Approximating shortest lattice vectors is not harder than approximating closest lattice vectors

Oded Goldreich* Daniele Micciancio† Shmuel Safra‡ Jean-Pierre Seifert§

January 21, 1999

Abstract

We show that given oracle access to a subroutine which returns approximate closest vectors in a lattice, one may find in polynomial-time approximate shortest vectors in a lattice. The level of approximation is maintained; that is, for any function f, the following holds: Suppose that the subroutine, on input a lattice $\mathcal{L}$ and a target vector $\mathbf{w}$ (not necessarily in the lattice), outputs $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v} - \mathbf{w}\| \leq f(n) \cdot \|\mathbf{u} - \mathbf{w}\|$ for any $\mathbf{u} \in \mathcal{L}$. Then, our algorithm, on input a lattice $\mathcal{L}$, outputs a nonzero vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| \leq f(n) \cdot \|\mathbf{u}\|$ for any nonzero vector $\mathbf{u} \in \mathcal{L}$. The result holds for any norm, and preserves the dimension of the lattice, i.e., the closest vector oracle is called on lattices of exactly the same dimension as the original shortest vector problem.

This result establishes the widely believed conjecture by which the shortest vector problem is not harder than the closest vector problem. The proof can be easily adapted to establish an analogous result for the corresponding computational problems for linear codes.

Keywords: Computational problems in integer lattices, reducibility among approximation problems, linear error-correcting codes.

*Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, ISRAEL. Email: oded@theory.lcs.mit.edu.

†Laboratory for Computer Science, MIT, Cambridge, USA. Email: miccianc@theory.lcs.mit.edu.

‡Department of Computer Science, School of Mathematics, Tel-Aviv University, Ramat-Aviv, ISRAEL. Email: safra@math.tau.ac.il.

§Department of Mathematics and Computer Science, University of Frankfurt, 60054 Frankfurt on the Main, P.O. Box 11 19 32, GERMANY. Email: seifert@cs.uni-frankfurt.de.

1 Introduction

Two basic computational problems regarding integer lattices are the Shortest Vector Problem (SVP), and the Closest Vector Problem (CVP). Loosely speaking, the input to SVP is a lattice, and one is required to find the shortest (non-zero) vector in the lattice. In CVP the input is a lattice together with a target vector, and one is required to find the lattice vector closest to the target. Lengths and distances may be measured in a variety of norms, but the case of the Euclidean (L_2) Norm is considered the most interesting one.

It is widely believed that SVP is not harder than CVP, and many even believe that SVP is strictly easier. Empirical evidence to these beliefs is provided by the gap between known hardness results for both problems. Whereas it is easy to establish the NP-Hardness of CVP (cf., [vEB]), the question of whether SVP is NP-Hard was open for almost two decades (until being recently resolved in the affirmative, for randomized reductions, by Ajtai [A]). Furthermore, approximating CVP in n-dimensional lattices to within a $2^{\log^{0.999} n}$ factor is NP-Hard (cf., [ABSS, DKS]), whereas SVP is only known to be NP-Hard to approximate to within constant factors smaller than $\sqrt{2}$ (cf., [M]).

Note that for all Euclidean norms ($L_p, p \geq 1$), SVP can be easily reduced to CVP using the NP-hardness of the latter. However, this general NP-completeness argument produces CVP instances of dimension much bigger than the original SVP problem. An interesting question is whether a direct reduction is possible that preserves the dimension. More importantly, the NP-hardness results do not elucidate on the relation between approximate SVP and approximate CVP when the approximation factor is polynomial (or super-polynomial) in the dimension, or the norm is different from the Euclidean ones. We recall that the only when the approximation factor is exponential the two problems are known to be solvable in polynomial time (cf. [LLL, B]).

The first non-empirical evidence that SVP is not harder than CVP (in the same dimension) was recently given by Henk [H], who showed that solving SVP (in the exact sense) is reducible to solving CVP (also in the exact sense). Moreover, the result in [H] holds for a wide variety of norms (not only Euclidean ones). Here we provide an analogous (and thus stronger) result for approximation, and unlike Henk's proof we do not employ any non-elementary result about lattices.

