A Fast Phase-Based Enumeration Algorithm for SVP Challenge through y-Sparse Representations of Short Lattice Vectors ⋆ ⋆⋆

Dan Ding[1], Guizhen Zhu[2], Yang Yu[1], Zhongxiang Zheng[1]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China P. R.
[2] Data Communication Science and Technology Research Institute, Beijing 100191, China P. R.

Abstract. In this paper, we propose a new phase-based enumeration algorithm based on two interesting and useful observations for y-sparse representations of short lattice vectors in lattices from SVP challenge benchmarks[24]. Experimental results show that the phase-based algorithm greatly outperforms other famous enumeration algorithms in running time and achieves higher dimensions, like the Kannan-Helfrich enumeration algorithm. Therefore, the phase-based algorithm is a practically excellent solver for the shortest vector problem (SVP).
Keywords: Lattice-Based Cryptography; Shortest Vector Problem(SVP); y-Sparse Representation; Phase-based Enumeration Algorithm.

1 Introduction

Lattice is a set of regularly arranged points in a Euclidean space, and it is widely used in both cryptanalysis and cryptography in recent years. As a promising candidate for the post-quantum cryptography, the lattice-based cryptography [19] attracts much attention from the cryptology community. The seminal paper in 1982 by A. K. Lenstra, H. W. Lenstra and L. Lavász [15] proposes the famous **LLL** algorithm for finding a short lattice basis. In the past 30 years after LLL algorithm, a variety of one-way functions are proposed based on the worst-case hardness of variants of lattice problems [1][17][18], and some new public-key cryptography [3][8][23] are put forward based on the hardness of lattice problems.

Therefore, the lattice problems are of prime importance to cryptography because the security of the lattice-based cryptography is based on the hardness of them. The two famous lattice problems are, the shortest vector problem (SVP), which is to, given a lattice basis, find the shortest nonzero vector in the lattice, and the closest vector problem (CVP), which is to, given a lattice basis and a target vector, find the lattice vector closest to the target vector.

The CVP has long been proved to be NP-hard by P. van Emde Baos in 1981 through classical Cook/Karp reduction[27], and the proof is refined by D. Miccancio et al[16], while, at the same time, the hardness of other lattice problem SVP remains an open problem until SVP is proved to be NP-hard under a randomized reduction by M. Ajtai in 1998[2]. Therefore, both CVP and SVP are hard enough to afford the security of lattice-based cryptography.

Since the hardness of both lattice problems, the algorithms to solve the two of them attracts interests of more and more researchers in cryptology community recently. In the last 30 years, a variety of algorithms have been proposed for both lattice problems, or, more specifically, for SVP. The SVP algorithms are of the following two categories: the theoretically sound algorithms, and the practically sound ones. The sieve algorithm for SVP [4] [22] [28] is proved to find the shortest vector within $2^{O(n)}$ time complexity with an exponential space complexity, which defies implementation, or, is not practical, especially for lattices of high dimensions. The algorithm based on Voronoi-cell computation [20] is a deterministic SVP algorithm, which is proved to be of $2^{2n+o(n)}$ time complexity, and not practical with an exponential

⋆ Supported by 973 Program (Grant No. 2013CB834205) and the National Natural Science Foundation of China (Grant No. 61133013).
⋆⋆ Supported by the National Development Foundation for Cryptological Research (No. MMJJ201401003).

space complexity. The blockwise Korkin-Zolotarev algorithm, or, **BKZ** algorithm [5][25], falls into the second category, and the **BKZ** algorithm is a basis reduction algorithm and is to find lattice basis with excellent properties (which will be discussed in Section 2), which is also, practically, good at finding short vectors. The SVP algorithm is amenable for implementing with polynomial space, but it is still unknown whether the **BKZ** algorithm will terminate within finite steps, though the basis is good enough after finite iterations[9][10] (not theoretically sound). A genetic algorithm for SVP is proposed [6] based on the y-sparse representation of short lattice vectors, with polynomial space complexity and excellent experimental results, but it is still an open problem to estimate its time complexity.

The enumeration algorithm is sound both practically (of polynomial space complexity) and theoretically (with delicate theoretical analysis). The most famous Kannan-Helfrich enumeration algorithm is proposed by R. Kannan [13] and B. Helfrich[12], which is theoretically analyzed as of $2^{O(n \log n)}$ time complexity[14][11]. Some further improved enumeration algorithms[7] are proposed afterwards obtaining good experimental results.

The contributions of this paper are twofold. First, we propose some interesting observations for the y-sparse representations of the short lattice vectors: dividing the vector $\mathbf{y}$ corresponding to the shortest vector in a lattice evenly into phases, the second half of the $\mathbf{y}$ takes on an ascending order in their ℓ_1-norm of each phases, and sparse as whole; second, based on the observations, we propose a new phase-based enumeration algorithm, and the results show that the algorithm dramatically reduces the running time compared to other famous enumeration algorithms, like the Kannan-Helfrich algorithm, without missing the shortest vector.

The rest of the paper is organized as follows: Section 2 presents some necessary preliminaries on lattices and y-sparse representations of short lattice vectors; Section 3 introduces the concept of "phase" to the y-sparse representations of short lattice vectors, and proposes some interesting observations for the short vectors of SVP challenge benchmarks; based on phases, we put forward a fast phase-based enumeration algorithm in Section 4; experimental results are reported in Section 5, and the conclusion is drawn in the following Section 6 and future work in Section 7.

