Hardness of Approximating the Closest Vector Problem with Pre-Processing

Mikhail Alekhnovich [*] Subhash A. Khot [†] Guy Kindler [‡] Nisheeth K. Vishnoi [§]

Abstract

We show that, unless $\mathsf{NP}\subseteq\mathsf{DTIME}(2^{\text{poly}\log(n)})$, *the closest vector problem with pre-processing, for* ℓ_p *norm for any* $p \geq 1$, *is hard to approximate within a factor of* $(\log n)^{1/p-\epsilon}$ *for any* $\epsilon > 0$. *This improves the previous best factor of* $3^{1/p} - \epsilon$ *due to Regev [19]. Our results also imply that under the same complexity assumption, the nearest codeword problem with pre-processing is hard to approximate within a factor of* $(\log n)^{1-\epsilon}$ *for any* $\epsilon > 0$.

1. Introduction

An n-dimensional (integer) lattice $\mathcal{L}$ is a set of vectors $\{\sum_{i=1}^{n} a_i\mathbf{b}_i \mid a_i \in \mathbb{Z}\}$, where $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n \in \mathbb{Z}^n$ is a set of linearly independent vectors, called the *basis* of the lattice. Equivalently, one may define a lattice as an arbitrary additive subgroup of the group $\mathbb{Z}^n$.

Lattices are important mathematical objects that have many applications in various fields of mathematics, including convex analysis, number theory and computer science. They have been studied since the early 19th century by Gauss [9], who gave an algorithm to compute the shortest vector in a two-dimensional lattice. Subsequently, lattices have been studied in the works of Dirichlet, Hermite and Minkowski. The original motivation came from number theoretic problems such as solving Diophantine equations and finding rational approximations for real numbers. In recent times, lattices have had several important applications in computational mathematics. The discovery of the celebrated LLL algorithm of Lenstra, Lenstra and Lovász [14], that approximates the shortest vector in a lattice, allowed one to construct efficient algorithms for many computational tasks, such as solving integer programs in a fixed number of variables [14, 15, 10], factoring polynomials over rationals [14], and breaking a knapsack based cryptosystem [13]. Interestingly, lattices are used in both ways in cryptography: As an algorithmic tool for breaking other cryptographic systems, as well as for obtaining hard cryptographic primitives. In particular, in a ground breaking work, [2] Ajtai constructed an example of the worst-case to average-case reduction based on the shortest vector problem in a lattice. For a comprehensive introduction to the computational theory of lattices we refer the reader to [17].

A central computational problem in the theory of lattices is the so called closest vector problem (CVP): Given an integer lattice, represented by a basis $\mathbf{B}$, and a target vector $\mathbf{t}$, the objective is to find a lattice point $\mathbf{Bx}$ that minimizes the distance $\|\mathbf{Bx} - \mathbf{t}\|$. The best known approximation factor for CVP achieved by a (randomized) polynomial time algorithm is $2^{O(n \log\log n/\log n)}$, due to Ajtai *et al.* [3]. The best known deterministic polynomial time algorithm is due to Schnorr [20] and achieves a factor of $2^{O(n(\log\log n)^2/\log n)}$. On the other hand, a result of Dinur *et al.* [7] establishes that it is NP-hard to approximate CVP within a factor better than $n^{O(1/\log\log n)}$.

In this paper, we investigate the complexity of the closest vector problem with *pre-processing*, referred to as CVPP. In this setting, the basis $\mathbf{B}$ of the lattice depends only on the input length, and hence can be assumed to be known beforehand. This allows the possibility of doing arbitrary pre-processing with the basis, and using the pre-computed information to solve CVP on the input $(\mathbf{B}, \mathbf{t})$. Although there is no computational restriction on the pre-processing step, given the input, which in this case is a target vector $\mathbf{t}$, the algorithm should run in time polynomial in the length of $\mathbf{t}$.

The motivation to study the complexity of such pre-processing problems comes from cryptography and coding theory. In this setting, typically, a publicly known lattice (or a linear error-correcting code) is being used to transmit messages across a malicious/faulty channel. The decoding/decrypting of the received word reduces to solving CVP for the lattice being used in the protocol. Since the basis of the lattice is publicly known beforehand, it is natural to ask if the performance of the decoding algorithm can be improved, or if the security of the cryptographic protocol can be compromised, using this prior knowledge of the

[*] Institute for Advanced Study, Princeton, NJ. Supported by CCR grant $\mathcal{N}$CCR-0324906.

[†] College of Computing, Georgia Tech, Atlanta GA 30332. Email: khot@cc.gatech.edu

[‡] Institute for Advanced Study, Princeton, NJ. Supported by CCR grant $\mathcal{N}$CCR-0324906 and $\mathcal{N}$DMS-0111298.

[§] IBM India Research Lab, Block-1 IIT Delhi, New Delhi 110016. Email: nvishnoi@in.ibm.com lattice. For more details, refer to [8, 19].

The pre-processed version of CVP seems to be easier than the original problem in some cases. For instance, using the so-called Korkine-Zolotarev basis, Lagarias *et al.* [12] constructed an $O(n^{1.5})$ factor approximation algorithm for CVPP, which is far better than the almost-exponential approximation factor known for CVP. This was further improved to n by Regev [19], and subsequently to $O(\sqrt{n/\log n})$ by Aharonov and Regev [1].

In light of the fact that one can find much better approximation algorithms for CVPP when compared to CVP, proving strong hardness of approximation results for CVPP is a more challenging task than that for CVP. Bruck and Naor [5] showed NP-hardness for the analogue of CVPP in coding theory, the nearest codeword problem with pre-processing (NCPP). In this problem, a binary error correcting code $\mathcal{C}$ is fixed in advance and the goal is, given a vector $\mathbf{v}$, to find the closest (in the hamming metric) codeword in $\mathcal{C}$. Subsequently, Micciancio [16] established the NP-hardness of CVPP. Both results hold only for the exact version of the problems. The first inapproximability result was due to Feige and Micciancio [8], who proved a $(5/3)^{1/p} - \epsilon$ factor hardness for CVPP for ℓ_p norm, for any $\epsilon > 0$. They proved this by showing a $5/3 - \epsilon$ factor hardness for NCPP. Regev [19] improved these to $3 - \epsilon$ and $3^{1/p} - \epsilon$ respectively, for any $\epsilon > 0$.

In this paper we show that CVPP is NP-hard to approximate within any constant factor. Under the stronger assumption that NP$\not\subseteq$DTIME$(2^{\mathrm{poly}\log(\mathrm{n})})$, we show that CVPP for ℓ_p norm, for any $p \geq 1$, is hard to approximate within a factor of $(\log n)^{1/p-\epsilon}$ for any $\epsilon > 0$. Further, our results imply that NCPP is hard to approximate to within a factor of $(\log n)^{1-\epsilon}$, unless NP$\subseteq$DTIME$(2^{\mathrm{poly}\log(\mathrm{n})})$.

The paper is organized as follows: Section 2 presents notations and problem definitions. The rest of the paper contains two independent proofs of our results. Section 3 contains our first (self-contained) result that establishes the NP-hardness of approximating CVPP and NCPP to within any constant factor. This result also implies that both problems are hard to approximate within a factor of $(\log n)^{O(1)}$, unless NP$\subseteq$DTIME$(2^{\mathrm{poly}\log(\mathrm{n})})$. In Sections 4 and 5 we present another proof that gives stronger hardness results. This proof requires a pre-processed version of the PCP Theorem which may be of independent interest. Due to space limitations, the proof of the pre-processed PCP theorem will only appear in the full version of this paper.

2. Preliminaries

This section presents formal descriptions of the problems, notions, and notations which are used in our reductions.

Notations. Vectors and matrices will be denoted by bold letters such as $\mathbf{B}, \mathbf{v}$ etc. Vectors will be assumed to be column vectors. For a matrix $\mathbf{C} \in \mathbb{F}_2^{n\times k}$, let $\mathcal{C}(\mathbf{C})$ denote the linear code $\{\mathbf{Cv} : \mathbf{v} \in \mathbb{F}_2^k\}$. For a matrix $\mathbf{B} \in \mathbb{Z}^{n\times k}$, let $\mathcal{L}(\mathbf{B})$ denote the lattice $\{\mathbf{Bv} : \mathbf{v} \in \mathbb{Z}^k\}$. For a code $\mathcal{C}$ and a $\mathbf{t} \in \mathbb{F}_2^n$, let $\Delta(\mathcal{C}, \mathbf{v}) := \min_{\mathbf{c}\in\mathcal{C}} \delta(\mathbf{c}, \mathbf{v})$. Here $\delta(\cdot, \cdot)$ is the hamming distance.

