Parameterized Inapproximability Hypothesis under Exponential Time Hypothesis*

Venkatesan Guruswami
Simons Institute for the Theory of Computing, and Departments of EECS and Mathematics
UC Berkeley
Berkeley, USA
venkatg@berkeley.edu

Bingkai Lin
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
lin@nju.edu.cn

Xuandi Ren
Department of EECS
UC Berkeley
Berkeley, USA
xuandi_ren@berkeley.edu

Yican Sun
School of Computer Science
Peking University
Beijing, China
sycpku@pku.edu.cn

Kewen Wu
Department of EECS
UC Berkeley
Berkeley, USA
shlw_kevin@hotmail.com

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.

CCS CONCEPTS

• **Theory of computation** → **Fixed parameter tractability**; **Problems, reductions and completeness**.

KEYWORDS

Fixed-parameter algorithms and complexity, Hardness of approximations, PCP theorems

ACM Reference Format:
Venkatesan Guruswami, Bingkai Lin, Xuandi Ren, Yican Sun, and Kewen Wu. 2024. Parameterized Inapproximability Hypothesis under Exponential Time Hypothesis. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC '24), June 24–28, 2024, Vancouver, BC, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3618260.3649771

*Research supported in part by NSF grants CCF-2228287, CCF-2211972, a Simons Investigator award, a Sloan Research Fellowship and NSF CAREER Award CCF-2145474.

STOC '24, June 24–28, 2024, Vancouver, BC, Canada

ACM ISBN 979-8-4007-0383-6/24/06
https://doi.org/10.1145/3618260.3649771

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 P ≠ NP assumption to obtain more fine-grained lower bounds for NP-hard problems, for example, Exponential Time Hypothesis (ETH) [42], Strong Exponential Time Hypothesis (SETH) [15, 42], Gap Exponential Time Hypothesis (Gap-ETH) [24].

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 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* [28, 29, 34]. In this theory, there are also alternative hypotheses as a strengthening of P ≠ NP. For example, W[1] ≠ 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, FPT approximation algorithms have been designed for various NP-complete problems. Examples include Vertex-Coloring [22, 61], Min-k-Cut [37, 38, 48, 58], k-Path-Deletion [49], k-Clustering [1], k-Means / k-Medians [2, 14,

17, 21, 44, 50], Max k-Hypergraph Vertex Cover [59, 63], Flow Time Scheduling [65].

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 [16, 20, 46, 47, 52, 55], k-SetIntersection [13, 51], k-Steiner Orientation [66], Max-k-Coverage [60], k-SVP, k-MDP and related problems [10, 11, 60]. An exciting recent line of work [16, 18, 45, 53, 54, 56] 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 [32] 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 [6, 7, 23], 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 [32], 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 [57] formalized and entitled the above question as *Parameterized Inapproximability Hypothesis (PIH)*. Here we present the following reformulation[1] of PIH due to [32]:

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 [41], Shortest Vector [10, 11], Direct Odd Cycle Transversal [57], Determinant Maximization and Grid Tiling [62], Baby PIH [41], k-MaxCover [46], 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., [32]). Indeed, it is often recognized that gap-preserving reductions are much easier than gap-producing reductions [31]. 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) [42], a fundamental gap-free hypothesis in the modern complexity theory and a weakening of the Gap-ETH assumption[3].

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.*

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.
- 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 [4, 60].

[1] The original statement of PIH in [57] replaces the runtime bound by $\mathsf{W}[1]$-hardness, and the reformulation by [32] 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.

[3] We thank Benny Applebaum for pointing his work [3] to us, which shows that smooth-ETH, a strengthening of ETH, implies Gap-ETH (and thus PIH). In contrast, we prove PIH directly from ETH, completely bypassing Gap-ETH.

