On Bounded Distance Decoding for General Lattices

Yi-Kai Liu* Vadim Lyubashevsky** Daniele Micciancio**

University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093-0404, USA
{y9liu,vlyubash,daniele}@cs.ucsd.edu

Abstract. A central problem in the algorithmic study of lattices is the *closest vector problem*: given a lattice $\mathcal{L}$ represented by some basis, and a target point $\boldsymbol{y}$, find the lattice point closest to $\boldsymbol{y}$. *Bounded Distance Decoding* is a variant of this problem in which the target is guaranteed to be close to the lattice, relative to the minimum distance $\lambda_1(\mathcal{L})$ of the lattice. Specifically, in the α-Bounded Distance Decoding problem (α-**BDD**), we are given a lattice $\mathcal{L}$ and a vector $\boldsymbol{y}$ (within distance $\alpha \cdot \lambda_1(\mathcal{L})$ from the lattice), and we are asked to find a lattice point $\boldsymbol{x} \in \mathcal{L}$ within distance $\alpha \cdot \lambda_1(\mathcal{L})$ from the target. In coding theory, the lattice points correspond to codewords, and the target points correspond to lattice points being perturbed by noise vectors. Since in coding theory the lattice is usually fixed, we may "pre-process" it before receiving any targets, to make the subsequent decoding faster. This leads us to consider α-**BDD** with pre-processing. We show how a recent technique of Aharonov and Regev [2] can be used to solve α-**BDD** with pre-processing in polynomial time for $\alpha = O\left(\sqrt{(\log n)/n}\right)$. This improves upon the previously best known algorithm due to Klein [13] which solved the problem for $\alpha = O(1/n)$. We also establish hardness results for α-**BDD** and α-**BDD** with pre-processing, as well as generalize our results to other ℓ_p norms.

1 Introduction

A lattice is the set of intersection points of a regular (but not necessarily orthogonal) n-dimensional grid. One of the most central problems in the algorithmic study of lattices is the *closest vector problem*: given a lattice $\mathcal{L}$ (typically represented by a basis, see Section 2 for details), and a target point $\boldsymbol{y}$, find the lattice point closest to $\boldsymbol{y}$. Beside having numerous applications in theoretical computer science, lattices are a central object in coding theory [1, 8]. In this setting, lattice points represent codewords, and the target point $\boldsymbol{y}$ represents a perturbed codeword (encoding a message being transmitted). In this scenario, the closest vector problem corresponds exactly to the maximum likelyhood decoding problem for white Gaussian noise channels. The closest vector problem is NP-hard to solve

* Supported by an ARO/NSA Quantum Computing Graduate Research Fellowship
** Supported by NSF CAREER 0093029 and NSF ITR 0313241

even approximately for any constant [5] and some sub-polynomial [9] approximation factors. On the positive side, the best general approximation algorithm to solve the closest vector problem in (random) polynomial time achieves only approximation factors almost exponential in the dimension of the lattice [3]. We remark that there are two fundamental differences between the closest vector problem as typically studied in complexity theory and coding theory:

1. In complexity theory, the lattice is considered as part of the input to the problem, while in coding theory the lattice (defining the error correcting code) is usually fixed once and for all.
2. In complexity theory the target point can be arbitrarily far from the lattice, while in coding theory it is usually assumed that the distance of the target from the lattice is less than half the minimum distance between lattice points.

The first issue has been recently addressed [15, 11, 18, 4], considering a version of the **CVP** *with pre-processing* (**CVPP**). In **CVPP**, the lattice is fixed and can be arbitrarily preprocessed, and the complexity of the **CVP** algorithm is measured without taking pre-processing time into account. In the sequence of papers [15, 11, 18, 4] it is shown that there are lattices such that **CVPP** is NP-hard to solve exactly, or even approximate within any constant factor.[1]

The second issue is equally important, but has so far received far less attention. The relevance of the second issue stems from the fact that the amount of error (i.e., the distance of the target from the lattice) depends on the properties of the communication channel, and it is usually known to the code designer. This allows the code designer to choose a lattice code whose minimum distance is, in some respect, "large" compared to the maximum error.

The variant of the closest vector problem where the target is guaranteed to be close to the lattice relative to the minimum distance, $\lambda_1(\mathcal{L})$, of the lattice is called the *Bounded Distance Decoding* problem (**BDD**) [23]. Specifically, in the α-Bounded Distance Decoding problem (α-**BDD**), we are given a lattice $\mathcal{L}$ and a vector $\boldsymbol{y}$ (within distance $\alpha \cdot \lambda_1(\mathcal{L})$ from the lattice), and are asked to find a lattice point $\boldsymbol{x} \in \mathcal{L}$ within distance $\alpha \cdot \lambda_1(\mathcal{L})$ from the target. Typically $\alpha = 1/2$, but other values of α can be interesting as well, as some decoding algorithms may not work up to the unique decoding radius $\lambda_1/2$, while in other cases even for $\alpha > 1/2$ it may be possible to come up with a relatively short (i.e., polynomially long) list of candidate lattice points. (The latter is called the "list decoding problem", and it has received an great deal of attention lately in the context of codes over finite fields. We are not aware of any result of the same kind for lattice codes, but the problem is certainly very interesting and natural.)

Our contribution. In this paper we investigate the bounded distance decoding problem α-**BDD** (with pre-processing), and prove both algorithmic and computational hardness results about this problem. On the algorithmic side, we

[1] The factor achieved by a **CVP** approximation algorithm is defined as the ratio between the distance (from the target) of lattice point output by the algorithm, over the distance of the optimal solution.

