Orthogonalized Lattice Enumeration for Solving SVP

Zhongxiang Zheng[1], Xiaoyun Wang[2], Guangwu Xu[3], Yang Yu[1]

[1] Department of Computer Science and Technology,Tsinghua University, Beijing 100084, China,
[2] Institute for Advanced Study,Tsinghua University, Beijing 100084, China,
[3] Department of Electrical Engineering and Computer Sciences, University of Wisconsin, Milwaukee, WI 53201, USA
xiaoyunwang@mail.tsinghua.edu.cn

Abstract. In 2014, the orthogonalized integer representation was proposed independently by Ding et al. using genetic algorithm and Fukase et al. using sampling technique to solve SVP. Their results are promising. In this paper, we consider sparse orthogonalized integer representations for shortest vectors and propose a new enumeration method, called orthognalized enumeration, by integrating such a representation. Furthermore, we present a mixed BKZ method, called MBKZ, by alternately applying orthognalized enumeration and other existing enumeration methods. Compared to the existing ones, our methods have greater efficiency and achieve exponential speedups both in theory and in practice for solving SVP. Implementations of our algorithms have been tested to be effective in solving challenging lattice problems. We also develop some new technique to reduce enumeration space which has been demonstrated to be efficient experimentally, though a quantitative analysis about its success probability is not available.

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 $\mathbf{b}_1, \ldots, \mathbf{b}_n$ 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 shortest Euclidean norm in each lattice. There are two famous computational problems in lattices:

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

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

Over the past two decades, these two hard problems SVP and CVP have been of prime importance to the lattice cryptography. 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 [26, 27, 34]. Another important work is the deterministic algorithm with $\tilde{O}(2^{2n})$ time and $\tilde{O}(2^{n})$ space given by Micciancio and Voulgaris [24]. 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 of [24].

For the class of polynomial-space algorithms, two popular techniques are used, one is *lattice reduction*, including the famous LLL alogrithm [21], HKZ reduction [18] and BKZ reduction [31]. The other important technique is *enumeration technique* which is an exact algorithm to find shortest vectors in a reduced space. These two techniques are complementary in the following sense. Usually, a reduction cannot output shortest vectors in high dimensional lattices, it is used to find vectors that are sufficiently short to ensure an enumeration search to work efficiently. Whereas the enumeration technique works as a subroutine and applies to sublattices of lower dimension repeatedly to greatly improve the output quality of reductions. The first polynomial-space algorithm was provided by Kannan [18] in 1980s. This theoretical enumeration algorithm is based on LLL-reduced basis and HKZ reduction and achieves worst-case time complexity of $2^{O(n \log n)}$. A more accurate analysis on Kannan's algorithm was given by Helfrich [16], with the complexity of $d^{\frac{d}{2}+o(d)}$; the complexity bound was further improved to $d^{\frac{d}{2e}+o(d)}$ by Hanrot and Stehlé in [13] (d is the dimension of the lattice and e represents the base of natural logarithm). Another popular polynomial-space algorithm is the Schnorr-Euchner enumeration based on the BKZ reduction, and its enumeration complexity is estimated by Gama, Nguyen and Regev in [11] as $\sum_{l=1}^{n} q^{(n-l)l/2} 2^{O(n)}$($q$ is a constant depending on the basis). Although the complexity of the Schnorr-Euchner enumeration seems higher than that of Kannan's, it is in fact a widely used practical method. For example, it is a fundamental tool in the popular mathematical library NTL [33]. It is noted that many security assessments [22, 25, 28, 32] and SVP searching methods [14, 20] of lattice cryptosystems are based on BKZ implementation of NTL. Therefore, a further improvement to the enumeration technique is of significant importance for SVP searching. Gama, Nguyen and Regev proposed an improved enumeration using the extreme pruning technique and the speedup is exponential [11]. Chen and Nguyen used the technique to design BKZ 2.0 which improves the

BKZ algorithm [7]. Further recent improvements of BKZ include methods based on progressive strategy [4] and result predictions [23].