2 Preliminaries

Let n be an integer, and let $\mathbb{R}^n$ be the n-dimensional Euclidean space. The Euclidean norm, ℓ_2-norm, of a vector $\mathbf{v}$ is defined as $\|\mathbf{v}\| = \sqrt{\sum_{i=1}^{n} {v_i}^2}$, and the ℓ_1-norm of $\mathbf{v}$ as $\|\mathbf{v}\|_1 = \sum_{i=1}^{n} |v_i|$, in which $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{R}^n$. The linear space spanned by a set of vectors is denoted by $\mathrm{span}(\cdot)$ and its orthogonal complement $\mathrm{span}(\cdot)^{\perp}$, and $\mathbf{B}^T$ is the transpose of a matrix $\mathbf{B}$. We denote $\lfloor \cdot \rfloor$ as the closest integer less than or equal to a real number, and $\lceil \cdot \rceil$ the upper closest integer. The closed sphere in $\mathbb{R}^n$ is denoted as $\mathcal{B}_n(\mathbf{O}, r)$ with $\mathbf{O}$ as its origin and r its radius. Finally, we denote the $\inf(\cdot)$ as the infimum of the sequence, $\min(\cdot, \cdot)$ as the smaller of the two parameters, and $\dim(\cdot)$ the dimension of the spanned space.

2.1 Lattices

A *lattice* $\mathcal{L}$ is an additive subgroup of the Euclidean space $\mathbb{R}^n$. The lattice can be defined as set of all the integral combinations of n linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$. If all the vectors are of dimensional n, the lattice is called *full-rank*. All the lattices considered in this paper are full-rank if not specified otherwise.

The basis $\mathbf{B}$ of a lattice $\mathcal{L}$ is the matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n] \in \mathbb{R}^{n \times n}$ with the n vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ as its columns. Then the lattice can be represented as

$$\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx} | \mathbf{x} \in \mathbb{Z}^n\} = \{\mathbf{v} \in \mathbb{R}^n \mid \mathbf{v} = \sum_{i=1}^{n} \mathbf{b}_i x_i, x_i \in \mathbb{Z}\}.$$

The i^{th} *successive minimum* $\lambda_i(\mathcal{L})$ (for $i = 1, \ldots, n$) of a lattice $\mathcal{L}$ is defined as the smallest radium of a sphere within which there are i linearly independent lattice points, i.e.,

$$\lambda_i(\mathcal{L}) = \inf\{r \in \mathbb{R}^n | \dim\{\text{span}(\mathcal{L} \cap \mathcal{B}_n(\mathbf{O}, r))\} = i\}.$$

Then, the first successive minima $\lambda_1(\mathcal{L})$ is the Euclidean norm, or length, of the shortest nonzero vector in the lattice $\mathcal{L}$. For a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ of a lattice $\mathcal{L}(\mathbf{B}) \in \mathbb{R}^{n \times n}$, its *Gram-Schmidt Orthogonalization* $\mathbf{B}^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$ is defined as,

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \mathbf{b}_j^*,$$

where

$$\mu_{ij} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}, \text{ for } 1 \leq j \leq i \leq n.$$

and the factor matrix $\mu = \{\mu_{ij}\}_{1 \leq i,j \leq n}$ in which $\mu_{ij} = 0$ for $i < j$.

We define $\pi_i : \mathbb{R}^n \mapsto \text{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})^\perp$ as the projection on the orthogonal complement of the span of the first $i-1$ bases of $\mathbf{B}$, for all $i \in \{1, 2, \ldots, n\}$. $\pi_i(\mathbf{b}_j)$ is expressed as

$$\pi_i(\mathbf{b}_j) = \mathbf{b}_j^* + \sum_{k=i}^{j-1} \mu_{jk} \mathbf{b}_k^*, \quad \text{if} \quad i < j.$$

And, we define $\mathcal{L}_i^{(k)}$ as the lattice of rank k generated by the basis $[\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_{i+k-1})]$ in which $i+k \leq n+1$. Clearly, it is true that $\mathcal{L}_i^{(n-i+1)} = \pi_i(\mathcal{L})$, which implies the lattice of rank $n-i+1$ generated by basis $[\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_n)]$.

Thereby, we define a basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ as a *β-blockwise Korkin-Zolotarev basis*, or ***BKZ**-reduced basis*, if the following conditions hold:

1. Its $|\mu_{ij}| \leq 1/2$, for $1 \leq j < i \leq n$;
2. and $\pi_i(\mathbf{b}_i)$ is the shortest vector of the lattice $\mathcal{L}_i^{(\min(\beta, n-i+1))}$ under the Euclidean norm, or length, for $1 \leq i \leq n$.

You can refer to [16] for details of lattices.

2.2 *y*-Sparse Representations of Short Lattice Vectors

As defined above, a lattice vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ can be represented as $\mathbf{v} = \mathbf{Bx}$, in which $\mathbf{x}$ is an integer vector. Then $\mathbf{x}$ can corresponds to a specific lattice vector $\mathbf{v}$ under a basis $\mathbf{B}$. The y-sparse representation is to regards the lattice $\mathbf{v}$ from another point of view, which is endowed with some excellent properties.