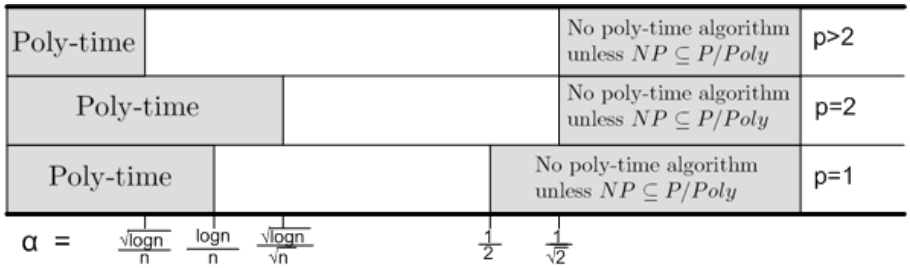

Fig. 1. Complexity of α-BDD with pre-processing for various l_p norms

show that α-**BDD** with pre-processing can be solved in polynomial time for $\alpha = O(\sqrt{\log n/n})$. Previously, the problem was known to be polynomial time solvable only for factors $O(1/n)$ [6, 13]. On the computational hardness side, we show (under standard complexity assumptions) that the α-**BDD** problem cannot be solved in polynomial time (even in its pre-processing variant) for any constant factor $\alpha > 1/\sqrt{2}$. Specifically, for the α-**BDD** problem with pre-processing, any polynomial time solution would imply that NP is contained in P/poly. We also adapt some of our results to other ℓ_p norms (see figure 1).

Related work. Our work is closely related, and builds upon, previous work of Aharonov and Regev [2] (for the algorithmic results) and Micciancio [16] (for the hardness results), as well as previous algorithms [6, 13] and NP-hardness results [4] for the **CVP** problem with pre-processing.

The well known *nearest plane algorithm* [6] (together with the bounds in [14]) yields a polynomial time solution to α-**BDD** with pre-processing for $\alpha = 2/n$. That solution had been subsequently improved to any $\alpha = O(1/n)$ in [13], which was still the best general solution prior to our work. Our algorithm uses a technique developed by Aharonov and Regev [2], and it is closely related to their work. In [2], Aharonov and Regev show how to compute (with pre-processing) a function $f(\boldsymbol{x})$ which approximates the distance of a target point $\boldsymbol{x}$ from a lattice within a factor of $O(\sqrt{n/\log n})$. Unfortunately, being able to approximate the distance of a target point from the lattice does not, in general, allow us to find lattice points which are close to the target.[2] So, the Aharonov and Regev's **CVP** pre-processing algorithm does not directly yield a solution to α-**BDD**. In this paper we observe that, for a certain region of $\mathbb{R}^n$ close to the lattice, the function f of Aharonov and Regev allows us to distinguish which of two points in $\mathbb{R}^n$ is

[2] Technically, there is no known approximation preserving reduction from the problem of approximating **CVP** in its search version (i.e., finding an approximately closest lattice point), to approximating **CVP** in its decision or distance estimation version (i.e., determining if a target point is close or far away from a lattice). (See [17, Chapter 3].) Such a reduction trivially exists for the exact versions of **CVP** and for small (subpolynomial) approximation factors by NP-hardness [5, 9], but finding such a reduction for polynomial approximation factors is an open problem.

closer to the lattice. So, by adding a noise vector to our target point, we can generate a nearby point which is closer to the lattice and verify that it actually is closer. Our α-**BDD** algorithm performs a "guided" walk starting from the target and moving closer and closer to the lattice, until we get within distance $O(\lambda_1/n)$ from it, at which point some other known algorithm for α-**BDD** (i.e. [6, 13]) can be used.

On the complexity front, no hardness result was known prior to our work because, interestingly, all known NP-hardness results for **CVP** (with or without pre-processing) [5, 9, 15, 11, 18, 4] employed lattices with very small minimum distance. We prove our hardness results using a technique of Micciancio [16] to embed a lattice $\mathcal{L}$ and target $\boldsymbol{y}$ into a higher dimensional space in such a way that the minimum distance of the lattice increases, without at the same time substantially increasing the distance of the target from the lattice. The hardness results for ℓ_p norms for $p > 2$ are obtained by an application of a recent technique of Regev and Rosen [19] to our result for the ℓ_2 norm.

Questions regarding the complexity of α-**BDD** (and related problems) had been previously considered in the setting of linear codes over finite fields. For example, Vardy [23] conjectured the problem to be NP-hard for $\alpha = 1/2$, and for the closely related *relatively near codeword* problem (RNC) NP-hardness results for any $\alpha > 1/2$ were proven by Dumer, Micciancio and Sudan [10], and later adapted by Regev [18] to the pre-processing variant of the problem.

2 Preliminaries

2.1 Lattices

The set of all integer combinations of vectors $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ defines a *lattice* $\mathcal{L}(\boldsymbol{B})$ in $\mathbb{R}^n$. $\boldsymbol{B}$ is said to form a basis of $\mathcal{L}(\boldsymbol{B})$ (when the basis is clear from context, we may write $\mathcal{L}$ instead of $\mathcal{L}(\boldsymbol{B})$). For any basis $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$, the *Gram-Schmidt* basis is denoted by $\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*$ where $\boldsymbol{b}_i^*$ is the component of $\boldsymbol{b}_i$ which is orthogonal to the vector space formed by $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{i-1}$. We denote by $\lambda_1(\mathcal{L})$ the length of the shortest vector of $\mathcal{L}$ (equivalently, the minimum distance of $\mathcal{L}$).

Lemma 1. *For every n-dimensional lattice $\mathcal{L}$, there exists a basis $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ such that* $\min_i \|b_i^*\| \geq \lambda_1(\mathcal{L})/n$.

Lattice Problems Given a lattice basis $\mathbf{B}$, vector $\boldsymbol{x}$ and a real t, the decisional version of the closest vector problem (**CVP**) asks whether $dist(\mathcal{L}(\boldsymbol{B}), \boldsymbol{x}) \leq t$. The approximate version of decisional **CVP** can be formulated as a promise problem $\mathbf{GapCVP}_\gamma$. Given a lattice basis $\mathbf{B}$, vector $\boldsymbol{x}$ and a real t, an algorithm for $\mathbf{GapCVP}_\gamma$ should answer "YES" if $dist(\mathcal{L}(\boldsymbol{B}), \boldsymbol{x}) \leq t$ and "NO" if $dist(\mathcal{L}(\boldsymbol{B}), \boldsymbol{x}) > \gamma t$. For all values in between, any answer is acceptable.