The main purpose of our paper is to propose a new enumeration method called *the othogonalized enumeration*. Our design is motivated by the integer sparse representation of the shortest vector with respect to the Gram-Schmidt basis. It is observed that for a BKZ-reduced basis, the norms of the othogonalized basis tend to decrease quickly as the component index gets large. This indicates that for a shortest lattice vector, its (rounding) coefficients with respect to the othogonalized basis are likely to be zero when their indices are not big enough. The idea of using sparse representation of shortest vectors with respect to the Gram-Schmidt basis can be traced back to Schnorr's Random Sampling Algorithm [30]. This idea was expanded independently by Ding , Zhu, and Wang [8] in a genetic algorithm and by Fukase and Kashiwabara [10] in a sampling algorithm. Genetic algorithm was initiated by Holland [17] in 1975, and it has been used to solve optimization problems such as timetabling, scheduling, and engineering problems [6, 9, 12]. The essence of the method is to transform a shortest lattice vector into a new integer vector corresponding to the Gram-Schimdt orthogonal basis. The new integer vector is sparse in the sense that most of its components are zero. Moreover, this kind of sparse integer vectors have some special properties such as those nonzero components are mostly ± 1 and they are located at the lower segment. With the help of the sparse representations, vectors act like chromosomes to start a genetic algorithm which performs searching of shortest lattice vectors successfully. Fukase and Kashiwabara [10] extended Schonrr's random sampling technique by considering integer sparse representation (also known as natural number representation); they combined this technique with restricting reduction techniques to solve SVP challenge in dimensions that are much higher than ever.

Our contributions. Our main contribution is to give a more efficient enumeration utilizing the sparse integer representation of shortest vectors. Firstly, we study the orthogonalized integer representation of the shortest vector. This representation enables us to describe a very natural enumeration method which is called the orthognalized enumeration. This enumeration takes a new input parameter k as a measure to control the number of nodes needed for enumeration. To be more specific, the main purpose of our method is to reduce the number of enumeration trials by considering some relationship between the sparse integer (rounding coefficients) vector $\mathbf{y} = (y_1, ..., y_n)$ under the Gram-Schmidt basis $\mathbf{B}^*$ and the (coefficients) vector $\mathbf{x} = (x_1, ..., x_n)$ with respect to the lattice basis $\mathbf{B}$ of a shortest lattice vector to be searched. By choosing a theoretical estimated threshold k and setting $y_i = 0$ $(1 \leq i \leq n - k)$ with high probability, we are able to cut the searching space for the shortest vector $\mathbf{v}$ into $(x_{n-k+1}, ..., x_n)$. For every $(x_{n-k+1}, ..., x_n)$ and its corresponding $(y_{n-k+1}, ..., y_n)$, we can compute the unique values of $x_i, \forall i = 1, ..., n-k$. This means that, our enumeration only searches k-dimensional subspace instead of that for n-dimensional space in those previous methods (e.g., full enumeration, linear enumeration, extreme enumeration etc). Another difference between our technique and other existing enumeration methods is that the integer sparse representation technique in our approach makes the number of nodes for enumeration to be strictly bounded above by the parameter k. Because of these features, our method is ideal for use in high dimension case or in the case with limited computing resources by adjusting the parameter k according to the actual problem's dimension or available computing resources. Furthermore, we propose a new BKZ method called MBKZ by alternately using orthognalized enumeration and traditional enumeration in this paper. By using the Monte-Carlo Simulation, we estimate the expectation of number of nodes under different enumeration methods, and the result shows that exponential speedup can be achieved by our new method, MBKZ. Implementation of our methods have been tested to solve challenging SVP problems with dimension up to 121, the experimental results are also consistent with our theoretical estimation.

We also develop an interesting technique to reduce the searching space of enumeration with non-negligible probability. Although a quantitative analysis about the success probability is not available at this moment, it works very 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 $\mathcal{L}$:

$$\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i, x_i \in \mathbb{Z}\}.$$

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

Shortest vector. A non-zero vector with the smallest Euclidean norm in a lattice $\mathcal{L}$ is called a shortest vector of $\mathcal{L}$. The length of a shortest vector is also called the first minimum and written as $\lambda_1(\mathcal{L})$. Let $\|\mathbf{v}\|$ denote the Euclidean norm of a vector $\mathbf{v} \in \mathbb{R}^m$, then $\lambda_1(\mathcal{L}) = \min_{\mathbf{v}\in\mathcal{L}\setminus\{\mathbf{0}\}} \|\mathbf{v}\|$.

