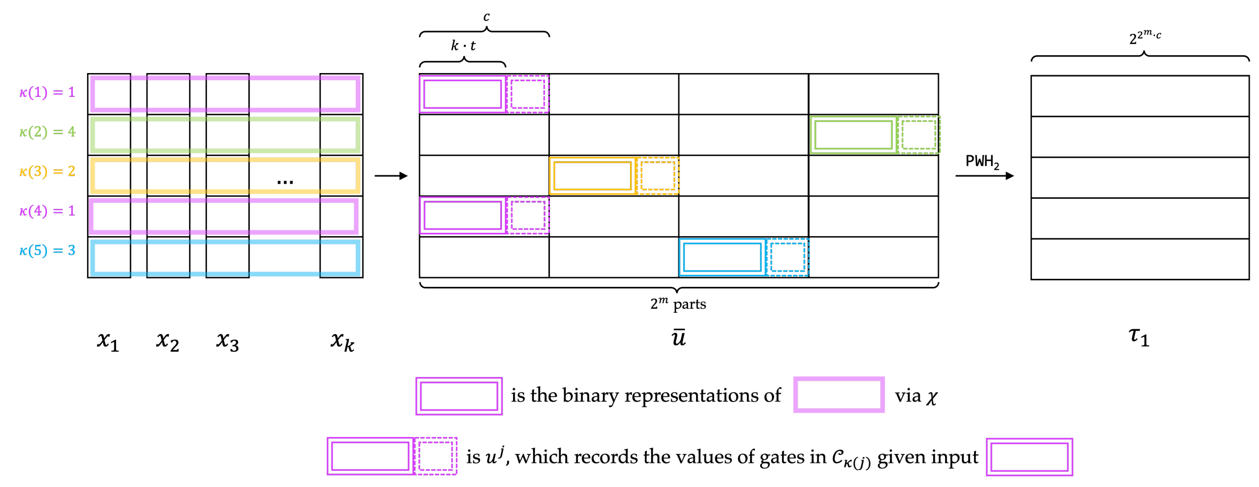


Figure 3: The construction of $\overline{u}$ and its encoding τ_1. Here $d = 5$ and $m = 2$.

τ_1 and τ_2 are supposed to be $\mathtt{PWH}_2(\overline{u})$ and $\mathtt{PWH}_2(\overline{w})$ for some $\overline{u} \in (\mathbb{F}_2^d)^{2^m \cdot c}$ and $\overline{w} \in (\mathbb{F}_2^d)^{2^m \cdot c^2}$ constructed as follows: for each $j \in [d]$, we use $u^j \in \mathbb{F}_2^c, w^j \in \mathbb{F}_2^{c \times c}$ to denote the proof[11] that the binary representations of σ^j satisfy the circuit $\mathcal{C}_{\kappa(j)}$. For $\overline{u}$ (resp., $\overline{w}$), we place u^j (resp., w^j) on the j-th coordinate and at the $\kappa(j)$-th length-c (resp., length-c^2) part, and leave all remaining parts zero. See Figure 3 for an illustration.

We remark that the alphabet of the verifier A here has different alphabets ($\mathbb{F}^d$ and $\mathbb{F}_2^d$) for π_1 and π_2. This is convenient for stating the tests and the analysis. To make it consistent with the definition of PPCPP (Definition 2.8), we can simply perform a black-box reduction that equips π_2 with alphabet $\mathbb{F}^d$ as well but rejects if any query result during the test is not from $\{0,1\}^d \cong \mathbb{F}_2^d$.

To get a sense of the detail inside τ_1 (or τ_2), we refer to Figure 4 where τ_1 supposedly faithfully stores $\mathtt{PWH}_2(\overline{u})$. This will be helpful in understanding the tests of A below.

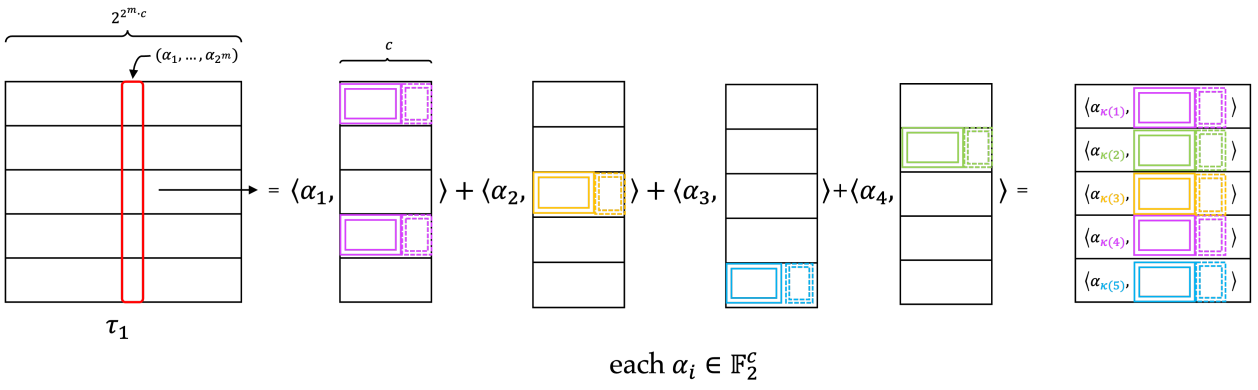

$2^{2^m \cdot c}$
$(\alpha_1, \ldots, \alpha_{2^m})$
τ_1
$= \langle \alpha_1,$
$\rangle + \langle \alpha_2,$
$\rangle + \langle \alpha_3,$
$\rangle + \langle \alpha_4,$
$\rangle =$
c
$\langle \alpha_{\kappa(1)},$
$\langle \alpha_{\kappa(2)},$
$\langle \alpha_{\kappa(3)},$
$\langle \alpha_{\kappa(4)},$
$\langle \alpha_{\kappa(5)},$
each $\alpha_i \in \mathbb{F}_2^c$

Figure 4: An illustration of $\tau_1[\alpha_1, \ldots, \alpha_{2^m}]$.

[11]Technically this proof is for QUADEQ instance $\Gamma_{\kappa(j)}$. But due to the correspondence in Fact 5.3 (or Claim 5.6), we view it as a proof for the satisfiability of $\mathcal{C}_{\kappa(j)}$. In fact, u^j is the values of gates in $\mathcal{C}_{\kappa(j)}$ and $w^j = u^j (u^j)^\top$.

Verification Procedure of A. The verifier A selects one of the following eight tests with equal probability. For ease of understanding, we group the tests according to their functions.

- LINEARITY TEST.