For a positive integer n, define $[n] := \{1, \ldots, n\}$. For two sets S, T of the same cardinality, by abuse of notation, $S \equiv T$ will mean that we have fixed a bijection between S and T, and hence, use S, T interchangeably as is convenient. Henceforth, $\|\cdot\|$ denotes the Euclidean norm $\|\cdot\|_2$. However, the definitions in this paper hold for any ℓ_p norm, for $p \geq 1$. When we want to make the norm explicit, we will use the notation $\|\cdot\|_p$.

Following is a list of problems that we consider in this paper. We also present *informal* descriptions of the problems we consider before their actual formalizations.

Nearest Codeword Problem (NCP). Given a code $\mathcal{C} \subseteq \mathbb{F}_2^n$, and a word $\mathbf{t} \in \mathbb{F}_2^n$, the goal is to find a codeword $\mathbf{c} \in \mathcal{C}$ which is *closest* to $\mathbf{t}$ in the hamming metric. Here we consider the case when the code is a linear subspace of $\mathbb{F}_2^n$.

Problem 2.1 *For a function $f \geq 0$, an instance of* GapNCP$_{f(\cdot)}$ *is denoted by $(\mathbf{C}, \mathbf{t}, d)$, where $\mathbf{C} \in \mathbb{F}_2^{n\times k}$, $\mathbf{t} \in \mathbb{F}_2^n$ and $d \in \mathbb{Z}^+$. It is a YES instance if $\Delta(\mathcal{C}(\mathbf{C}), \mathbf{t}) \leq d$, and a NO instance if $\Delta(\mathcal{C}(\mathbf{C}), \mathbf{t}) > f(n) \cdot d$.*

Minimum Satisfiability in Linear Space (MINSAT). Here, we are given a CNF formula $\phi = C_1 \vee \cdots \vee C_m$ over the variables $\{x_1, \ldots, x_n\}$, and a linear subspace $\mathcal{L} \subseteq \mathbb{F}_2^n$. The goal is to find an assignment to the variables *from* the linear space $\mathcal{L}$ which satisfies as few clauses of ϕ as possible. This is a variant of the standard satisfiability problem: (1) This is a minimization problem, rather than a maximization problem. (2) The space from which one is allowed to choose an assignment is a part of the input, rather than $\mathbb{F}_2^n$.

Problem 2.2 *For a function $f \geq 0$, an instance of* GapMINSAT$_{f(\cdot)}$ *is denoted by $(\mathbf{V}, \mathcal{L}, \mathbf{E}, d)$, where $\mathbf{V}, \mathbf{E} \in \{0,1\}^{m\times n}$ and correspond to a CNF formula ϕ with variables $\{x_1, \ldots, x_n\}$ as explained later, $\mathcal{L}$ is a linear subspace of $\mathbb{F}_2^n$, and $d \in \mathbb{Z}^+$. The CNF formula ϕ corresponding to $\mathbf{V}, \mathbf{E}$ is the following: Let $\mathbf{V}_{ij}, \mathbf{E}_{ij}$ denote the i, j-th entry of $\mathbf{V}$, $\mathbf{E}$ respectively. ϕ has the form $C_1 \wedge \cdots \wedge C_m$, where each C_i is the boolean OR (over $j = 1, \ldots, n$) of the literals $x_j^{\mathbf{E}_{ij}}$ for which $\mathbf{V}_{ij} = 1$ (the notation x_j^ϵ means that the variable x_j is negated if and only if $\epsilon = 0$). It is a YES instance if there is a $\mathbf{v} \in \mathcal{L}$ which satisfies at-most d clauses of ϕ, and a NO instance if every $\mathbf{v} \in \mathcal{L}$ satisfies more than $f(n) \cdot d$ clauses of ϕ. Let $\rho(\phi)$ denote the minimum, over the assignments $\mathbf{v} \in \mathcal{L}$, of the clauses of ϕ satisfied by $\mathbf{v}$.*

k-Hypergraph Vertex Cover (HVC(k)). Given a hypergraph $\mathcal{H}(V, E)$, where each edge of $\mathcal{H}$ has cardinality k, the goal is to find a minimum size subset of V which intersects with (*covers*) all the edges in E.

Problem 2.3 *For $f \geq 0$, an instance of* GapHVC(k)$_{f(\cdot)}$ *is denoted by $(\mathcal{H}(V, E), d)$, where $\mathcal{H}$ is a hypergraph with vertex set $V = [n]$, and edge set E consisting of edges $e \subseteq V$, with each edge of cardinality k, while $d \in \mathbb{Z}^+$. It is a YES instance if there is a set $C \subseteq V$, with $|C| \leq d$, such that $|C \cap e| \geq 1$ for all $e \in E$. It is a NO instance if for every $C \subseteq V$, with $|C| \leq f(n, k) \cdot d$, there is an edge $e \in E$ with $C \cap e = \emptyset$.*

Closest Vector Problem (CVP). Given a lattice $\mathcal{L} \subseteq \mathbb{Z}^n$, and a vector $\mathbf{t} \in \mathbb{Z}^n$, the goal is to find a lattice point $\mathbf{v} \in \mathcal{L}$ which is *closest* to $\mathbf{t}$ in the ℓ_2 distance. The lattice is typically generated by a (full-rank) basis matrix $\mathbf{B} \in \mathbb{Z}^{n \times n}$: $\mathcal{L} = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}$. This is a variant of the closest vector problem, but for our purposes, this turns out to be more convenient to work with instead of CVP. Roughly, a hardness result for MISP implies an equivalent hardness result for CVP.

Problem 2.4 *For a function $f \geq 0$, an instance of* GapCVP$_{f(\cdot)}$ *is denoted by $(\mathbf{B}, \mathbf{t}, d)$, where $\mathbf{B} \in \mathbb{Z}^{n \times n}$, $\mathbf{t} \in \mathbb{Z}^n$ and $d \in \mathbb{Z}^+$. It is a YES instance if there exists a $\mathbf{x} \in \mathbb{Z}^n$ such that $\|\mathbf{Bx} - \mathbf{t}\| \leq d$, and a NO instance if for all $\mathbf{x} \in \mathbb{Z}^n$, $\|\mathbf{Bx} - \mathbf{t}\| > f(n) \cdot d$.*

Minimum Integral Solution Problem (MISP). The input to this problem consists of a set of *fixed* linear forms, described by $\mathbf{B}_f \in \mathbb{Z}^{k_1 \times n}$, a set of *variable* linear forms $\mathbf{B}_v \in \mathbb{Z}^{k_2 \times n}$, and a target vector $\mathbf{t} \in \mathbb{Z}^{k_1}$. The goal is to find an *integral* solution $\mathbf{x} \in \mathbb{Z}^n$ to the system $\mathbf{B}_f\mathbf{x} = \mathbf{t}$, which is of least l_2 norm with respect to $\mathbf{B}_v$; or minimizes $\|\mathbf{B}_v\mathbf{x}\|$.