We show how to reduce the task of finding an f-approximation for SVP to the task of finding an f-approximation for CVP (in the same dimension and with the same approximation factor). This resolves a decade old question of László Babai [B], who actually suggested as a challenge to reduce the task of approximating SVP to within any sub-exponential factor to the task of approximating CVP (in the same dimension) quite well (e.g., upto a constant factor $c > 1$). Our result holds for any function f (and thus, in particular, for $f \equiv 1$), for any norm, and for both the decision and the search versions.

In section 2 we introduce some notation and formally define the problems. In sections 3 and 4 we establish the above claims. Section 5 adapts the proof techniques to establish an analogous result for linear codes. Section 6 concludes with some remarks and open problems.

2 Preliminaries

In the following, we use lowercase letters for scalars, boldface lowercase letters for vectors, and capital letters for sets, matrices, and sequences of vectors. The sets of reals, integers and natural numbers are denoted by $\mathbb{R}$, $\mathbb{Z}$, and $\mathbb{N}$, respectively.

$\mathbb{R}^m$ is the m-dimensional Euclidean real vector space, and $\|\cdot\|$ is an arbitrary norm $\mathbb{R}^m \mapsto \mathbb{R}$. A lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^m$. Its rank, denoted by $\mathrm{rank}(\mathcal{L})$, is the dimension of the $\mathbb{R}$-subspace, denoted $\text{span}(\mathcal{L})$, that it spans. Each lattice $\mathcal{L}$ of rank n has a basis, i.e., a sequence $[\mathbf{b}_1, \ldots, \mathbf{b}_n]$ of n elements of $\mathcal{L}$ that generate $\mathcal{L}$ as an Abelian group. Thus, the lattice is obtained by all *integer* linear combinations of the basis vectors, whereas the span of the lattice corresponds to all real linear combinations of the basis vectors.

In the following definitions we state two fundamental computational problems regarding lattices. Both problem are stated with respect to the same (arbitrary) norm $\|\cdot\|$. We always assume that a lattice $\mathcal{L}$ is given by a basis $[\mathbf{b}_1, \ldots, \mathbf{b}_n]$ generating $\mathcal{L}$. The approximation factor is measured in terms of n (the rank of the lattice).

Definition 1 (Shortest Vector Problem): *In the f-Shortest Vector Problem, denoted $\mathtt{SVP}_f$, one is given a lattice $\mathcal{L}$ and the task is to find a non-zero vector $\mathbf{v} \in \mathcal{L}$ so that*

$$\|\mathbf{v}\| \leq f(n) \cdot \|\mathbf{u}\|$$

for any other non-zero vector $\mathbf{u} \in \mathcal{L}$. In the decision version, denoted $\mathtt{GapSVP}_f$, one should distinguish pairs $(\mathcal{L}, d)$ for which the shortest (non-zero) lattice vector has length at most d from pairs for which the shortest (non-zero) lattice vector has length greater than $f(n) \cdot d$.

Definition 2 (Closest Vector Problem): *In the f-Closest Vector Problem, denoted $\mathtt{CVP}_f$, one is given a lattice $\mathcal{L}$ and a vector $\mathbf{w} \in \text{span}(\mathcal{L})$ and the task is to find a vector $\mathbf{v} \in \mathcal{L}$ so that*

$$\|\mathbf{v} - \mathbf{w}\| \leq f(n) \cdot \|\mathbf{u} - \mathbf{w}\|$$

for any other vector $\mathbf{u} \in \mathcal{L}$. In the decision version, denoted $\mathtt{GapSVP}_f$, one should distinguish between triples $(\mathcal{L}, \mathbf{w}, d)$ for which there exists a lattice vector within distance d from $\mathbf{w}$ and triples for which there exists no lattice vector within distance $f(n) \cdot d$ from $\mathbf{w}$.

3 Reducing approximate SVP to approximate CVP

There are two differences between the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP). On one hand, SVP asks for a lattice point close to the all-zero vector, while CVP asks for a lattice point close to an arbitrary target vector; on the other hand, SVP disallows the all-zero solution whereas CVP accepts the target vector as an admissible solution (provided it belongs to the lattice). Thus, the two problems are not trivially related. In particular, the trivial reduction from SVP to CVP (i.e., $\mathcal{L} \mapsto (\mathcal{L}, 0^m)$) may not work since the CVP oracle may return the all-zero vector. Our aim is to prevent this possibility. The intuitive idea is the following (see Figure 1). First of all, instead of looking for a lattice point close to the all-zero vector, we look for a lattice point close to some other lattice vector $\mathbf{w} \in \mathcal{L}$. Moreover, to avoid $\mathbf{w}$ being returned as a solution, we run the CVP oracle on a sub-lattice $\mathcal{L}' \subset \mathcal{L}$ not containing $\mathbf{w}$. The problem is now how to select a sub-lattice $\mathcal{L}' \subset \mathcal{L}$ without removing the $\mathcal{L}$-vectors closest to $\mathbf{w}$. We start with the following observation.

