Orthogonalized Lattice Enumeration for Solving SVP

Zhongxiang Zheng[1], Xiaoyun Wang[2], Yang Yu[1]

[1] Department of Computer Science and Technology,Tsinghua University, 100084, Beijing, China,
[2] Institute for Advanced Study,Tsinghua University, 100084, Beijing, China,
`xiaoyunwang@mail.tsinghua.edu.cn`

Abstract. In 2014, the orthogonalized integer representation is presented indepently by Dan Ding etc and Fukase etc to solve SVP respectively by genetic algorithm and sampling technique, and both work have achieved very good results. In this paper, we first study the probability distribution of the shortest vector with orthogonalized integer representations, and give a new enumeration method based on the representations, called orthognalized enumeration. Besides, we present a new BKZ method, called MBKZ, which alternately uses orthognalized enumeration and traditional enumeration. Our method has the obvious advantage in time complexity compared to the previous ones which achieved exponential speedups both in theory and in practice. Furthermore, a new technique of reducing enumeration space is described, we can't find a quantitative analysis about success probability but it is effective in practice.

Key words: Lattice-based, SVP, sparse representations, enumeration, BKZ

1 Introduction

A lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^m$. It is generated by n linearly independent vectors $\mathbf{b_1}, \dots, \mathbf{b_n}$ in $\mathbb{R}^m$, and the integer n is the dimension of $\mathcal{L}$. The discreteness of lattices implies that there exists a nonzero vector with the smallest non-zero Euclidean norm in each lattice, denoted as λ_1. There are two famous computational problems in lattices:

– Shortest Vector Problem (SVP): Given a basis of lattice $\mathcal{L}$, find the shortest nonzero vector in the lattice.

– Closest Vector Problem (CVP): Given a basis of lattice $\mathcal{L}$ and a target vector, find the lattice vector closest to the target.

The two hard problems SVP and CVP are of prime importance to the lattice cryptography in the past 20 years. There are two main types of algorithms for solving SVP and CVP. One is the exponential space algorithms, and the other is algorithms with polynomial space. The first exponential algorithm much attracted cryptographic community is the randomized sieve algorithm proposed in 2001 by Ajtai, Kumar and Sivakumar[3]. The sieve method reduces the upper bound of the time to $2^{O(n)}$ at the cost of $2^{O(n)}$ space. It has been developed into some improved sieves including heuristic methods in recent years[25][26][34]. Another important work is the deterministic algorithm with $O(2^{2n})$ time and $O(2^n)$ space given by Micciancio and Voulgaris[23]. The latest progress is the randomized algorithm with $O(2^n)$ time using the discrete Gaussian sampling method[1], which is the first randomized algorithm(without heuristic assumption) faster than the deterministic algorithm.

For polynomial-space algorithms, two popular techniques are used, one is *lattice reduction*, including the famous LLL alogrithm[20], HKZ reduction[17] and BKZ reduction[30]. The other important technique is *enumeration technique* which is an exact algorithm to find the shortest vectors in the reduced space. Two techniques are complementary. Usually, the reductions cannot output the shortest vectors in high dimension lattices, it is used to find sufficient short vectors to ensure the enumeration search work efficiently. Whereas the enumeration technique on sublattices with low dimensions is repeatedly used as a subroutine, and aims to improve the output quality of the reductions greatly. The first polynomial-space algorithm was provided by Kannan[17] in 1980s which is based on LLL-reduced basis and HKZ reduction, the complexity of Kannan's enumeration is $2^{O(nlogn)}$. The more accurate analysis complexity with $d^{\frac{d}{2}+o(d)}$ was given by Helfrich [15], and the further improved complexity bound of $d^{\frac{d}{2e}+o(d)}$ was proved by Hanrot and Stehl*é* in [12]. Another popular polynomial-space algorithm is Schnorr-Euchner enumeration based on the BKZ reduction, and its enumeration complexity is estimated by Gama, Nguyen and Regev in [10] as $\sum_{l=1}^{n} q^{(n-l)l/2} 2^{O(n)}$($q$ is a constant decided by basis). Although the complexity of Schnorr-Euchner enumeration seems higher than Kannan's enumeration, in fact it is widely used in practical implementations, and becomes a fundamental tool in the popular mathematical library NTL[33]. Many security estimates[21][24][27][31] and SVP searching[13][19] of lattice cryptosystems are based on BKZ implementation of NTL. Therefore, the further improvement of the enumeration becomes very important for SVP searching. Gama, Nguyen and Regev proposed an improvement to enumeration with extreme pruning technique and the speedup is exponential[10]. Chen and Nguyen used the technique to improve the BKZ algorithm named BKZ 2.0[6]. Further improvements of BKZ are also achieved in progressive strategy[4] and result predictions[22] in recent years.

The main purpose of our paper is to describe a new enumeration method, and we call it othogonalized enumeration. Our idea is motivated by the integer sparse representation of the shortest vector regarding the Gram-Schmidt basis. The idea of using the sparsity of the shortest vector's representation regarding the Gram-Schmidt basis can be trace back to Schnorr's Random Sampling Algorithm[29], and was expanded independently by Dan Ding etc[7] in a genetic algorithm and Fukase etc[9] in a sampling algorithm. Genetic algorithm is originated by Holland[16] in 1975, and has been used to solve optimization problems, like timetabling, scheduling and engineering problems[5][8][11]. The main clue of the method is to transform the shortest lattice vector into a new integer vector corresponding to the Gram-Schimdt orthogonal basis. The new integer vector is sparse, which means that it often consists of $-1, 0, 1$; most of elements in the vector are 0s; most nonzero elements always locate on the tail of the vector. With the help of the sparse representations, vectors can act as chromosomes to start a genetic algorithm and search SVP successfully. While Random Sampling Reduction was proposed by Schnorr in 2003[29], and M. Fukase and K. Kashiwabara [9] extended Schonrr's sampling technique by the integer sparse representation (also denoted as natural number representation), they used the technique combined with restricting reduction techniques to solve SVP challenge in a higher demension than ever.

Our contributions. Our contribution is to give a more efficient enumeration utilizing the sparse integer representation of shortest vectors . Firstly, we study the distribution of the shortest vector with the orthogonalized integer representation, and get a probability distribution through Monte-Carlo Simulation. With the support of the probability distribution, we describe a natural enumeration method which is called orthognalized enumeration. Orthognalized enumeration demands for a new input parament k. In more details, the main purpose of our method is to reduce the enumeration trials by considering the relationship between the sparse integer vector $\mathbf{y} = (y_1, ..., y_n)$ regarding the Gram-Schmidt basis $\mathbf{B}^*$ and the original searched vector $\mathbf{x} = (x_1, ..., x_n)$ with the basis $\mathbf{B}$. By choosing a theoretical estimated threshold k such that $y_i (1 \leq i \leq n-k)$ equals to 0 with high probability, we can cut the searching space for the shortest vector $\mathbf{v}$ into $(x_{n-k+1}, ..., x_n)$. For every $(x_{n-k+1}, ..., x_n)$, we can compute $(y_{n-k+1}, ..., y_n)$ according to the one-to-one correspondence between $\mathbf{x}$ and $\mathbf{y}$, and then compute the unique values of $x_i, \forall i = 1, ..., n-k$. This means that, our enumeration only searches k-dimension subspace instead of the previous method (full enumeration, linear enumeration, extreme enumertaion etc) with n-dimension space. This feature makes the method ideal for use in high dimension. Furthermore, we introduce a new BKZ method called MBKZ by alternately using orthognalized enumeration and traditional enumeration. Monte-Carlo Simulation is used to estimate the expectations of number of nodes by using different enumeration methods, and the result shows that exponential speedup can be achieved by our new method. We make experiments in basis with dimension up to 140, and experimental results are consistent with our theoretical estimation.

Furthermore, we describe an interesting technique to reduce the searching space of enumeration with non-negligible probability. Although we cannot find a quantitative analysis about success probability, it works well in experiments.