Problem 2.5 *For a function $f \geq 0$, an instance of* GapMISP$_{f(\cdot)}$ *is denoted by $(\mathbf{B}_f, \mathbf{B}_v, \mathbf{t}, d)$, where $\mathbf{B}_f \in \mathbb{Z}^{k_1 \times n}$, $\mathbf{B}_v \in \mathbb{Z}^{k_2 \times n}$, $\mathbf{t} \in \mathbb{Z}^{k_1}$ and $d \in \mathbb{Z}^+$. It is a YES instance if there exists a $\mathbf{x} \in \mathbb{Z}^n$ such that $\mathbf{B}_f\mathbf{x} = \mathbf{t}$ and $\|\mathbf{B}_v\mathbf{x}\| \leq d$, and a NO instance if for all $\mathbf{x} \in \mathbb{Z}^n$ satisfying $\mathbf{B}_f\mathbf{x} = \mathbf{t}$, $\|\mathbf{B}_v\mathbf{x}\| > f(n) \cdot d$. $\mathbf{B}_f$ will be referred to as "fixed" linear forms on the variables, while $\mathbf{B}_v$ will be referred to as "variable" linear forms.*

Label Cover Problem (LCP). The input to this problem consists of: (1) A bipartite graph $G(V, W, E)$. (2) Two integers R and S; the intention being to assign vertices in V labels from $[S]$, and to assign vertices in W labels from $[R]$. (3) The labeling has to satisfy certain *constraints* given by functions $\pi_{(v,w)}$ for each edge $(v, w) \in E$. Given a label for $w \in W$, the map $\pi_{(v,w)}$ fixes the label that $v \in V$ should be assigned in order to *satisfy* the edge (v, w). (4) Further, for every $w \in W$, a *partition* $\mathcal{P}_w$ of $[R]$ is specified and a *permissible* set from the partition P_w is provided. w can only be assigned labels from the set $P_w \subseteq [R]$. The goal is to find an assignment of labels to vertices of G so as to maximize the number of satisfied edges.
The reason why we have a partition $\mathcal{P}_w$ for every vertex $w \in W$ is technical and will become clear in subsequent sections. It may be useful as of now to ignore the partition and just think of $P_w \subseteq [R]$ as the only labels that one is allowed to assign to vertices in w. Thus, compared to the vertices in V, each vertex in W has its own (different) set of labels, but each is a subset of a common ground set $[R]$.

Problem 2.6 *For a function $f \geq 0$, an instance of* GapLCP$_{f(\cdot)}$ *is*

$$\mathcal{U}\left(G(V, W, E), [R], [S], n, m, \{\pi_e\}_{e \in E}, \{\mathcal{P}_w, P_w\}_{w \in W}\right),$$

where $G = (V, W, E)$ is a a bipartite graph, with $|V| = n$, $|W| = m$, E is the set of edges, $[S]$ is the set of labels for vertices in V, for every $w \in W$, $\mathcal{P}_w = \bigsqcup_l R_{w,l}$ is a partition of the set $[R]$, while $P_w \in \{R_{w,l}\}$ is a set in the partition that represents all "permissible" labels from which a label can be assigned to $w \in W$, and for every $e \in E$, $\pi_e : [R] \mapsto [S]$. A "labeling" is a pair of maps $L_W : W \mapsto [R]$, $L_V : V \mapsto [S]$. An edge $e = (v, w)$ is "satisfied" by a labeling (L_V, L_W) if $L_W(w) \in P_w$ and $\pi_e(L_W(w)) = L_V(v)$. $\mathcal{U}$ is a YES instance if there is a labeling that satisfies all its edges. It is a NO instance if every labeling satisfies at-most $f(n)$ fraction of the edges.

We note that in the standard definition of this problem, the partition is trivial: $\mathcal{P}_w = \{[R]\}$ for all $w \in W$, and hence, the set of permissible labels $P_w = [R]$ for all $w \in W$.

2.1. Pre-processing versions of Gap problems

Let us formally define what pre-processing problems are, and some specific pre-processing versions of problems. We actually focus our attention on *uniform* pre-processing problems, whose hardness can be based on uniform assumptions.

Pre-processing Problems. Consider a gap problem GapX$_{f(\cdot)}$ where inputs are tuples $(A_1, \ldots, A_l)$ of a fixed length, and have a size parameter n which is polynomially related to the actual size of the input. In a pre-processing version of GapX$_{f(\cdot)}$, we consider subproblems where part of the input depends only on n, and therefore can be assumed to be known in advance.

Formally, a pre-processing instance of GapX$_{f(\cdot)}$ is specified by an algorithm which, given the string 1^n, generates in polynomial time a partial input $(A_1, \ldots, A_k)$ for some fixed $k < l$. We refer to this algorithm as a *uniform partial input generator*, and we call $(A_1, \ldots, A_k)$ the *uniform input*.

An algorithm which *solves* $\mathsf{GapX}_{f(\cdot)}$ *with preprocessing* in time $T(n)$, is actually an arbitrary function $\mathbb{P}$, which has no complexity constraints. Given a preprocessing instance of $\mathsf{GapX}_{f(\cdot)}$ (namely the code of a uniform partial input generator), $\mathbb{P}$ outputs an algorithm which solves the $\mathsf{GapX}_{f(\cdot)}$ problem in polynomial time on any input $(A_1, \ldots, A_l)$ with size parameter n, on condition that the partial tuple $(A_1, \ldots, A_k)$ is the one obtained by the partial input generator when run on the string 1^n.

One can easily observe that in order to show hardness of solving $\mathsf{GapX}_{f(\cdot)}$ with preprocessing, it suffices to come up with the following ingredients: the first is a preprocessing instance of $\mathsf{GapX}_{f(\cdot)}$, and the other is a reduction from a hard problem to $\mathsf{GapX}_{f(\cdot)}$, which generates inputs $(A_1, \ldots, A_l)$ of size n, where $(A_1, \ldots, A_k)$ is the output of the partial instance generator on input 1^n.

Note that once the $\mathsf{GapX}_{f(\cdot)}$ problem is defined, its preprocessing version is defined by specifying k, namely the entries of the partial inputs that are uniformly generated and are allowed to be pre-processed. In the following list we define pre-processing versions of problems by specifying what the uniform part of their input should be.

- **Nearest Codeword Problem with Pre-processing ($\mathsf{GapNCPP}_{f(\cdot)}$)**
  From the input tuple $(\mathbf{C}, \mathbf{t}, d)$ to $\mathsf{GapNCP}_{f(\cdot)}$, $(\mathbf{C})$ is the uniform input to $\mathsf{GapNCPP}_{f(\cdot)}$.

- **Minimum Satisfiability in Linear Space with Pre-processing ($\mathsf{GapMINSATP}_{f(\cdot)}$)**
  From the input to $\mathsf{GapMINSAT}_{f(\cdot)}$, which is a tuple $(\mathbf{V}, \mathcal{L}, \mathbf{E}, d)$, $(\mathbf{V}, \mathcal{L})$ is the uniform input to $\mathsf{GapMINSATP}_{f(\cdot)}$.

- **Closest Vector Problem with Pre-processing ($\mathsf{GapCVPP}_{f(\cdot)}$)**
  From the input tuple $(\mathbf{B}, \mathbf{t}, d)$ to $\mathsf{GapCVP}_{f(\cdot)}$, $(\mathbf{B})$ is the uniform input to $\mathsf{GapCVPP}_{f(\cdot)}$.

- **Minimum Integral Solution Problem with Pre-processing ($\mathsf{GapMISPP}_{f(\cdot)}$)**
  From the input to $\mathsf{GapMISP}_{f(\cdot)}$, which is a tuple $(\mathbf{B}_f, \mathbf{B}_v, \mathbf{t}, d)$, $(\mathbf{B}_f, \mathbf{B}_v)$ is the uniform input to $\mathsf{GapMISPP}_{f(\cdot)}$.

- **Label Cover Problem with Pre-processing ($\mathsf{GapLCPP}_{f(\cdot)}$)**
  Recall that an input to $\mathsf{GapLCP}_{f(\cdot)}$ is a tuple $\mathcal{U}$:
  $$(G(V, W, E), [R], [S], n, m, \{\pi_e\}_e, \{\mathcal{P}_w, P_w\}_{w \in W}) .$$
  The uniform part consist of $G(V, W, E)$, n, m, the set of *candidate* labels $[R]$ for vertices in W, $[S]$, and the projection maps π_e. Further, for every $w \in W$, a partition of $[R]$, $\mathcal{P}_w = \bigsqcup_l R_{w,l}$ is fixed, which also depends just on the length of the input. The input to $\mathsf{GapLCPP}_{f(\cdot)}$ now consists of a set $P_w \in \{R_{w,l}\}_l$, for every $w \in W$. Recall that this is the set of permissible labels for w.

When we do not wish to talk about the gap, we will refer to these problems as NCP, MINSAT, CVP, MISP, LCP, NCPP, MINSATP, CVPP, LCPP and MISPP.

The following proposition states that MISPP can be reduced to CVPP.

Proposition 2.7 *There exists a polynomial time reduction from* $\mathsf{GapMISPP}_{f(\cdot)}$ *to* $\mathsf{GapCVPP}_{f'(\cdot)}$, *where* $f'(n) := f(n^\epsilon)$, *for some constant* $\epsilon > 0$.