For constant $\rho > 1$, the hardness of $(k, \rho \cdot k)$-ExactCover was only proved under assumptions with inherent gaps [41, 60], imitating the reduction in the non-parameterized world [30]. 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., [41][4]) 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 $\mathrm{DOCT}(D)$, 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 $\mathrm{DOCT}(D) \leq k$, from those with $\mathrm{DOCT}(D) \geq \rho \cdot k$. The parameter of this problem is the optimum k. This problem is a generalization of several well studied problems including Directed Feedback Vertex Set and Odd Cycle Transversal. For a brief history of this problem, we refer to the previous work [57]. In their work, the authors prove the following hardness of $(k, \rho \cdot k)$-DOCT.

Theorem 1.5 ([57]). *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 [6, 7, 40]. 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 [6], the Reed-Muller code [7], or the long code [40]), 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 codeword length must be 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 [56]. 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 some additional features. 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,

[4] [41] proves the hardness of $(k, \rho \cdot k)$-ExactCover under a weaker version of PIH, namely, Average Baby PIH with rectangular constraints.

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 [5, 6] 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 $1_{u_e = M_e v_e}$.

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 [5], 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 $M_e v$. See Section 6 for details.

1.3 Related Works and Discussions

Related Works. As mentioned above, prior to our work, PIH was only known to hold under Gap-ETH [16, 26]. 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 [54, 56] proposed to prove PIH via a strong lower bound for constant-gap k-Clique. This is reminiscent of [8], where the NP-hardness of constant-gap Clique leads to a free-bit PCP. However, the construction in [8] does not apply in the parameterized setting since the proof length will be too long. In addition, the framework of [56] 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 [41] 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. First, 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 [16, 32].

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.

Seond, 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. In light of this, all we need is to establish the $\mathsf{W}[1]$-hardness for the *exact* version of vector-valued CSPs. We remark that our vector-valued CSP instances are closely related to an $\mathsf{M}[1]$-complete problem Mini-3SAT [33] 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

W[1] ≠ FPT. We refer interested readers to [19] 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.

Due to space limitations, we relegate the detailed proof into the full version [39].

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 $1_{\mathcal{E}}$ as the indicator function. 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 and 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 $\langle A, B\rangle = \sum_{i,j\in[d]} A_{i,j}B_{i,j}$ 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 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. We use $\mathrm{val}(G, \sigma) = \frac{1}{|E|}\sum_{e\in E}\Pi_e(\sigma(u_e), \sigma(v_e))$ to denote the *satisfiability value* for an assignment σ. The satisfiability value for G is $\mathrm{val}(G) = \max_{\sigma\colon V\to\Sigma}\mathrm{val}(G, \sigma)$. We say that an assignment σ is a *solution* to a CSP instance G if $\mathrm{val}(G, \sigma) = 1$, and G is *satisfiable* iff G has a solution. When the context is clear, we omit σ in the description of a constraint, i.e., $\Pi_e(u_e, v_e)$ stands for $\Pi(\sigma(u_e), \sigma(v_e))$.

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 $\mathrm{val}(G) = 1$ or $\mathrm{val}(G) < 1 - \varepsilon$.

We refer to [35] for the backgrounds on fixed parameter tractability and FPT reductions. In this paper, for convenience, we use the following variant due to the sparsification lemma [43] and Tovey's reduction [64], which gives 3SAT additional structure.

Hypothesis 2.1 (ETH). *No algorithm can decide* 3SAT *within runtime* $2^{o(n)}$*, where each variable is contained in at most four clauses and each clause contains exactly three distinct variables.*[5]

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.

Definition 2.2 (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 $\mathtt{PWH}(A)[b] = Ab$.

Note that the parallel Walsh-Hadamard code is also known as interleaved Hadamard code and linear transformation code [25, 36].

When $d = 1$, the parallel Walsh-Hadamard code coincides with the standard Walsh-Hadamard code. It is clear that PWH has the relative distance $\delta = 1 - \frac{1}{|\mathbb{F}|}$, which is at least $\frac{1}{2}$ since $|\mathbb{F}| \geq 2$.

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 PWH, we perform the famous *BLR test* [12], 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 2.3. *If* $\mathbf{Pr}_{a,b\in\mathbb{F}^k}[w[a] + w[b] = w[a + b]] \geq 1-\varepsilon$*, then* $\Delta(x, \mathrm{Im}(\mathtt{PWH})) \leq 6\varepsilon$.