An algorithm that solves $\mathbf{GapCVP}_\gamma$ with pre-processing works in two steps. First, it is given a basis $\boldsymbol{B}$. The algorithm then outputs an advice string A. The time that the algorithm expends in obtaining A does not count towards

its running time. Then, it is given a vector $\boldsymbol{x}$ and a real t. With the ability to use the advice string A, it should answer "YES" if $dist(\mathcal{L}(\boldsymbol{B}), \boldsymbol{x}) \leq t$ and "NO" if $dist(\mathcal{L}(\boldsymbol{B}), \boldsymbol{x}) > \gamma t$. For all values in between, any answer is acceptable. Alekhnovich, et. al. [4] showed that the $\mathbf{GapCVP}_\gamma$ with pre-processing problem is NP-hard for any constant γ. Aharonov and Regev [2] showed a polynomial algorithm for this problem for $\gamma = O\left(\sqrt{n/\log n}\right)$.

In the alpha-Bounded Distance Decoding problem (α-**BDD**), we are given a lattice basis $\mathbf{B}$ and a vector $\boldsymbol{x}$ and are asked to find a lattice vector $\mathbf{y} \in \mathcal{L}(\mathbf{B})$ such that $dist(\mathbf{x}, \mathbf{y}) \leq \alpha \cdot \lambda_1(\mathcal{L}(\mathbf{B}))$ (if such a vector exists). In Section 4 we show that this problem is NP-hard for $\alpha > 1/\sqrt{2}$.

As for $\mathbf{GapCVP}_\gamma$ with pre-processing, an algorithm for α-**BDD** with pre-processing works in two steps. First, it is given a basis $\boldsymbol{B}$. The algorithm then outputs an advice string A. The time that the algorithm expends in obtaining A does not count towards its running time. Then, it is given a vector $\boldsymbol{x}$. With the ability to use the advice string A, it should find a lattice vector $\mathbf{y} \in \mathcal{L}(\mathbf{B})$ (if one exists) such that $dist(\mathbf{x}, \mathbf{y}) \leq \alpha \cdot \lambda_1(\mathcal{L}(\mathbf{B}))$. In Section 3, we provide a polynomial algorithm for values of $\alpha = O\left(\sqrt{\log n/n}\right)$ and in Section 4, we show that if a polynomial algorithm exists for $\alpha > 1/\sqrt{2}$, then $NP \subseteq P/poly$.

We define one last lattice problem. This problem will be useful in Section 4 to prove the hardness results. Given a full rank n-dimensional lattice basis $\mathbf{B}$, vector $\boldsymbol{x}$ and a real t, an algorithm for $\mathbf{GapCVP}'_\gamma$ should answer "YES" if $\exists \boldsymbol{z} \in \{0,1\}^n$ such that $dist(\boldsymbol{Bz}, \boldsymbol{x}) \leq t$ and "NO" if $dist(\mathcal{L}(\boldsymbol{B}), s\boldsymbol{x}) > \gamma t$ for all $s \in \mathbb{Z} \setminus \{0\}$. In all other cases, any answer is acceptable. Arora, et. al. [5] showed this problem to be NP-hard for all constants $\gamma \geq 1$.

2.2 Gaussian Functions on Lattices

In [2], Aharonov and Regev considered the following function f on any $\boldsymbol{x} \in \mathbb{R}^n$,

$$f(\boldsymbol{x}) = \frac{\sum_{\boldsymbol{y} \in \mathcal{L}} e^{-\pi \|\boldsymbol{x}-\boldsymbol{y}\|^2}}{\sum_{\boldsymbol{y} \in \mathcal{L}} e^{-\pi \|\boldsymbol{y}\|^2}}$$

and showed that with polynomial advice, one can estimate its value on exponentially many points in the quotient group $\mathbb{R}^n/\mathcal{L}$.

Lemma 2. *([2, Lemma 1.3]) Let $\mathcal{L}$ be an n-dimensional lattice. For any set S consisting of $2^{poly(n)}$ points in the group $\mathbb{R}^n/\mathcal{L}$ and any constant $c > 0$, there exists an advice string of size $poly(n)$ that allows one to evaluate the function f in polynomial time with error at most n^{-c} on every point in S.*

A property of the function f is that if $\boldsymbol{x} \in \mathbb{R}^n$ has only one lattice point close to it, then the value of f will almost be entirely determined by the distance of $\boldsymbol{x}$ from this point.

Lemma 3. *Let $\mathcal{L}$ be an n-dimensional lattice whose shortest vector has length greater than $\sqrt{\frac{n}{2\pi}}$ and $\boldsymbol{x} \in \mathbb{R}^n$. If all points in $\mathcal{L}$ other than $\boldsymbol{y}'$ are at a distance more than $\sqrt{\frac{n}{2\pi}}$ from $\boldsymbol{x}$, then $f(\boldsymbol{x}) = e^{-\pi \|\boldsymbol{x}-\boldsymbol{y}'\|^2} \pm 2^{-\Omega(n)}$.*

The next lemma shows that the function f is very sensitive at a distance less than $\sqrt{\log n}$ away from the lattice when the length of the shortest vector of the lattice is greater than $\sqrt{\frac{n}{2\pi}}$. Thus, if we are at a point $\boldsymbol{x} \in \mathbb{R}^n$ and move closer to the lattice, there will be a noticeable change in the value of the function.