Proposition 3.1 *Let $\mathbf{v} = \sum_{i=1}^{n} c_i \mathbf{b}_i$ be a shortest non-zero vector in $\mathcal{L}$. Then, there exists an i so that c_i is odd.*

Proof: Otherwise, all c_i's are even, and $\frac{1}{2} \cdot \mathbf{v} = \sum_{i=1}^{n} \frac{c_i}{2} \mathbf{b}_i$ is a shorter vector in $\mathcal{L}$. ■

We now show how to reduce the shortest vector problem to the solution of n instances of the closest vector problem.

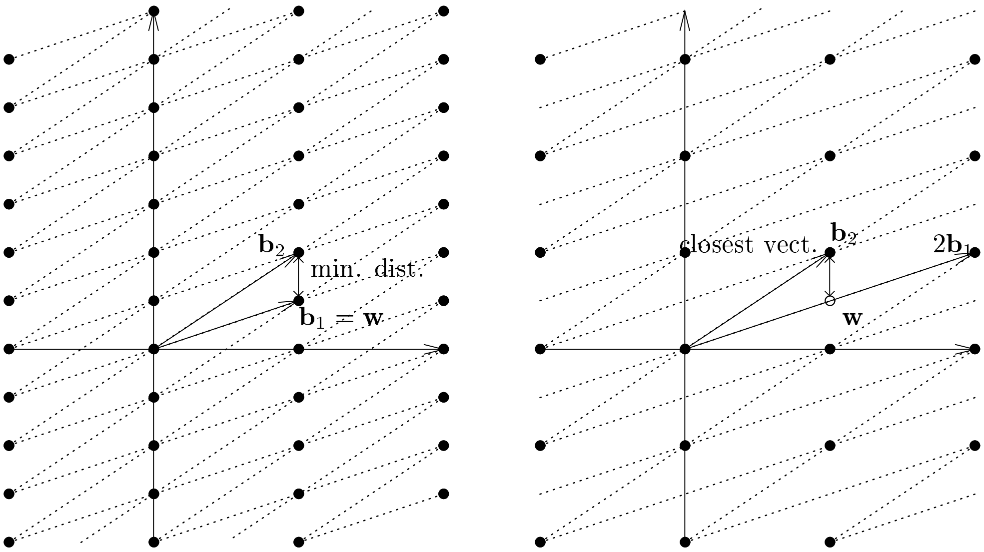

Figure 1: Reducing SVP to CVP

input: A pair (B, d), where $B = [\mathbf{b}_1, ..., \mathbf{b}_n]$ and $d \in \mathbb{R}$.

For $j = 1$ to n, invoke the oracle on input $(B^{(j)}, \mathbf{b}_j, d)$, where $B^{(j)}$ is as in Eq. (1).

output: the OR of all oracle replies.

Figure 2: The reduction – decision version

The reduction: Given a basis $B = [\mathbf{b}_1, ..., \mathbf{b}_n]$ to the lattice $\mathcal{L}(B) = \{\sum_{i=1}^{n} c_i \mathbf{b}_i : c_1, ..., c_n \in \mathbb{Z}\}$, we construct n instances of CVP. The j^{th} instance consists of the basis

$$B^{(j)} \stackrel{\text{def}}{=} [\mathbf{b}_1, ..., \mathbf{b}_{j-1}, 2\mathbf{b}_j, \mathbf{b}_{j+1}, ..., \mathbf{b}_n] \tag{1}$$

and the target vector $\mathbf{b}_j$. In the search version we uses these n instances of CVP in n corresponding queries to the $\mathtt{CVP}_f$ oracle, and output the shortest *difference* returned in all these calls (i.e., if $\mathbf{v}_j$ is the vector returned by the j^{th} call on input $(B^{(j)}, \mathbf{b}_j)$, we return the shortest of the vectors $\mathbf{v}_1 - \mathbf{b}_1, \ldots, \mathbf{v}_n - \mathbf{b}_n$). In the decision version, we augment these queries by the same parameter d given in the `GapSVP` instance, and return YES if and only if one of the oracle calls was answered by YES. The reduction for the decision version is depicted in Fig. 2.