Assume w is η-close to an actual codeword w^* of 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 2.4. *If* w *is* η*-close to some* $w^* \in \mathrm{Im}(\mathtt{PWH})$*, then*

$$\Pr_{a\in\mathbb{F}^k}\left[w[x + a] - w[a] = w^*[x]\right] \geq 1 - 2\eta.$$

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

2.3 Probabilistic Checkable Proofs with Proximity

Probabilistic Checkable Proofs of Proximity (PCPP, also known as assignment testers) [9, 27] are essential gadgets when proving the PCP theorem [5, 7, 23]. 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 2.5 (($q, \delta, \varepsilon, f, g$)-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 2.6. 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 $|\pi_1 \circ \pi_2| \leq |\mathbb{F}|^k + f(k)$, $R_A \leq g(k)$, 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 2.7. *The reduction described in Definition 2.6 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 $|V'| \leq |\mathbb{F}|^k + f(k) + g(k)$.*
- *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 $\Delta(\sigma'(V'_1), \mathtt{PWH}(\sigma)) \leq \delta$.*

3 PROOF OF THE MAIN THEOREM

In this section, we prove the following quantitative version of our main theorem (Theorem 1.2).

Theorem 3.1. *Assuming ETH, no algorithm can decide $\frac{1}{9600}$-Gap k-CSP in $f(k) \cdot n^{o(\sqrt{\log\log k})}$ time 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 $k \geq 1$,* 3SAT *has a PCP 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 $\Pi_e(u, v) = 1_{u=M_e v}$.
  - Parallel. There exists a sub-constraint $\Pi^{sub}_e : \mathbb{F} \times \mathbb{F} \to \{0, 1\}$ and a subset of coordinates $Q_e \subseteq [d]$ such that Π_e checks Π^{sub}_e for every coordinate in Q_e, i.e., $\Pi_e(u, v) = \bigwedge_{i\in Q_e} \Pi^{sub}_e(u_i, v_i)$.
- Each variable is related to at most one parallel constraint.

[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.

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, we can prove Theorem 3.1 and Theorem 3.2 by combining the two steps above.

3.1 Reduction from 3SAT to VecCSPs

In this step, we reduce 3SAT to VecCSPs. By Hypothesis 2.1, 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 from VecCSPs 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 with $m(k) \ge 1$,*

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. 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 $\mathrm{val}(G^*) < 1 - \varepsilon^*$.*

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.*

[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.

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.*

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 $\varepsilon \in \left(0, \frac{1}{400}\right)$, 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. 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 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.*

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^*. In detail, set $G^* = (V^*, E^*, \Sigma^*, \{\Pi_e^*\}_{e\in E^*})$ to be the CSP instance obtained by applying the reduction in Definition 2.6 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 2.7. Putting everything together, we can prove Theorem 3.1 and Theorem 3.2 by combining Theorem 3.4 and Theorem 3.5.

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.

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 $C_1, \ldots, C_\ell$ where each 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 $C \in C_1 \dot\cup \cdots \dot\cup C_\ell$, we identify $\mathbb{F}_8 = \{0,1\}^3$ as the set of partial assignments to the clause 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 $\mathbb{F}_8$.

Now we define six tests as sub-constraints to be used later. For $j \in [3]$ and $b \in \{0,1\}$, we define $\Pi_{j,b} : \mathbb{F}_8 \times \mathbb{F}_8 \to \{0,1\}$ that checks: (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 $\mathbb{F}_8^{|C_p|}$, 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 $\in C_p$, 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 C_p and entries of $w_{p,q,j,s,b}$ by variables in $\mathcal{V}_q$. Since $d = \max\{\lceil m/\ell \rceil, \lceil n/\ell \rceil\} = \max\{|C_p|, |\mathcal{V}_q|\}$, 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 C_p and $\mathcal{V}_q$ respectively. 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 $[4]$.