Given a lattice basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ and its Gram-Schmidt orthogonalization $\mathbf{B}^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$ with its factor matrix $\mu = \{\mu_{ij}\}_{1 \leq i,j \leq n} \in \mathbb{R}^{n \times n}$ such that $\mathbf{B} = \mathbf{B}^* \mu^T$, for any vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$, or $\mathbf{v} = \mathbf{Bx}$, in which $\mathbf{x} = [x_1, \ldots, x_n] \in \mathbb{Z}^n$, we define another vector $\mathbf{t} = [t_1, \ldots, t_n] \in \mathbb{R}^n$ as, for $1 \leq i \leq n$,

$$t_i = \begin{cases} 0 & \text{for } i = n, \\ \sum_{j=i+1}^{n} \mu_{ji} x_j & \text{for } i < n. \end{cases}$$

and another vector $\mathbf{y} = (y_1, y_2, \ldots, y_n) \in \mathbb{Z}^n$ as, for $1 \leq i \leq n$,

$$y_i = \lfloor x_i + t_i \rceil.$$

Thereby, the definition establishes a one-to-one correspondence between a lattice vector $\mathbf{v}$ and its $\mathbf{y}$ as below:

$$\mathbf{y} \xleftrightarrow{\mathbf{y}=\mathbf{x}+\lfloor \mathbf{t} \rceil} \mathbf{x} \xleftrightarrow{\mathbf{v}=\mathbf{Bx}} \mathbf{v},$$

We call $\mathbf{v}$ is correspondent to $\mathbf{y}$, or, $\mathbf{v} \sim \mathbf{y}$.

We call such a representation as sparse because most of the elements in $\mathbf{y}$ corresponding to short lattice vectors under a **BKZ**-reduced basis are zero's. For example, the shortest vector of the 40-dimensional lattice (generated by $seed = 0$) in SVP challenge $\mathbf{v}$ is (-398 -305 -268 125 96 214 284 -108 37 -2 402 228 -243 -33 -76 -265 -3 558 323 552 -419 -408 217 2 440 375 -153 108 79 80 -299 -81 385 -80 -53 -294 -170 380 164 172), (and $\|\mathbf{v}\| = 1702$), and its corresponding $\mathbf{y}$ under its 5-**BKZ** reduced basis is (0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 -1 -1 0 0 0 1). We can see that only 4 nonzero elements in $\mathbf{y}$ and they are all distributed in the second half with absolute value of 1.

Actually, most y-sparse representations of the short vectors in the lattice under a **BKZ**-reduced basis shares this excellent property. Therefore, for an integer vector $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ corresponding to a short vector in a lattice under a **BKZ**-reduced basis, we have the following two heuristics as follows:

1. The first half integer elements $y_1, \ldots, y_{\lfloor n/2 \rfloor}$ in $\mathbf{y}$ are all zero's;
2. The absolute value $y_{\lfloor n/2 \rfloor+1}, \ldots, y_n$ of the second half integers in $\mathbf{y}$ is bounded by $\sqrt{\frac{\lambda_1}{\|\mathbf{b}_i^*\|}}$ instead of $\frac{\lambda_1}{\|\mathbf{b}_i^*\|}$ as stated in the theorem in [6], for $\lfloor n/2 \rfloor + 1 \leq i \leq n$;

For a more rigorous treatment of y-sparse representation, refer to [6].

3 y-Phase: Some Interesting Observations

In this section, we discuss the phase of the y-sparse representation of short lattice vectors and some interesting and useful observations.

As discussed in Section II, the y that is correspondent to a short vector $\mathbf{v}$ is sparse, i.e., only a fraction of the elements of $\mathbf{y}$ is nonzero, and, more precisely, the absolute value of the nonzero elements are small, say 1, 2, or 3, and most nonzero elements fall into the second half of vector $\mathbf{y}$. Like the y corresponding to the shortest vector of the random lattice of dimension 40 of SVP challenge, under a **BKZ**-reduced basis, as shown in Fig. 1, only 4 elements are nonzero (of absolute value 1), and they are all distributed in the second half (of index 21-40) of $\mathbf{y}$.

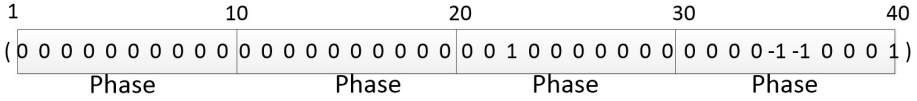

Fig. 1. The y-Phases of the 40-Dimensional Random Lattice of SVP Challenge

If we consider the y corresponding more deeply, we can see that nonzero elements are not distributed evenly in the last half elements: the last ten elements includes 3 nonzero elements while only 1 nonzero element falls between index 21 and 30 (the first half of the last 20 elements in $\mathbf{y}$). Therefore, we can divide the 40 elements of $\mathbf{y}$ evenly into 4 *phases*, each of which contains 10 elements. As in Fig. 1, the first two phases are all 0's, and the third phase has 1 nonzero element while the last phase obtains 3. We can notate the first phase as $\mathbf{y}_{1,\ldots,10}$, and $\mathbf{y}_{11,\ldots,20}$, $\mathbf{y}_{21,\ldots,30}$, and $\mathbf{y}_{31,\ldots,40}$ as the following 3 phases, or $\mathbf{y}_{l,\ldots,l+9}$ for the low index $l = 1, 11, 21, 31$. On the phases, we can define the ℓ_1-norm of the phases, like $\|\mathbf{y}_{l,\ldots,l+9}\|_1$, as a measure for the number of nonzero elements in the phase. Therefore, the ℓ_1-norm of the 4 phases of $\mathbf{y}$ corresponding the shortest vector of dimension 40 takes on an ascending order as $(0, 0, 1, 3)$.