The validity of the reduction follows from the correspondence between solutions to the input SVP instance and solutions to the CVP instances used in the queries. Specifically:

Proposition 3.2 *Let* $\mathbf{v} = \sum_{i=1}^{n} c_i \mathbf{b}_i$ *be a lattice vector in* $\mathcal{L}(B)$ *so that* c_j *is odd. Then* $\mathbf{u} = \frac{c_j+1}{2}(2\mathbf{b}_j) + \sum_{i \neq j} c_i \mathbf{b}_i$ *is a lattice vector in* $\mathcal{L}(B^{(j)})$ *and the distance of* $\mathbf{u}$ *from the target* $\mathbf{b}_j$ *equals the length of* $\mathbf{v}$.

Proof: Firstly, note that $\mathbf{u} \in \mathcal{L}(B^{(j)})$ since $\frac{c_j+1}{2}$ is an integer (as c_j is odd). Secondly, observe that

$$\begin{aligned}\mathbf{u} - \mathbf{b}_j &= \frac{c_j+1}{2} 2\mathbf{b}_j + \sum_{i \neq j} c_i \mathbf{b}_i - \mathbf{b}_j \\ &= c_j \mathbf{b}_j + \sum_{i \neq j} c_i \mathbf{b}_i \;=\; \mathbf{v}\end{aligned}$$

and the proposition follows. ■

Proposition 3.3 *Let* $\mathbf{u} = c'_j(2\mathbf{b}_j) + \sum_{i \neq j} c_i \mathbf{b}_i$ *be a lattice vector in* $\mathcal{L}(B^{(j)})$*. Then* $\mathbf{v} = (2c'_j - 1)\mathbf{b}_j + \sum_{i \neq j} c_i \mathbf{b}_i$ *is a non-zero lattice vector in* $\mathcal{L}(B)$ *and the length of* $\mathbf{v}$ *equals the distance of* $\mathbf{u}$ *from the target* $\mathbf{b}_j$*.*

Proof: Firstly, note that $\mathbf{v}$ is non-zero since $2c'_j - 1$ is an odd integer. Secondly, observe that

$$\begin{aligned}\mathbf{v} &= (2c'_j - 1)\mathbf{b}_j + \sum_{i \neq j} c_i \mathbf{b}_i \\ &= c'_j(2\mathbf{b}_j) + \sum_{i \neq j} c_i \mathbf{b}_i - \mathbf{b}_j \;=\; \mathbf{u} - \mathbf{b}_j\end{aligned}$$

and the proposition follows. ■

Combining Propositions 3.1 and 3.2, we conclude that one of the CVP-instances has an optimum which is at most the optimum of the given SVP-instance. On the other hand, by Proposition 3.3, the optimum of each of the CVP-instances is lower bounded by the optimum of the given SVP-instance. Details follow.

Theorem 3 *For every function* $f : \mathbb{N} \mapsto \{r \in \mathbb{R} : r \geq 1\}$, $\mathtt{SVP}_f$ *(resp.,* $\mathtt{GapSVP}_f$*) is Cook-reducible to* $\mathtt{CVP}_f$ *(resp.,* $\mathtt{GapCVP}_f$*). Furthermore, the reduction is non-adaptive, and all queries maintain the rank of the input instance.*

Proof: We prove the theorem for the decisional version. The search version is analogous. Let (B, d) be a $\mathtt{GapSVP}_f$ instance, and define $\mathtt{GapCVP}_f$ instances $(B^{(j)}, \mathbf{b}_j, d)$ for $j = 1, \ldots, n$, where $B^{(j)}$ is as in Eq. (1). We want to prove that if (B, d) is a YES instance, then $(B^{(j)}, \mathbf{b}_j, d)$ is a YES instance for some $j = 1, \ldots, n$, and if (B, d) is a NO instance, then $(B^{(j)}, \mathbf{b}_j, d)$ is a NO instance for all $j = 1, \ldots, n$.