The rest of the paper is organized as follows: In Section 2, we provide some necessary backgrounds on lattice and describe the orthogonalized integer representations. In Section 3, we introduce our basic orthogonalized enumeration, and estimate the success probability. Section 4 introduces the details of MBKZ. A further improvement of enumeration is given in Section 5. Finally a conclusion is given in Section 6.

2 Preliminaries

Lattice. A lattice $\mathcal{L}$ is defined as the set of all integral combinations of n linearly independent vectors $\mathbf{b_1}, \ldots, \mathbf{b_n}$ in $\mathbb{R}^m (m \geq n)$, these linearly independent vectors are a basis of $\mathcal{L}$. The integer n is the dimension of $\mathcal{L}$ and $vol(\mathcal{L})$ is the volume or determinant of $\mathcal{L}$:

$$\mathcal{L}(\mathbf{b_1}, \ldots, \mathbf{b_n}) = \{\sum_{i=1}^{n} x_i \mathbf{b_i}, x_i \in \mathbb{Z}\}.$$

The basis of $\mathcal{L}$ is not unique, but they have the same number n of elements and the same volume $vol(\mathcal{L})$. When $m = n$, the lattice is called full-rank.

Shortest vector. $\|\mathbf{v}\|$ is denoted as the Euclidean norm of a vector $\mathbf{v} \in \mathbb{R}^m$, the non-zero vector with smallest Euclidean norm in a lattice $\mathcal{L}$, which is called the shortest vector, and its norm is the first minimum $\lambda_1(\mathcal{L})$.

$Gram - Schmidt\ orthogonalization$. The Gram-Schmidt orthogonalization is a method for orthogonalizing a set of vectors in an inner product space, most commonly the Euclidean space $\mathbb{R}^n$. For a basis $\mathbf{B} = [\mathbf{b_1}, \ldots, \mathbf{b_n}]$, the Gram-Schmidt process generates an orthogonal set $\mathbf{B}^* = [\mathbf{b_1^*}, \ldots, \mathbf{b_n^*}]$ as follows:

$$\mathbf{b_i^*} = \mathbf{b_i} - \sum_{j=1}^{i-1} \mu_{ij} \mathbf{b_j^*}. \tag{1}$$

where $u_{ij} = \frac{\langle \mathbf{b_i}, \mathbf{b_j^*} \rangle}{\langle \mathbf{b_j^*}, \mathbf{b_j^*} \rangle},\ for\ 1 \leq j < i \leq n.$

The Gram-Schmidt procedure projects each $\mathbf{b_i}$ to the space orthogonal to the space spanned by $\mathbf{b_1^*}, \ldots, \mathbf{b_{i-1}^*}$, and keeps the determinant unchanged, $det(\mathbf{B}) = \prod_{i=1}^{n} \|\mathbf{b_i^*}\|$.

BKZ. BKZ is a *lattice reduction* technique with blockwise algorithms[30]. It applies successive elementary transformations to an input basis, and outputs a BKZ-reduced basis whose vectors are shorter and more orthogonal. For a blocksize $\beta \geq 2$ and a basis $\mathbf{B} = (\mathbf{b_1}, \ldots, \mathbf{b_n})$ of a lattice, it firstly applies

LLL to $\mathbf{B}$ and then applies enumeration to each block $\mathbf{B}_{[j,min(j+\beta-1,n)]}$ which is denoted as $L_{[j,min(j+\beta-1,n)]}$ for $1 \le j \le n$ to find $\mathbf{v} = (v_1, \cdots, v_n)$ which ensures $\|\pi_j(\sum_{i=j}^{min(j+\beta-1,n)} v_i\mathbf{b_i})\| = \lambda_1(L_{[j,min(j+\beta-1,n)]})$, π_j is the orthogonal projection on $(\mathbf{b_1}, \cdots, \mathbf{b_{i-1}})^{\perp}$. And after finding vectors shorter than any vectors in the basis, LLL is called to update the basis. These steps would be repeated several times until no vector shorter than the basis vectors can be found in each block, and the final basis is the output. While it is a popular observation that the output basis seems to have $\|\mathbf{b_i^*}\|/\|\mathbf{b_{i+1}^*}\| \approx q$, and q depends on the quality of BKZ, which is also mentioned in [10]. All the lattice basis $\mathbf{B}$ discussed throughout this paper are BKZ-reduced lattices unless specified otherwise.

Gaussian Heuristic. The Gaussian Heuristic is used to estimate the distribution of vectors in a lattice. It assumes that the number of points in a set is related to its volume. Given a lattice $\mathcal{L}$ and a subset $\mathcal{S} \subseteq \mathcal{L}$, the number of points in $\mathcal{L} \cap \mathcal{S}$ is approximately $vol(\mathcal{S})/vol(\mathcal{L})$.

Heuristic 2. The distribution of the coordinates of the shortest vector $\mathbf{v}$, when written in the normalized Gram-Schmidt basis $(\mathbf{b_1^*}/\|\mathbf{b_1^*}\|, ..., \mathbf{b_n^*}/\|\mathbf{b_n^*}\|)$ of the input basis, look like those of a uniformly distributed vector of norm $\|\mathbf{v}\|$.

Heuristic 3. The distribution of the normalized Gram-Schmidt orthogonalization $(\mathbf{b_1^*}/\|\mathbf{b_1^*}\|, ..., \mathbf{b_n^*}/\|\mathbf{b_n^*}\|)$ of a random reduced basis $(\mathbf{b_1}, \ldots, \mathbf{b_n})$ looks like that of a uniformly distributed orthogonal matrix.

Here we use the same heuristic 2 and heuristic 3 as [10], so as to assume the coordinates $\mathbf{x}$ of the target vector $\mathbf{v}$ in the orthogonal basis $(\mathbf{b_n^*}/\|\mathbf{b_n^*}\|, ..., \mathbf{b_1^*}/\|\mathbf{b_1^*}\|)$ distribute like a uniform vector where $\|\mathbf{x}\| = \|\mathbf{v}\|$.

Orthogonalized Integer Representations. A vector $\mathbf{v}$ can be represented as the integral combination of the basis vectors $\mathbf{v} = \mathbf{Bx}$. With the help of orthogonalized integer representations[7][9], $\mathbf{x}$ can be transformed into an integer vector $\mathbf{y}$ with $\mathbf{B}^*$ by the following way: Given a basis $\mathbf{B} = [\mathbf{b_1}, \ldots, \mathbf{b_n}]$ and its Gram-Schimdt orthogonalization $\mathbf{B}^* = [\mathbf{b_1^*}, \ldots, \mathbf{b_n^*}]$ with the matrix $\mathbf{R} = [\mu_{ij}]_{1 \le i,j \le n} \in \mathbb{R}^{n \times n}$, such that $\mathbf{B} = \mathbf{B}^*\mathbf{R}$. For any vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$, and $\mathbf{v} = \mathbf{Bx}$, in which $\mathbf{x} = (x_1, \ldots, x_n)$, define a vector $\mathbf{t} = (t_1, \ldots, t_n) \in \mathbb{R}^n$ as

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

and compute $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{Z}^n$ as,

$$y_i = \lfloor x_i^* \rceil = \lfloor x_i + t_i \rceil = x_i + \lfloor t_i \rceil, for\ 1 \le i \le n.$$

Since $x_i \in \mathbb{Z}$, we can establish a one-to-one correspondence between $\mathbf{x}$ and $\mathbf{y}$, and also a one-to-one correspondence between $\mathbf{v}$ and $\mathbf{y}$. We call $\mathbf{y}$ the orthogonalized integer representations in this paper:

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

3 Enumeration

In this section, we first recall Full Enumeration and Extreme Prunging Enumeration, and then present our main contribution, the Orthogonalized Enumeration.