Let us take a look at two more examples. As shown in Table 1, the $\mathbf{y}$ corresponding to the short vector of 95-dimensional random lattice, under a **BKZ**-reduced basis, is divided into 10 phases (with each consecutive 10 elements as a phase). We can see that the first 6 phases are of ℓ_1-norm 0, and that the ℓ_1-norm of the last 4 phases are 1,2,3,4 (an ascending order). Similarly, Fig. 2 shows that the $\mathbf{y}$ corresponding to the shortest vector $\mathbf{v}$ of the 134-dimensional random lattice takes on an ascending order of the ℓ_1-norms of its 14 phases. The ℓ_1-norm of the phases are $(0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 3, 5, 6, 7)$. Actually,

Phases	$\mathbf{y}_{l\ldots \min(l+10,n)}$	ℓ_1-norm of the Phases
$y_1, \ldots, y_{10}$	(0 0 0 0 0 0 0 0 0 0	0
$y_{11}, \ldots, y_{20}$	0 0 0 0 0 0 0 0 0 0	0
$y_{21}, \ldots, y_{30}$	0 0 0 0 0 0 0 0 0 0	0
$y_{31}, \ldots, y_{40}$	0 0 0 0 0 0 0 0 0 0	0
$y_{41}, \ldots, y_{50}$	0 0 0 0 0 0 0 0 0 0	0
$y_{51}, \ldots, y_{60}$	0 0 0 0 0 0 0 0 0 0	0
$y_{61}, \ldots, y_{70}$	1 0 0 0 0 0 0 0 0 0	1
$y_{71}, \ldots, y_{80}$	0 0 0 0 1 -1 0 0 0 0	2
$y_{81}, \ldots, y_{90}$	0 0 0 1 0 0 -1 0 0 -1	3
$y_{91}, \ldots, y_{95}$	1 0 0 1 -2)	4

Table 1. y-Phases of the Random Lattice of Dimension 95 ($\|\mathbf{v}\| = 2584$)

most of $\mathbf{y}$ corresponding to the short vectors in random lattices of SVP challenges obtains the same property as we observe above, Therefore, we concludes the two useful observations as follows:

Phases	$\mathbf{y}_{l\ldots \min(l+10,n)}$	ℓ_1-norm of the Phases
$y_1, \ldots, y_{10}$	(0 0 0 0 0 0 0 0 0 0	0
$y_{11}, \ldots, y_{20}$	0 0 0 0 0 0 0 0 0 0	0
$y_{21}, \ldots, y_{30}$	0 0 0 0 0 0 0 0 0 0	0
$y_{31}, \ldots, y_{40}$	0 0 0 0 0 0 0 0 0 0	0
$y_{41}, \ldots, y_{50}$	0 0 0 0 0 0 0 0 0 0	0
$y_{51}, \ldots, y_{60}$	0 0 0 0 0 0 0 0 0 0	0
$y_{61}, \ldots, y_{70}$	0 0 0 0 0 0 0 0 0 0	0
$y_{71}, \ldots, y_{80}$	0 0 0 1 0 0 0 0 0 0	1
$y_{81}, \ldots, y_{90}$	0 0 0 0 0 0 0 1 0 0	1
$y_{91}, \ldots, y_{100}$	0 0 0 0 0 1 0 0 0 0	1
$y_{101}, \ldots, y_{110}$	0 -1 0 0 1 0 0 1 0 0	3
$y_{111}, \ldots, y_{120}$	0 0 1 -1 0 0 2 1 0 0	5
$y_{121}, \ldots, y_{130}$	0 0 0 2 -1 0 -1 0 -1 1	6
$y_{131}, \ldots, y_{134}$	3 2 -1 -1)	7

Table 2. y-Phases of the Random Lattice of Dimension 134 ($\|\mathbf{v}\| = 2976$)

Observation 1 *Under a **BKZ**-reduced basis, only a fraction of the y-representation of the short vectors are nonzero elements (only $\frac{1}{10} \sim \frac{1}{6}$). And as stated in [6], the nonzero elements almost all fall into the second half of its y-representation with small absolute value of the nonzero elements. In other words, the ℓ_1-norm of the whole $\mathbf{y}$ is approximately between $\frac{n}{10}$ and $\frac{n}{6}$.*

Observation 2 *Divided into phases of length 10, the y-representation of the short vectors of random lattices in SVP challenge, under a **BKZ**-reduced basis, is of an ascending order under ℓ_1-norm.*

The two observations are useful in designing fast enumeration algorithm for SVP challenge as in the next section.

4 A Phase-Based Enumeration Algorithm for SVP challenge

In this section, we discuss in detail our phase-based enumeration algorithm, which applies the two useful observations in the last section.