First assume (B, d) is a YES instance and let $\mathbf{v} = \sum_{i=1}^{n} c_i \mathbf{b}_i$ be the shortest non-zero lattice vector in $\mathcal{L}(B)$. We know $\|\mathbf{v}\| \leq d$, and (by Proposition 3.1) c_j is odd for some j. The vector $\mathbf{u}$ as defined in Proposition 3.2 belongs to $\mathcal{L}(B^{(j)})$ and satisfies $\|\mathbf{u} - \mathbf{b}_j\| = \|\mathbf{v}\| \leq d$, proving that $(B^{(j)}, \mathbf{b}_j, d)$ is a YES instance.

Now assume $(B^{(j)}, \mathbf{b}_j, d)$ is not a NO instance for some j. There exists a vector $\mathbf{u}$ in $\mathcal{L}(B^{(j)})$ such that $\|\mathbf{u} - \mathbf{b}_j\| \leq f(n) \cdot d$. The vector $\mathbf{v}$ defined in Proposition 3.3 is a non-zero lattice vector in $\mathcal{L}(B)$ and satisfies $\|\mathbf{v}\| = \|\mathbf{u} - \mathbf{b}_j\| \leq f(n) \cdot d$, proving that (B, d) is not a NO instance. ■

4 A Randomized Reduction

In the previous section we showed that solving an instance of $\mathtt{SVP}_f$ can be deterministically reduced to solving n instances of $\mathtt{CVP}_f$, where n is the rank of the lattices. A natural question if whether it is possible to reduce an `SVP` problem to a single instance of `CVP`. The proof of Theorem 3 suggests that this is possible for randomized reductions. Namely, on input (B, d), choose $j \in \{1, \ldots, n\}$ at random and output $(B^{(j)}, \mathbf{b}_j, d)$. We notice that YES instances are mapped to YES instances with probability at least $1/n$, and NO instances are always mapped to NO instances. We now show that we can actually do better than that. We give a probabilistic reduction from $\mathtt{SVP}_f$ to $\mathtt{CVP}_f$ that succeeds with probability at least $1/2$.

Theorem 4 *For every function $f : \mathbb{N} \mapsto \{r \in \mathbb{R} : r \geq 1\}$, there is a probabilistic many-one reduction from $\mathtt{SVP}_f$ (resp., $\mathtt{GapSVP}_f$) to $\mathtt{CVP}_f$ (resp., $\mathtt{GapCVP}_f$) that has one-sided error probability bounded above by $1/2$. Furthermore, the `CVP` instance produced has the same rank as the original `SVP` problem.*

Proof: Again, we prove the theorem for the decisional version, and the search version is analogous. Let (B, d) be an SVP instance, where $B = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$. Output CVP instance $(B', \mathbf{b}_1, d)$ where $B' = [\mathbf{b}'_1, \ldots, \mathbf{b}'_n]$ is defined as follows. Let $c_1 = 1$ and choose $c_i \in \{0, 1\}$ $(i = 2, \ldots, n)$ uniformly and independently at random. For all i, let $\mathbf{b}'_i = \mathbf{b}_i + c_i \mathbf{b}_1$. We want to prove that if (B, d) is a YES instance then $(B', \mathbf{b}_1, d)$ is a YES instance with probability at least $1/2$, while if (B, d) is a NO instance then $(B', \mathbf{b}_1, d)$ is always a NO instance. Notice that $\mathcal{L}(B')$ is a sub-lattice of $\mathcal{L}(B)$ and that $\mathbf{b}_1$ is not in $\mathcal{L}(B')$.

Let's start with the NO case first. Assume $(B', \mathbf{b}_1, d)$ is not a NO instance. By definition, there exists a vector $\mathbf{u}$ in $\mathcal{L}(B')$ such that $\|\mathbf{u} - \mathbf{b}_1\| \leq f(n) \cdot d$. Since $\mathcal{L}(B')$ is a sub-lattice of $\mathcal{L}(B)$ and $\mathbf{b}_1$ is not in $\mathcal{L}(B')$, $\mathbf{v} = \mathbf{u} - \mathbf{b}_1$ is a non-zero vector in $\mathcal{L}(B)$ of length at most $f(n) \cdot d$, proving that (B, d) is not a NO instance.