The proof of Proposition 2.7 can be obtained by a slight modification of Lemma 10 in [7], where a reduction is shown from the shortest integral solution problem, which is a special case of MISP.

3. Reduction from Hypergraph Vertex Cover

In this section we present a complete proof of the following result:

Theorem 3.1 *For every constant* $c > 0$, $\mathsf{GapNCPP}_c$ *is* NP*-complete. Further, under the assumption that* $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{\mathrm{poly}\log(\mathrm{n})})$, NCPP *is hard to approximate within a factor of* $(\log n)^\delta$, *for some constant* $\delta > 0$.

As an immediate corollary, via a reduction from $\mathsf{GapNCPP}_{f(\cdot)}$ to $\mathsf{GapCVPP}_{\sqrt{f(\cdot)}}$ (see [8, 19] for details), we obtain the following.

Corollary 3.2 *For every constant* $c > 0$, $\mathsf{GapCVPP}_c$ *is* NP*-complete. Further, under the assumption that* $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{\mathrm{poly}\log(\mathrm{n})})$, CVPP *is hard to approximate within a factor of* $(\log n)^\delta$, *for some constant* $\delta > 0$.

The result relies on the following theorem about the hardness of approximation of HVC(k). (This problem is referred to as Ek-Vertex-Cover in [6].)

Theorem 3.3 (Dinur *et al.* [6]) *For every* $\epsilon > 0$ *and every* $k \geq 3$, $\mathsf{GapHVC(k)}_{k-1-\epsilon}$ *is* NP*-complete.*

The proof of Theorem 3.1 involves two steps: (1) Lemma 3.4 gives an approximation preserving reduction from HVC(k) to MINSATP. The instance of MINSATP produced is of size poly(n^k). (2) Lemma 3.5 shows that, assuming that MINSATP is NP-hard, NCPP is hard to approximate within some (fixed) constant factor. Hence, a direct application of Theorem 3.3, with a large enough value of k proves Theorem 3.1.

Lemma 3.4 *For every odd integer k, there exists a $\mathrm{poly}\big(n^{O(k)}\big)$ time reduction that maps an instance $\mathcal{H}(V,E)$ (with $|V| = n$) of* HVC(k) *to an instance $(\mathbf{V},\mathcal{L},\mathbf{E})$ of* MINSATP*, such that if $\phi_{\mathcal{H}}$ denotes the CNF formula corresponding to $(\mathbf{V},\mathbf{E})$*

1. *The matrices $\mathbf{V},\mathbf{E}$ are of size $n \times \binom{n}{k}$.*
2. *$\mathbf{V}$ and $\mathcal{L}$ depend only on n and k.*
3. *$\rho(\phi_{\mathcal{H}})$ is equal to the size of a minimum vertex cover of $\mathcal{H}$.*

Proof: We construct $\phi_{\mathcal{H}}$ in the following way: The matrices $\mathbf{V}$ and $\mathbf{E}$ can be read off from the description of $\phi_{\mathcal{H}}$. The variables of $\phi_{\mathcal{H}}$ are $y_{I,i}$, for all $I \subseteq [n]$ with $|I| = k$, and all $i \in [n]$. For each $i \in [n]$, $\phi_{\mathcal{H}}$ contains a clause $C_i := \bigvee_{I \ni i} (y_{I,i})^{1-\chi_{\mathcal{H}}(I)}$, where $\chi_{\mathcal{H}}(I) = 1$ if and only if $\mathcal{H}$ contains the edge I. (Recall that the notation x^{ϵ} means that x is negated if and only if $\epsilon = 0$.) $\mathcal{L}$ is defined as the linear space over $\mathbb{F}_2$ spanned by the equations $\{\bigoplus_{i\in I} y_{I,i} = 0\}_{I\subseteq[n],|I|=k}$. Note that neither the underlying matrix $\mathbf{V}$ of $\phi_{\mathcal{H}}$, nor $\mathcal{L}$ depend on $\mathcal{H}$. Only $\mathbf{E}$ depends on $\mathcal{H}$. Now we prove (3). This is done in two parts:
1. $\rho(\phi_{\mathcal{H}})$ is less than or equal to the size of a minimum vertex cover of $\mathcal{H}$.
Let $C \subseteq [n]$ be a vertex cover of $\mathcal{H}$. For every edge $I \in E$, fix a vertex $\ell_I \in I$ which belongs to C. Set the variable $y_{I,i}$ to 1 if and only if $I \in E$ and $i \neq \ell_I$, and to 0 otherwise. Then, $y_{I,i}$ satisfies exactly the clauses that correspond to vertices in C. Also, since k is odd, this solution lies in the linear space $\mathcal{L}$.
2. $\rho(\phi_{\mathcal{H}})$ is greater than or equal to the size of a minimum vertex cover of $\mathcal{H}$.
Given an assignment for $y_{I,i}$, denote $C = \{i | C_i \text{ is satisfied}\}$. We claim that C is a vertex cover: Indeed, consider some edge $I \in E$. Since k is odd, and $\{y_{I,i}\}$ lie in the linear space $\mathcal{L}$, $y_{I,i}$ is 0 for at-least one $i \in I$. This implies that $i \in C$. This completes the proof of the lemma. ∎

Lemma 3.5 *There exists a constant $\mu > 0$, such that for every odd integer $k \geq 3$, there exists a polynomial reduction that maps an instance $(\mathbf{V},\mathcal{L},\mathbf{E})$ ($\mathbf{V},\mathbf{E} \in \{0,1\}^{m\times n}$ and each row of $\mathbf{V}$ has exactly k 1s) of* MINSATP *to an instance $(\mathbf{C},\mathbf{v})$ of* NCPP*, such that, if ϕ denotes the CNF formula corresponding to $(\mathbf{V},\mathbf{E})$, then,*

1. *$\mathbf{C}$ depends only on $\mathbf{V}$ and $\mathcal{L}$.*
2. *$\Delta(\mathbf{v},\mathcal{C}(\mathbf{C})) \leq \rho(\phi) l_1$.*
3. *$\Delta(\mathbf{v},\mathcal{C}(\mathbf{C})) \geq \mu\rho(\phi) l_1$.*

Here $l_1 := \mathrm{poly}(n, 1/\mu)$ and $\mathbf{C} \subset \mathbb{F}_2^{l_1}$.

Proof: Let $\phi = \bigwedge_{i=1}^{m} C_i$, where C_i are clauses over the variable set $\{x_1,\ldots,x_n\}$. First, we define a linear code $\mathcal{C}$ (depending on $\mathbf{E}$ and $\mathcal{L}$) that will be used in the reduction. For an integer parameter $l_1 = \mathrm{poly}(n, 1/\mu)$, fix an assymptotically good linear error correcting code $\mathcal{C}' \subseteq \{0,1\}^{l_1}$ of dimension n with a generating matrix $\mathbf{G} \in \{0,1\}^{l_1\times n}$, such that for all $\mathbf{v} \in \mathcal{C}'$, $|\mathbf{wt}(\mathbf{v})/l_1| \geq \mu$ for some positive constant μ.
For a clause $C_i = \bigvee_{\{j:\mathbf{V}_{ij}=1\}} x_j^{\mathbf{E}_{ij}}$, define the matrix $\mathbf{G}_i$ in the following way: The j-th column of $\mathbf{G}_i$ is the j-th column of $\mathbf{G}$ if $\mathbf{V}_{ij} = 1$, and the all zeroes vector, $\mathbf{0} \in \{0,1\}^{l_1}$, otherwise. Note that $\mathbf{G}_i$ depends only on the list of variables in C_i, which depends only on $\mathbf{V}$. The generating matrix $\mathbf{C}$ of $\mathcal{C}$ contains the product $\mathbf{G}_i \cdot \mathbf{G}_{\mathcal{L}}$ for each clause C_i :

$$\mathbf{C} = \left(\mathbf{G}_1 \cdot \mathbf{G}_{\mathcal{L}},\ \cdots, \mathbf{G}_m \cdot \mathbf{G}_{\mathcal{L}}\right)^{\mathsf{T}},$$