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 $T_{\zeta,0}$ (and $T_{\zeta,1}$) as all pairs $(C, x) \in C_p \times \mathcal{V}_q$ where the s-th appearance of variable x is the j-th literal in clause C as x (and $\neg x$, resp.).

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 $C_p \times \mathcal{V}_q \subseteq [d] \times [d]$ 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 $\kappa_{\zeta,b} : [d] \to [d]$ 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.

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 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.

First, 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, 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 C_p, we impose the linear constraint that $1_{\widehat{z}=\widetilde{z}}$, which is a linear constraint.

Next, 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. 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 $1_{\widehat{w}=M_{\widehat{w},\widetilde{w}}\widetilde{w}}$ where $M_{\widehat{w},\widetilde{w}} \in \{0,1\}^{d\times d}$ is the permutation matrix of the permutation $\kappa_{\widehat{w}} \circ \kappa_{\widetilde{w}}^{-1}$, which is also a linear constraint.

5 PARALLEL PCPPS FOR VECTOR-VALUED CSPS WITH PARALLEL CONSTRAINTS

This section is devoted to proving Proposition 3.7. 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 NP $\subseteq$ PCP[poly$(n), O(1)$] [6].

5.1 An Exposition of the Quadeq Problem

In the following, we give a brief exposition of Quadeq for referencing purpose in our actual construction.

Definition 5.1 (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. The Quadeq problem also admits a constant-query PCPP based on the Walsh-Hadamard Code. We refer to [5] for details.

5.2 The Parallel Quadeq Problem

We will generalize the PCP verifier for Quadeq 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 $m \leq k/2$. 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 $\Pi_\ell : \mathbb{F} \times \mathbb{F} \to \{0,1\}$ 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., $\chi(a)+\chi(b)=\chi(a+b)$.
- For each sub-constraint $\Pi_\ell\colon \mathbb{F}\times\mathbb{F}\to\{0,1\}$, 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 $j\in[d]$, 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 C_S to compute the conjunction of the sub-constraints $\overline{\Pi}_\ell$ for $\ell\in S$. Formally, C_S is the Boolean function mapping $\mathbb{F}_2^{k\cdot t}$ to $\{0,1\}$ such that

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

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 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 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 C_S's. This is formalized in Claim 5.2. 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 5.2. Let $\sigma\colon V\to\mathbb{F}^d$ be an assignment of V. Then σ is a solution of G iff $C_{\kappa(j)}(y_1^j,\ldots,y_k^j)=1$ 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 C_S as the satisfiability of a quadratic equation system, which is formalized in Claim 5.3.

Claim 5.3. 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 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.

Moreover, u^j represents the values of gates in $C_{\kappa(j)}$ 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 Parallel PCPPs for Parallel Quadeq

In light of Claim 5.3, 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 5.3 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 5.3) reduced from circuit C_S (see Claim 5.2). 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 $\mathbb{F}_2^d$. 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[10] that the binary representations of σ^j satisfy the circuit $C_{\kappa(j)}$. 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.

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 2.5), 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 $\{0,1\}^d\cong\mathbb{F}_2^d$.

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.

(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 $\mathtt{PWH}(\sigma)$, $\mathtt{PWH}_2(\overline{u})$ and $\mathtt{PWH}_2(\overline{w})$ for some $\sigma\in(\mathbb{F}^d)^k,\overline{u}\in(\mathbb{F}_2^d)^{2^m\cdot c}$ and $\overline{w}\in(\mathbb{F}_2^d)^{2^m\cdot c^2}$.

(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 $\beta_1,\ldots,\beta_{2^m}\in\mathbb{F}_2^c$ and obtain $v:=\tau_1[\beta_1,\ldots,\beta_{2^m}]+\tau_1[\alpha_1+$

[10] Technically this proof is for Quadeq instance $\Gamma_{\kappa(j)}$. But due to Claim 5.3, we view it as a proof for the satisfiability of $C_{\kappa(j)}$. In fact, u^j is the values of gates in $C_{\kappa(j)}$ and $w^j=u^j(u^j)^\top$.