Now assume (B, d) is a YES instance and let $\mathbf{v} = \sum_{i=1}^{n} x_i \mathbf{b}_i$ be the shortest vector in $\mathcal{L}(B)$. From Proposition 3.2, x_j is odd for some j. Let $\alpha = x_1 + 1 - \sum_{i>1} c_i x_i$. Notice that if x_i is even for all $i > 1$, then x_1 must be odd and α is even. On the other hand, if x_i is odd for some $i > 1$ then α is even with probability $1/2$. In both cases, with probability at least $1/2$, α is even and $\mathbf{u} = \frac{\alpha}{2}\mathbf{b}'_1 + \sum_{i>1} x_i \mathbf{b}'_i$ is a lattice vector in $\mathcal{L}(B')$. Finally notice that

$$
\begin{aligned}
\mathbf{u} - \mathbf{b}_1 &= \left(\alpha \mathbf{b}_1 + \sum_{i>1} x_i(\mathbf{b}_i + c_i \mathbf{b}_1)\right) - \mathbf{b}_1 \\
&= \left(x_1 - \sum_{i>1} c_i x_i\right) \mathbf{b}_1 + \sum_{i>1} x_i \mathbf{b}_i + \sum_{i>1} x_i c_i \mathbf{b}_1 \quad = \quad \mathbf{v}
\end{aligned}
$$

and therefore $\|\mathbf{u} - \mathbf{b}_1\| \leq d$, proving that $(B', \mathbf{b}_1, d)$ is a YES instance. ■

5 Adaptation to Linear Codes

Two well-known problems in coding theory, analogous to SVP and CVP for lattices, are the Minimum Distance Problem (MDP) and the Nearest Codeword Problem (NCP), for linear codes. In the Minimum Distance Problem, the input is a linear code over a finite field $\mathbb{F}$ (the alphabet) and one must find a non-zero codeword of minimum Hamming weight. In the Nearest Codeword Problem, the input is a linear code and a target string (over the same alphabet), and one must find the codeword closest (in the Hamming metric) to this string.

A linear code of *length* n over a finite field $\mathbb{F}$ is a linear subspace $\mathcal{C}$ of $\mathbb{F}^n$. The *rate* of a code $\mathcal{C}$ is its dimension as a vector space over $\mathbb{F}$. Codes can be represented by a generator matrix, analogous to the basis representation of lattices. The most interesting case is when the alphabet $\mathbb{F} = \mathbb{Z}/2\mathbb{Z}$ is the binary field. In this case, a code is given by a full rank m-by-n Boolean matrix C and the codewords are all linear combinations of the columns of C (where the sum is taken modulo 2). The Hamming weight of a word $\mathbf{w} \in \mathbb{F}^n$, denoted $\mathrm{wt}(\mathbf{w})$ is the number of non-zero elements in $\mathbf{w}$. The distance between words is usually measured by the Hamming metric $d(\mathbf{v}, \mathbf{w}) = \mathrm{wt}(\mathbf{v} - \mathbf{w})$.

The Minimum Distance Problem and the Nearest Codeword Problem are obviously related to the problems of finding good error correcting codes and decoding them respectively. Although, historically, the prevailing approach in coding theory has been to study the complexity of code construction, while completely ignoring the complexity of decoding the resulting code (cf. [FFMMV, V]), the relation between the two problems is clear: we would like to find good linear codes that can also be efficiently decoded. As in the lattice case, empirical evidence shows that MDP is not harder than NCP: whereas it is easy to establish the NP-hardness of NCP (cf. [BMT]), the question for MDP was open until recently being resolved in the affirmative by Vardy (cf. [V]). Furthermore, the NP-hardness of approximating NCP to within any constant factor was proved in [ABSS], whereas MDP was proved NP-hard to approximate within any constant only recently (cf. [DMS]).

However, to the best of out knowledge, the exact relationship between the complexity of these two fundamental coding problems, has never been investigated. We prove a result for coding problems analogous to the result on lattices: approximating the Minimum Distance of a code is not harder than approximating the Nearest Codeword to a target string. In light of the result (cf. [TV], p. 77) that *almost all* linear codes are good (in the sense that they attain the Gilbert-Varshamov bound $R = 1 - H(d)$, where R is the rate, d the relative distance and H the binary entropy function), we have the following interesting implication: if an efficient algorithm to (approximately) solve the decoding problem (for linear codes) exists, then we can also efficiently find good codes. Interestingly, algebraic geometry codes performing better than the Gilbert-Varshamov bound have been used to prove the NP-hardness of approximating the Minimum Distance Problem (cf. [DMS]).