where $\mathbf{G}_{\mathcal{L}}$ is any generator matrix of $\mathcal{L}$ (i.e. a matrix such that $\mathbf{y} \in \mathcal{L}$ iff there is an $\mathbf{x}$ such that $\mathbf{y} = \mathbf{G}_{\mathcal{L}}\mathbf{x}$). We are ready to define the target vector $\mathbf{v} \in \{0,1\}^{l_1 m}$. For each clause C_i, define the i-th block of $\mathbf{v}$ (the block of co-ordinates from $(i-1)n+1,\ldots,in$) as $\mathbf{v}_i := \bigoplus_{j:\mathbf{V}_{ij}=1,\mathbf{E}_{ij}=0} \mathbf{G}[j]$, where $\mathbf{G}[j]$ is the j-th column of $\mathbf{G}$. Now, (2) and (3) follow from the following two statements:
1. If $\rho(\phi) \leq k$, then $\Delta(\mathbf{v},\mathcal{C}(\mathbf{C})) \leq k l_1$.
Consider the vector $\mathbf{u}$ such that $\mathbf{u} \in \mathcal{L}$, and $\mathbf{u}$ satisfies at-most k clauses of ϕ, let $\mathbf{u} = \mathbf{G}_{\mathcal{L}}\mathbf{x}$ Then, the hamming distance from the codeword $\mathbf{Cx}$ to $\mathbf{v}$ is $\Delta(\mathbf{v},\mathbf{Cx}) = \sum_{i=1}^{m} \Delta(\mathbf{G}_i\mathbf{u},\mathbf{v}_i)$. Note that for all i, $\mathbf{v}_i - \mathbf{G}_i\mathbf{u}$ is an element of $\mathcal{C}'$, because $\mathbf{v}_i \in \mathcal{C}'$, and every column of $\mathbf{G}_i$ is in $\mathcal{C}'$. Also, if the clause C_i is unsatisfied by $\mathbf{u}$, then $\mathbf{v}_i = \mathbf{G}_i\mathbf{u}$. Thus,

$$\Delta(\mathbf{v},\mathbf{u}) = \sum_{i=1}^{m} \Delta(\mathbf{G}_i\mathbf{u},\mathbf{v}_i) = \sum_{C_i \text{ is satisfied by } \mathbf{u}} \mathbf{wt}(\mathbf{G}_i\mathbf{u}-\mathbf{v}) \leq k l_1.$$

2. If $\rho(\phi) \geq k$, then $\Delta(\mathbf{v},\mathcal{C}(\mathbf{C})) \geq \mu k l_1$
Assume on the contrary that there exists a vector $\mathbf{x}$ such that $\Delta(\mathbf{v},\mathbf{Cx}) < \mu k l_1$, let $\mathbf{u} = \mathbf{G}_{\mathcal{L}}\mathbf{x}$. Then, $\mathbf{u} \in \mathcal{L}$. Note that $\mathbf{v}_i - G_i\mathbf{u}$ is either 0 (if and only if C_i is not satisfied by $\mathbf{u}$), or some non-zero codeword in $\mathcal{C}'$. Thus, one may write

$$\Delta(\mathbf{v},\mathbf{u}) = \sum_{i=1}^{m} \Delta(\mathbf{G}_i\mathbf{u},\mathbf{v}_i) = \sum_{C_i \text{ is satisfied by } \mathbf{u}} \mathbf{wt}(\mathbf{G}_i\mathbf{u}-\mathbf{v}) \geq \mu k l_2,$$

which contradicts our assumption that $\Delta(\mathbf{v}, \mathbf{Cu}) < \mu k l_1$. ∎

The first part of Theorem 3.1 now follows from Theorem 3.3, Lemma 3.4 and Lemma 3.5. The second part of Theorem 3.1 follows similarly from a strengthening of Theorem 3.3 when we choose $k = (\log n)^{O(1)}$.

4. PCPs for Constraint Satisfaction Problems, 2-Prover Games and the Label Cover Problem

A Quadratic Constraint Satisfaction Problem. The following constraint satisfaction problem (CSP) will be the starting point for our result.

Quadratic CSP over $\mathbb{F}_2$ (QCSP(2)). Given a set of quadratic polynomial equations over $\mathbb{F}_2[x_1, \ldots, x_n]$, the goal is to find an assignment to the variables $\{x_1, \ldots, x_n\} \in \mathbb{F}_2^n$ which satisfies as many equations as possible. Each equation is of the form $p(x_1, \ldots, x_n) = c$, where the degree of p is 2 and $c \in \mathbb{F}_2$. Further, each polynomial is known to depend only on at-most 3 variables.

Problem 4.1 *For a function $f \geq 0$, an instance of* GapQCSP(2)$_{f(\cdot)}$ *is denoted by $\left(\{p_j\}_{j=1}^m, \{c_j\}_{j=1}^m\right)$, where each p_j is a quadratic polynomial in $\mathbb{F}_2[x_1, \ldots, x_n]$ which depends on at-most three variables, and $c_j \in \mathbb{F}_2$. It is a YES instance if there exists an assignment $(a_1, \ldots, a_n) \in \mathbb{F}_2^n$ to the variables such that $p_j(a_1, \ldots, a_n) = c_j$, for all $1 \leq j \leq m$. It is a NO instance if for every assignment $(a_1, \ldots, a_n) \in \mathbb{F}_2^n$ to the variables, the fraction of equations which are satisfied, that is, the fraction of j for which $p_j(a_1, \ldots, a_n) = c_j$ is less than $f(n)$. The corresponding decision problem where the objective is to decide whether there is an assignment which satisfies all the equations or not is referred to as* QCSP(2)*.*

Quadratic CSP over $\mathbb{F}_2$ with Pre-processing (QCSPP(2), GapQCSPP(2)$_{f(\cdot)}$)
From the input $\left(\{p_j\}_{j=1}^m, \{c_j\}_{j=1}^m\right)$ to QCSP(2) and GapQCSP(2)$_{f(\cdot)}$, $\left(\{p_j\}_{j=1}^m\right)$ is the uniform input to QCSPP(2) and GapQCSPP(2)$_{f(\cdot)}$ respectively.

A PCP Theorem for QCSPP(2). The main reduction of this paper uses a PCP Theorem for QCSPP(2). But first, we need to establish NP-completeness of QCSPP(2).

Theorem 4.2 QCSPP(2) *is* NP*-complete.*

Proof: We reduce 3SAT to QCSPP(2). For this proof, it is convenient to view the formulation for 3SAT which has been used in the definition of MINSAT: The input is $(\mathbf{V}, \mathbf{E})$, where $\mathbf{V}, \mathbf{E} \in \{0,1\}^{m \times n}$ and corresponds to a 3SAT formula $\phi = C_1 \wedge \cdots \wedge C_m$ with variables $\{x_1, \ldots, x_n\}$. Further, each row of $\mathbf{V}$ has exactly 3 1s. Since 3SAT is in NP, for every n, there is a circuit $\mathcal{C}_n$ which takes as input $(\mathbf{V}, \mathbf{E})$ and an assignment $\mathbf{a} \in \{0,1\}^n$, such that, $\mathcal{C}_n(\mathbf{a}, \mathbf{V}, \mathbf{E}) = (1, \mathbf{V}, \mathbf{E})$ if $\mathbf{a}$ is a satisfying assignment for ϕ, and $(0, \mathbf{V}, \mathbf{E})$ otherwise.

Now we present the reduction. Let $(\mathbf{V}, \mathbf{E})$ be the input corresponding to a 3SAT instance ϕ. We may assume that every gate in $\mathcal{C}_n$ has fan-in 2 and fan-out 1. For every bit in the input $(\mathbf{a}, \mathbf{V}, \mathbf{E})$ to $\mathcal{C}_n$, there is a variable: x_i is supposed to be assigned the i-th bit of $\mathbf{a}$. $x_{i,j}$ is supposed to be assigned $\mathbf{V}_{ij}$, while $x'_{i,j}$ is supposed to be assigned $\mathbf{E}_{i,j}$.