$\beta_1, \ldots, \alpha_{2^m} + \beta_{2^m}]$ 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 $\beta_1, \ldots, \beta_{2^m} \in \mathbb{F}_2^{c\times c}$ and obtain $v := \tau_2[\beta_1, \ldots, \beta_{2^m}] + \tau_2[\alpha_1+\beta_1, \ldots, \alpha_{2^m}+\beta_{2^m}]$ 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 $\kappa(j) = S$.

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

$$\tau_1[r_1, \ldots, r_{2^m}] \odot \tau_1[r'_1, \ldots, r'_{2^m}] = \tau_2[Z_1] + \tau_2[Z_2] \tag{1}$$

with four queries, where $Z_1 = y_1, \ldots, y_{2^m}, Z_2 = y_1 + r_1 r_1'^{\top}, \ldots, y_{2^m} + r_{2^m} r_{2^m}'^{\top}$, $\odot$ is the bit-wise multiplication. This simultaneously performs the Tensor Test of Quadeq on all d coordinates.

(P7) Pick a random subset H of $[q]$ and uniformly random $\beta_1, \ldots, \beta_{2^m} \in \mathbb{F}_2^{c\times c}$. For each $S \in [2^m]$, define $\alpha_S = \sum_{z\in H} D_{S,z} \in \mathbb{F}_2^{c\times c}$. Obtain $y := \tau_2[\beta_1, \ldots, \beta_{2^m}] + \tau_2[\alpha_1 + \beta_1, \ldots, \alpha_{2^m} + \beta_{2^m}]$ 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 of Quadeq on all d coordinates simultaneously, where on the j-th coordinate we check the constraints with respect to $C_{\kappa(j)}$.

(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 $\xi_1, \ldots, \xi_{2^m} \in \mathbb{F}_2^c$. 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

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

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

Then check if

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

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 part, we prove Proposition 3.7 with the following three lemmas (Lemma 5.4, Lemma 5.5, and Lemma 5.6), which are devoted to bound the parameters, and show completeness and soundness.

Lemma 5.4 (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.*

Lemma 5.5 (Completeness). *If G has a solution $\sigma : V \to \mathbb{F}^d$, then there is a proof $\pi_1 \circ \tau_1 \circ \tau_2$ which A accepts with probability 1.*

Lemma 5.6 (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 $\Delta(\pi_1, \mathsf{PWH}(\sigma)) \le 48\varepsilon$.*

Proposition 3.7 follows from Lemma 5.4, Lemma 5.5 and Lemma 5.6.

6 PARALLEL PCPPS FOR VECTOR-VALUED CSPS WITH LINEAR CONSTRAINTS

This section is devoted to proving Proposition 3.8.

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 $\Pi_e(u_e, v_e) = 1_{u_e = M_e v_e}$.

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 [9], with an additional introduction of some subtle auxiliary variables.

Auxiliary Variables. Label variables V by $\{1, 2, \ldots, k\}$ and constraints by $\{1, 2, \ldots, m\}$. For every $p \in V$ and $e \in E$, define an auxiliary variable $z_{p,e}$ with alphabet $\mathbb{F}^d$. Given an assignment $\sigma(p)$, the assignment to $z_{p,e}$ should equal $z_{p,e} = M_e\sigma(p)$ [11].

Note that we introduce an auxiliary 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 $\mathsf{PWH}(\sigma)$, 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 $\mathsf{PWH}(\sigma)$ for some solution σ of G. With equal probability, A selects one of the following four tests:

(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.

(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, these two tests ensure that both π_1 and π_2 are close to a codeword of PWH.

(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

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

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

Intuitively, this 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.

(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, this ensures $\sigma(u_e) = z_{v_e,e}$ for every $e \in E$.