  (P1) Pick uniformly random $\alpha, \beta \in \mathbb{F}^k$ and check if $\pi_1[\alpha] + \pi_1[\beta] = \pi_1[\alpha + \beta]$ with three queries.

  (P2) Pick uniformly random $\alpha, \beta \in \mathbb{F}_2^{2^m \cdot c}$ and check if $\tau_1[\alpha] + \tau_1[\beta] = \tau_1[\alpha + \beta]$ with three queries.

  (P3) Pick uniformly random $\alpha, \beta \in \mathbb{F}_2^{2^m \cdot c^2}$ and check if $\tau_2[\alpha] + \tau_2[\beta] = \tau_2[\alpha + \beta]$ with three queries.

  These three tests ensure that π_1, τ_1, τ_2 are close to $\mathtt{PWH}(\sigma), \mathtt{PWH}_2(\overline{u}), \mathtt{PWH}_2(\overline{w})$ for some $\sigma \in (\mathbb{F}^d)^k, \overline{u} \in (\mathbb{F}_2^d)^{2^m \cdot c}$ and $\overline{w} \in (\mathbb{F}_2^d)^{2^m \cdot c^2}$, respectively.

- ZERO TEST.

  (P4) Take a random subset T of $[2^m]$, generate a random $\alpha_i \in \mathbb{F}_2^c$ for each $i \in T$ and set $\alpha_i = 0$ for each $i \notin T$. Then pick uniformly random $\beta_1, \dots, \beta_{2^m} \in \mathbb{F}_2^c$ and obtain $v := \tau_1[\beta_1, \dots, \beta_{2^m}] + \tau_1[\alpha_1 + \beta_1, \dots, \alpha_{2^m} + \beta_{2^m}]$ by two queries. Reject if for some $j \in [d]$, we have $\kappa(j) \notin T$ but the j-th coordinate of v is non-zero.

  (P5) Take a random subset T of $[2^m]$, generate a random $\alpha_i \in \mathbb{F}_2^{c \times c}$ for each $i \in T$ and set $\alpha_i = 0$ for each $i \notin T$. Then pick uniformly random $\beta_1, \dots, \beta_{2^m} \in \mathbb{F}_2^{c \times c}$ and obtain $v := \tau_2[\beta_1, \dots, \beta_{2^m}] + \tau_2[\alpha_1 + \beta_1, \dots, \alpha_{2^m} + \beta_{2^m}]$ by two queries. Reject if for some $j \in [d]$, we have $\kappa(j) \notin T$ but the j-th coordinate of v is non-zero.

  These two tests ensure that $\overline{u}$ and $\overline{w}$ are of the forms we want, i.e., for every $S \in [2^m]$, the S-th length-c part (respectively, length-c^2 part) has non-zero values on the j-th coordinate only if $\kappa(j) = S$.

- TENSOR TEST.

  (P6) Pick uniformly random $r_1, \dots, r_{2^m}, r'_1, \dots, r'_{2^m} \in \mathbb{F}_2^c$ and $y_1, \dots, y_{2^m} \in \mathbb{F}_2^{c \times c}$, and check whether

  $$\tau_1[r_1, \dots, r_{2^m}] \odot \tau_1[r'_1, \dots, r'_{2^m}] = \tau_2[y_1, \dots, y_{2^m}] + \tau_2[y_1 + r_1 r_1'^{\top}, \dots, y_{2^m} + r_{2^m} r_{2^m}'^{\top}] \quad (1)$$

  with four queries, where $\odot$ is the coordinate-wise multiplication.

  This performs the TENSOR TEST (Item 2) of QUADEQ on all d coordinates simultaneously.

- CONSTRAINT TEST.

  (P7) Pick a random subset H of $[q]$ and uniformly random $\beta_1, \dots, \beta_{2^m} \in \mathbb{F}_2^{c \times c}$. For each $S \in [2^m]$, define $\alpha_S = \sum_{z \in H} D_{S,z} \in \mathbb{F}_2^{c \times c}$. Obtain $y := \tau_2[\beta_1, \dots, \beta_{2^m}] + \tau_2[\alpha_1 + \beta_1, \dots, \alpha_{2^m} + \beta_{2^m}]$ by two queries and reject if for some $j \in [d]$, the j-th coordinate does not equal to $\sum_{z \in H} b_{\kappa(j),z}$.

  This performs the CONSTRAINT TEST (Item 3) of QUADEQ on all d coordinates simultaneously, where on the j-th coordinate we check the constraints with respect to $\mathcal{C}_{\kappa(j)}$.

- CONSISTENCY TEST.

(P8) Pick a random subset D of $[k]$ and a uniformly random $\beta \in \mathbb{F}^k$. Pick a random linear function $\psi\colon \mathbb{F}_2^t \to \mathbb{F}_2$ and uniformly random $\xi_1, \dots, \xi_{2^m} \in \mathbb{F}_2^c$. Define $\alpha \in \mathbb{F}^k$ to be the indicator vector of D, i.e., $\alpha_i = 1$ for $i \in D$ and $\alpha_i = 0$ for $i \notin D$. Let

$$\gamma = (\psi(1,0,\dots,0), \psi(0,1,\dots,0), \dots, \psi(0,0,\dots,1)) \in \mathbb{F}_2^t$$

and

$$\eta = (\underbrace{\gamma_1, \dots, \gamma_k}_{k \text{ of } t \text{ bits}}, \underbrace{0, \dots, 0}_{\text{remaining } c-kt \text{ bits}}) \in \mathbb{F}_2^c \quad \text{where} \quad \gamma_i = \begin{cases} \gamma & \text{if } i \in D, \\ 0^t & \text{otherwise.} \end{cases}$$

Then check if

$$\psi \circ \chi(\pi_1[\beta] + \pi_1[\alpha + \beta]) = \tau_1[\xi_1, \dots, \xi_{2^m}] + \tau_1[\eta + \xi_1, \dots, \eta + \xi_{2^m}], \tag{2}$$

where $\psi \circ \chi\colon \mathbb{F} \to \mathbb{F}_2$ is applied coordinate-wise.

This test checks if for every $j \in [d]$, the first $k \cdot t$ bits in u^j equal to the binary representations of σ^j specified by the isomorphism χ.

5.4 Analysis of Parallel PCPPs

In this subsection, we prove Proposition 3.7 with the following three lemmas (Lemma 5.7, Lemma 5.8, and Lemma 5.9), which are devoted to bound the parameters, and show completeness and soundness, respectively.