Lemma 4. *Let $\mathcal{L}$ be an n-dimensional lattice whose shortest vector has length greater than $\sqrt{\frac{n}{2\pi}}$, and let $\boldsymbol{y} \in \mathcal{L}$. Suppose $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^n$ are points such that $\|\boldsymbol{x} - \boldsymbol{y}\| = D$, $\|\boldsymbol{x}' - \boldsymbol{y}\| \leq (D + n^{-4})\sqrt{1 - \frac{1}{n}}$ where $\frac{1}{n} \leq D \leq \sqrt{\log n}$, and for all $\boldsymbol{y}' \in \mathcal{L} \setminus \{\boldsymbol{y}\}$, $\|\boldsymbol{x} - \boldsymbol{y}'\|, \|\boldsymbol{x}' - \boldsymbol{y}'\| > \sqrt{\frac{n}{2\pi}}$. Then $f(\boldsymbol{x}') - f(\boldsymbol{x}) > n^{-6.5}$.*

The below lemma is a partial converse of lemma 4.

Lemma 5. *Let $\mathcal{L}$ be an n-dimensional lattice whose shortest vector has length greater than $\sqrt{\frac{n}{2\pi}}$, and let $\boldsymbol{y} \in \mathcal{L}$. Suppose $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^n$ are points such that $\|\boldsymbol{x} - \boldsymbol{y}\|, \|\boldsymbol{x}' - \boldsymbol{y}\| \leq \sqrt{\log n}$, and for all $\boldsymbol{y}' \in \mathcal{L} \setminus \{\boldsymbol{y}\}$, $\|\boldsymbol{x} - \boldsymbol{y}'\|, \|\boldsymbol{x}' - \boldsymbol{y}'\| > \sqrt{\frac{n}{2\pi}}$, and $f(\boldsymbol{x}') - f(\boldsymbol{x}) > n^{-7.1}$. Then $\|\boldsymbol{x}' - \boldsymbol{y}\| \leq \sqrt{1 - 1/n^8}\|\boldsymbol{x} - \boldsymbol{y}\|$.*

3 Finding the Closest Lattice Vector

The following theorem was proved by Klein in [13]:

Theorem 1. *There is an algorithm that, when given an n-dimensional lattice $\mathcal{L}$ represented by basis vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$, and a target $\boldsymbol{x} \in \mathbb{R}^n$ that's at distance D away from $\mathcal{L}$, will find the closest lattice vector to $\boldsymbol{x}$, in time $n^{D^2/\min_i \|b_i^*\|^2}$.*

Combining Theorem 1 with Lemma 1, implies that whenever the target point is within $O(\lambda_1(\mathcal{L})/n)$ of the lattice, there is a basis that can be used as advice to find the nearest lattice point in polynomial time.

Without loss of generality, we may assume that our lattice is scaled such that $\lambda_1(\mathcal{L}) > \sqrt{n}$. If the target that we're given is within distance $1/\sqrt{n}$ of the lattice (and thus within distance $\lambda_1(\mathcal{L})/n$ of the lattice), we can just find the closest vector by applying Theorem 1. If the target point is not that close to the lattice but is still within $\sqrt{\log n}$ of it, we will find another point closer to the lattice that is also close to the target point. We will proceed in like manner by finding points closer to the lattice until we get within $1/\sqrt{n}$ of the lattice at which time we will apply the algorithm in Theorem 1.

Theorem 2. *Let $\mathcal{L}$ be a lattice with shortest vector at least $\sqrt{n}$ and let A be the polynomial size advice string as in Lemma 2 that allows us to approximate the function f with error at most n^{-8}. Then there is a polynomial time algorithm using advice A that, when given a point $\boldsymbol{x} \in \mathbb{R}^n$ that is within distance $\sqrt{\log n}$ of a lattice point $\boldsymbol{y} \in \mathcal{L}$, will find a point $\boldsymbol{x}'$ that is within distance $1/\sqrt{n}$ of $\boldsymbol{y}$.*

Proof. Let f_A be the function that uses advice A and approximates f to within n^{-8} on exponentially many points (as in Lemma 2). To be precise, we would need to put a grid on $\mathbb{R}^n$ everywhere within $\sqrt{\log n}$ of the lattice, and only be able to approximate the function f at the intersection points of the grid. Since both f

and f_A are symmetric with respect to the lattice, it will suffice to consider only grid points within distance $\sqrt{\log n}$ of the origin. Within this region we can make the grid very fine (i.e the diagonal of a grid square can be n^{-c} for any constant c), which is good enough for our purposes. For simplicity, we will assume that for all $\boldsymbol{x} \in \mathbb{R}^n$, where $\boldsymbol{x}$ is within $\sqrt{\log n}$ of the lattice, $|f_A(\boldsymbol{x}) - f(\boldsymbol{x})| < n^{-8}$. Consider the following algorithm: (in the algorithm $\boldsymbol{u}_i$ is the ith standard unit vector)

GetCloser$(\boldsymbol{x}, f_A)$
  while$(f_A(\boldsymbol{x}) < (e^{-\pi/n} + n^{-8}))$
    compute $D_A = \sqrt{\frac{-\log f_A(\boldsymbol{x})}{\pi}}$
    construct set $S = \{\boldsymbol{x} - j(D_A/\sqrt{n})\boldsymbol{u}_i \ : i \in \{1, \ldots, n\}, j \in \{-1, 1\}\}$
    set $\boldsymbol{x} \leftarrow \operatorname*{argmax}_{\boldsymbol{x}' \in S} f_A(\boldsymbol{x}')$
  return $\boldsymbol{x}$

First we will show that at each iteration of the while loop, D_A is very close to the correct distance D between $\boldsymbol{x}$ and the lattice point $\boldsymbol{y}$. Then we will show that one of the $2n$ elements of the set S is a vector that is within distance $(D + n^{-4})\sqrt{1 - 1/n}$ of the lattice. This will imply that the element $\boldsymbol{x}' \in S$ for which the value $f_A(\boldsymbol{x}')$ is the largest is within distance $D\sqrt{1 - 1/n^8}$ of the lattice. And by continuing to loop, we will eventually get within $1/\sqrt{n}$ of the lattice point.

Lemma 6. *At every step of the algorithm,* $|D_A - \|\boldsymbol{x} - \boldsymbol{y}\|| \leq n^{-4}$.