6.2 Analysis of Parallel PCPPs

In this part, 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.*

Lemma 6.2 (Completeness). *If 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.*

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 $\Delta(\pi_1, \mathtt{PWH}(\sigma)) \le 24\varepsilon$.*

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

REFERENCES

[1] Fateme Abbasi, Sandip Banerjee, Jaroslaw Byrka, Parinya Chalermsook, Ameet Gadekar, Kamyar Khodamoradi, Dániel Marx, Roohani Sharma, and Joachim Spoerhase. 2023. Parameterized Approximation Schemes for Clustering with General Norm Objectives. *FOCS* (2023).

[2] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. 2020. Better Guarantees for k-Means and Euclidean k-Median by Primal-Dual Algorithms. *SIAM J. Comput.* 49, 4 (2020). https://doi.org/10.1137/18M1171321

[3] Benny Applebaum. 2017. Exponentially-hard gap-csp and local PRG via local hardcore functions. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 836–847.

[4] Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. 1997. The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations. *J. Comput. Syst. Sci.* 54, 2 (1997), 317–331. https://doi.org/10.1006/JCSS.1997.1472

[5] Sanjeev Arora and Boaz Barak. 2009. *Computational Complexity - A Modern Approach*. Cambridge University Press. http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264

[6] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. 1998. Proof Verification and the Hardness of Approximation Problems. *J. ACM* 45, 3 (1998), 501–555. https://doi.org/10.1145/278298.278306

[7] Sanjeev Arora and Shmuel Safra. 1998. Probabilistic Checking of Proofs: A New Characterization of NP. *J. ACM* 45, 1 (1998), 70–122. https://doi.org/10.1145/273865.273901

[8] Mihir Bellare, Oded Goldreich, and Madhu Sudan. 1998. Free Bits, PCPs, and Nonapproximability-Towards Tight Results. *SIAM J. Comput.* 27, 3 (1998), 804–915.

[9] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. 2006. Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding. *SIAM J. Comput.* 36, 4 (2006), 889–974. https://doi.org/10.1137/S0097539705446810

[10] Huck Bennett, Mahdi Cheraghchi, Venkatesan Guruswami, and João Ribeiro. 2023. Parameterized Inapproximability of the Minimum Distance Problem over all Fields and the Shortest Vector Problem in all ℓ_p Norms. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, Barna Saha and Rocco A. Servedio (Eds.). ACM, 553–566. https://doi.org/10.1145/3564246.3585214

[11] Arnab Bhattacharyya, Édouard Bonnet, László Egri, Suprovat Ghoshal, Karthik C. S., Bingkai Lin, Pasin Manurangsi, and Dániel Marx. 2021. Parameterized Intractability of Even Set and Shortest Vector Problem. *J. ACM* 68, 3 (2021), 16:1–16:40. https://doi.org/10.1145/3444942

[12] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. 1993. Self-testing/correcting with applications to numerical problems. *J. Comput. System Sci.* 47, 3 (1993), 549–595.

[13] Boris Bukh, Karthik C. S., and Bhargav Narayanan. 2021. 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*. IEEE, 237–244. https://doi.org/10.1109/FOCS52979.2021.00032

[14] Jaroslaw Byrka, Thomas W. Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. 2017. An Improved Approximation for k-Median and Positive Correlation in Budgeted Optimization. *ACM Trans. Algorithms* 13, 2 (2017), 23:1–23:31. https://doi.org/10.1145/2981561

[15] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. 2009. The Complexity of Satisfiability of Small Depth Circuits. In *Parameterized and Exact Computation*, Jianer Chen and Fedor V. Fomin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 75–85.

[16] Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. 2017. From Gap-ETH to FPT-Inapproximability: Clique, Dominating Set, and More. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, Chris Umans (Ed.). IEEE Computer Society, 743–754. https://doi.org/10.1109/FOCS.2017.74

[17] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. 2002. A Constant-Factor Approximation Algorithm for the k-Median Problem. *J. Comput. Syst. Sci.* 65, 1 (2002), 129–149. https://doi.org/10.1006/jcss.2002.1882

[18] Yijia Chen, Yi Feng, Bundit Laekhanukit, and Yanlin Liu. 2023. Simple Combinatorial Construction of the $k^{o(1)}$-Lower Bound for Approximating the Parameterized k-Clique. *CoRR* abs/2304.07516 (2023). https://doi.org/10.48550/arXiv.2304.07516 arXiv:2304.07516

[19] Yijia Chen and Martin Grohe. 2007. An isomorphism between subexponential and parameterized complexity theory. *SIAM J. Comput.* 37, 4 (2007), 1228–1258.