Lemma 5.7 (Parameters). *The verifier A takes as input two proofs π_1 and π_2, where π_1 has length $|\mathbb{F}|^k$ and π_2 has length at most $f(k) = 2^{2^k \cdot |\mathbb{F}|^{O(1)}}$. A then uses at most $g(k) = 2^{2^k \cdot |\mathbb{F}|^{O(1)}}$ randomness, and queries at most four positions of the proofs. Furthermore, the list of queries made by A can be generated in FPT time.*

Proof. The length of π_1 is $|\mathbb{F}|^k$ by definition, and the length of π_2 is $2^{2^m \cdot c} + 2^{2^m \cdot c^2} \le 2^{2^k \cdot |\mathbb{F}|^{O(1)}}$, where we recall that $m \le k/2$ and $c = k \cdot |\mathbb{F}|^{O(1)}$.

The amount of randomness is calculated as follows. Item **(P1)** has $|\mathbb{F}|^{2k}$ uniform possibilities, Item **(P2)** has $2^{2c \cdot 2^m}$, Item **(P3)** has $2^{2c^2 \cdot 2^m}$, Item **(P4)** has[12] $2^{2^m} \cdot 2^{c \cdot 2^m} \cdot 2^{c \cdot 2^m}$, Item **(P5)** has $2^{2^m} \cdot 2^{c^2 \cdot 2^m} \cdot 2^{c^2 \cdot 2^m}$, Item **(P6)** has $2^{2c \cdot 2^m} \cdot 2^{c^2 \cdot 2^m}$, Item **(P7)** has $2^q \cdot 2^{c^2 \cdot 2^m}$, and Item **(P8)** has[13] $2^k \cdot |\mathbb{F}|^k \cdot 2^t \cdot 2^{c \cdot 2^m}$. Recall that $2^t = |\mathbb{F}|$, $m \le k/2$, $c = k \cdot |\mathbb{F}|^{O(1)}$, and $q = k \cdot |\mathbb{F}|^{O(1)}$. Hence we may duplicate integer multiples for each of them and assume that they all have $2^{2^k \cdot |\mathbb{F}|^{O(1)}}$ uniform possibilities. Then the total randomness sums up to $8 \cdot 2^{2^k \cdot |\mathbb{F}|^{O(1)}} \le g(k)$ as desired.

It's easy to see that A makes at most four queries in any case, and the list of queries under all randomness can be generated in FPT time. □

Lemma 5.8 (Completeness). *Suppose there is a solution $\sigma : V \to \mathbb{F}^d$ of G, then there is a proof $\pi_1 \circ \tau_1 \circ \tau_2$ which A accepts with probability 1.*

Proof. By Claim 5.5, for each $j \in [d]$, $\mathcal{C}_{\kappa(j)}$ outputs 1 when taking the binary representation of σ^j, i.e., $(\chi(\sigma^j(x_1)), \dots, \chi(\sigma^j(x_k)))$, as input. Thus by Claim 5.6, for each $j \in [d]$, we have a solution u^j to the QUADEQ instance $\Gamma_{\kappa(j)}$, where the first $k \cdot t$ bits of u^j equal to $\chi(\sigma^j(x_1)), \dots, \chi(\sigma^j(x_k))$.

[12] The second $2^{c \cdot 2^m}$ comes from additionally sampling random elements for $i \notin T$ for padding to make sure they are uniform possibilities. Similar for Item **(P5)**.

[13] The second 2^t comes from the randomness in ψ, which is a random *linear* function from $\mathbb{F}_2^t$ to $\mathbb{F}_2$.

We set $\pi_1 = \texttt{PWH}(\sigma(x_1), \ldots, \sigma(x_k))$, $\tau_1 = \texttt{PWH}_2(\overline{u})$, and $\tau_2 = \texttt{PWH}_2(\overline{w})$ where $\overline{u} \in \left(\mathbb{F}_2^d\right)^{2^m \cdot c}$ and $\overline{w} \in \left(\mathbb{F}_2^d\right)^{2^m \cdot c^2}$ defined to be consistent with Figure 3:

- For every $j \in [d]$, the j-th coordinate of $\overline{u}$, viewed as a length-$(2^m \cdot c)$ binary string, has u^j on the $\kappa(j)$-th length-c part, and zero everywhere else.
- For every $j \in [d]$, the j-th coordinate of $\overline{w}$, viewed as a length-$(2^m \cdot c^2)$ binary string, has $u^j(u^j)^\top$ on the $\kappa(j)$-th length-c^2 part, and zero everywhere else.

Since π_1, τ_1 and τ_2 are all parallel Walsh-Hadamard codewords, they pass the linearity tests in Item **(P1)**, Item **(P2)**, Item **(P3)** naturally.

Given $\overline{u}$ defined as above, any query to $\tau_1[\alpha_1, \ldots, \alpha_{2^m}]$, where each $\alpha_i \in \mathbb{F}_2^c$, gives us a vector $v \in \mathbb{F}_2^d$, whose j-th coordinate stores $\langle \alpha_{\kappa(j)}, u^j \rangle$. See Figure 4 for an illustration. Thus for any subset T of $[2^m]$, if we set $\alpha_i = 0$ for all $i \notin T$ as in Item **(P4)** and Item **(P5)**, the resulting v equals zero on all coordinates $j \in [d]$ with $\kappa(j) \notin T$, which passes the tests.

To verify Item **(P6)** and Item **(P7)**, we simply observe that

- For every $j \in [d]$ and $r, r' \in \mathbb{F}_2^c$, we have $\langle w^j, rr'^\top \rangle = \langle u^j(u^j)^\top, rr'^\top \rangle = (r^\top u^j)(r'^\top u^j)$.
- For every $j \in [d]$ and $H \subseteq [q]$, we have

$$\left\langle w^j, \sum_{z \in H} D_{\kappa(j),z} \right\rangle = (u^j)^\top \left(\sum_{z \in H} D_{\kappa(j),z} \right) u^j = \sum_{z \in H} (u^j)^\top D_{\kappa(j),z} u^j = \sum_{z \in H} b_{\kappa(j),z},$$

  since u^j is a solution to the QUADEQ instance $\Gamma_{\kappa(j)}$.