4.1 Overview

Table 3. The Phase-Based Enumeration Algorithm for SVP

Input: A β-**BKZ** reduced basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ of a lattice $\mathcal{L}$.
Output: The Shortest Nonzero Vector $\mathbf{v}'$ in the lattice $\mathcal{L}(\mathbf{B})$

1. Compute $\mathbf{B}$'s Gram-Schmidt Orthogonalizations $\mathbf{B}^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$
and its factor matrix $\mu = \{\mu_{ij}\}_{1 \le i,j \le n}$;
2. Estimate the first minima λ_1 by Gaussian Heuristic;
3. Initialize $\alpha = (\alpha_1, \ldots, \alpha_n)$ as $\alpha_i \leftarrow 0$, for $1 \le i < \lfloor \frac{n}{2} \rfloor$,
and $\alpha_i \leftarrow \sqrt{\frac{\lambda_1}{\|\mathbf{b}_i^*\|}}$, for $\lfloor \frac{n}{2} \rfloor \le i \le n$;
4. Let $\mathbf{y} \leftarrow (0, 0, \ldots, 0)$, and $\mathbf{v}' \leftarrow \mathbf{b}_1$;
5. Let $m \leftarrow \lceil \frac{n}{2}/10 \rceil$; $\mathbf{d} \leftarrow \{d_1, d_2, \ldots, d_m\}$;//max ℓ_1-norms of phases
6. PhaseEnumeration$(0, \lfloor \frac{n}{2} \rfloor, \min(\lfloor \frac{n}{2} \rfloor + 10, n), \mathbf{y}, \mathbf{v}')$;
7. **Return** $\mathbf{v}'$;

The low ℓ_1-norm of y-representation of short vectors as in Observation 1 shows us that we need not waste time enumerating all the feasible lattice vectors, and that all we need to do is to enumerate all the $\mathbf{y}$ such that the ℓ_1-norm of the last half of $\mathbf{y}$ is $\frac{n}{10}$ to $\frac{n}{6}$, i.e., $\left\|\mathbf{y}_{\lfloor \frac{n}{2} \rfloor, \ldots, n}\right\|_1 \in [\frac{n}{10}, \frac{n}{6}]$. Moreover, Observation 2 reveals that the nonzero elements are distributed unevenly among the phases in the last half of $\mathbf{y}$, and, more precisely, the ℓ_1-norm of the phases in the second half takes on an ascending order. Then, we can enumerate phase by phase the $\mathbf{y}$ with ascending ℓ_1-norms for phases in the second half, which highlights the main idea of our phase-based enumeration algorithm.

Our phase-based enumeration predefines an integer sequence $\mathbf{d} = (d_1, \ldots, d_m)$ (m is the number of phases) of the maximum ℓ_1-norm for each phases (the sum of all the ℓ_1-norm are only $\frac{n}{10}$ to $\frac{n}{6}$), and it is clear that the sequence is in an ascending order. Then, the algorithm enumerates $\mathbf{y}$ recursively phase-after-phase with the i^{th} phase of ℓ_1-norm less than d_i. Finally, the algorithm returns the shortest lattice vector. Clearly, the enumeration procedure based on phase searches much less lattice vectors than the common enumeration algorithm which runs over all the lattice vectors, and the two observations in the last section ensure that the shortest vector will not be omitted in such phase-based method.

Table 3 shows the pseudo-codes of the main procedure of our phase-based enumeration algorithm. As in the table, given a lattice basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ (which is BKZ-reduced), the procedure first computes the Gram-Schmidt orthogonalzation $\mathbf{B}^*$ and its μ. Estimating the first minima λ_1 of the lattice, the procedure calculates the bounds $\alpha = (\alpha_1, \ldots, \alpha_n)$ of $\mathbf{y}$ as 0 for the first half of the elements and $\sqrt{\frac{\lambda_1}{\|\mathbf{b}_i^*\|}}$ for the second half, based on theorem and heuristics in [6]. Then, the procedure initialize the integer vector $\mathbf{y}$ as all-zero and the shortest vector $\mathbf{v}'$ as the first vector $\mathbf{b}_1$ of the basis $\mathbf{B}$. Note that we only take the phases in the second half into consideration, and, so, the number of phases m is set as $\lceil \frac{n}{2}/10 \rceil$. After predefining the integer sequence $\mathbf{d} = (d_1, \ldots, d_m)$ in an ascending order with $\sum_{i=1}^{m}(d_i) \le \frac{n}{6}$, we starts to enumerates all the $\mathbf{y}$ such that the ℓ_1-norm of the i^{th} phase is less than d_i using the subroutine PhaseEnumeration(), which will be described in the following subsection.

4.2 PhaseEnumeration() and PhaseEnumerationBottom()

In this subsection, we discuss two subroutines PhaseEnumeration() and PhaseEnumerationBottom(), which constitute the main body of our phase-base enumeration algorithm.