3.1 Full Enumeration

Given a Gram-Schmidt orthogonalized basis $\mathbf{B}^*$ and an upper bound R, the full enumeration method[30] enumerates $x_n, x_{n-1}..., x_1$ of $\mathbf{x}$ successively which satisfy(the enumeration algorithm can be found in appendix A):

$$x_n^2 \|\mathbf{b_n^*}\|^2 \leq R^2,$$

$$(x_{n-1} + \mu_{n,n-1}x_n)^2 \|\mathbf{b_{n-1}^*}\|^2 \leq R^2 - (x_n)^2 \|\mathbf{b_n^*}\|^2,$$

$$(x_i + \sum_{j=i+1}^{n} \mu_{j,i}x_j)^2 \|\mathbf{b_i^*}\|^2 \leq R^2 - \sum_{j=i+1}^{n} l_j.$$

Here $l_i = (x_i + \sum_{j>i} x_j \mu_{j,i})^2 \|\mathbf{b_i^*}\|^2$.

The number of nodes that need to be searched is decided by the size of enumeration tree. The total number of tree nodes N_e is estimated as $N_e \approx \sum_{l=1}^{n} H_l$[10], where H_l is the estimated number of nodes at level l, and:

$$H_l = \frac{1}{2} \cdot \frac{V_l(R)}{\prod_{i=n+1-l}^{n} \|\mathbf{b_i^*}\|} \approx q^{(n-l)l/2} 2^{O(n)}.$$

where $V_l(R) = R^l \cdot \frac{\pi^{l/2}}{\Gamma(l/2+1)}$ and $\|\mathbf{b_i^*}\| / \|\mathbf{b_{i+1}^*}\| \approx q$. And when $l = n/2$, H_l gets the maximum value as $q^{n^2/8} 2^{O(n)}$.

3.2 Extreme Pruning Enumeration

Extreme Pruning Enumeration improves Full Enumeration by replacing the bound R by a serial of bounding functions $R_1, ..., R_n$. Two strategies of choosing bounding functions are often used. One is linear pruning with success probability about $1/n$, and the other is extreme pruning with success probability extreme small.

The number of nodes in enumerating tree of extreme pruning is:

$$N_{ext} = 1/2 \sum_{t=1}^{n} \frac{V_{R_1,\cdots,R_t}}{\prod_{i=n+1-t}^{n} \|\mathbf{b_i^*}\|},$$

where $V_{R_1,\cdots,R_t} = V_t(R_t) \cdot \underset{\mathbf{u} \sim \mathbf{Ball_t}}{\mathbf{Pr}} (\forall j \in [1,t], \sum_{i=1}^{j} u_i^2 \leq \frac{R_j^2}{R_t^2})$.

For orthogonalized enumeration with pruning, the number of nodes in enumerating tree is :

$$N_{ort} = 1/2 \sum_{t=1}^{n-k} \frac{V_{R_1,\cdots,R_t}}{\prod_{i=n+1-t}^{n} \|\mathbf{b_i^*}\|}.$$

where $V_{R_1,\cdots,R_t} = V_t(R_t) \cdot \underset{\mathbf{u \sim Ball_t}}{\mathbf{Pr}} (\forall j \in [1,t], \sum_{i=1}^{j} u_i^2 \leq \frac{R_j^2}{R_t^2})$.

The linear pruning can reduce the number of nodes searched by a factor of 1.189^n and extreme pruning can achieve a speedup of 1.414^n compared to full enumeration according to [10].

3.3 Orthogonalized Enumeration Algorithm

The idea of the orthogonalized enumeration is to make use of orthogonalized integer representations, which has been used in solving SVP in many methods including sampling[9] and genetic algorithm[7]. However, as a new efficient method, enumeration based on orthogonalized integer representations hasn't ever been designed. So we provide the orthogonalized enumeration in order to make a further improvement for enumeration.

The orthogonalized enumeration demands for a new input k, and the number of nodes enumerated is limited by k, which makes it different from previous enumeration methods. By choosing a proper k such that $y_i (1 \leq i \leq n-k)$ equals to 0 with high probability, enumeration is conducted among $(x_{n-k+1}, ..., x_n)$. For every $(x_{n-k+1}, ..., x_n)$, we can compute the corresponding $(y_{n-k+1}, ..., ..., y_n)$, and then compute the unique values of $x_i, \forall i = 1, ..., n-k$ with the condition that $y_i = 0, \forall i = 1, ..., n-k$. The algorithm of Orthogonalized Enumeration is described in Algorithm 1.

3.4 Running Time and Success Probability Analysis

The running time of enumeration algorithm is given by:

$$T_{node} \cdot N$$

Where T_{node} is the average amount time used in processing one node, and N is the number of nodes which is needed to search. As we can see in Algorithm 1 and Algorithm 2, enumerations are conducted among $(x_{n-k+1}, ..., x_n)$ while other x_is are directly computed. And the expected number of nodes N can be computed as follow.

$$N = 3 \cdot \prod_{i=n-k+1}^{n-1} poss\_v\_cnt_i$$

Algorithm 1 Orthogonalized Enumeration Algorithm

```
Input: BKZ-reduced basis: $\mathbf{B}$, an upper bound of $\|\mathbf{v}\|^2$: $R_b$, $k$
Output: the shortest vector $\mathbf{v}$ with $\|\mathbf{v}\|^2 < R_b$
1: For the input basis $\mathbf{B}$, compute Gram-Schmidt orthogonalization of it as $\mathbf{B}^*$ and
   $\mu_{i,j}$ as the elements of the lower-triangular matrix where $\mathbf{b_i} = \mathbf{b_i^*} + \sum_{j=1}^{i-1} \mu_{i,j}\mathbf{b_j^*}$.
2: Compute the $\mathbf{d} := [d_1, ..., d_n] = [R_b^{0.5} n^{-0.5}/\|b_1^*\|, ..., R_b^{0.5} n^{-0.5}/\|b_n^*\|]$.
3: $\mathbf{sv_{1\times n}} := \mathbf{0}, slen := 0$
4: $\mathbf{un_{n\times n}} := \mathbf{0}, \mathbf{ylen_{1\times n}} := \mathbf{0}, \mathbf{uvec_{1\times n}} := \mathbf{0}$
5: $\mathbf{poss\_v_{n\times 5}} := \mathbf{0}, \mathbf{poss\_v\_cnt_{1\times n}} := \mathbf{0}, \mathbf{poss\_v\_ind_{1\times n}} := \mathbf{0}$
6: for $x_n = 2, 1, 0$ do
7:   $uvec_n := x_n$
8:   $un_{n,i} := x_n \cdot \mu_{n,i}\ \forall i = 1, ..., n-1$
9:   $ylen_n := x_n^2 \|\mathbf{b_i^*}\|^2$
10:  for $t = n-1, ..., 1$ do
11:    if $poss\_v\_cnt_t = 0$ then
12:      $(\mathbf{poss\_v_t}, poss\_v\_cnt_t) :=$ COMPUTE_POSSIBLE_VALUE$(t, k, d_t, un_{t+1,t})$
13:      $poss\_v\_ind_t := 1$
14:    else
15:      $poss\_v\_ind_t := poss\_v\_ind_t + 1$
16:    end if
17:    $uvec_t := poss\_v_{t, poss\_v\_ind_t}$
18:    $un_{t,i} := un_{t+1,i} + uvec_t \cdot \mu_{t,i}\ \forall i = 1, ..., t-1$
19:    $ylen_t := ylen_{t+1} + (uvec_t + un_{t+1,t})^2 \|\mathbf{b_i^*}\|^2$
20:    if t=1 then
21:      update $slen$ to store the shortest $ylen_1$ and update $\mathbf{sv}$ to store the corresponding $uvec_1$
22:      for $i = t, ..., n-1$ do
23:        if $poss\_v\_ind_i < poss\_v\_cnt_i$ then
24:          $t := i$, $break$
25:        else
26:          $poss\_v\_cnt_i := 0$
27:        end if
28:      end for
29:      $t := t + 1$
30:    end if
31:  end for
32: end for
33: $\mathbf{v} = \mathbf{sv}$
34: return $\mathbf{v}$
```