The reduction from MDP to NCP is basically the same as the lattice one. Actually, it is even easier to establish for binary codes, as the analogue of Proposition 3.1 is trivial (and in fact holds for any non-zero codeword). Eq. (1) simplifies too, since here multiplying a column by 2 results in the all-zero column (which may in fact be omitted altogether). Finally, the analogues of Propositions 3.2 and 3.3 follow easily as above (actually, even more easily). We conclude that

Theorem 5 *For every function $f : \mathbb{N} \mapsto \{r \in \mathbb{R} : r \geq 1\}$, the problem of approximating the distance of a given Boolean linear code upto factor f is Cook-reducible to the problem of approximating the distance of a given string from a given Boolean linear code upto factor f.*

The above theorem actually holds for linear codes over an arbitrary finite field $\mathbb{F} = GF(q)$.

Theorem 6 *For every function $f : \mathbb{N} \mapsto \{r \in \mathbb{R} : r \geq 1\}$ and any field $\mathbb{F} = GF(q)$, the problem of approximating the minimum distance of a given linear code over $\mathbb{F}$ upto factor f is Cook-reducible to the problem of approximating the distance of a given string from a linear code over $\mathbb{F}$ within the same approximation factor. Moreover, the reduction preserves the length and decreases the rate of the code.*

Proof: Let $C = [\mathbf{c}_1, \ldots, c_n]$ be a linear code over $GF(q)$. For all $i = 1, \ldots, n$, define the sub-code $C^{(i)} = [\mathbf{c}_1, \ldots, \mathbf{c}_{i-1}, \mathbf{c}_{i+1}, \ldots, \mathbf{c}_n]$ and look for the codeword in $C^{(i)}$ (approximately) closest to $\mathbf{c}_i$. Let $\mathbf{d}_i$ be this codewords. Return the C-codeword $\mathbf{d}_i - \mathbf{c}_i$ of minimum weight.

Since $\mathbf{c}_i$ does not belong to the code $C^{(i)}$, the result is always a non-zero C-codeword. We now prove that for any codeword $\mathbf{u} = \sum_{i=1}^{n} x_i \mathbf{c}_i$ in C, there exists an $i \in \{1, \ldots, n\}$ such that $\mathbf{c}_i$ is at distance at most $\|\mathbf{u}\|$ from $C^{(i)}$. Since $\mathbf{u}$ is non-zero, it must be $x_i \neq 0$ for some i. Let y be the multiplicative inverse of $-x_i$ in $\mathbb{F}$ (i.e., $yx_i = -1$), and define the codeword $\mathbf{v} = \sum_{j\neq i}(yx_j)\mathbf{c}_j \in C^{(i)}$. Then,

$$\begin{aligned}\mathbf{v} - \mathbf{c}_i &= yx_1\mathbf{c}_i + \sum_{j\neq i}(yx_j)\mathbf{c}_j \\ &= y\sum_{j=1}^{n} x_j\mathbf{c}_j \;=\; y\mathbf{u}\end{aligned}$$

Thus, we have $\mathrm{wt}(\mathbf{v} - \mathbf{c}_i) = \mathrm{wt}(y\mathbf{u}) = \mathrm{wt}(\mathbf{u})$ (since multiplication by a non-zero scalar does not change the Hamming weight of a vector). ■

As in the previous section, we can use randomness to reduce the shortest codeword problem to a single instance of the nearest codeword problem. This time the success probability (on YES instances) can be made as high as $1 - 1/q$ (while the zero-error on NO instances is preserved).