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

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 [31]. It applies successive elementary transformations to an input basis, and outputs a BKZ-reduced basis whose vectors are shorter and more orthogonal. More specifically, 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 lattice $L_{[j,\min(j+\beta-1,n)]}$ generated by the block $\mathbf{B}_{[j,\min(j+\beta-1,n)]} = [\pi_j(\mathbf{b}_j), \pi_j(\mathbf{b}_{j+1}), \cdots, \pi_j(\mathbf{b}_{min(j+\beta-1,n)})]$, where $\pi_j(\mathbf{x}) = \sum_{i=j}^{n} \frac{\langle \mathbf{x}, \mathbf{b}^*{}_i \rangle}{\langle \mathbf{b}^*{}_i, \mathbf{b}^*{}_i \rangle} \mathbf{b}^*{}_i$ is the orthogonal projection on $\text{span}(\mathbf{b}_1, \cdots, \mathbf{b}_{j-1})^{\perp}$. As a result, an integer vector $\mathbf{v} = (v_j, \cdots, v_{min(j+\beta-1,n)})$ is found such that $\|\pi_j(\sum_{i=j}^{\min(j+\beta-1,n)} v_i \mathbf{b}_i)\| = \lambda_1(L_{[j,\min(j+\beta-1,n)]})$. After finding vectors that are shorter than any base vectors, *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. It is observed that the output basis seems to obey $\|\mathbf{b}^*{}_i\|/\|\mathbf{b}^*{}_{i+1}\| \approx q$ with q depending on the quality of BKZ, see also [11]. All of the lattice bases $\mathbf{B}$ discussed in this paper are BKZ-reduced bases unless specified otherwise.

In the rest of our discussion, we shall use the same set of heuristics as that in [11]. These heuristics are listed as follows:

Gaussian Heuristic. The Gaussian Heuristic is used to estimate the number 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 (measurable) subset $\mathcal{S} \subseteq \mathbb{R}^m$, 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, looks 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.

Orthogonalized Integer Representations. A lattice vector $\mathbf{v}$ can be represented as a combination of basis vectors $\mathbf{v} = \mathbf{B}\mathbf{x}$. According to the orthogonalized integer representation [8, 10], $\mathbf{x}$ can be transformed into an integer vector $\mathbf{y}$ with respect to $\mathbf{B}^*$ through the following manner: the basis $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ and its Gram-Schimdt orthogonalization $\mathbf{B}^* = [\mathbf{b}^*{}_1, \ldots, \mathbf{b}^*{}_n]$ are related by $\mathbf{B} = \mathbf{B}^*\mathbf{R}$ where $\mathbf{R} = [R_{ij}]$ with $R_{ij} = \begin{cases} \mu_{ij} & \text{if } 1 \le j < i \le n \\ 1, & \text{if } 1 \le i = j \le n \\ 0 & \text{if } 1 \le i < j \le n. \end{cases}$ For any vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$, assume that $\mathbf{v} = \mathbf{B}\mathbf{x}$ with $= (x_1, \ldots, x_n)$. We define $\mathbf{y}$ to be the rounding integer vector of $\mathbf{R}\mathbf{x}$. More precisely, we first 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 have established a one-to-one correspondence between $\mathbf{x}$ and $\mathbf{y}$, and also a one-to-one correspondence between $\mathbf{v}$ and $\mathbf{y}$:

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

We shall call $\mathbf{y}$ the orthogonalized integer representations in the rest of this paper.

3 Enumeration

In this section, we first recall Full Enumeration and Extreme Prunging Enumeration. We shall present our main contribution, the Orthogonalized Enumeration, in the latter part of this section.

3.1 Full Enumeration

Given a Gram-Schmidt orthogonalized basis $\mathbf{B}^*$ and an upper bound R, the full enumeration method [31] enumerates $x_n, x_{n-1} ..., x_1$ of $\mathbf{x}$ successively under the following constraints:

$$x_n^2 \|\mathbf{b}^*{}_n\|^2 \le R^2,$$

$$(x_{n-1} + \mu_{n,n-1} x_n)^2 \|\mathbf{b}^*{}_{n-1}\|^2 \le R^2 - (x_n)^2 \|\mathbf{b}^*{}_n\|^2,$$

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

Here $l_i = (x_i + \sum_{j>i} x_j \mu_{j,i})^2 \|\mathbf{b}^*{}_i\|^2$. We include this enumeration algorithm in appendix A for the sake of completeness.

The number of nodes that need to be searched is determined by the size of enumeration tree. The total number of tree nodes N_e is estimated as $N_e \approx \sum_{l=1}^{n} H_l$ [11], with the summand H_l being the estimated number of nodes at level l:

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

where $V_l(R) = R^l \cdot \frac{\pi^{l/2}}{\Gamma(l/2+1)}$ and $\|\mathbf{b}^*{}_i\| / \|\mathbf{b}^*{}_{i+1}\| \approx q$. It is noted that 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 extremely small.