[20] Yijia Chen and Bingkai Lin. 2019. The Constant Inapproximability of the Parameterized Dominating Set Problem. *SIAM J. Comput.* 48, 2 (2019), 513–533. https://doi.org/10.1137/17M1127211

[21] Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li. 2019. Tight FPT Approximations for k-Median and k-Means. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece (LIPIcs, Vol. 132)*, Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 42:1–42:14. https://doi.org/10.4230/LIPIcs.ICALP.2019.42

[22] Erik D. Demaine, Mohammad Taghi Hajiaghayi, and Ken-ichi Kawarabayashi. 2005. 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*. IEEE Computer Society, 637–646. https://doi.org/10.1109/SFCS.2005.14

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

[24] Irit Dinur. 2016. Mildly exponential reduction from gap 3SAT to polynomial-gap label-cover. *Electron. Colloquium Comput. Complex.* 23 (2016), 128.

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

[26] Irit Dinur and Pasin Manurangsi. 2018. ETH-Hardness of Approximating 2-CSPs and Directed Steiner Network. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA (LIPIcs, Vol. 94)*, Anna R. Karlin (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 36:1–36:20. https://doi.org/10.4230/LIPICS.ITCS.2018.36

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

[28] Rodney G. Downey and Michael R. Fellows. 1995. Fixed-Parameter Tractability and Completeness I: Basic Results. *SIAM J. Comput.* 24, 4 (1995), 873–921. https://doi.org/10.1137/S0097539792228228

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

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

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

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

[33] Michael R Fellows. 2003. Blow-ups, win/win's, and crown rules: Some new directions in FPT. In *Graph-Theoretic Concepts in Computer Science: 29th International Workshop, WG 2003. Elspeet, The Netherlands, June 19-21, 2003. Revised Papers 29*. Springer, 1–12.

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

[35] Jörg Flum and Martin Grohe. 2006. *Parameterized Complexity Theory*. Springer. https://doi.org/10.1007/3-540-29953-X

[36] Parikshit Gopalan, Venkatesan Guruswami, and Prasad Raghavendra. 2011. List Decoding Tensor Products and Interleaved Codes. *SIAM J. Comput.* 40, 5 (2011), 1432–1462.

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

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

[39] Venkatesan Guruswami, Bingkai Lin, Xuandi Ren, Yican Sun, and Kewen Wu. 2023. Parameterized Inapproximability Hypothesis under ETH. arXiv:2311.16587 [cs.CC]

[40] Venkatesan Guruswami, Jakub Opršal, and Sai Sandeep. 2020. Revisiting Alphabet Reduction in Dinur's PCP. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 176)*, Jarosław Byrka and Raghu Meka (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 34:1–34:14. https://doi.org/10.4230/LIPIcs.APPROX/RANDOM.2020.34

[41] Venkatesan Guruswami, Xuandi Ren, and Sai Sandeep. 2023. Baby PIH: Parameterized Inapproximability of Min CSP. *arXiv preprint arXiv:2310.16344* (2023).

[42] Russell Impagliazzo and Ramamohan Paturi. 2001. On the Complexity of k-SAT. *J. Comput. System Sci.* 62 (2001), 367–375.

[43] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 2001. Which problems have strongly exponential complexity? *J. Comput. System Sci.* 63, 4 (2001), 512–530.