For Item **(P8)**, on the left hand side of (2), we have

$$\psi \circ \chi(\pi_1[\beta] + \pi_1[\alpha + \beta]) = \psi \circ \chi(\pi_1[\alpha]) = \psi \circ \chi \left(\sum_{i \in S} \sigma(x_i) \right) = \sum_{i \in S} \psi(\chi(\sigma(x_i))),$$

where $\psi \circ \chi$ is applied coordinate-wise and the second equality is due to the linearity of ψ and the fact that χ is a additive isomorphism. On the right hand side of (2), by our choice of η and the fact that the first $k \cdot t$ bits of each u^j are just $(\chi(\sigma^j(x_1)), \ldots, \chi(\sigma^j(x_k)))$, we also get $\sum_{i \in S} \psi(\chi(\sigma(x_i)))$. □

Lemma 5.9 (Soundness). *Suppose there is a proof $\pi_1 \circ \tau_1 \circ \tau_2$ which A accepts with probability at least $1 - \varepsilon$, then there is a solution σ to G such that $\Delta(\pi_1, \texttt{PWH}(\sigma)) \le 48\varepsilon$.*

Proof. Given such a proof, each individual test passes with probability at least $1 - 8\varepsilon$. By the soundness of BLR testing (Theorem 2.6), passing the linearity test in Item **(P1)** with probability at least $1 - 8\varepsilon$ implies there exists $\sigma \in (\mathbb{F}^d)^k$ such that $\Delta(\pi_1, \texttt{PWH}(\sigma)) \le 48\varepsilon$.

Similarly for Item **(P2)** and Item **(P3)**, τ_1, τ_2 are (48ε)-close to $\texttt{PWH}_2(\overline{u})$ and $\texttt{PWH}_2(\overline{w})$ for some $\overline{u} \in (\mathbb{F}_2^d)^{2^m \cdot c}$ and $\overline{w} \in (\mathbb{F}_2^d)^{2^m \cdot c^2}$ respectively.

Next we prove that, for every $j \in [d]$, the j-th coordinate of $\overline{u}$, viewed as a length-$(2^m \cdot c)$ binary string, has non-zero values only in the $\kappa(j)$-th length-c part. Suppose it is not, and it is non-zero on the ℓ-th length-c part for some $\ell \neq \kappa(j)$. Then in Item **(P4)**, with probability $\frac{1}{4}$, we have $\ell \in T$ but $\kappa(j) \notin T$. Now for any $\{\alpha_i\}_{i \neq \ell}$, by the random subsum principle (Lemma 5.1), for at least $\frac{1}{2}$ of the choices of α_ℓ, the j-th coordinate of $\texttt{PWH}_2(\overline{u})[\alpha_1, \ldots, \alpha_{2^m}]$ is non-zero. Furthermore, since $\kappa(j) \notin T$, our test will reject whenever the j-th entry is non-zero. Thus, as long as the local correction procedure correctly decodes $\texttt{PWH}_2(\overline{u})[\alpha_1, \ldots, \alpha_{2^m}]$, which happens with probability at least $1 - 96\varepsilon$, the test fails. The overall rejection probability in this test is then at least $\frac{1}{8} - 96\varepsilon$, which is greater than the 8ε as $\varepsilon < \frac{1}{832}$. Thus a contradiction. Similar analysis also works for Item **(P5)**.

At this point, for each $j \in [d]$, define $u^j \in \mathbb{F}_2^c$ (resp., $w^j \in \mathbb{F}_2^{c \times c}$) to be the j-th coordinate of the $\kappa(j)$-th length-c (resp., length-c^2) part of $\overline{u}$ (resp., $\overline{w}$). By the analysis above, any query $\tau_1[\alpha_1, \dots, \alpha_{2^m}]$ gives us a vector $v \in \mathbb{F}_2^d$, whose j-th coordinate stores $\langle u^j, \alpha_{\kappa(j)} \rangle$; and the same holds for τ_2. See Figure 4 for an illustration. We then prove that $w^j = u^j (u^j)^\top$ holds for every $j \in [d]$, and ${u^j}^\top D_{\kappa(j),z} u^j = b_{\kappa(j),z}$ holds for every $j \in [d], z \in [q]$. This shows that they form a solution of the quadratic equation system in Claim 5.6.

- If for some $j \in [d]$, we have $w^j \neq u^j \otimes u^j$. Then by Lemma 5.1 twice, for at least $\frac{1}{4}$ fraction of choices of (r, r'), we have $r^\top w^j r' \neq (r^\top u^j)({r'}^\top u^j)$. Suppose the four queried points in Item **(P6)** are indeed as if on $\texttt{PWH}_2(\overline{u}), \texttt{PWH}_2(\overline{w})$, which happens with probability at least $1 - 192\varepsilon$. Then the left hand side of (1) is a vector v whose j-th coordinate is $(r^\top u^j)({r'}^\top u^j)$, while the right hand side of (1) is a vector v' whose j-th coordinate is $r^\top w^j r'$. Thus $v \neq v'$ and Item **(P6)** rejects. Now that the rejection probability is at least $\frac{1}{4} - 192\varepsilon$, it is greater than 8ε when $\varepsilon < \frac{1}{800}$, which provides a contradiction.

- If for some $j \in [d], z \in [q]$, we have ${u^j}^\top D_{\kappa(j),z} u^j \neq b_{\kappa(j),z}$. By Lemma 5.1, for $\frac{1}{2}$ fraction of choices of $H \subseteq [q]$, we have $\sum_{z \in H} {u^j}^\top D_{\kappa(j),z} u^j \neq \sum_{z \in H} b_{\kappa(j),z}$. Suppose the two queried points in Item **(P7)** are indeed as if on $\texttt{PWH}_2(\overline{w})$, which happens with probability at least $1 - 96\varepsilon$. Then Item **(P7)** rejects. Now that the rejection probability is at least $\frac{1}{2} - 96\varepsilon$, it is greater than 8ε as $\varepsilon < \frac{1}{208}$, which provides a contradiction.

Finally, we prove that for every $j \in [d]$, the first $k \cdot t$ bits in u^j, which we denote as $(u_1^j, \dots, u_k^j) \in \mathbb{F}_2^{k \cdot t}$, equal to $(\chi(\sigma^j(x_1)), \dots, \chi(\sigma^j(x_k)))$. This shows that the binary representation of σ certifies the satisfiability of the circuits $\mathcal{C}_S$'s as well as the QUADEQ instances Γ_S's by Claim 5.5 and Claim 5.6, which means σ is a solution of G.