Algorithm 2 COMPUTE_POSSIBLE_VALUE Algorithm

Input: $t, k, d_t, un_{t+1,t}$

Output: a set **c** and the number of elements in **c**

```
1:  $\mathbf{c} := \varnothing$
2:  $\mathbf{c} \leftarrow \lfloor -un_{t+1,t} \rceil$
3:  if $t \geq n - k + 1$ then
4:      $\mathbf{c} \leftarrow \lfloor d_t - un_{t+1,t} \rceil$
5:      if $|d_t - un_{t+1,t} - \lfloor d_t - un_{t+1,t} \rceil| > 0.4$ then
6:          if $d_t - un_{t+1,t} > \lfloor d_t - un_{t+1,t} \rceil$ then
7:              $\mathbf{c} \leftarrow \lfloor d_t - un_{t+1,t} \rceil + 1$
8:          else
9:              $\mathbf{c} \leftarrow \lfloor d_t - un_{t+1,t} \rceil - 1$
10:         end if
11:     end if
12:     $\mathbf{c} \leftarrow \lfloor -d_t - un_{t+1,t} \rceil$
13:     if $| - d_t - un_{t+1,t} - \lfloor -d_t - un_{t+1,t} \rceil| > 0.4$ then
14:         if $-d_t - un_{t+1,t} > \lfloor -d_t - un_{t+1,t} \rceil$ then
15:             $\mathbf{c} \leftarrow \lfloor -d_t - un_{t+1,t} \rceil + 1$
16:         else
17:             $\mathbf{c} \leftarrow \lfloor -d_t - un_{t+1,t} \rceil - 1$
18:         end if
19:     end if
20: end if
21: return $\mathbf{c}$, $card(\mathbf{c})$
```

For a basis $\mathbf{B} = [\mathbf{b_1}, \ldots, \mathbf{b_n}]$ and its Gram-Schimdt orthogonalization $\mathbf{B}^* = [\mathbf{b_1^*}, \ldots, \mathbf{b_n^*}]$, let $\mathbf{x}^* = (x_1^*, ..., x_n^*)$ be the coordinates of the shortest vector $\mathbf{v}$ in $\mathbf{B}^*$.

$$\mathbf{v} = \sum_{i=1}^{n} \mathbf{v_i} = \sum_{i=1}^{n} x_i^* \mathbf{b_i^*}.$$

Let $\mathbf{y} = (y_1, ..., y_n) = (\lfloor x_1^* \rceil, ..., \lfloor x_n^* \rceil)$ stands for its orthogonalized integer representation. We have $\mathbf{v_i} = x_i^* \mathbf{b_i^*} = x_i^* \|\mathbf{b_i^*}\| \cdot \mathbf{b_i^*} / \|\mathbf{b_i^*}\|$. According to the Heuristic 2 and Heuristic 3, we assume that $(\|\mathbf{v_1}\|, ..., \|\mathbf{v_n}\|)$ distributes uniformly. And success probability of orthogonalized enumeration with parament k can be estimated as:

$$P_{succ}(n,k) = \Pr_{\mathbf{v} \sim Ball_n(\|\mathbf{b_1}\|)} \left(\forall j \in [1,n], \lfloor \|\mathbf{v_j}\| / \|\mathbf{b_j^*}\| \rceil \in \mathbf{poss\_v_j} - \lfloor -un_{j+1,j} \rceil \right)$$

We obtain the relationship between dimension n ranging from 40 to 130 and $P_{succ}(n,k)$ with $k \in [8,20]$ through Monte-Carlo Simulation. The results can be found in Figure 1.

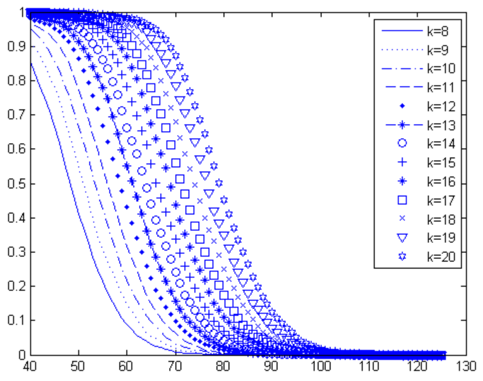

Figure 1. Relationship between Dimension n and $P_{succ}(n,k)$ with $k \in [8,20]$

3.5 Comparison between Orthogonalized Enumeration and Previous Enumeration

Based on N and $P_{succ}(n,k)$ of the orthogonalized enumeration obtained in Section 3.4, we can get the expected number of nodes needed to search a n dimension basis using orthogonalized enumeration by $N/P_{succ}(n,k)$ with different k, and then get the expected number of nodes needed to search a n dimension basis using orthogonalized enumeration with a proper k, denoted as N_{orth}. We also estimate the expected number of nodes when using full enumeration, linear pruning enumeration and extreme pruning enumeration, denoted as N_{full}, N_{linear}, $N_{extreme}$, and results can be found in Figure 2.

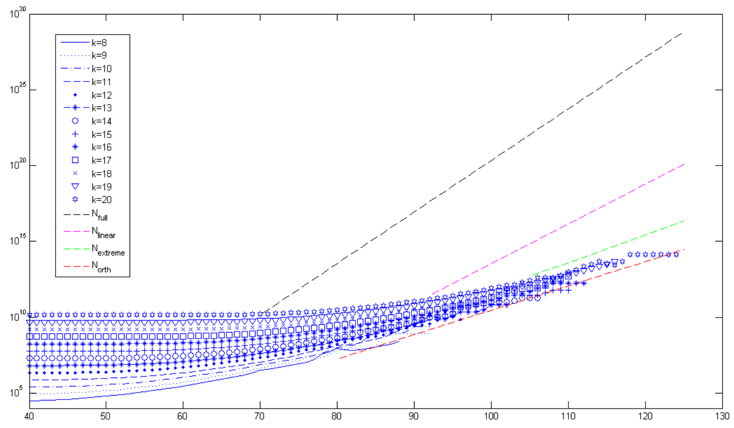

Figure 2. Relationship between Dimension n and Expected Number of Nodes Needed for Different Enumeration Methods

Compared to full enumeration, linear pruning enumeration and extreme pruning enumeration achieve a speedup of 1.189^n and 1.414^n, while the orthogonalized enumeration improves full enumeration by a factor of 1.512^n. The improvement compared to extreme pruning enumeration is sharp, but not as large as that n-dimension search reduced to k-dimension search as we expected. This indicates that though doesn't select an unique $(x_1, ..., x_{n-k})$ as the orthogonalized enumeration does in enumeration, extreme pruning has pruned the searching space of $(x_1, ..., x_{n-k})$ to a very small space which makes it an extremely effective method since it's presented. While the orthogonalized enumeration has a smaller searching space which is limited by k, and the introduction of k also makes it more flexible to control the searching process. These features make the orthogonalized enumeration a more efficient method than previous methods and one of our biggest innovations.

4 MBKZ

4.1 Description of Algorithm

The main idea of Mixed BKZ (denoted as MBKZ) is to alternately use orthognalized enumeration and traditional enumeration(full enumeration, linear pruning enumeration, extreme pruning enumeration etc). Another difference of MBKZ is that we set the blocksize of orthognalized enumeration to n, in order to make good use of the fact that the number of nodes needed in the orthognalized enumeration is limited by k. The detail of MBKZ can be found in Algorithm 3. Besides, the orthognalized enumeration algorithm is adjusted a little to speedup the effectiveness in MBKZ, which is shown in Algorithm 4.

The design of MBKZ is due to the following reason. According to [6], probability enumeration can speedup the search but may not return any vector or maybe not the shortest one. As a result, in BKZ 2.0, randomizing technique is used to ensure the enumeration process can find a shorter vector in acceptable time. Though the technique is useful, but it also brings unavoidable overheads according to [4], because the bases are not good after they have been randomized in practice and an extra reduction process should be called to reduce the randomized bases before enumeration. Though no quantitative analysis about the proportion of the extra overheads has been proposed, it is non-negligible in practice. While in MBKZ, we use a new technique to avoid randomizing bases and ensure enumeration success probability at the same time. Experimental data shows that this technique is effective which makes MBKZ a more efficient method compared to the previous ones. The details are as follow.