Associated to the output of the i-th internal gate[1] in $\mathcal{C}_n$ is a variable z_i. Further, let y_0 be the variable corresponding to the gate which outputs whether an assignment $\mathbf{a}$ satisfies ϕ or not, and denote by $y_{i,j}$ and $y'_{i,j}$ the output of the gates outputting the i,j-th entry of $\mathbf{V}$ and $\mathbf{E}$ respectively. We just show the arithmetization (over $\mathbb{F}_2$) for the AND gate (similar arithmetizations hold for the NOT and the OR gate): If the input to an AND gate are variables z, z', and the output z'', then, one can write the equation $z'' - zz' = 0$ corresponding to it.

We write such an equation for every gate, internal or output, in $\mathcal{C}_n$. Each equation is of degree at-most 2 and has at-most 3 variables. Note that every such equation depends only on the description of $\mathcal{C}_n$. To see the connection with QCSPP(2), we think of $\mathbf{a}, \mathbf{V}, \mathbf{E}$ as variables in this set of polynomial equations and add the additional set of equations $y_0 = 1$, $y_{i,j} = \mathbf{V}_{ij}$ and $y'_{i,j} = \mathbf{E}_{ij}$. Hence, we get a QCSPP(2) instance over the set of variables

$$\{x_i : 1 \leq i \leq n\} \cup \{x_{i,j} : 1 \leq i,j \leq n\} \\ \cup \{x'_{i,j} : 1 \leq i,j \leq n\} \cup \{z_i : 1 \leq i \leq \text{size}(\mathcal{C}_n)\} \cup y_0 \\ \cup \{y_{i,j} : 1 \leq i,j \leq n\} \cup \{y'_{i,j} : 1 \leq i,j \leq n\},$$

(Notice that $\mathcal{C}_n$ can be generated by a polynomial time algorithm which is given as input 1^n. Hence, this reduction is a polynomial time reduction.) By definition, it follows that this quadratic system has a solution if and only if ϕ has a satisfying solution. This completes the proof of the lemma. ∎

PCP Theorem for QCSPP(2). We start with an instance $\mathcal{U}$ of QCSPP(2) and construct another instance $\mathcal{U}'$ of it in which the *unsatisfiability is amplified*: If $\mathcal{U}$ is satisfiable, so is $\mathcal{U}'$. However, if $\mathcal{U}$ is not satisfiable, then no assignment to the variables of $\mathcal{U}'$ satisfies more than a θ fraction of the constraints in $\mathcal{U}'$. Here, $\theta > 0$ is some fixed constant. The reduction is summarized in the following theorem:

[1] A gate is said to be internal if its output is not an output of the circuit.

Theorem 4.3 *Let $\left(\{p_j\}_{j=1}^m, \{c_j\}_{j=1}^m\right)$ be the input to* QCSP(2) *over the set of variables $\{x_1, \ldots, x_n\}$. There is a polynomial time constructible mapping, and a constant $\theta > 0$, such that:*

1. *The mapping takes a pair $\left(\{p_j\}_{j=1}^m, \{c_j\}_{j=1}^m\right)$ to $\left(\{p'_j\}_{j=1}^{m'}, \{c'_j\}_{j=1}^{m'}\right)$,*

2. *The set of polynomials $\{p'_j\}_{j=1}^{m'}$ are quadratic over $\mathbb{F}_2[x_1, \ldots, x_{n'}]$ and depend only on $\{p_j\}_{j=1}^m$.*

3. *If the system $\{p_j = c_j\}_{j=1}^m$ has a satisfying solution, so does the system $\{p'_j = c'_j\}_{j=1}^{m'}$.*

4. *If $\{p_j = c_j\}_{j=1}^m$ is not satisfiable, then for any assignment to its variables $\{x_1, \ldots, x_{n'}\}$, more than a θ fraction of the equations $\{p'_j = c'_j\}_{j=1}^{m'}$ are not satisfied.*

5. *If each p_j depends on at-most 3 variables, so does each p'_j.*

6. *There is a fixed integer n_0 such that each variable in $\{x_1, \ldots, x_{n'}\}$ appears in exactly n_0 of the polynomials $\{p'_j\}_{j=1}^{m'}$.*

Hence, there is an absolute constant $\theta > 0$ such that GapQCSPP(2)$_\theta$ *is* NP*-hard.*

The proof of this theorem, which is a laborious and an almost exact mimic of the proof of the PCP Theorem, is beyond the scope of this version of the paper.

2-Prover Games and the Label Cover Problem(s). Now we explain the standard framework for outer PCPs. Albeit, instead of starting with a gap instance of 3SAT, we start with an instance of GapQCSPP(2)$_\theta$ as in Theorem 4.3. The first step is to construct a 2-Prover-1-Round Game from the instance $\left(\{p'_j\}_{j=1}^{m'}, \{c'_j\}_{j=1}^{m'}\right)$ of GapQCSPP(2)$_\theta$. In the 2-Prover Game, the first prover is supposed to provide the assignments to the variables in each equation, while the second prover is supposed to provide an assignment to the variables which would satisfy all the equations. The verifier then picks an equation at random, say $p(x_1, x_2, x_3) = c$, reads the answer of the first prover: (a_1, a_2, a_3), a supposed assignment to the variables (x_1, x_2, x_3), checks if $p(a_1, a_2, a_3) = c$. If so, she then picks a random variable from among $\{x_1, x_2, x_3\}$, say x_3, and accepts only if both provers are consistent in their answers for x_3. It is easy to see that if the instance $\left(\{p'_j\}_{j=1}^{m'}, \{c'_j\}_{j=1}^{m'}\right)$ is a YES instance, then there is a strategy for both the provers to succeed with probability 1. While, if it is a NO instance, then there is no strategy for the 2 provers to succeed with probability more than $2/3 + \theta/3$.

Given an instance $\phi := \left(\{p'_j\}_{j=1}^{m'}, \{c'_j\}_{j=1}^{m'}\right)$ of QCSPP(2)$_\theta$ over the variable set $\{x_1, \ldots, x_{n'}\}$, construct an instance

$$\mathcal{U}_\phi(G(V, W, E), [8], [2], n', m', \{\pi_e\}_{e \in E}, \{\mathcal{P}_w, P_w\}_{w \in W})$$

as follows: The graph $G(V, W, E)$ has $V := \{x_1, \ldots, x_{n'}\}$, $W := \{p'_1, \ldots, p'_{m'}\}$, and $(x_i, p'_j) \in E$ if and only if p'_j depends on x_i. Note that the degree of a vertex in V is 3, while of that in W is n_0. We identify the set [8] with $\{0,1\}^3$, and the set [2] with $\{0,1\}$. For a vertex $p'_j \in W$, let $P_{p'_j} \subseteq \{0,1\}^3 \equiv [8]$ be the set of solutions to the equation $p'_j = c'_j$. The partition $\mathcal{P}_{p'_j}$ is just $P_{p'_j} \sqcup ([8] \backslash P_{p'_j})$. Now we define the projection maps $\pi_{(x_i, p'_j)}$. Since we have fixed a bijection between $\{0,1\}$ and [2], and one between $\{0,1\}^3$ and [8], it suffices to describe the map from $P_{p'_j}$ to $\{0,1\}$. For an edge $e = (x_i, p'_j)$, where p'_j depends on variables (x_i, x_j, x_k), and an assignment (a, b, c) in $P_{p'_j}$, the mapping is $\pi_e((a, b, c)) = a$. Here a is the value assigned to x_i. Notice that $\mathcal{U}_\phi$ is an instance of GapLCPP$_{2/3+\theta/3}$. By Theorem 4.3 and the discussion preceding the reduction, it follows that GapLCPP$_{2/3+\theta/3}$ is NP-complete.

Amplifying the gap. Starting with an instance $\mathcal{U}_\phi(G(V, W, E), [8], [2], n, m, \{\pi_e\}_{e \in E}, \{\mathcal{P}_w, P_w\}_{w \in W})$ of GapLCPP$_{2/3+\theta/3}$, for an integer $k > 0$, one can use Raz's Parallel Repetition Theorem [18] to construct an instance $\mathcal{U}_\phi^{\otimes k}$:

$$(G'(V', W', E'), [R'], [S'], n', m', \{\pi'_e\}_e, \{\mathcal{P}'_w, P'_w\}_{w \in W'})$$

with $n' = n^k$, $m' = m^k$ and R', S' are $2^{O(k)}$. It follows from Raz's result that that $\mathcal{U}_\phi^{\otimes k}$ is an instance of GapLCPP$_{\theta'^k}$, where θ' depends only on θ.