Lemma 7. *Let* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *be points in* $\mathbb{R}^n$ *such that* $\|\boldsymbol{x} - \boldsymbol{y}\| = D$ *and* $n > 6$. *For any* $c > 0$ *and* $D_A \in [D - c, D + c]$, *the set*

$$S = \{\boldsymbol{x} - j(D_A/\sqrt{n})\boldsymbol{u}_i \ : i \in \{1, \ldots, n\}, j \in \{-1, 1\}\}$$

contains a vector $\boldsymbol{x}'$ *such that* $\|\boldsymbol{x}' - \boldsymbol{y}\| < (D + c)\sqrt{1 - 1/n}$.

Proof. Without loss of generality, assume that $\boldsymbol{y} = \boldsymbol{0}$. Then $D^2 = \|\boldsymbol{x}\|^2 = \sum_i x_i^2 > (D - c)^2$. Thus, there must exist an i, such that $|x_i| \geq (D - c)/\sqrt{n}$. Let $j \in \{-1, 1\}$ have the same as the sign of x_i. Then the vector $\boldsymbol{x}' = \boldsymbol{x} - j\frac{D_A}{\sqrt{n}}\boldsymbol{u}_i$ is in S and

$$\|\boldsymbol{x}'\|^2 = \|(x_1, \ldots, x_{i-1}, x_i - j\tfrac{D_A}{\sqrt{n}}, x_{i+1}, \ldots, x_n)\|^2 = D^2 - 2|x_i|\tfrac{D_A}{\sqrt{n}} + \tfrac{D_A^2}{n}$$

Since $|x_i| \geq (D - c)/\sqrt{n}$ and $(D - c) \leq D_A \leq (D + c)$, we have

$$\|\boldsymbol{x}'\|^2 \leq D^2 - 2\tfrac{(D-c)^2}{n} + \tfrac{(D+c)^2}{n} < (D + c)^2(1 - 1/n)$$

where the last inequality holds for $n > 6$.

Lemma 6 says that the value of D_A that we calculate using f_A is within n^{-4} of the actual distance D. Lemma 7 says that the set S contains a point that is

a distance at most $(D + n^{-4})\sqrt{1 - 1/n}$ away from the lattice. By Lemma 4, we know that if $\|\boldsymbol{x}' - \boldsymbol{y}\| \leq (\|\boldsymbol{x} - \boldsymbol{y}\| + n^{-4})\sqrt{1 - 1/n}$, then $f(\boldsymbol{x}') - f(\boldsymbol{x}) > n^{-6.5}$, which by the triangular inequality implies that $f_A(\boldsymbol{x}') - f_A(\boldsymbol{x}) > n^{-6.5} - 2n^{-8} > n^{-7}$. Thus, there is a point $\boldsymbol{x}' \in S$ such that $f_A(\boldsymbol{x}') - f_A(\boldsymbol{x}) > n^{-7}$. By the triangular inequality, this implies that $f(\boldsymbol{x}') - f(\boldsymbol{x}) > n^{-7} - 2n^{-8} > n^{-7.1}$, which by Lemma 5 means that $\|\boldsymbol{x}' - \boldsymbol{y}\| \leq \sqrt{1 - 1/n^8}\|\boldsymbol{x} - \boldsymbol{y}\|$. So at every step, we are guaranteed to be getting closer to the lattice by a factor of $\sqrt{1 - 1/n^8}$. The loop ends once $f_A(\boldsymbol{x}) > (e^{-\pi/n} + n^{-8})$, which means that $f(\boldsymbol{x}) > e^{-\pi/n}$. Since at every step of the loop we made sure we were getting closer to the lattice, it must be that the point $\boldsymbol{x}$ is within distance $\sqrt{\log n}$ of the lattice, and thus by Lemma 3, $f(\boldsymbol{x}) = e^{-\pi\|\boldsymbol{x}-\boldsymbol{y}'\|} \pm 2^{-\Omega(n)}$. Since $f(\boldsymbol{x}) > e^{-\pi/n}$, it must be that $\|\boldsymbol{x} - \boldsymbol{y}\| < 1/\sqrt{n} + 2^{-\Omega(n)}$. To calculate the running time of the algorithm, we note that at every step of the loop, the value of the function f_A increases by n^{-7} with constant probability. Since the starting value of f_A is greater than $e^{-\pi \log n} - n^{-8} \approx n^{-\pi}$, the running time of the algorithm will be $O(n^5)$ multiplied by the time it takes to get a closer point $\boldsymbol{x}'$ (which is $O(n)$), multiplied by the time it takes to evaluate f_A (which is $poly(n)$).

3.1 Other ℓ_p Norms

Our algorithm can be adapted to solve α-**BDD** for the ℓ_1 norm, with $\alpha = \log n/n$. (The naive approach, using the ℓ_2 algorithm to solve the problem, works for $\alpha = \sqrt{\log n}/n$.) We believe our algorithm should also work for ℓ_p norms, $1 < p < 2$, with $\alpha = \sqrt[p]{\log n/n}$; but we are unable to give a rigorous proof in this case. When $p > 2$, however, the algorithm no longer works, and new ideas are probably necessary.

First, we recall how the technique of Aharonov and Regev [2] works. For any lattice $\mathcal{L}$, we define a periodic function $F(\boldsymbol{x}) = \sum_{\boldsymbol{y}\in\mathcal{L}} f(\boldsymbol{x} - \boldsymbol{y})$. $F(\boldsymbol{x})$ can be written in terms of its Fourier coefficients $\hat{F}(\boldsymbol{w})$, $\boldsymbol{w} \in \mathcal{L}^*$, as follows: $F(\boldsymbol{x}) = \sum_{w\in\mathcal{L}^*} \hat{F}(\boldsymbol{w})e^{2\pi i\langle \boldsymbol{w},\boldsymbol{x}\rangle}$. Note that $\hat{F}(\boldsymbol{w}) = (1/\det(\mathcal{L}))\hat{f}(\boldsymbol{w})$, where $\hat{f}$ is the Fourier transform of f. Provided that the Fourier coefficients are non-negative, we can view them as a probability distribution over the dual lattice; and given some "advice" consisting of points in the dual lattice sampled according to this distribution, we can approximately compute $F(\boldsymbol{x})$.