In BKZ process, enumeration is called to successively search a better vector $\mathbf{v}$ which is consist of a combination of $(\mathbf{b_i}, ..., \mathbf{b_j})$ to replace $\mathbf{b_i}$ for all i from 1 to $n-1$, where $j = min(i+\beta-1, n)$ and β denotes the blocksize. However, the searching of orthognalized enumeration is conducted among the last k dimensions $(\mathbf{b_{j-k+1}}, ..., \mathbf{b_j})$. And in orthognalized enumeration the blocksize β is set to n which indicates that we are always searching the shortest vector $\mathbf{v}$ to replace $\mathbf{b_i}$ among the same space $(\mathbf{b_{n-k+1}}, ..., \mathbf{b_n})$ for all i. All nodes needed to enumerate for replacing $\mathbf{b_i}$ are usually included by those enumerated for replacing $\mathbf{b_{i-1}}$ if no changes have been made to the basis after the enumeration for $\mathbf{b_{i-1}}$. Therefore if the enumeration for $\mathbf{b_{i-1}}$ is failed, we can avoid repeating enumeration process by reusing the intermediate results to search $\mathbf{b_i}$. As a result, we can run orthognalized enumeration when $i = 1$, store a best result in each depth and decide which $\mathbf{b_i}$ should be replaced after enumeration. And this is the main idea of MBKZ.

Algorithm 3 The Mixed Block Korkin-Zolotarev Algorithm

```
Input: A basis $\mathbf{B} = (\mathbf{b_1}, ..., \mathbf{b_n})$, a blocksize $\beta \in 2, ..., n$, the Gram-Schmidt triangular
    matrix $\mu$, $\|\mathbf{b_1^*}\|^2, ..., \|\mathbf{b_n^*}\|^2$ and orthognalized enumeration parament $k$
Output: A $BKZ - \beta$ reduced basis $(\mathbf{b_1}, ..., \mathbf{b_n})$
 1: $\mathbf{sv_{n\times n}} := \mathbf{0}, \mathbf{slen_{1\times n}} := \mathbf{0}$
 2: $z := 0$, $jj := 0$, $cnt := 0$, $LLL(\mathbf{b_1}, ..., \mathbf{b_n}, \mu)$
 3: while $z < n - 1$ do
 4:   $jj := jj$ mod $(n - 1) + 1$
 5:   if $jj = 1$ then
 6:     $cnt := cnt + 1$
 7:   end if
 8:   if $cnt$ mod $2 = 0$ and $jj = 1$ then
 9:     $kk := n$, $h := min(k + 1, n)$, $\mathbf{v} := (1, 0, ..., 0)$
10:     $(\mathbf{sv}, \mathbf{slen}) = \mathbf{Orthognalized\_Enum\_for\_MBKZ}(\mu_{[jj,kk]}, \|\mathbf{b_{jj}^*}\|^2, ..., \|\mathbf{b_{kk}^*}\|^2, k)$
11:     for $i = jj, ..., kk$ do
12:       if $slen_i < \|\mathbf{b_i^*}\|^2$ then
13:         $\mathbf{v} = \mathbf{sv_i}$, $jj = i$, break;
14:       end if
15:     end for
16:   else
17:     $kk := min(jj + \beta - 1, n)$, $h := min(k + 1, n)$
18:     $\mathbf{v} = \mathbf{Traditonal\_Enum}(\mu_{[jj,kk]}, \|\mathbf{b_{jj}^*}\|^2, ..., \|\mathbf{b_{kk}^*}\|^2)$
19:   end if
20:   if $\mathbf{v} \neq (1, 0, ..., 0)$ then
21:     $z := 0$
22:     update basis by $LLL(\mathbf{b_1}, ..., \sum_{i=jj}^{kk} v_i \mathbf{b_i}, \mathbf{b_j}, ..., \mathbf{b_h}, \mu)$
23:   else
24:     $z := z + 1$
25:     reduce the next block by $LLL(\mathbf{b_1}, ..., \mathbf{b_h}, \mu)$
26:   end if
27: end while
```

Algorithm 4 Orthognalized_Enum_for_MBKZ

```
Input: $\mu$, $\|\mathbf{b_1^*}\|^2, ..., \|\mathbf{b_n^*}\|^2$, $k$
Output: $\mathbf{sv_{n\times n}}, \mathbf{slen_{1\times n}}$
 1: Compute the $\mathbf{d} := [d_1, ..., d_n] = [n^{-0.5}\|\mathbf{b_1^*}\|/\|b_1^*\|, ..., n^{-0.5}\|\mathbf{b_1^*}\|/\|b_n^*\|]$.
 2: $\underline{\mathbf{sv_{n\times n}} := \mathbf{0}, \mathbf{slen_{1\times n}} := \mathbf{0}}$
 3: $\mathbf{un_{n\times n}} := \mathbf{0}, \mathbf{ylen_{1\times n}} := \mathbf{0}, \mathbf{uvec_{1\times n}} := \mathbf{0}$
 4: $\mathbf{poss\_v_{n\times 5}} := \mathbf{0}, \mathbf{poss\_v\_cnt_{1\times n}} := \mathbf{0}, \mathbf{poss\_v\_ind_{1\times n}} := \mathbf{0}$
 5: for $x_n = 2, 1, 0$ do
 6:   $uvec_n := x_n$
 7:   $un_{n,i} := x_n \cdot \mu_{n,i}\ \forall i = 1, ..., n-1$
 8:   $ylen_n := x_n^2\|\mathbf{b_i^*}\|^2$
 9:   for $t = n-1, ..., 1$ do
10:     if $poss\_v\_cnt_t = 0$ then
11:       $(\mathbf{poss\_v_t}, poss\_v\_cnt_t) :=$ COMPUTE_POSSIBLE_VALUE$(t, k, d_t, un_{t+1,t})$
12:       $poss\_v\_ind_t := 1$
13:     else
14:       $poss\_v\_ind_t := poss\_v\_ind_t + 1$
15:     end if
16:     $uvec_t := poss\_v_{t,poss\_v\_ind_t}$
17:     $un_{t,i} := un_{t+1,i} + uvec_t \cdot \mu_{t,i}\ \forall i = 1, ..., t-1$
18:     $ylen_t := ylen_{t+1} + (uvec_t + un_{t+1,t})^2\|\mathbf{b_i^*}\|^2$
19:     update $\mathbf{slen_t}$ to store the shortest $ylen_t$ and update $\mathbf{sv_t}$ to store the corresponding $uvec_t$
20:     if t=1 then
21:       for $i = t, ..., n-1$ do
22:         if $poss\_v\_ind_i < poss\_v\_cnt_i$ then
23:           $t := i$, break
24:         else
25:           $poss\_v\_cnt_i := 0$
26:         end if
27:       end for
28:       $t := t+1$
29:     end if
30:   end for
31: end for
32: return sv, slen
```

4.2 Running Time and Success Probability Analysis of Orthognalized Enumeration in MBKZ

According to the MBKZ algorithm, the success probability of the orthognalized enumeration in MBKZ, denoted as $P_{succ\_MBKZ}(m,k)$, should be computed as follow.

$$P_{succ\_MBKZ}(m,k) = P_{succ}(m,k) \prod_{i=m+1}^{\eta} (1 - P_{succ}(i,k))$$

where η is the largest j such that $P_{succ}(j,k) \neq 0$.

The results of $P_{succ\_MBKZ}(m,k)$ with $k \in [8,20]$ can be found in Figure 3. After conducting an orthognalized enumeration, the probability of successfully finding a better vector to replace $\mathbf{b_{n-m+1}}$ is indicated by $P_{succ\_MBKZ}(m,k)$. And if we don't use orthognalized enumeration, the work demands for running a traditional enumeration in m dimensions instead.