Smoothening the projection maps in LCPP**.** For our reduction we will need an additional property from the projection maps $\{\pi_e\}_e$: For a $\delta > 0$, for any $w \in W$, and any pair of distinct labels $i, i' \in [R]$,

$$\Pr_{v \in_R N(w)} \left[\pi_{(v,w)}(i) = \pi_{(v,w)}(i')\right] \leq \delta.$$

Here the probability is over picking a random neighbor of w. We refer to this property as δ-smoothness, and in general is not guaranteed by the instances of GapLCPP$'^k_\theta$ produced by parallel repetition. Khot [11] proposed a modification of Raz's parallel verifier with parameter k. His construction has parameters T, k (think of $T, k \gg 1$, and can be chosen independently of each other) and allows one to construct instances which have the δ-*smoothness* property for $\delta = 1/T$. More precisely, in our setting, starting with an instance $\mathcal{U}_\phi(G(V, W, E), [8], [2], n, m, \{\pi_e\}_e, \mathcal{P}_w, \{P_w\}_{w \in W})$ of

GapLCPP$_{2/3+\theta/3}$, and integer parameters T, k, Khot's 2-Prover Game can be used to construct an instance $\mathcal{U}_\phi^{T,k}$:

$$(G'(V', W', E'), [R'], [S'], n', m', \{\pi'_e\}_e, \{\mathcal{P}'_w, P'_w\}_{w \in W'})$$

with $n' = n^{O(Tk)}$, $m' = m^{O(Tk)}$ and R', S' are $2^{O(Tk)}$. Using Raz's result, he shows that $\mathcal{U}_\phi^{T,k}$ is an instance of GapLCPP$_{2^{-\gamma k}}$, where $0 < \gamma < 1$ depends only on θ, and hence, is fixed. Further, he shows that $\mathcal{U}_\phi^{T,k}$ has the $1/T$-smoothness property.

Some explanation is in order regarding the partition $\mathcal{P}'_w$ and the set of permissible labels P'_w. We obtain an instance of LCPP by combining an instance promised by the PCP Theorem for QCSPP(2) with Khot's 2-Prover Game. Hence, the set of satisfying assignments of a polynomial change based on the right hand side of the equation it is in. Hence, for a k-tuple of polynomials, based on the 2^k possible values of the right hand sides of the equations, there are 2^k sets. These correspond to the candidate labels for a vertex w, which in turn corresponds to the k-tuple of polynomials under consideration. From these candidate labels, based on the right hand side of the polynomial equations, the input; the set of permissible labels for w, is specified. This partition of candidate labels, of course, depends on the right hand sides of the polynomials.

5. Reduction from Label Cover: $(\log n)^{1/2-\epsilon}$ hardness

Theorem 5.1 *For every $\epsilon > 0$,* GapMISPP$_{(\log n)^{1/2-\epsilon}}$ *is* NP*-complete unless* NP$\subseteq$DTIME$(2^{\text{poly}\log(n)})$.

This implies, via Proposition 2.7, the hardness of approximation of CVPP.

Corollary 5.2 *For every $\epsilon > 0$,* GapCVPP$_{(\log n)^{1/2-\epsilon}}$ *is* NP*-complete unless* NP$\subseteq$DTIME$(2^{\text{poly}\log(n)})$.

Remark 5.3 *The proof of Theorem 5.1 (and hence, of Corollary 5.2) can be easily seen to imply a hardness of $(\log n)^{1/p-\epsilon}$ for* MISPP *and* CVPP *over ℓ_p norm, for any $p \geq 1$.*

Noticing that our reduction produces a MISPP instance with $0/1$ values, *a la* Arora *et al.* [4], we infer the following corollary:

Corollary 5.4 *For every $\epsilon > 0$,* GapNCPP$_{(\log n)^{1-\epsilon}}$ *is* NP*-complete unless* NP$\subseteq$DTIME$(2^{\text{poly}\log(n)})$.

Overview. The proof of Theorem 5.1 proceeds by reducing a *smoothened* label cover instance to an instance of MISPP. Recall that for arbitrary parameters T and k, starting with a QCSPP(2) instance of size $\text{poly}(n)$, the PCP Theorem for QCSPP(2) (Theorem 4.2) combined with Khot's 2-Prover Game produces an instance of LCPP, namely $\mathcal{U}^{T,k}$, of size $n^{O(Tk)}$ in which the constraint maps are $1/T$-smooth. Further, it is also guaranteed that deciding whether there is a labeling which satisfies all the constraints or no labeling satisfies more than a $2^{-\gamma k}$ fraction of the constraints is NP-hard, for some fixed $\gamma > 0$.
The reduction takes $\mathcal{U}^{T,k}$ and, in time polynomial in the size of $\mathcal{U}^{T,k}$, converts it into a MISPP instance $\mathcal{B}(T, k, n)$ such that:

1. If there is a labeling which satisfies all the constraints of $\mathcal{U}^{T,k}$, then $\mathcal{B}(T, k, n)$ has *short* integral solutions of cost at-most $C(n)$.
2. If no labeling of $\mathcal{U}^{T,k}$ satisfies more than $2^{-\gamma k}$ fraction of the constraints, then every integral solution to $\mathcal{B}(T, k, n)$ is of cost at-least $2^{\frac{\gamma k}{2}} C(n)$.

Thus, picking $k = \Omega(\log \log N)$, $T = 2^{\Omega(k)}$, where N is the size of $\mathcal{B}(T, k, n)$, we obtain about $\sqrt{\log N}$ hardness for MISPP. Now we move on to the description of the precise reduction and the choice of parameters.

The reduction. For parameters T, k to be decided later, consider the following instance ($\mathcal{U}^{T,k}$) of GapLCPP$_{2^{-\gamma k}}$ from Section 4:

$$(G(V, W, E), [R], [S], n, m, \{\pi_e\}_{e \in E}, \{\mathcal{P}_w, P_w\}_{w \in W}),$$

where $n = |V|, m = |W|$ are $n^{O(Tk)}$, and R', S' are $2^{O(Tk)}$. The only part of the input which is not uniform (or does not depend on n) is $\{P_w\}_{w \in W}$. Recall that for every $w \in W$, $\bigsqcup_l R_{w,l}$ a partition of $[R]$, while the input is the set of permissible labels for each w, $P_w \in \{R_{w,l}\}_l$. The instance is $\delta := 1/T$ smooth. Recall that this means that for any $w \in W$ and any pair of distinct labels $i, i' \in [R]$,

$$\Pr_{v \in_R N(w)} \left[\pi_{(v,w)}(i) = \pi_{(v,w)}(i')\right] \leq \delta.$$

Now we define the corresponding GapMISP instance. The variables are:

$$\begin{aligned} x_{w,i} &\quad : \quad \forall\, w \in W,\ \forall i \in [R] \\ y_{v,j} &\quad : \quad \forall\, v \in V,\ \forall j \in [S] \end{aligned}$$

The *fixed* linear forms are

$$\sum_{i \in P_w} x_{w,i} = 1 \qquad \forall\, w \in W \tag{1}$$

$$\sum_{i \in R_{w,l}} x_{w,i} = 0 \qquad \forall\, w{\in}W,\ R_{w,l}{\neq}P_w \tag{2}$$

$$\sum_{j \in [S]} y_{v,j} = 1 \qquad \forall\, v \in V \tag{3}$$

$$\Big(\sum_{i : \pi_{(v,w)}[i]=j} x_{w,i}\Big) - y_{v,j} = 0 \qquad \begin{matrix}\forall e{=}(v,w){\in}E, \\ \forall j{\in}[S]\end{matrix} \tag{4}$$

The *variable* linear forms are

$$\begin{aligned} \sqrt{n}\cdot x_{w,i} &\quad:\quad \forall\, w\in W,\ \forall i\in[R] \\ \sqrt{m}\cdot y_{v,j} &\quad:\quad \forall\, v\in V,\ \forall j\in[S] \end{aligned}$$

Since the partition $\mathcal{P}_w = \bigsqcup_l R_{w,l}$ depends only on n, the only part that depends on $\{P_w\}_{w\in W}$ is the r.h.s. of (1) and (2), and hence, this is an instance of GapMISPP. Now we analyze the gap of this reduction and its tradeoff with the size of the instance produced.