[44] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. 2004. A local search approximation algorithm for k-means clustering. *Comput. Geom.* 28, 2-3 (2004), 89–112. https://doi.org/10.1016/j.comgeo.2004.03.003

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

[46] Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. 2019. On the Parameterized Complexity of Approximating Dominating Set. *J. ACM* 66, 5 (2019), 33:1–33:38. https://doi.org/10.1145/3325116

[47] Karthik C. S. and Inbal Livni Navon. 2021. On Hardness of Approximation of Parameterized Set Cover and Label Cover: Threshold Graphs from Error Correcting Codes. In *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, Hung Viet Le and Valerie King (Eds.). SIAM, 210–223. https://doi.org/10.1137/1.9781611976496.24

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

[49] Euiwoong Lee. 2019. Partitioning a graph into small pieces with applications to path transversal. *Math. Program.* 177, 1-2 (2019), 1–19. https://doi.org/10.1007/s10107-018-1255-7

[50] Shi Li and Ola Svensson. 2016. Approximating k-Median via Pseudo-Approximation. *SIAM J. Comput.* 45, 2 (2016), 530–547. https://doi.org/10.1137/130938645

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

[52] Bingkai Lin. 2019. A Simple Gap-Producing Reduction for the Parameterized Set Cover Problem. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece (LIPIcs, Vol. 132)*, Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 81:1–81:15. https://doi.org/10.4230/LIPIcs.ICALP.2019.81

[53] Bingkai Lin. 2021. Constant approximating k-clique is W[1]-hard. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, Samir Khuller and Virginia Vassilevska Williams (Eds.). ACM, 1749–1756. https://doi.org/10.1145/3406325.3451016

[54] Bingkai Lin, Xuandi Ren, Yican Sun, and Xiuhan Wang. 2022. On Lower Bounds of Approximating Parameterized k-Clique. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France (LIPIcs, Vol. 229)*, Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 90:1–90:18. https://doi.org/10.4230/LIPIcs.ICALP.2022.90

[55] Bingkai Lin, Xuandi Ren, Yican Sun, and Xiuhan Wang. 2023. Constant Approximating Parameterized k-SETCOVER is W[2]-hard. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, Nikhil Bansal and Viswanath Nagarajan (Eds.). SIAM, 3305–3316. https://doi.org/10.1137/1.9781611977554.ch126

[56] Bingkai Lin, Xuandi Ren, Yican Sun, and Xiuhan Wang. 2023. Improved Hardness of Approximating k-Clique under ETH. *FOCS* (2023).

[57] Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. 2020. Parameterized Complexity and Approximability of Directed Odd Cycle Transversal. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, Shuchi Chawla (Ed.). SIAM, 2181–2200.

[58] Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan. 2020. A Parameterized Approximation Scheme for Min k-Cut. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, Sandy Irani (Ed.). IEEE, 798–809.

[59] Pasin Manurangsi. 2019. A Note on Max k-Vertex Cover: Faster FPT-AS, Smaller Approximate Kernel and Improved Approximation. In *2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San Diego, CA, USA (OASIcs, Vol. 69)*, Jeremy T. Fineman and Michael Mitzenmacher (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 15:1–15:21. https://doi.org/10.4230/OASIcs.SOSA.2019.15

[60] Pasin Manurangsi. 2020. 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*. SIAM, 62–81.

[61] Dániel Marx. 2008. Parameterized Complexity and Approximation Algorithms. *Comput. J.* 51, 1 (2008), 60–78. https://doi.org/10.1093/comjnl/bxm048

[62] Naoto Ohsaka. 2022. On the Parameterized Intractability of Determinant Maximization. In *33rd International Symposium on Algorithms and Computation (ISAAC 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[63] Piotr Skowron and Piotr Faliszewski. 2017. Chamberlin-Courant Rule with Approval Ballots: Approximating the MaxCover Problem with Bounded Frequencies in FPT Time. *J. Artif. Intell. Res.* 60 (2017), 687–716. https://doi.org/10.1613/jair.5628

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

[65] Andreas Wiese. 2018. Fixed-Parameter Approximation Schemes for Weighted Flowtime. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA (LIPIcs, Vol. 116)*, Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 28:1–28:19. https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2018.28

[66] Michał Włodarczyk. 2020. 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.

Received 13-NOV-2023; accepted 2024-02-11