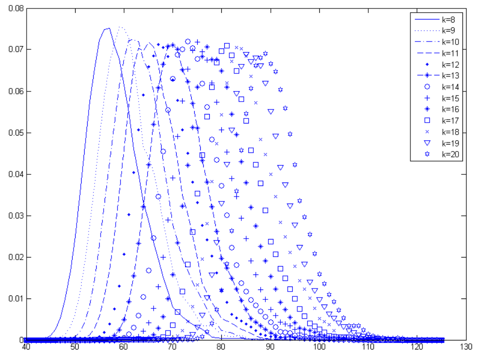

Figure 3. Relationship between m and $P_{succ\_MBKZ}(m,k)$ with $k \in [8,20]$

We compute an expected number of nodes that orthognalized enumeration needs with different k, denoted as $N_{orth,k}$, by using the method described in Section 3.4. To make comparison between orthognalized enumeration and other enumeration methods, we compute expected number of nodes needed for different methods. Let $\phi_{full}(m)$, $\phi_{linear}(m)$ and $\phi_{extreme}(m)$ denote the expected number of nodes it takes to find a better vector in m dimensions with full enumeration, linear pruning enumeration and extreme pruning enumeration. Respectively we can get expected number of nodes it takes to finish what a round of orthognalized enumeration with the parament k can do with three enumeration methods, denoted as $N_{full,k}$, $N_{linear,k}$ and $N_{extreme,k}$. Results with $k \in [8, 20]$ are shown in Figure 4.

$$N_{full,k} = \sum_{\forall m} (P_{succ\_MBKZ}(m,k) \cdot \phi_{full}(m))$$

$$N_{linear,k} = \sum_{\forall m} (P_{succ\_MBKZ}(m,k) \cdot \phi_{linear}(m))$$

$$N_{extreme,k} = \sum_{\forall m} (P_{succ\_MBKZ}(m,k) \cdot \phi_{extreme}(m))$$

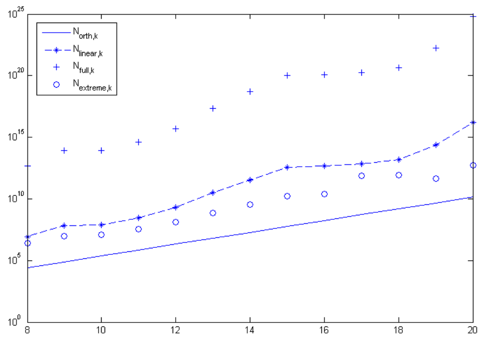

Figure 4. Number of Nodes Needed for Different Enumeration Methods

According to the comparison above, the design for the orthognalized enumeration in MBKZ brings another speedup of $O(n)$ compared to the original orthognalized enumeration. We conduct experiments in basis of [28] with dimensions up to 140, the results are consistent with our analysis. Besides, there are still some important notions we get through experiments need to be known.

1. Orthognalized enumeration can exponentially speedup traditional enumeration, but which $\mathbf{b_i}$ will be replaced is uncertain. As a result, combining orthognalized enumeration and traditional enumeration method in MBKZ can improve previous BKZ method sharply. However when using orthognalized enumeration independently as the enumeration process in a BKZ algorithm, things are not as good as the ideal. Because in this situation, k should be set large enough to keep the probability of updating $\mathbf{b_1}$ considerable, which may introduce extra overhead in enumeration.

2. The output of MBKZ generally has better quality compared to BKZ or BKZ 2.0 with the same blocksize (note that blocksize in MBKZ denotes the blocksize of traditional enumeration, the orthognalized enumeration's blocksize

is always set to n), the shortest vector in dimension $100-110$ can be directly found by MBKZ with the blocksize about $40-42$, while BKZ or BKZ 2.0 need a much larger blocksize (*i.e* in BKZ 2.0, blocksize should be set about 75 according to [6]) to do so.

3. When we choose linear pruning or extreme pruning as the traditional enumeration method in MBKZ, randomizing technique is not as necessary as it is in BKZ 2.0, because the orthognalized enumeration and traditional enumeration methods have different searching spaces and are continuously updating it for each other. Though hard to make quantitative analysis, this is thought to be an effective way to reduce duplicate searching and improve the effectiveness further.

5 Further Improvements

In this section, we describe an interesting technique to reduce the searching space of enumeration with non-negligible probability. We can't give an exact quantitative success probability analysis for the method, but it is practicable in experiments.

5.1 Description of Method

The main idea of the method is to utilize the property of basis transformation. Let $\mathbf{v} = \sum x_i b_i$ represent the shortest vector in $\mathbf{B}$. Matrix $\mathbf{U}$ is an upper triangular matrix with elements on the main diagonal are all 1s, so in a new basis $\mathbf{B}' = \mathbf{UB}$, $\mathbf{v} = \sum x_i' b_i'$ is still the shortest vector. We set U as:

$$\mathbf{U} = \begin{pmatrix} 1\,0\,0\ldots0\,0 & a_1 \\ 0\,1\,0\ldots0\,0 & a_2 \\ \vdots\,\vdots\,\vdots\;\vdots\;\vdots\,\vdots & \vdots \\ 0\,0\,0\ldots0\,1 & a_{n-1} \\ 0\,0\,0\ldots0\,0 & 1 \end{pmatrix}$$

Then we can know that $\mathbf{B}' = [\mathbf{b_1'}, ..., \mathbf{b_n'}] = \mathbf{UB} = [\mathbf{b_1}+a_1\mathbf{b_n}, ..., \mathbf{b_i}+a_i\mathbf{b_n}, ..., \mathbf{b_n}]$, so $\mathbf{b_n'} = \mathbf{b_n}$ and $\mathbf{b_i'} = \mathbf{b_i} + a_i\mathbf{b_n}$ $\forall i \in [1, n-1]$. And we have:

$$\mathbf{v} = \sum x_i'\mathbf{b_i'} = \sum_{i=1}^{n-1} x_i'(\mathbf{b_i} + a_i\mathbf{b_n}) + x_n'\mathbf{b_n} = \sum_{i=1}^{n-1} x_i'\mathbf{b_i} + (\sum_{i=1}^{n-1} a_i x_i' + x_n')\mathbf{b_n}$$

$$= \sum x_i\mathbf{b_i}$$

which indicates:

$$x_i' = x_i \ \forall i \in [1, n-1]$$

$$x'_n = -\sum_{i=1}^{n-1} a_i x_i + x_n$$

Let's make two extremely strong assumptions first to describe the idea:

Extremely Strong Assumption 1. We know x'_n exactly.

Extremely Strong Assumption 2. $a_i = \omega^i$ and $\omega > 2|x_i|\ \forall i \in [1, n-1]$

Based on the two extremely strong assumptions, we can get $(x_1, ..., x_{n-1})$ by Algorithm 5:

Algorithm 5 The x_i Recovery Algorithm

```
Input: $x'_n$, $a_i = \omega^i$
Output: $(x_1, ..., x_{n-1})$
 1: $s := x'_n$
 2: for $i = 1, ..., n-1$ do
 3:   if $s > 0$ then
 4:     $x_i := \lfloor s/a_i \rfloor$
 5:   else
 6:     $x_i := \lceil s/a_i \rceil$
 7:   end if
 8:   $s := s \bmod a_i$
 9: end for
10: for $i = n-1, ..., 1$ do
11:   if $|x_i| > \omega/2$ then
12:     if $x_i > 0$ then
13:       $x_i = x_i - \omega$, $x_{i-1} := x_{i-1} + 1$
14:     else
15:       $x_i = x_i + \omega$, $x_{i-1} := x_{i-1} - 1$
16:     end if
17:   end if
18: end for
19: return $(x_1, ..., x_{n-1})$
```

5.2 Success Probability Analysis

The result is amazing. But in actual case, we don't know x'_n, and we can't compute a_i of that size (when $n = 100$, a_{n-1} is over 10^{195}). So we choose to make two new assumptions that are more practical:

Assumption 1. We can estimate $|x'_n|$ with an acceptable error.