Table 4 shows that pseudo-codes of PhaseEnumeration(). As shown in the table, it is an enumeration procedure using "recursion". Given the parameters of phase number p, the low index of l, the high index h, and the current $\mathbf{y}$, which has been set before the low index l, the procedure searches the current phase such that the ℓ_1-norm of current phase is less than the predefined d_p, i.e., $\|\mathbf{y}_{l,\ldots,h-1}\|_1 \le$

Table 4. PhaseEnumeration()

Input: the phase p, low index l, high index h, an integer vector $\mathbf{y} = (y_1, \ldots, y_n)$ and a lattice vector $\mathbf{v}'$;
Output: The shortest nonzero vector $\mathbf{v}' = (v'_1, \ldots, v'_n)$.

1. **If** $h \geq n$ **then** PhaseEnumerationBottom(l, $\mathbf{y}$, $\mathbf{v}'$, l);
//Enter the bottom procedure
2. **else for** $i \in \{l, \ldots, h\}$ **do**
   (a) **if** $i = h$ **then** PhaseEnumeration($p+1$, h, $\min(h+10, n)$, $\mathbf{y}$, $\mathbf{v}'$); //Enter the next phase
   (b) **else for** $j \in \{-\alpha_i, \ldots, \alpha_i\}$ **do**
      (i) **if** $\|\mathbf{y}_{h-10,\ldots,i-1}\|_1 + |j| \leq d_p$ **then**
         (1) Let $y_i \leftarrow j$;
         (2) **if** $\|\mathbf{y}_{h-10,\ldots,i}\|_1 = d_p$ **then** //enter the next phase
            PhaseEnumeration($p+1$, h, $\min(h+10, n)$, $\mathbf{y}$, $\mathbf{v}'$);
         (3) **else** //continue with the current phase
            PhaseEnumeration(p, $i+1$, h, $\mathbf{y}$, $\mathbf{v}'$);
         (4) Let $y_i \leftarrow 0$;
3. **Return** $\mathbf{v}'$.

$d_p - \|\mathbf{y}_{h-10,\ldots,l-1}\|_1$, and, then, it invokes itself or the bottom procedure PhaseEnumerationBottom() to search the next phase.

Entering the procedure PhaseEnumeration(), it invokes the bottom subroutine PhaseEnumerationBottom() if the high index h is larger than the lattice rank n. If not, the procedure has not arrived at the last phase, and (at Line 2) it searches all the index between l and h. If the variable i grabs the high index h, the procedure has enters the phase that follows, and, then, it calls itself with parameters of $p+1$, h, $h+10$. If i grabs an index less than h, the procedure chooses a j as the nonzero value for y_i within bounds $[-\alpha_i, \alpha_i]$. If the newly chosen has not made the ℓ_1-norm of the current phase larger than d_p, y_i is set as j, and the procedure either enters another phase if the ℓ_1-norm of the current phase is equal to d_p, or, otherwise, continues with current phase by invoking itself with the low index l replaced by $i+1$ (searches between index $i+1$ and h). Thereby, the procedure searches all the $\mathbf{y}$ that satisfies the requirements of the predefined maximum ℓ_1-norm $\mathbf{d}$.

Table 5. PhaseEnumerationBottom()

Input: the low index l, an integer vector $\mathbf{y} = (y_1, \ldots, y_n)$, a lattice vector $\mathbf{v}'$, and the low index l' of the last phase;
Output: The shortest nonzero vector $\mathbf{v}' = (v'_1, \ldots, v'_n)$.

1. **for** $i \in \{l, \ldots, n\}$ **do**
   (a) **for** $j \in \{-\alpha_i, \ldots, \alpha_i\}$ **do**
      (i) **if** $\|\mathbf{y}_{l',\ldots,i}\|_1 + |j| \leq d_m$ **then**
         (1) Let $y_i \leftarrow j$;
         (2) Compute $\mathbf{v} \sim \mathbf{y}$ using $\mathbf{B}^*$ and μ;
         (3) **if** $\|\mathbf{v}\| < \|\mathbf{v}'\|$ **then** $\mathbf{v}' \leftarrow \mathbf{v}$;
         (4) **if** $\|\mathbf{y}_{l',\ldots,i}\|_1 < d_m$ **then**
            PhaseEnumerationBottom($i+1$, $\mathbf{y}$, $\mathbf{v}'$, l');
         (5) Let $y_i \leftarrow 0$.
2. **Return** $\mathbf{v}'$.

Table 5 shows the bottom procedure of our phase-based enumeration algorithm. Given the low index l, the bottom procedure searches all the index between l and the lattice rank n, and updates the shortest vector $\mathbf{v}'$ if it finds shorter vectors. Entering the bottom procedure, the index i runs over $[l, \ldots, n]$ and j

chooses all the value between $-\alpha_i$ and α_i for y_i. If the newly-generated nonzero element does not make the ℓ_1-norm of the current phase go beyond d_m, the y_i is set as j and we go on with updating the shortest vector $\mathbf{v}'$ if better solution comes to light. After that, if the ℓ_1-norm of the current phase is still less than d_m, or, in other words, the current phase can still contain more nonzero elements, the procedure continues with the phase by invoking itself with a new low index $i+1$.

The two subroutines recursively searches all the $\mathbf{y} = (y_1, \ldots, y_n)$ which satisfies that the ℓ_1-norm of each phases is less than a predefined maximum ℓ_1-norm sequence $\mathbf{d} = (d_1, \ldots, d_n)$, thereby finding the shortest vector in the lattice by searching relatively few lattice vectors.