For general ℓ_p norms, it is natural to use the function $f(\boldsymbol{x}) = e^{-\|\boldsymbol{x}\|_p^p}$. Note that f is a product of one-dimensional functions, $f(\boldsymbol{x}) = g(x_1)\cdots g(x_n)$, where $g(x) = e^{-|x|^p}$. Hence its Fourier transform is $\hat{f}(\boldsymbol{k}) = \hat{g}(k_1)\cdots\hat{g}(k_n)$. In order to use the technique of Aharonov and Regev, we would like to show that $\hat{f}(\boldsymbol{k}) \geq 0$; functions f with this property are called "positive definite" [22].

In addition, both our work and [2] make use of a technical lemma due to Banaszczyk [7] (see also [21]). We need to prove a generalization of this result for ℓ_p norms. (It is this step that determines the ratio $\alpha = \sqrt[p]{\log n/n}$.) It turns out that this requires us to show that $\hat{f}(\boldsymbol{k})$ is a non-increasing function of $\|\boldsymbol{k}\|$.

Algorithm for the ℓ_1 norm. In this case, $g(x) = e^{-|x|}$, and $\hat{g}(k) = 2/(1+(2\pi k)^2)$. So it is easy to see that $\hat{f}(\boldsymbol{k}) \geq 0$ and is a decreasing function of $\|\boldsymbol{k}\|$. Our algorithm then works with minor modifications.

Algorithm for the ℓ_p norms with $1 < p < 2$. It is known that $\hat{g}(k) \geq 0$ [20]. Numerical investigation suggests that $\hat{g}(k)$ is a decreasing function of $|k|$, but we have not been able to prove this analytically. If this is true, then $\hat{f}(\boldsymbol{k})$ has the required properties, and our algorithm works.

What goes wrong for ℓ_p norms with $p > 2$. In this case, it is not true that $\hat{g}(k) \geq 0$ [20]. Hence $\hat{f}(\boldsymbol{k})$ can be negative for some $\boldsymbol{k}$. We can still interpret $|\hat{F}(\boldsymbol{w})|$ as a probability distribution; however, note that $F(\mathbf{0}) < \sum_w |\hat{F}(\boldsymbol{w})|$. Heuristic arguments suggest that when we normalize the probability distribution, $F(\mathbf{0})$ will be exponentially small, so our algorithm breaks down. (For instance, it is easy to see that $f(\mathbf{0}) = 2^{-\Omega(n)} \int_{\mathbb{R}^n} |\hat{f}(\boldsymbol{k})| d\boldsymbol{k}$.)

4 Hardness Results

In this section we prove the hardness of the α-**BDD** and α-**BDD** with pre-processing problems. The proofs are by reduction from a version of the closest vector problem $\mathbf{GapCVP}'_\gamma$, and are based on techniques developed by Micciancio [16] to prove that the shortest vector problem is NP-hard to approximate within any constant factor smaller than $\sqrt{2}$. In fact, the $\alpha > 1/\sqrt{2}$ requirement in our proof comes from exactly the same limiting factor that makes Micciancio's proof [16] work only for approximation ratios bounded by $\sqrt{2}$. It is an interesting open question whether employing techniques used in the proof of stronger in-approximability results for **SVP** [12] it is possible to improve our NP-hardness result for α-**BDD** to $\alpha = 1/2$ or maybe even $\alpha = \Omega(1)$ or $\alpha = o(1)$.

Theorem 3. *For any ℓ_p norm ($p \geq 1$) and $\alpha > 1/\sqrt[p]{2}$ and $\gamma > 1/\sqrt[p]{1-\alpha^{-p}/2}$, there is a probabilistic polynomial time reduction from $\mathbf{GapCVP}'_\gamma$ to α-**BDD**. Moreover, the reduction has the following two additional properties:*

1. *On input $\mathbf{GapCVP}'_\gamma$ instance $(\mathbf{B}, \boldsymbol{y}, t)$, the reduction makes a single call to the α-**BDD** oracle on a lattice that depends only on $\mathbf{B}$ and t (but not $\boldsymbol{y}$).*
2. *Randomness is only used in the construction of the α-**BDD** lattice, and with high probability the constructed lattice is good for all target points $\boldsymbol{y}$.*

*In particular, there is a probabilistic polynomial time reduction from $\mathbf{GapCVP}'_\gamma$ with pre-processing, to α-**BDD** with pre-processing such that randomness is only used during the pre-processing stage (and therefore can be equivalently be replaced by a non-uniform advice).*

Proof: Throughout the proof, $\|\cdot\|$ always denotes the ℓ_p norm. Let $(\mathbf{B}, \boldsymbol{y}, t)$ be a $\mathbf{GapCVP}'_\gamma$ instance. We want to determine if $\boldsymbol{y}$ is close or far from the lattice. The idea is to use the α-**BDD** oracle to find a lattice vector close to

$\boldsymbol{y}$. If the oracle fails or returns a vector far away from the target $\boldsymbol{y}$, we would like to conclude that $\boldsymbol{y}$ is indeed far from the lattice. The problem is that theα-**BDD** oracle is guaranteed to work only when the distance of $\boldsymbol{y}$ is small, relative to the minimum distance of the lattice $\lambda_1(\mathbf{B})$. Following [16], we address this problem by embedding $\mathbf{B}$ and $\boldsymbol{y}$ in a higher dimensional space, with the effect of increasing the minimum distance of the lattice, without substantially increasing the distance of the target from the lattice. The proof in [16] is based on the construction of a lattice basis $\mathbf{L}$ with some very special properties as described in the following lemma.