Assumption 2. $a_i = 0\ \forall i \in [1, m-1]$, $a_i = \omega^{i-m+1}$ and $\omega > 2|x_i|\ \forall i \in [m, n-1]$.

From the heuristic 2 and heuristic 3, we assume the coordinates $\mathbf{x}$ of the target vector $\mathbf{v}$ in the orthogonal basis $(\mathbf{b_n^*}/\|\mathbf{b_n^*}\|, ..., \mathbf{b_1^*}/\|\mathbf{b_1^*}\|)$ distribute like a uniform vector where $\|\mathbf{x}\| = \|\mathbf{v}\|$. And we assume it is so in $(\mathbf{b_n'^*}/\|\mathbf{b_n'^*}\|, ..., \mathbf{b_1'^*}/\|\mathbf{b_1'^*}\|)$ where we have changed directions of $(\mathbf{b_m'^*}, ..., \mathbf{b_n'^*})$ compared to the original $\mathbf{B}$.

Heuristic 4. The distribution of the normalized Gram-Schmidt orthogonalization $(\mathbf{b_1'^*}/\|\mathbf{b_1'^*}\|, ..., \mathbf{b_n'^*}/\|\mathbf{b_n'^*}\|)$ of $\mathbf{B'} = \mathbf{UB}$ where $\mathbf{B}$ is a random reduced basis, $\mathbf{U}$ is an upper triangular matrix with elements on the main diagonal are all 1s and $\mathbf{b_1} = \mathbf{b_1'}$, looks like that of a uniformly distributed orthogonal matrix.

We estimate $|x_n'|$ by $n^{-0.5}\|\mathbf{b_1'}\|/\|\mathbf{b_n'^*}\|$, let $E_{|x_n'|}$ denote the error of the estimation and $P_{error}(\theta)$ denote the probability that $E_{|x_n'|} \leq \theta$, we can get the probability distribution by Monte-Carlo simulation. Results can be found in Figure 5.

$$E_{|x_n'|} = \frac{||x_n'| - n^{-0.5}\|\mathbf{b_1'}\|/\|\mathbf{b_n'^*}\||}{|x_n'|}$$

$$P_{error}(\theta) = Pr(E_{|x_n'|} \leq \theta)$$

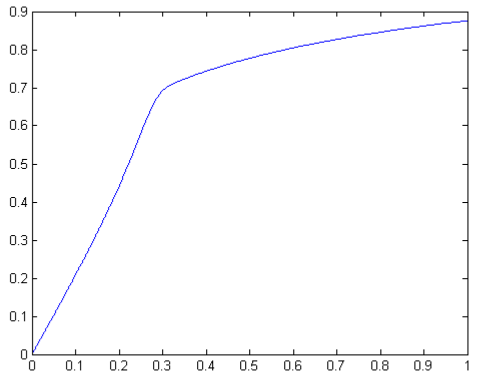

Figure 5. Relationship between θ and P_{error}

Errors in the estimation of $|x_n'|$ bring error of computing x_i (note that we use x_n' but not $|x_n'|$ below because of the symmetry of $\mathbf{v}$ and $-\mathbf{v}$):

$$n^{-0.5}\|\mathbf{b_1'}\|/\|\mathbf{b_n'^*}\| = x_n' \pm \theta|x_n'| = -\sum_{i=m}^{n-1}\omega^{i-m+1}x_i + x_n \pm \theta|x_n'|$$

The threshold that the computation of x_i is not influenced by the error θ, denoted as $th(\omega, i)$, is calculated by $\omega^{-(n-1-i)} - \omega^{-(n-i)}$(the maximum and minimum of $th(\omega, i)$ is $\omega^{-(n-1-i)}$ and $\omega^{-(n-1-i)} - 2\omega^{-(n-i)}$, we take the average of them). For $\omega = 3, 5$ and 7, $th(\omega, i)$ is shown in Table 1.

Table 1. $th(\omega, i)$ for different i with $\omega = 3, 5$ and 7

	$i = n-1$	$i = n-2$	$i = n-3$	$i = n-4$	$i = n-5$	$i = n-6$
$\omega = 3$	0.66667	0.22222	0.07407	0.02469	0.00823	0.00274
$\omega = 5$	0.8	0.16	0.032	0.0064	0.00128	0.00026
$\omega = 7$	0.85714	0.11429	0.02286	0.00457	0.00091	0.00018

With $P_{error}(\theta)$ and $th(\omega, i)$, we can get $P_{nf}(\omega, i)$ which represents the probability that we can successfully calculate $(x_i, ..., x_{n-1})$ and the result is not influenced by the error caused by the estimation of $|x'_n|$. Table 2 shows P_{nf} with $\omega = 3, 5$ and 7, and $P_{nf}(\omega, i)$ is calculated by $P_{nf}(\omega, i) = P_{error}(th(\omega, i))$.

Table 2. $P_{nf}(\omega, i)$ for different i with $\omega = 3, 5$ and 7

	$i = n-1$	$i = n-2$	$i = n-3$	$i = n-4$	$i = n-5$	$i = n-6$
$\omega = 3$	0.81966	0.50113	0.15408	0.0517	0.01736	0.00573
$\omega = 5$	0.84558	0.34396	0.06726	0.01342	0.0028	0.0006
$\omega = 7$	0.85554	0.24013	0.04787	0.00954	0.00215	0.00038

From $P_{nf}(\omega, i)$, we know that the larger ω is, the harder it is to get more x_is because of the error caused by the estimation of $|x'_n|$.

Now we have $P_{nf}(\omega, i)$ which shows the influence of the error brought by the estimation of $|x'_n|$, the remaining thing to do is to choose a proper ω. From the analysis before, we know that the larger ω is, the smaller $P_{nf}(\omega, i)$ becomes. While $\omega > 2|x_i|\ \forall i \in [m, n]$ is the precondition of the method, let $P_{|x|}(\omega, m)$ denote the probability that $|x_i| < \omega/2\ \forall i \in [m, n]$. The success probability of the method, denoted as $P_s(\omega, m)$, that we can calculate $(x_m, ..., x_{n-1})$ successfully is expressed as:

$$P_s(\omega, m) = P_{|x|}(\omega, m) \cdot P_{nf}(\omega, m)$$

We can't give an exact probability analysis about $P_{|x|}(\omega, m)$, but qualitative analysis can be done. Many experiments based on orthogonalized integer representations[9][7] have shown the orthogonalized integer representation $\mathbf{y} = (y_1, ..., y_n)$ of the shortest vector is consist of small y_i where $y_i = x_i + \lfloor \sum_{j=i+1}^{n} \mu_{j,i} x_j \rceil$. And for a BKZ-reduced basis we generally have $|\mu| < 0.5$, so when i is close to n, the x_i which can be got by a small y_i minus a limited number of $\mu_{j,i}x_j$ usually has a limited absolute value. However, when i gets smaller, $\lfloor\sum_{j=i+1}^{n}\mu_{j,i}x_j\rceil$ tends to get larger and so it is with $|x_i|$. The analysis leads to a notion that $P_{|x|}(\omega,m)$ has positive correlation with ω and m, and when ω and m is large $P_{|x|}(\omega,m)$ is close to 1. We conduct experiments in random basis, the experiment results are consistent with our analysis, and also show that $P_{|x|}(\omega,m)$ is considerable with $m\in[n-1,n-6]$ which makes $P_s(\omega,m)$ negligible with $m\in[n-1,n-4]$.

With some x_is computed, enumerations can be conducted in a reduced space, which brings an obvious speedup to all enumeration methods including orthogonalized enumeration and previous ones.