5 Experimental Results

In this section, we compare the running times of our phase-based enumeration algorithm with the seminal Kannan-Helfrich Enumeration algorithm [14] and ℓ_1-norm based enumeration algorithm, and all the three algorithms are implemented using C++ with Victor Shoup's Number Theory Library (NTL) version 6.0.0 [26]. Experiments are performed on a workstation with 16 Intel Xeon 2.4Ghz CPUs and 16G RAM under a Red Hat Linux Server release 5.4. All the experiments are run on the SVP challenge benchmarks [24] of dimension 20-95, and all the random bases are generated using their random lattice generator with random $seed = 0$. All the bases are preprocessed by a **BKZ** subroutine with their block size $\beta < \frac{n}{4}$ (n is their rank of the lattices). The Kannan-Helfrich enumeration algorithm is, actually, to search all the feasible lattice vectors in the hypersphere of $\sum_{i=1}^{n} y_i^2 \left\|\mathbf{b}_i^*\right\|^2 \leq \lambda_1$, which has been deeply researched into in the recent years [12][11][21], and the ℓ_1-norm based enumeration algorithm is to search $\mathbf{y}$ with the last half of ℓ_1-norm less than $\frac{n}{10} \sim \frac{n}{6}$ (based on Observation 1), or, in other words, the 1-phase enumeration algorithm. Our phase-based enumeration algorithm runs with the predefined maximum ℓ_1-norm for the second half phases $\mathbf{d}$ as $(1, 2, \ldots, m)$.

Table 6. Running Time Comparison of Enumeration Algorithms under Dimension-40 Random Lattices

Running Time	Enumeration Algorithms
3017.3270s	the Kannan-Helfrich Enumeration Algorithm
12.0048s	the ℓ_1-Norm Based Enumeration Algorithm
1.47209s	the Phase-Based Enumeration Algorithm

The three algorithms are run under the random lattice basis of dimension 40 (with a preprocessing of 5-BKZ reduction), the running time of the three are compared in Table 5. As shown in the table, the Kannan-Helfrich enumeration algorithm find the shortest vector (of Euclidean norm 1702) with over 3000 seconds, and the ℓ_1-norm based enumeration algorithm uses approximately 12 seconds, and our phase-enumeration algorithm only 1.4 seconds, which is over 3000 faster than Kannan-Helfrich algorithm and 10 times faster than ℓ_1-norm based algorithm.

We continue to run the three algorithms on the random lattices of dimension 20-95 and the running time comparison is given in Fig. 2. As shown in the figure, the Kannan-Helfrich enumeration algorithm consumes the most running time: it consumes over hundreds of seconds under lattice basis of dimension 20, and it only runs up to lattice basis of dimension 70 with unbearably long time. The ℓ_1-norm enumeration algorithm runs faster: nearly 1.3 seconds for lattice of dimension 20, and runs up to the 80-dimensional lattice. The second algorithm runs over some lattices of dimension 20-80 and approximately tens or hundreds of times faster than Kannan-Helfrich algorithm. Finally, our phase-based enumeration algorithm outperforms the other two enumeration algorithms: it runs less than 1 seconds for lattice of dimension 20, and runs through most lattices of dimension 20-95, and it enjoys a thousands of times speedup over Kannan-Helfrich algorithm, and also much better than the 1-phase enumeration. It is clear that the running time of our phase-based algorithm depends heavily on the predefined integer sequence $\mathbf{d}$ of the

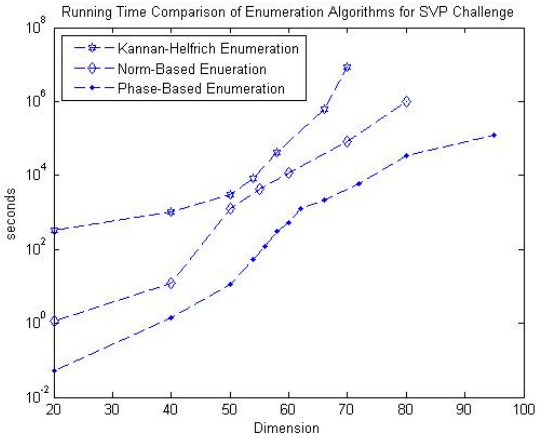

Fig. 2. Running Time Comparison of the Enumeration Algorithms for SVP Challenge

maximum ℓ_1-norm for phases, and we believe that an improvement can be achieved by choosing a better $\mathbf{d}$.

The experimental results implies that the phase-based enumeration algorithm gains a great advantage over the other two enumeration algorithm in running time, and it never misses any optimum solution though searching only a fraction of the feasible lattice vectors.

6 Conclusion

In this paper, we propose a novel phase-based enumeration algorithm for shortest vector problems based on the two interesting and useful observations for the short lattice vectors. The experimental results show that the phase-based enumeration greatly outperforms the other famous enumeration algorithms in time complexity under the most random lattice bases in SVP challenge benchmarks[24]. In conclusion, it is practically an excellent algorithm for SVP challenge.

7 Future Work

In the future, we will attempt to give a theoretical and quantitative analysis of time complexity of our phase-based enumeration algorithm compared to Kannan-Helfrich algorithm, and, at the same time, we will run our phase-based enumeration algorithm under the lattices of much higher dimensions, like of dimension 136, with more delicately-chosen predefined maximum phase-wise ℓ_1-norms.