Suppose for some $i \in [k], j \in [d]$, we have $\chi(\sigma^j(x_i)) \neq u_i^j$. Since ψ is a random linear function mapping to $\mathbb{F}_2$, with probability $\frac{1}{2}$, they still differ after ψ applied on. Then by Lemma 5.1, for $\frac{1}{2}$ of the choices of $S \subseteq [k]$, we have

$$\sum_{i \in S} \psi(\chi(\sigma^j(x_i))) \neq \sum_{i \in S} \psi(u_i^j).$$

Suppose the four queried points in Item **(P8)** are indeed as if on $\texttt{PWH}(\sigma), \texttt{PWH}_2(\overline{u})$, which happens with probability at least $1 - 192\varepsilon$. Then the left hand side of (2) is

$$\psi \circ \chi \left(\sum_{i \in S} \sigma(x_i) \right) = \sum_{i \in S} \psi(\chi(\sigma(x_i))),$$

where $\psi \circ \chi$ is applied coordinate-wise and the equality holds due to the linearity of ψ and the fact that χ is a additive isomorphism. The right hand side of (2) is $\sum_{i \in S} \psi(u_i^j)$ by our construction of η. Therefore, Item **(P8)** rejects with probability at least $\frac{1}{4} - 192\varepsilon$, which is greater than 8ε when $\varepsilon < \frac{1}{800}$, leading to a contradiction again.

In conclusion, we have shown that if A accepts with probability at least $1 - \varepsilon$, then π_1 must be (48ε)-close to $\texttt{PWH}(\sigma)$, where σ is a solution of G. □

Proposition 3.7 follows from a combination of Lemma 5.7, Lemma 5.8 and Lemma 5.9.

6 Parallel PCPPs for Vector-Valued CSPs with Linear Constraints

This section is devoted to proving Proposition 3.8, which we recall below.

Proposition (Proposition 3.8 Restated)**.** *Let h and m be two computable functions. Let G be a VecCSP instance with k variables where (1) the alphabet is $\mathbb{F}^d$ and $|\mathbb{F}| \le h(k)$, (2) all constraints are linear constraints, and (3) there are at most $m(k)$ constraints. Then for every $\varepsilon \in \left(0, \frac{1}{400}\right)$, there is a $\left(4, 24\varepsilon, \varepsilon, f(k) = |\mathbb{F}|^{k\cdot m(k)}, g(k) = |\mathbb{F}|^{8k\cdot m(k)}\right)$-PPCPP verifier for G.*

6.1 Construction of Parallel PCPPs

Fix a VecCSP instance $G = (V, E, \Sigma, \{\Pi_e\}_{e\in E})$ from Proposition 3.8. Recall that $k = |V|$ and we set $m = |E| \le m(k)$. By Definition 3.3, since all constraints are linear, for each constraint $e \in E$ we denote

- its two endpoints by u_e and v_e,
- the matrix for this linear constraint by $M_e \in \mathbb{F}^{d\times d}$,
- the semantics of this constraint by $\Pi_e(u_e, v_e) = \mathbb{1}_{u_e = M_e v_e}$.

For ease of presentation, we call u_e the *head* of the constraint e, and v_e the *tail* of e, respectively.

Our construction of the PPCPP verifier A is similar to the Walsh-Hadamard-based one in [BGH+06], with an additional introduction of some subtle auxilary variables.

Auxilary Variables. Label variables V by $\{1, 2, \dots, k\}$ and constraints by $\{1, 2, \dots, m\}$. For every $p \in V$ and $e \in E$, we define an auxiliary variable $z_{p,e}$ with alphabet $\mathbb{F}^d$. Given an assignment $\sigma(p)$ to the variable p, the assignment to $z_{p,e}$ should equal $z_{p,e} = M_e\sigma(p)$ [14].

Note that we introduce an auxilary variable for every pair $(p, e) \in V \times E$, *even if e is not adjacent to p.* This way, we can check both the inner constraints $z_{p,e} = M_e\sigma(p)$ and the conjunction of all linear constraints $\sigma(u_e) = z_{v_e,e}$ with constant queries, soundness, and proximity.

Below, we describe the details of the PPCPP verifier A for G.

Input of A. The verifier A takes as input $\pi_1 \circ \pi_2$, where:

- π_1 is indexed by vectors in $\mathbb{F}^k$ and has alphabet $\mathbb{F}^d$. It is supposed to be $\texttt{PWH}(\sigma)$, the parallel Walsh-Hadamard encoding of an assignment σ to V.
- π_2 is indexed by vectors in $\mathbb{F}^{km}$ and has alphabet $\mathbb{F}^d$. It is supposed to be the parallel Walsh-Hadamard encoding of the collection $\{z_{p,e}\}_{p\in V, e\in E}$, treated as a vector of $(\mathbb{F}^d)^{km}$.

Verification Procedure of A. Here is how A verifies whether π_1 is close to $\texttt{PWH}(\sigma)$ for some solution σ of G. With equal probability, A selects one of the following four tests:

- Linearity Test.

  (L1) Pick uniformly random $a_1, a_2 \in \mathbb{F}^k$ and check $\pi_1[a_1] + \pi_1[a_2] = \pi_1[a_1 + a_2]$ by three queries.

[14] Here we abuse the notation and use $z_{p,e}$ also to denote the value assigned to it.

(L2) Pick uniformly random $b_1, b_2 \in \mathbb{F}^{km}$ and check $\pi_2[b_1] + \pi_2[b_2] = \pi_2[b_1 + b_2]$ by three queries.

Intuitively, Item **(L1)** and Item **(L2)** ensure that both π_1 and π_2 are close to a codeword of PWH.

- Matrix Test.

(L3) Pick uniformly random $\lambda \in \mathbb{F}^k$ and $\mu \in \mathbb{F}^m$ and set $\gamma = (\lambda_1\mu_1, \lambda_1\mu_2, \ldots, \lambda_k\mu_m) \in \mathbb{F}^{km}$. Assume μ is indexed by constraints $e \in E$ and define matrix $M_0 = \sum_{e\in E} \mu_e M_e$. Note that we can compute M_0 efficiently without any query.

Then pick uniformly random $a \in \mathbb{F}^k, b \in \mathbb{F}^{km}$, query $\pi_1[a], \pi_1[a+\lambda], \pi_2[b], \pi_2[b+\gamma]$, and check if

$$\pi_2[b+\gamma] - \pi_2[b] = M_0(\pi_1[a+\lambda] - \pi_1[a]). \tag{3}$$

Intuitively, Item **(L3)** ensures that π_2 encodes the collection $\{z_{p,e}\}_{p\in V, e\in E}$ where all inner constraints $z_{p,e} = M_e\sigma(p)$ are satisfied.

- Constraint Test.

(L4) Pick uniformly random $\mu \in \mathbb{F}^m$ and assume μ is indexed by constraints $e \in E$. Define a vector $\lambda \in \mathbb{F}^k$ by setting $\lambda_p = \sum_{e\in E:\, u_e = p} \mu_e$ for $p \in V$, where we assume that λ is indexed by vertices $p \in V$. In other words, λ_p is the sum of μ_e's for constraint $e \in E$ whose head is p.