Completeness

If $\mathcal{U}^{T,k}$ is a YES instance of $\mathsf{GapLCPP}_{2^{-\gamma k}}$, then there is an assignment to the variables of the corresponding GapMISPP instance such that the objective is at-most $\sqrt{2mn}$. Consider a labeling which satisfies all the edges of $\mathcal{U}^{T,k}$. Now we construct a solution to the GapMISPP with objective at-most $\sqrt{2mn}$ as follows: If the label i is assigned to the vertex $w\in W$, then, assign 1 to $x_{w,i}$, and assign 0 to all $x_{w,i'}$, for $i'\neq i$. This makes sure that the constraints (1) and (2) are satisfied. Similarly, if the label j is assigned to the vertex $v\in V$, then, assign 1 to $y_{v,j}$, and assign 0 to $y_{v,j'}$, for $j'\neq j$. This makes sure that the constraints (3) are satisfied. Further, if labels i and j are assigned to w and v respectively in this satisfying assignment, then for the edge (v,w), $\pi_{(v,w)}(i)=j$, and hence, the constraints (4) are also satisfied.

Soundness

This is where we need the $\delta = 1/T$-smoothness of $\mathcal{U}^{T,k}$. We will establish factor h hardness, where h, as well as other parameters, are fixed in the end. Assume that there is a solution to the GapMISPP instance with objective equal to $h\sqrt{2mn}$. Then,

$$\sum_{w\in W, i\in[R]} x_{w,i}^2 < 2mh^2 \quad\text{and}\quad \sum_{v\in V, j\in[S]} y_{v,j}^2 < 2nh^2. \tag{5}$$

We define the block of variables $B_w := \{x_{w,i} : 1\le i\le R\}$, for every $w\in W$, and $A_v := \{y_{v,j} : 1\le j\le S\}$, for every $v\in V$. By (5), the average number of non-zero variables in the B-blocks, as well as in the A-blocks, is at most $2h^2$. We throw away all $v\in V$ and $w\in W$ whose respective blocks contain more than $200h^2$ non-zero entries. We do not care about satisfying the edges incident to such vertices, as they make up for at-most a 0.01-fraction of the edges.

We can therefore assume that for each of the remaining vertices, there are at-most $200h^2$ non-zero variables in its block. Let A_v^+ (resp. B_w^+) denote the set of non-zero variables in the respective blocks. From (1), we have that there is at-least one $x_{w,i}\in P_w$ which is non-zero, or in our notation, $B_w^+\cap P_w\neq\emptyset$. We therefore arbitrarily assign, for every $w\in W$, a label l_w from $B_w^+\cap P_w$. Similarly, for every $v\in V$, there is at-least one non-zero $y_{v,j}$. We assign v with a label chosen *at random* from A_v^+.

Now using the δ-smoothness property, and a union bound, we have that for all the remaining vertices $w\in W$ (that have not been thrown away)

$$\begin{aligned} \Pr_{v\in N(w)} &\left[\exists\, l'\neq l_w : \pi_{(v,w)}(l') = \pi_{(v,w)}(l_w)\right] \\ &\le \delta|B_w^+| \le 200\delta h^2. \end{aligned}$$

Call an edge (v,w) *good*, if for all $l'\neq l_w$, $\pi_{(v,w)}(l')\neq\pi_{(v,w)}(l_w)$. For every w, there are at-least $1-200\delta h^2$ fraction of edges (v,w) which are good. For a vertex v neighbouring w on a good edge, the constraint (4) implies that $y_{v,\pi_{(v,w)}(l_w)} = x_{w,l_w}$. Since x_{w,l_w} is non-zero, so is $y_{v,\pi_{(v,w)}(l_w)}$, and hence, with probability at-least $1/|A_v^+|\ge 1/200h^2$, the edge (v,w) is satisfied. Hence, the fraction of edges satisfied is at-least $1/(200h^2)\left(1-0.01-200\delta h^2\right)$. This has to be at-most $2^{-\gamma k}$, as $\mathcal{U}^{T,k}$ is a NO instance of $\mathsf{GapLCPP}_{2^{-\gamma k}}$.

Choice of parameters. We choose δ as a function of h, such that $200\delta h^2 \le 0.01$. Since this leaves the fraction of edges satisfied by the above labeling at-least $0.981/(200h^2)$, it must be that $h = 2^{\frac{\gamma}{2}k-a}$, for some fixed constant a. This implies that $\delta = 2^{-\gamma k-b}$, for some fixed constant b. Hence, $T(=1/\delta) = 2^{\gamma k+b}$. The size of the instance N is n^{cTk} for some fixed constant c. Let $k = \frac{1}{\gamma\epsilon}\log\log n$. Hence, $n = 2^{2^{\gamma\epsilon k}}$, and $N = 2^{2^{\gamma\epsilon k+\gamma k+b+\log k+\log c}}$. Hence, $h = 2^{\frac{\gamma}{2}k-a} \ge (\log N)^{1/2-\epsilon}$. This establishes that the size of the instance of GapMISPP produced is $N^{\mathrm{polylog}(N)}$ and the hardness factor is $(\log N)^{1/2-\epsilon}$, and hence, Theorem 5.1 follows.

6 Acknowledgements

We are grateful to Daniele Micciancio for his simplification of Lemma 3.5.

References

[1] D. Aharonov and O. Regev. Lattice problems in NP intersect coNP. In *Annual Symposium on Foundations of Computer Science*, number 45, pages 362–371, 2004.

[2] M. Ajtai. The shortest vector problem in ℓ_2 is NP-hard for randomized reductions. In *Proceedings of the ACM Symposium on the Theory of Computing*, number 30, pages 10–19, 1998.

[3] M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the ACM Symposium on the Theory of Computing*, number 33, pages 601–610, 2001.

[4] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. Hardness of approximate optima in lattices, codes, and linear systems. *Journal of Computer and System Sciences*, 54(2):317–331, 1997.

[5] J. Bruck and M. Naor. The hardness of decoding linear codes with preprocessing. *IEEE Transactions on Information Theory*, 36(2):381–385, 1990.

[6] I. Dinur, V. Guruswami, S. Khot, and O. Regev. A new multilayered pcp and the hardness of hypergraph vertex cover. *SIAM J. Comput.*, 34(5):1129–1146, 2005.

[7] I. Dinur, G. Kindler, R. Raz, and S. Safra. Approximating cvp to within almost-polynomial factors is np-hard. *Combinatorica*, 23(2):205–243, 2003.

[8] U. Feige and D. Micciancio. The inapproximability of lattice and coding problems with preprocessing. *Journal of Computer and System Sciences*, 69(1):45–67, 2004.

[9] C. Gauss. *Disquisitiones arithmeticae*. Yale Univ. Pres, 1966. English translation by A. A. Clarke.

[10] R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the ACM Symposium on the Theory of Computing*, number 15, pages 193–206, 1983.

[11] S. Khot. Hardness results for coloring 3-colorable 3-uniform hypergraphs. In *Annual Symposium on Foundations of Computer Science*, number 43, pages 23–32, 2002.

[12] J. Lagarias, H. Lenstra, and C. Schnorr. Korkine-Zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica*, 10:333–348, 1990.

[13] J. Lagarias and A. Odlyzko. Solving low-density subset sum problems. *Journal of the ACM*, 32(1):229–246, 1985.

[14] A. Lenstra, H. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:513–534, 1982.

[15] H. Lenstra. Integer programming with a fixed number of variables. Technical Report 81-03, Univ. of Amsterdam, Amsterdam, 1981.

[16] D. Micciancio. The hardness of the closest vector problem with preprocessing. *IEEE Transactions on Information Theory*, 47:1212–1215, 2001.

[17] D. Micciancio and S. Goldwasser. *Complexity of lattice problems: A cryptographic perspective*, volume 671. Kluwer Academic Publishers, 2002.

[18] R. Raz. A parallel repetition theorem. *SIAM Journal of Computing*, 27(3):763–803, 1998.

[19] O. Regev. Improved inapproximability of lattice and coding problems with preprocessing. *IEEE Transactions on Information Theory*, 50(9):2031–2037, 2004.

[20] C. Schnorr. A hierarchy of polynomial-time basis reduction algorithms. In *Proceedings of Conference on Algorithms*, pages 375–386, Péecs, Hungary, 1985.