References

1. Ajtai, M. Generating hard instances of lattice problems (extended abstract). In *STOC* (1996), pp. 99–108.
2. Ajtai, M. The shortest vector problem in ℓ_2 is np-hard for randomized reductions. *Proceeding of the* 30^{th} *Symposium on the Theory of Computing (STOC 1998)*, 284-406 (1998).
3. Ajtai, M., and Dwork, C. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC* (1997), pp. 284–293.
4. Ajtai, M., Kumar, R., and Sivaumar, D. A sieve algorithm for the shortest lattice vector problem. *Proceedings of the* 33^{th} *annual ACM symposium on Theory of computing (STOC'01) 33*, 601-610 (2001).
5. Chen, Y., and Nyuyen, P. Q. Bkz2.0: Better lattice security estimates. *Advances in Cryptology C ASIACRYPT 2011, Lecture Notes in Computer Science 7073*, 1-20 (2011).
6. Ding, D., Zhu, G., and Wang, X. A genetic algorithm for searching shortest lattice vector of svp challenge. Cryptology ePrint Archive, Report 2014/489, 2014. `http://eprint.iacr.org/`.

7. Gamma, N., Nguyen, P. Q., and Regev, O. Lattice enumeration using extreme pruning. *Advances in Cryptology C EUROCRYPT 2010, Lecture Notes in Computer Science 6110*, 257-278 (2010).
8. Goldreich, O., Goldwasser, S., and Halevi, S. Public-key cryptosystems from lattice reduction problems. In *CRYPTO* (1997), pp. 112–131.
9. Hanrot, G., Pujol, X., and Stehlé, D. Analyzing blockwise lattice algorithms using dynamical systems. *Advances in Cryptology-CRYPTO 6841*, 447-464 (2011).
10. Hanrot, G., and Stehlé, D. Analyzing blockwise lattice algorithms using dynamical systems. *Advances in Cryptology-CRYPTO 4622*, 170-186 (2007).
11. Hanrot, G., and Stehlé, D. Improved analysis of kannans shortest lattice vector algorithm. *Advances in Cryptology - CRYPTO 2007, Lecture Notes in Computer Sciences 4622*, 170-186 (2007).
12. Helfrich, B. Algorithms to construct minkowski reduced and hermit reduced bases. *Theoretical Computer Sciences 41*, 125-139 (1985).
13. Kannan, R. Improved algorithms for integer programming and related lattice problems. *Proc. of the* 15^{th} *Symposium on the Theory of Computing (STOC1983) 15*, 99-108 (1983).
14. Kannan, R. Minkowski's convex body theorem and integer programming. *Mathemaatics of Operations Research 12*, 415-440 (1987).
15. Lenstra, A. K., Lenstra, H. W., and Lavász, L. Factoring polynomials with rational coefficients. *Mathematische Annalen 261*, 513-534 (1982).
16. Micciancio, D., and Goldwasser, S. *Complexity of Lattice Problems: A Cryptographic Perspective*. Kluwer Academic Publishers, 2002.
17. Micciancio, D., and Regev, O. Worst-case to average-case reductions based on Gaussian measure. In *Proceedings of the 45rd annual symposium on foundations of computer science - FOCS 2004* (Rome, Italy, Oct. 2004), IEEE, pp. 371–381. Journal verion in SIAM Journal on Computing.
18. Micciancio, D., and Regev, O. Worst-case to average-case reductions based on Gaussian measure. *SIAM Journal on Computing 37*, 1 (2007), 267–302. Preliminary version in FOCS 2004.
19. Micciancio, D., and Regev, O. Lattice-based cryptography. *Proceeding of the Post-Quantum Cryptography (PQC'09)*, 147-191 (2009).
20. Micciancio, D., and Voulgaris, P. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. *Proceedings of the* 42^{th} *annual ACM symposium on Theory of computing (STOC'10) 42*, 351-358 (2010).
21. Micciancio, D., and Walter, M. Fast lattice point enumeration with minimal overhead. Cryptology ePrint Archive, Report 2014/569, 2014. `http://eprint.iacr.org/`.
22. Nguyen, P. Q., and Vidick, T. Sieve algorithms for the shortest vector problem are practical. *Journal of Mathematical Cryptography 2*, 2(181C207) (2008).
23. Regev, O. New lattice-based cryptographic constructions. *J. ACM 51*, 6 (2004), 899–942.
24. Schneider, M., and Gamma, N. Svp challenge, 2010. `http://www.latticechallenge.org/svp-challenge/`.
25. Schnorr, C. P. A hierarchy of polynomial lattice basis reduction algorithms. *Theoretical Computer Sciences 53*, 201-224 (1987).
26. Shoup, V. Number theory c++ library (ntl) vesion 6.0.0, 2010. `http://www.shoup.net/ntl/`.
27. van Emde Boas, P. Another np-complete partition problem and the complexity of computing short vectors in a lattice. *Technical Report, Mathematisch Instituut, Universiteit van Amsterdam 81-04* (1981).
28. Wang, X., Liu, M., Tian, C., and Bi, J. Improved nguyen-vidick heuristic sieve algorithm for shortest vector problem. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security* (2011), ACM, pp. 1–9.