In addition, define a vector $\gamma \in \mathbb{F}^{km}$, indexed by a vertex-constraint pair $(p,e) \in V \times E$, by

$$\gamma_{p,e} = \begin{cases} \mu_e & v_e = p, \\ 0 & \text{otherwise.} \end{cases}$$

In other words, $\gamma_{p,e}$ stores μ_e if the tail of the constraint e is vertex p.

Note that the two vectors μ and γ can be computed efficiently without any query.

Then pick uniformly random $a \in \mathbb{F}^k, b \in \mathbb{F}^{km}$, query $\pi_1[a], \pi_1[a+\lambda], \pi_2[b], \pi_2[b+\gamma]$, and check if

$$\pi_2[b+\gamma] - \pi_2[b] = \pi_1[a+\lambda] - \pi_1[a].$$

Intuitively, Item **(L4)** ensures $\sigma(u_e) = z_{v_e,e}$ for every constraint $e \in E$.

6.2 Analysis of Parallel PCPPs

In this subsection, we prove Proposition 3.8 with the following three lemmas (Lemma 6.1, Lemma 6.2 and Lemma 6.3), which are devoted to bounding the parameters, and establishing the completeness and soundness of the verifier, respectively.

Lemma 6.1 (Parameters)**.** *The verifier A takes as input two proofs π_1 and π_2, where π_1 has length $|\mathbb{F}|^k$ and π_2 has length $f(k) = |\mathbb{F}|^{km}$. A then uses at most $g(k) = |\mathbb{F}|^{8km}$ randomness, and queries at most four positions of the proofs. Furthermore, the list of queries made by A can be generated in FPT time.*

Proof. The length of π_1 and π_2 are $|\mathbb{F}|^k$ and $|\mathbb{F}|^{km}$ respectively by definition.

In terms of randomness, Item **(L1)** has $|\mathbb{F}|^{2k}$ uniform possibilities, Item **(L2)** has $|\mathbb{F}|^{2km}$, Item **(L3)** has $|\mathbb{F}|^{2k+m+km}$, and Item **(L4)** has $|\mathbb{F}|^{k+m+km}$. Note that we may duplicate integer multiples for each of them and assume that they all have $|\mathbb{F}|^{4km}$ uniform possibilities. Then the total randomness sums up to $4 \cdot |\mathbb{F}|^{4km} \leq g(k)$ as desired.

It's easy to see that A makes at most four queries in any case, and the list of queries under all randomness can be generated in FPT time. $\square$

Lemma 6.2 (Completeness)**.** *Suppose there is a solution $\sigma : V \to \mathbb{F}^d$ of G, then there is a proof $\pi_1 \circ \pi_2$ which A accepts with probability 1.*

Proof. Fix such a solution σ. We assign the value $M_e\sigma(p)$ to the auxiliary variable $z_{p,e}$ and treat $\{z_{p,e}\}_{p\in V,e\in E}$ as a km-dimensional vector. We set π_1 as $\mathtt{PWH}(\sigma)$ and π_2 as $\mathtt{PWH}(\{z_{p,e}\}_{p\in V,e\in E})$. Since both π_1 and π_2 are codewords of PWH, they naturally pass the tests in Item **(L1)** and Item **(L2)**.

For Item **(L3)**, note that for every $\lambda \in \mathbb{F}^k, \mu \in \mathbb{F}^m, a \in \mathbb{F}^k, b \in \mathbb{F}^{km}$, we have

$$\begin{aligned}
\pi_2[b+\gamma] - \pi_2[b] = \pi_2[\gamma] &= \sum_{p\in V,e\in E} \lambda_p\mu_e z_{p,e} && \text{(by the definition of } \pi_2 \text{ and } \gamma) \\
&= \sum_{p,e} \lambda_p\mu_e \cdot M_e\sigma(p) = \left(\sum_{e\in E}\mu_e M_e\right)\left(\sum_{p\in V}\lambda_p\sigma(p)\right) && \text{(by the definition of } z_{p,e}) \\
&= M_0\pi_1[\lambda] = M_0(\pi_1[a+\lambda] - \pi_1[a]), && \text{(by the definition of } M_0 \text{ and } \pi_1)
\end{aligned}$$

which means that the test in Item **(L3)** passes.

Finally, we turn to Item **(L4)**. For every $\mu \in \mathbb{F}^m, a \in \mathbb{F}^k, b \in \mathbb{F}^{km}$, we have

$$\begin{aligned}
\pi_2[b+\gamma] - \pi_2[b] = \pi_2[\gamma] &= \sum_{e\in E}\mu_e z_{v_e,e} = \sum_{e\in E}\mu_e \cdot M_e\sigma(v_e) && \text{(by the definition of } \pi_2, \gamma, z_{v_e,e}) \\
&= \sum_{e\in E}\mu_e\sigma(u_e) && \text{(by } M_e\sigma(v_e) = \sigma(u_e) \text{ as } \sigma \text{ is a solution)} \\
&= \sum_{p\in V}\sigma(p)\cdot\left(\sum_{e\in E:\, u_e=p}\mu_e\right) && \text{(by rearranging the summation)} \\
&= \sum_{p\in V}\lambda_p\sigma(p) && \text{(by the definition of } \lambda) \\
&= \pi_1[\lambda] = \pi_1[a+\lambda] - \pi_1[a], && \text{(by the definition of } \pi_1)
\end{aligned}$$

which passes the test in Item **(L4)**. In all, A accepts $\pi_1 \circ \pi_2 = \mathtt{PWH}(\sigma) \circ \pi_2$ with probability 1. $\square$

Lemma 6.3 (Soundness)**.** *Suppose there is a proof $\pi_1 \circ \pi_2$ which A accepts with probability at least $1-\varepsilon$, then there is a solution σ to G such that $\Delta(\pi_1, \mathtt{PWH}(\sigma)) \leq 24\varepsilon$.*

Proof. First, note that π_1 fails at most 4ε fraction of tests in Item **(L1)**. By the soundness of BLR testing (Theorem 2.6), there exists some $\sigma \in (\mathbb{F}^d)^k$ such that $\Delta(\pi_1, \mathtt{PWH}(\sigma)) \leq 24\varepsilon$. Similarly by Item **(L2)**, we can find some $\sigma_2 \in (\mathbb{F}^d)^{km}$ such that $\Delta(\pi_2, \mathtt{PWH}(\sigma_2)) \leq 24\varepsilon$. Below, we treat σ as a mapping from V to $\mathbb{F}^d$, and σ_2 as a mapping from $V \times E \to \mathbb{F}^d$.