Theorem 7 *For every function $f : \mathbb{N} \mapsto \{r \in \mathbb{R} : r \geq 1\}$ and any finite field $\mathbb{F} = GF(q)$, there exists a probabilistic polynomial time algorithm that reduces the problem of approximating the minimum distance of a given linear code over $GF(q)$ upto factor f to solving a single instance of approximating the distance of a given string from a given linear code over $\mathbb{F}$ within the same approximation factor.* YES *instances are mapped to* YES *instances with probability $1 - \frac{1}{q}$, while* NO *instances are always mapped to* NO *instances. Moreover, the reduction preserves the length and decreases the rate of the code.*

Proof: We only describe the reduction. The rest of the proof is analogous to that of Theorem 4. Let $C = [\mathbf{c}_1, \ldots, \mathbf{c}_n]$ be the input code. Output target string $\mathbf{c}_1$ and code $C' = [\mathbf{c}'_2, \ldots, \mathbf{c}'_n]$ defined as follows. Choose $\alpha_i \in GF(q)$ $(i = 2, \ldots, n)$ uniformly and independently at random and let $\mathbf{c}'_i = \mathbf{c}_i + \alpha_i\mathbf{c}_1$. ■

6 Discussion

We proved that approximating the Shortest Vector Problem can be reduced in polynomial time to approximating the Closest Vector Problem. Our reduction preserves the approximation factor and the rank of the lattice, and can be adapted to other problems with similar structure, like the Minimum Distance Problem and the Nearest Codeword Problem for linear codes. In both cases, we reduced a *homogeneous* problem to the corresponding *inhomogeneous* one.

The results in [M] and [DMS] are in a certain sense a converse to our result. In [M] and [DMS] the Shortest Vector Problem and the Minimum Distance Problem are proved NP-hard to approximate by reduction from the Closest Vector Problem and the Nearest Codeword Problem respectively. Therefore the inhomogeneous problem is reduced to the corresponding homogeneous one. However, these reductions do not preserve the approximation factor, and produce instances much bigger than the original problems. It is an interesting open problem whether an approximation and size preserving reduction is possible from the Closest Vector Problem to the Shortest Vector Problem.

Another open problem is whethere there exists a Karp-reduction (*deterministic many-to-one* polynomial-time reduction) of the approximate SVP problem to the approximate CVP problem.

References

[A] M. Ajtai, "The Shortest Vector Problem is NP-Hard for Randomized Reductions", *Proc. 30th Symposium on Theory of Computing* 1998, pp. 10-19.

[ABSS] S. Arora, L. Babai, J. Stern, Z. Sweedyk, "The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations", *Journal of Computer and System Sciences* Vol. 54, pp. 317-331, 1997.

[BMT] E.R. Berlekamp, R.J. MacEliece, H.C.A. van Tilborg, "On the Inherent Intractability of Certain Coding Problems", IEEE Transactions on Information Theory, vol. IT-24, n. 3, May 1978.

[B] L. Babai, "On Lovasz' lattice reduction and the nearest lattice point problem", *Combinatorica* Vol. 6, pp. 1-13, 1986.

[DKS] I. Dinur, G. Kindler, S. Safra, "Approximating CVP to Within Almost-Polynomial Factors is NP-Hard", in *Proc. 39th Symposium on Foundations of Computer Science* 1998, pp. 99-109.

[DMS] I. Dumer, D. Micciancio, M. Sudan. "On the hardness of approximating the minimum distance of a linear code", Manuscript.

[FFMMV] J. Feigenbaum, G.D. Forney Jr., B.H. Marcus, R.J. McEliece, A. Vardy, "Introduction to the Special Issue on Codes and Complexity." *IEEE Transactions on Information Theory.* Vol. 42, No. 6. November 1996.

[H] M. Henk, "Note on shortest and nearest lattice vectors", *Information Processing Letters* Vol. 61, pp. 183-188, 1997.

[LLL] A.K. Lenstra, H.W. Lenstra, L. Lovász. "Factoring polynomials with rational coefficients". *Mathematische Annalen* 261, 515-534 (1982).

[M] D. Micciancio, "The Shortest Vector in a Lattice is Hard to Approximate to within Some Constant", in *Proc. 39th Symposium on Foundations of Computer Science* 1998, pp. 92-98.

[TV] M.A. Tsfasman, S.G. Vlăduts. "Algebraic Geometry Codes", Dodrecht: Kluwer Academic, 1991.

[V] A. Vardy. "Algorithmic Complexity in Coding Theory and the Minimum Distance Problem". *Proc. 29th Annual ACM Symposium on Theory of Computing* 1997, pp. 92-109.

[vEB] P. van Emde Boas. " Another NP-complete problem and the complexity of computing short vectors in a lattice". Report 81-04, Mathematische Instituut, Uni. Amsterdam, 1981.