6 Conclusion

In this paper, we describe a new enumeration algorithm based on orthogonalized integer representations of the shortest vector, and give a success probability analysis through Monte-Carlo Simulation. According to the result of analysis, we can set a suitable threshold, and reduce the enumerated space greatly which brings an exponential speedup compared to the existing enumeration algorithms based on BKZ reduction. Another meaningful work is to present a new BKZ method named MBKZ. Besides the speedup in enumeration nodes, MBKZ uses a new technique which reduces the duplicate work caused by probability enumeration and avoids the overheads brought by randomizing technique at the same time. In addition, MBKZ generally outputs better basis than other BKZ methods with the same blocksize in practice. What's more, a new technique to reduce enumeration space with non-negligible probability is given, though lack of quantitative analysis about success probability, it is effective in experiments.

Next we will work on a simulation algorithm to predict the performance of MBKZ in terms of running time and output quality, which will help a lot in theoretical analysis and conducting experiments in high dimension.

References

1. Aggarwal D, Dadush D, Regev O, et al. Solving the Shortest Vector Problem in 2^n Time Using Discrete Gaussian Sampling[C]//Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing. ACM, 2015: 733-742.
2. Agrell E, Eriksson T, Vardy A, et al. Closest point search in lattices[J]. Information Theory, IEEE Transactions on, 2002, 48(8): 2201-2214.
3. Ajtai M, Kumar R, Sivakumar D. A sieve algorithm for the shortest lattice vector problem[C]//Proceedings of the thirty-third annual ACM symposium on Theory of computing. ACM, 2001: 601-610.

4. Aono Y, Wang Y, Hayashi T, et al. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2016: 789-819.
5. Booker L B, Goldberg D E, Holland J H. Classifier systems and genetic algorithms[J]. Artificial intelligence, 1989, 40(1): 235-282.
6. Chen Y, Nguyen P Q. BKZ 2.0: Better lattice security estimates[M]//Advances in CryptologyCASIACRYPT 2011. Springer Berlin Heidelberg, 2011: 1-20.
7. Ding D, Zhu G, Wang X. A Genetic Algorithm for Searching the Shortest Lattice Vector of SVP Challenge[C]//Proceedings of the 2015 on Genetic and Evolutionary Computation Conference. ACM, 2015: 823-830.
8. Eiben A E, Aarts E H L, Van Hee K M. Global convergence of genetic algorithms: A Markov chain analysis[M]//Parallel problem solving from nature. Springer Berlin Heidelberg, 1991: 3-12.
9. Fukase M, Kashiwabara K. An Accelerated Algorithm for Solving SVP Based on Statistical Analysis[J]. Journal of information processing, 2015, 23(1): 67-80.
10. Gama N, Nguyen P Q, Regev O. Lattice enumeration using extreme pruning[M]//Advances in CryptologyCEUROCRYPT 2010. Springer Berlin Heidelberg, 2010: 257-278.
11. Goldberg D E, Holland J H. Genetic algorithms and machine learning[J]. Machine learning, 1988, 3(2): 95-99.
12. Hanrot G, Stehlé D. Improved analysis of Kannans shortest lattice vector algorithm[M]//Advances in Cryptology-CRYPTO 2007. Springer Berlin Heidelberg, 2007: 170-186.
13. Haque M M, Rahman M O, Pieprzyk J. Analysing Progressive-BKZ Lattice Reduction Algorithm[J]. 1st National Conference on Intelligent Computing and Information Technology
14. Hastad J, Just B, Lagarias J C, et al. Polynomial time algorithms for finding integer relations among real numbers[J]. SIAM Journal on Computing, 1989, 18(5): 859-881.
15. Helfrich B. Algorithms to construct Minkowski reduced and Hermite reduced lattice bases[J]. Theoretical Computer Science, 1985, 41: 125-139.
16. Holland J H. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence[M]. U Michigan Press, 1975.
17. Kannan R. Improved algorithms for integer programming and related lattice problems[C]//Proceedings of the fifteenth annual ACM symposium on Theory of computing. ACM, 1983: 193-206.
18. Kannan R. Minkowski's convex body theorem and integer programming[J]. Mathematics of operations research, 1987, 12(3): 415-440.
19. Kuo P C, Schneider M, Dagdelen Ö, et al. Extreme Enumeration on GPU and in Clouds[M]//CHES 2011. Springer Berlin Heidelberg, 2011: 176-191.
20. Lenstra A K, Lenstra H W, Lovász L. Factoring polynomials with rational coefficients[J]. Mathematische Annalen, 1982, 261(4): 515-534.

21. Lindner R, Peikert C. Better key sizes (and attacks) for LWE-based encryption[M]//Topics in CryptologyCCT-RSA 2011. Springer Berlin Heidelberg, 2011: 319-339.
22. Micciancio D, Walter M. Practical, predictable lattice basis reduction[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2016: 820-849.
23. Micciancio D, Voulgaris P. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations[J]. SIAM Journal on Computing, 2013, 42(3): 1364-1391.
24. Micciancio D, Regev O. Lattice-based cryptography[M]//Post-quantum cryptography. Springer Berlin Heidelberg, 2009: 147-191.
25. Nguyen P Q, Vidick T. Sieve algorithms for the shortest vector problem are practical[J]. Journal of Mathematical Cryptology, 2008, 2(2): 181-207.
26. Pujol X, Stehl*é* D. Solving the Shortest Lattice Vector Problem in Time $2^{2.465n}$[J]. IACR Cryptology ePrint Archive, 2009, 2009: 605.
27. R*ü*ckert M, Schneider M. Estimating the Security of Lattice-based Cryptosystems[J]. IACR Cryptology ePrint Archive, 2010, 2010: 137.
28. Schneider, M., and Gamma, N. http://www.latticechallenge.org/svp-challenge/.
29. Schnorr C P. Lattice reduction by random sampling and birthday methods[C]//STACS. 2003, 2607: 145-156.
30. Schnorr C P, Euchner M. Lattice basis reduction: improved practical algorithms and solving subset sum problems[J]. Mathematical programming, 1994, 66(1-3): 181-199.
31. Schnorr C P, H*ö*rner H H. Attacking the Chor-Rivest cryptosystem by improved lattice reduction[C]//Advances in CryptologyEurocrypt95. Springer Berlin Heidelberg, 1995: 1-12.
32. Schnorr C P, Lattice Reduction by Random Sampling and Birthday Methods,STACS 2003, LNCS, Vol.2607, pp.145C156, Springer-Verlag (2003).
33. Shoup, V. Number theory c++ library (ntl) vesion 6.0.0, available at http://www.shoup.net/ntl/.
34. Wang X, Liu M, Tian C, et al. Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem[C]//Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security. ACM, 2011: 1-9.

A Appendix

Algorithm 6 The full enumeration algorithm

```
Input: An integral lattice basis $(\mathbf{b_1}, \cdots, \mathbf{b_d})$, a bound A $\in \mathbb{Z}$
Output: All vectors in $L(\mathbf{b_1}, \cdots, \mathbf{b_d})$ that are of squared norm $\leq$ A
1: Compute the rational $\mu_{i,j}$'s and $\|\mathbf{b_i^*}\|^2$'s
2: $x := 0, l := 0, S := \emptyset$
3: $i := 1$. While $i \leq d$, do
4:     $l_i := (x_i + \sum_{j>i} x_j \mu_{j,i})^2 \|\mathbf{b_i^*}\|^2$.
5:     If $i = 1$ and $\sum_{j=1}^{d} l_j \leq A$, then $S := S \cup \{\mathbf{x}\}, x_1 := x_1 + 1$
6:     If $i \neq 1$ and $\sum_{j \geq i} l_j \leq A$, then
7:         $i := i - 1$, $x_i := \lceil -\sum_{j>i}(x_j \mu_{j,i}) - \sqrt{\frac{A - \sum_{j>i} l_j}{\|\mathbf{b_i^*}\|^2}} \rceil$.
8:     If $\sum_{j>i} l_j > A$, then $i := i + 1$, $x_i := x_i + 1$.
9: return S.
```