Now we prove that for every $p \in V, e \in E$, we have $\sigma_2(p,e) = M_e\sigma(p)$. Recall the notation from Item **(L3)**. For any fixed λ, μ (and thus γ, M_0 are also fixed), by Fact 2.7 and a union bound, with probability at least $1 - 96\varepsilon$ over random a, b, both of the following two equations hold

$$\pi_2[b+\gamma] - \pi_2[b] = \sum_{p\in V,e\in E}\lambda_p\mu_e\sigma_2(p,e), \tag{4}$$

$$\pi_1[a+\lambda] - \pi_1[a] = \sum_{p\in V}\lambda_p\sigma(p). \tag{5}$$

Recall the definition of M_0. By taking a difference between the LHS and RHS of (3) and plugging in (4) and (5), we can deduce that, with probability at least $1-96\varepsilon$,

$$(\pi_2[b+\gamma]-\pi_2[b])-M_0(\pi_1[a+\lambda]-\pi_1[a])=\sum_{p\in V,e\in E}\lambda_p\mu_e\left(\sigma_2(p,e)-M_e\sigma(p)\right)=\lambda^\top(M_1-M_2)\mu,$$

where we define matrix M_1 by setting the (p,e)-th entry as $\sigma_2(p,e)$ and matrix M_2 by setting the (p,e)-th entry as $M_e\sigma(p)$.

If $\sigma_2(p,e)\neq M_e\sigma(p)$ for some $p\in V, e\in E$, then $M_1\neq M_2$. By Lemma 5.1 with $\ell=|V|$ and $\ell'=|E|$, we have $\mathbf{Pr}_\lambda\left[\lambda^\top M_1\neq\lambda^\top M_2\right]\geq 1-\frac{1}{|\mathbb{F}|}$. Then by another round of Lemma 5.1 with $\ell=|E|$ and $\ell'=1$, we have $\mathbf{Pr}_{\lambda,\mu}\left[\lambda^\top M_1\mu\neq\lambda^\top M_2\mu\right]\geq\left(1-\frac{1}{|\mathbb{F}|}\right)^2\geq\frac{1}{4}$ as $|\mathbb{F}|\geq 2$. By a union bound, for at least $\frac{1}{4}-96\varepsilon$ fraction of the tests in Item **(L3)**, both (4) and(5) hold, yet the difference above is non-zero. This means at least $\frac{1}{4}-96\varepsilon$ fraction of tests in Item **(L3)** are violated. On the other hand, since A accepts $\pi_1\circ\pi_2$ with probability at least $1-\varepsilon$, at most 4ε fraction of the tests in Item **(L3)** can be violated. This gives a contradiction as $\varepsilon<\frac{1}{400}$.

Finally, we prove that σ is a solution of G. We focus on tests in Item **(L4)** and recall the notation there. By a union bound, with probability at least $1-96\varepsilon$ over random a,b for any fixed μ (and thus λ,γ are also fixed), both equations below hold:

$$\pi_2[b+\gamma]-\pi_2[b]=\sum_{e\in E}\mu_e\sigma_2(v_e,e)=\sum_{e\in E}\mu_e M_e\sigma(v_e),\tag{6}$$

$$\pi_1[a+\lambda]-\pi_1[a]=\sum_{e\in E}\mu_e\sigma(u_e),\tag{7}$$

where the second equality in (6) follows from our analysis for Item **(L3)** above. If $\sigma(u_e)\neq M_e\sigma(v_e)$ for some $e\in E$, then by Lemma 5.1, we have

$$\mathop{\mathbf{Pr}}_{\mu}\left[\pi_2[b+\gamma]-\pi_2[b]\neq\pi_1[a+\lambda]-\pi_1[a]\right]\geq 1-\frac{1}{|\mathbb{F}|}\geq\frac{1}{2}.$$

By a union bound, for at least $\frac{1}{2}-96\varepsilon$ fraction of tests in Item **(L4)**, both (6) and (7) hold, yet their difference is non-zero. This means $\pi_1\circ\pi_2$ violates at least $\frac{1}{2}-96\varepsilon$ fraction of tests in Item **(L4)**, contradicting the fact that $\pi_1\circ\pi_2$ can only violate at most 4ε fraction of tests, recalling again our assumption that $\varepsilon<\frac{1}{400}$. Thus σ is indeed a solution of G, completing the proof. □

Proposition 3.8 immediately follows from the combination of Lemma 6.1, Lemma 6.2 and Lemma 6.3.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[ABB$^+$23] Fateme Abbasi, Sandip Banerjee, Jaroslaw Byrka, Parinya Chalermsook, Ameet Gadekar, Kamyar Khodamoradi, Dániel Marx, Roohani Sharma, and Joachim Spoerhase. Parameterized approximation schemes for clustering with general norm objectives. *FOCS*, 2023.

[ABSS97] Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. Syst. Sci.*, 54(2):317–331, 1997.

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[ANSW20] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and Euclidean k-median by primal-dual algorithms. *SIAM J. Comput.*, 49(4), 2020.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[BBE$^+$21] Arnab Bhattacharyya, Édouard Bonnet, László Egri, Suprovat Ghoshal, Karthik C. S., Bingkai Lin, Pasin Manurangsi, and Dániel Marx. Parameterized intractability of even set and shortest vector problem. *J. ACM*, 68(3):16:1–16:40, 2021.

[BCGR23] Huck Bennett, Mahdi Cheraghchi, Venkatesan Guruswami, and João Ribeiro. Parameterized inapproximability of the minimum distance problem over all fields and the shortest vector problem in all ℓ_p norms. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 553–566. ACM, 2023.

[BGH$^+$06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM J. Comput.*, 36(4):889–974, 2006.

[BGS98] Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs, and nonapproximability-towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998.

[BKN21] Boris Bukh, Karthik C. S., and Bhargav Narayanan. Applications of random algebraic constructions to hardness of approximation. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 237–244. IEEE, 2021.

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993.

[BPR$^+$17] Jaroslaw Byrka, Thomas W. Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median and positive correlation in budgeted optimization. *ACM Trans. Algorithms*, 13(2):23:1–23:31, 2017.

[CCK$^+$17] Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From gap-ETH to FPT-inapproximability: Clique, dominating set, and more. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 743–754. IEEE Computer Society, 2017.

[CFLL23] Yijia Chen, Yi Feng, Bundit Laekhanukit, and Yanlin Liu. Simple combinatorial construction of the $k^{o(1)}$-lower bound for approximating the parameterized k-clique. *CoRR*, abs/2304.07516, 2023.