Lemma 8. *For any ℓ_p norm ($p \geq 1$) and any constant $\sigma \in [1, \sqrt[p]{2})$ there exists a (probabilistic) algorithm that on input $k \in \mathbb{Z}^+$ outputs, in $k^{O(1)}$ time, two positive integers $m, r \in \mathbb{Z}^+$, a lattice basis $\mathbf{L} \in \mathbb{Z}^{(m+1)\times m}$, a vector $\boldsymbol{s} \in \mathbb{Z}^{m+1}$, and a linear integer transformation $\mathbf{T} \in \mathbb{Z}^{k\times m}$ such that*

1. *$\lambda(\mathcal{L}(\mathbf{L})) > \sigma \cdot r$,*
2. *with probability at least $1 - 1/n^{O(k)}$ for all $\boldsymbol{x} \in \{0,1\}^k$ there exists a $\boldsymbol{z} \in \mathbb{Z}^m$ such that $\mathbf{T}\boldsymbol{z} = \boldsymbol{x}$ and $\|\mathbf{L}\boldsymbol{z} - \boldsymbol{s}\| \leq r$.*

Informally the lemma states that lattice $\mathcal{L}(\mathbf{L})$ contains an unusually dense cluster of lattice points around the center $\boldsymbol{s}$.

Our reduction first invokes Lemma 8 with $\sigma = 1/(\alpha\sqrt[p]{1-\gamma^{-p}})$ and $k = n$ to obtain $\mathbf{L}$, $\boldsymbol{s}$ and r. (Notice that under the assumptions $\alpha > 1/\sqrt[p]{2}$ and $\gamma > 1/\sqrt[p]{1-\alpha^{-p}/2}$, we get $\sigma < \sqrt[p]{2}$ as required by Lemma 8.) Then, it combines $(\mathbf{B}, \boldsymbol{y})$ and $(\mathbf{L}, \boldsymbol{s})$ to define a α-**BDD** instance

$$\mathbf{B}' = \begin{bmatrix} \mathbf{BT} & \mathbf{0} \\ \beta\mathbf{L} & \mathbf{0} \\ \mathbf{0} & \beta\sigma r \end{bmatrix} \qquad \boldsymbol{y}' = \begin{bmatrix} \boldsymbol{y} \\ \beta\boldsymbol{s} \\ 0 \end{bmatrix},$$

where β is an appropriate normalization factor to be chosen. Finally, the α-**BDD** oracle is invoked on input $(\mathbf{B}', \boldsymbol{y}')$. The $\mathbf{GapCVP}'_\gamma$ instance is accepted if and only if the α-**BDD** oracle returns a lattice point $\boldsymbol{v} = [(\mathbf{BT}\boldsymbol{z})^T, (\mathbf{L}\boldsymbol{z})^T, (\beta\sigma r)\cdot z]^T$ (where $\boldsymbol{z} \in \mathbb{Z}^m$ and $z \in \mathbb{Z}$) such that $\mathbf{BT}\boldsymbol{z}$ is within distance γt from target $\boldsymbol{y}$.

We now prove that the reduction is correct. Notice that if the reduction accepts, then there is a lattice vector $\mathbf{BT}\boldsymbol{z}$ within distance γt from the target $\boldsymbol{y}$, so $(\mathbf{B}, \boldsymbol{y}, t)$ is certainly not a NO instance of $\mathbf{GapCVP}'_\gamma$, and YES is a valid answer for the reduction. (Remember that when an instance does not satisfy the promise, any answer is acceptable.) All that remains to be shown is that when $(\mathbf{B}, \boldsymbol{y}, t)$ is a YES instance, then the reduction is guaranteed to accept (provided Lemma 8 was invoked successfully). So, assume the input to the reduction is a YES instance, i.e., there exists a binary vector $\boldsymbol{x} \in \{0,1\}^k$ such that $\|\mathbf{B}\boldsymbol{x} - \boldsymbol{y}\| \leq t$. First of all, we bound the minimum distance of the lattice $\mathbf{B}'$, so we can argue that the target is within the decoding radius of the α-**BDD** oracle. Considering only the second block of coordinates, we see that any lattice vector that is not a multiple of the last column has norm at least $\beta\lambda_1(\mathbf{L}) > \beta\sigma r$. It follows that the last column in the basis matrix is the (unique, up to sign change) shortest vector

in the lattice and $\lambda_1(\mathbf{B}) = \beta\sigma r$. Now, consider the distance of $\boldsymbol{y}'$ from the lattice $\mathbf{B}'$. We know from Lemma 8 that there exists an integer vector $\boldsymbol{z} \in \mathbb{Z}^m$ such that $\mathbf{T}z = \boldsymbol{x}$ and $\|\mathbf{L}\boldsymbol{z} - \boldsymbol{s}\| \leq r$. Multiplying the basis matrix $\mathbf{B}'$ by $[\boldsymbol{z}^T, 0]^T$, we get a lattice vector within distance

$$\sqrt[p]{\|\mathbf{B}(\mathbf{T}\boldsymbol{z}) - \boldsymbol{y}\|^p + \beta^p \|\mathbf{L}\boldsymbol{z} - \boldsymbol{s}\|^p} \leq \sqrt[p]{t^p + \beta^p r^p}$$

from the target $\boldsymbol{y}'$. So, the α-**BDD** promise $\mathrm{dist}(\mathbf{B}', \boldsymbol{y}') \leq \alpha\lambda_1(\mathbf{B}') = \alpha\beta\sigma r$ is satisfied whenever

$$\sqrt[p]{t^p + \beta^p r^p} \leq \alpha\beta\sigma r. \tag{1}$$

Moreover, if (1) is satisfied, then the α-**BDD** oracle returns a vector $\boldsymbol{v} = [(\mathbf{B}\mathbf{T}\boldsymbol{z})^T, (\mathbf{L}\boldsymbol{z})^T, (\beta\sigma r) \cdot z]^T$ within distance $\alpha\lambda_1(\mathbf{B}') \leq \alpha\beta\sigma r$ from the target. Since the distance of $\boldsymbol{v}$ from $\boldsymbol{y}$ is at least $\|\mathbf{B}\mathbf{T}\boldsymbol{z} - \boldsymbol{y}\|$, we conclude that the reduction accepts, provided conditions (1) and

$$\alpha\beta\sigma r \leq \gamma t \tag{2}$$

are satisfied. Both (1) and (2) are easily verified setting $\beta = t\gamma/\alpha\sigma r$ and substituting $\sigma/(\alpha\sqrt[p]{1-\gamma^{-p}})$. □