[CG07] Yijia Chen and Martin Grohe. An isomorphism between subexponential and parameterized complexity theory. *SIAM Journal on Computing*, 37(4):1228–1258, 2007.

[CGK$^+$19] Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li. Tight FPT approximations for k-median and k-means. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 42:1–42:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[CGTS02] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *J. Comput. Syst. Sci.*, 65(1):129–149, 2002.

[CIP09] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation*, pages 75–85, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[CL19] Yijia Chen and Bingkai Lin. The constant inapproximability of the parameterized dominating set problem. *SIAM J. Comput.*, 48(2):513–533, 2019.

[DF95a] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM J. Comput.*, 24(4):873–921, 1995.

[DF95b] Rodney G Downey and Michael R Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 141(1-2):109–131, 1995.

[DGKS08] Irit Dinur, Elena Grigorescu, Swastik Kopparty, and Madhu Sudan. Decodability of group homomorphisms beyond the Johnson bound. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 275–284, 2008.

[DHK05] Erik D. Demaine, Mohammad Taghi Hajiaghayi, and Ken-ichi Kawarabayashi. Algorithmic graph minor theory: Decomposition, approximation, and coloring. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 637–646. IEEE Computer Society, 2005.

[Din07] Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007.

[Din16] Irit Dinur. Mildly exponential reduction from gap 3sat to polynomial-gap label-cover. *Electron. Colloquium Comput. Complex.*, 23:128, 2016.

[DM18] Irit Dinur and Pasin Manurangsi. Eth-hardness of approximating 2-csps and directed steiner network. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 36:1–36:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[DR06] Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. *SIAM Journal on Computing*, 36(4):975–1024, 2006.

[Fei98] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[Fel03] 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*, pages 1–12. Springer, 2003.

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[FGL$^+$96] Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM (JACM)*, 43(2):268–292, 1996.

[FKLM20] Andreas Emil Feldmann, Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6):146, 2020.

[GGR11] Parikshit Gopalan, Venkatesan Guruswami, and Prasad Raghavendra. List decoding tensor products and interleaved codes. *SIAM J. Comput.*, 40(5):1432–1462, 2011.

[GLL18a] Anupam Gupta, Euiwoong Lee, and Jason Li. Faster exact and approximate algorithms for k-cut. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 113–123. IEEE, 2018.

[GLL18b] Anupam Gupta, Euiwoong Lee, and Jason Li. An fpt algorithm beating 2-approximation for k-cut. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2821–2837. SIAM, 2018.

[Gol16] Oded Goldreich. Lecture notes on linearity (group homomorphism) testing, 2016.

[GOS20] Venkatesan Guruswami, Jakub Opršal, and Sai Sandeep. Revisiting alphabet reduction in Dinur's PCP. In Jarosław Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*, volume 176 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[GRS23] Venkatesan Guruswami, Xuandi Ren, and Sai Sandeep. Baby PIH: Parameterized inapproximability of Min CSP. *arXiv preprint arXiv:2310.16344*, 2023.

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62:367–375, 2001.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[KK22] CS Karthik and Subhash Khot. Almost polynomial factor inapproximability for parameterized k-clique. In *37th Computational Complexity Conference (CCC 2022)*, volume 234, 2022.

[KL20] Ken-ichi Kawarabayashi and Bingkai Lin. A nearly 5/3-approximation FPT algorithm for min-k-cut. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 990–999. SIAM, 2020.

[KLM19] Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. *J. ACM*, 66(5):33:1–33:38, 2019.

[KMN$^+$04] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. *Comput. Geom.*, 28(2-3):89–112, 2004.

[KN21] Karthik C. S. and Inbal Livni Navon. On hardness of approximation of parameterized set cover and label cover: Threshold graphs from error correcting codes. In Hung Viet Le and Valerie King, editors, *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 210–223. SIAM, 2021.

[Lee19] Euiwoong Lee. Partitioning a graph into small pieces with applications to path transversal. *Math. Program.*, 177(1-2):1–19, 2019.

[Lin18] Bingkai Lin. The parameterized complexity of the k-biclique problem. *Journal of the ACM (JACM)*, 65(5):1–23, 2018.

[Lin19] Bingkai Lin. A simple gap-producing reduction for the parameterized set cover problem. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 81:1–81:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[Lin21] Bingkai Lin. Constant approximating k-clique is w[1]-hard. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1749–1756. ACM, 2021.

[LRSW22] Bingkai Lin, Xuandi Ren, Yican Sun, and Xiuhan Wang. On lower bounds of approximating parameterized k-clique. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 90:1–90:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[LRSW23a] Bingkai Lin, Xuandi Ren, Yican Sun, and Xiuhan Wang. Constant approximating parameterized k-SETCOVER is W[2]-hard. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 3305–3316. SIAM, 2023.

[LRSW23b] Bingkai Lin, Xuandi Ren, Yican Sun, and Xiuhan Wang. Improved hardness of approximating k-clique under ETH. *FOCS*, 2023.

[LRSZ20] Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Parameterized complexity and approximability of directed odd cycle transversal. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2181–2200. SIAM, 2020.

[LS16] Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. *SIAM J. Comput.*, 45(2):530–547, 2016.

[LSS20] Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan. A parameterized approximation scheme for min k-cut. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 798–809. IEEE, 2020.

[Man19] Pasin Manurangsi. A note on max k-vertex cover: Faster FPT-AS, smaller approximate kernel and improved approximation. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San Diego, CA, USA*, volume 69 of *OASIcs*, pages 15:1–15:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[Man20] Pasin Manurangsi. Tight running time lower bounds for strong inapproximability of maximum k-coverage, unique set cover and related problems (via t-wise agreement testing theorem). In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 62–81. SIAM, 2020.

[Mar08] Dániel Marx. Parameterized complexity and approximation algorithms. *Comput. J.*, 51(1):60–78, 2008.

[Ohs22] Naoto Ohsaka. On the parameterized intractability of determinant maximization. In *33rd International Symposium on Algorithms and Computation (ISAAC 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[SF17] Piotr Skowron and Piotr Faliszewski. Chamberlin-courant rule with approval ballots: Approximating the maxcover problem with bounded frequencies in FPT time. *J. Artif. Intell. Res.*, 60:687–716, 2017.

[Tov84] C. Tovey. A simplified NP-complete satisfiability problem. *Discret. Appl. Math.*, 8:85–89, 1984.

[Wie18] Andreas Wiese. Fixed-parameter approximation schemes for weighted flowtime. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, volume 116 of *LIPIcs*, pages 28:1–28:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[Wło20] Michał Włodarczyk. Parameterized inapproximability for steiner orientation by gap amplification. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.