Since the $\mathbf{GapCVP}'_\gamma$ problem is NP-hard for all constants $\gamma \geq 1$, we immediately get the following corollary:

Corollary 1. *For all $\alpha > 1/\sqrt[p]{2}$, α-**BDD** in the ℓ_p norm is NP-hard.*

In addition, the reduction in the proof of Theorem 3 only depends on $\boldsymbol{B}$ and t, and not on the actual vector $\boldsymbol{y}$. Thus it should be possible to use the hardness result of Alekhnovich, et. al. [4] to show the hardness of α-**BDD** with pre-processing. Two minor details arise, though. First, the reduction in [4], proves the hardness of $\mathbf{GapCVP}_\gamma$ with pre-processing, not $\mathbf{GapCVP}'_\gamma$ with pre-processing. But it can be extracted from the proof in [4] that the special version $\mathbf{GapCVP}'_\gamma$ with pre-processing is NP-hard as well. Another point is that in the proof of [4], the algorithm is allowed to pre-process only the basis $\mathbf{B}$, while in our reduction, the algorithm would get both $\mathbf{B}$ and t for pre-processing before getting the vector $\boldsymbol{y}$. This problem can be solved by observing that in [4] the NP-hardness of $\mathbf{GapCVP}_\gamma$ is proved by a reduction from the coding problem **NCPP**. As a result, there are only polynomially many values for t, and thus the advice string for $\mathbf{GapCVP}_\gamma$ can contain advice for all possible values of t.

Corollary 2. *For any constant $\alpha > 1/\sqrt[p]{2}$, if there exists a polynomial time algorithm that can solve α-**BDD** with pre-processing in the ℓ_p norm , then $NP \subseteq P/poly$.*

In [19], Regev and Rosen showed reductions from problems in the ℓ_2 norm to problems in the ℓ_p norm by using the fact that for any p, there exist embedding functions $f : \mathbb{R}^n \to \mathbb{R}^m$ (where m is $poly(n)$) such that for any $\boldsymbol{x} \in \mathbb{R}^n$ $\|\boldsymbol{x}\|_2 \approx \|f(\boldsymbol{x})\|_p$. Using the same idea, we can obtain the following corollary:

Corollary 3. *For any $p > 2$ and constant $\alpha > 1/\sqrt{2}$, if there exists a polynomial time algorithm that can solve α-**BDD** with pre-processing in the ℓ_p norm, then $NP \subseteq P/poly$.*

References

1. E. Agrell, T. Eriksson, A. Vardy, and K. Zeger. Closest point search in lattices. *IEEE Trans. on Inf. Theory*, 48(8):2201–2214, 2002.
2. D. Aharonov and O. Regev. Lattice problems in NP ∩ coNP. *Journal of the ACM*, 52(5):749–765, 2005.
3. M. Ajtai, R. Kumar, and D. Sivakumar. Sampling short lattice vectors and the closest lattice vector problem. In *CCC*, pages 53–57, 2002.
4. M. Alekhnovich, S. Khot, G. Kindler, and N. Vishnoi. Hardness of approximating the closest vector problem with pre-processing. In *FOCS*, 2005.
5. S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54(2):317–331, 1997.
6. L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.
7. W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–635, 1993.
8. A. H. Banihashemi and A. K. Khandani. On the complexity of decoding lattices using the Korkin-Zolotarev reduced basis. *IEEE Trans. on Inf. Theory*, 44(1):162–171, January 1998.
9. 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.
10. I. Dumer, D. Micciancio, and M. Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Trans. on Inf. Theory*, 49(1):22–37, January 2003.
11. U. Feige and D. Micciancio. The inapproximability of lattice and coding problems with preprocessing. *Journal of Computer and System Sciences*, 69(1):45–67.
12. S. Khot. Hardness of approximating the shortest vector problem in lattices. In *FOCS*, pages 126–135, 2004.
13. P. Klein. Finding the closest lattice vector when it's unusually close. In *SODA*, pages 937–941, 2000.
14. J. C. Lagarias, H. W. Lenstra Jr., and C. P. Schnorr. Korkin-zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica*, 10(4):333–348, 1990.
15. D. Micciancio. The hardness of the closest vector problem with preprocessing. *IEEE Trans. on Inf. Theory*, 47(3):1212–1215, 2001.
16. D. Micciancio. The shortest vector problem is NP-hard to approximate to within some constant. *SIAM J. on Computing*, 30(6):2008–2035, 2001.
17. D. Micciancio and S. Goldwasser. *Complexity Of Lattice Problems: A Cryptographic Perspective*. Kluwer Academic Publishers, 2002.
18. O. Regev. Improved inapproximability of lattice and coding problems with preprocessing. *IEEE Trans. on Inf. Theory*, 50(9):2031–2037, 2004.
19. O. Regev and R. Rosen. Lattice problems and norm embeddings. In *STOC*, 2006.
20. I.J. Schoenberg. Metric spaces and positive definite functions. *Trans. Amer. Math. Soc.*, 44(3):522–536, 1938.
21. D. Stefankovic. Fourier transforms in computer science. Master's thesis, University of Chicago, 2000.
22. J. Stewart. Positive definite functions and generalizations, an historical survey. *Rocky Mountain J. Math*, 6(3), 1976.
23. A. Vardy. Algorithmic complexity in coding theory and the minimum distance problem. In *STOC*, pages 92–109, 1997.