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

Analysis given by [11] shows that (1) with well-chosen bounding functions, the linear pruning can reduce the number of nodes searched by a factor of 1.189^n over the full enumeration; and (2) furthermore, a well-chosen extreme pruning can achieve a speedup of 1.414^n compared to the full enumeration.

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 [10] and genetic algorithm [8]. However, to the best of our knowledge, a new efficient enumeration method based on orthogonalized integer representations has not ever been designed. It is therefore one of the purpose of this paper to develop the orthogonalized enumeration in order to make a further improvement for enumeration.

For the orthogonalized enumeration, we introduce a new input k to control the number of nodes enumerated. This is one of the main differences between our method and existing enumeration methods. By choosing a proper k and setting $y_i = 0$ $(1 \leq i \leq n-k)$ with high probability, an enumeration is performed among $(x_{n-k+1}, ..., x_n)$. For every $(x_{n-k+1}, ..., x_n)$ and its corresponding $(y_{n-k+1}, ..., y_n)$, we can compute the unique values of $x_i, \forall i = 1, ..., n-k$ under the condition that $y_i = 0, \forall i = 1, ..., n-k$. In Algorithm 1, we present a detailed description of Orthogonalized Enumeration.

Besides the difference in searching space, our strategy of choosing nodes to be searched by orthogonalized enumeration is also different from others. Instead of scoping a range where x_i may lie in the existing enumeration methods, we decrease the scope of search into several nodes. In particular, enumeration for each x_i is conducted among the following two types of special values: zero point and balance point. Zero point represents the value that makes $|x_i^* b_i^*|$ smallest and balance points are values that make $|x_i^* b_i^*|$ closest to the average value computed based on heuristic 2 and heuristic 3. It is clear that the zero point is always unique but there are two balance points, namely the positive one and the negative one. Note that the average value obtained by heuristics might be erroneous, we set the tolerance bound to be 0.4 (the distance between $|x_i^* b_i^*|$ and the average value has an upper bound 0.5). If a balance point cannot make $|x_i^* b_i^*|$ close enough (according to the tolerance bound) to the average value, we extend the balance point to the values that make $|x_i^* b_i^*|$ the second closest to the average value. This enumeration strategy ensures that there are at most 5 choices for each x_i $(n-k < i < n)$ and 3 choices for x_n during the enumeration, the latter because negative balance points are not considered for choosing x_n due to symmetry. This leads to a conclusion that a process of orthogonalized enumeration will search at most $3 \cdot 5^{k-1}$ nodes. This is an important routine used by Algorithm 1 and is detailed in CPV (COMPUTE_POSSIBLE_VALUE) Procedure.

Next, we give some explanation of the terms used in Algorithm 1. The variable $\mathbf{d}$ is a vector to store the average values. Variables **sv** and *slen* are a vector and an integer that store the shortest vector and its norm respectively. The $n \times n$ matrix **un** and two vectors **ylen**, **uvec** are used to store intermediate results during the depth first search of the enumeration process to avoid repeated calculation. In addition, the $n \times 5$ matrix **poss_v** and vectors **poss_v_cnt**, **poss_v_ind** work together to store the choices for enumeration and decide which one is the next to search. More specifically, Procedure 1 puts the choices in **poss_v** and the number of the choices in **poss_v_cnt**, while **poss_v_ind** indicates which choice is the next to go.

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 $\|\mathbf{v}\|^2 < R_b$
```

Continued on next page

Algorithm 1 – continued from previous page

```
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 $\mathbf{d} := [d_1, ..., d_n] = [R_b^{0.5} n^{-0.5}/\|\mathbf{b}^*{}_1\|, ..., R_b^{0.5} n^{-0.5}/\|\mathbf{b}^*{}_n\|]$. // the average values
3: $\mathbf{sv}_{1\times n} := \mathbf{0}, slen := 0$ // sv stores the shortest vector and slen stores its norm
4: $\mathbf{un}_{n\times n} := \mathbf{0}, \mathbf{ylen}_{1\times n} := \mathbf{0}, \mathbf{uvec}_{1\times n} := \mathbf{0}$ // store intermediate results
5: $\mathbf{poss\_v}_{n\times 5} := \mathbf{0}, \mathbf{poss\_v\_cnt}_{1\times n} := \mathbf{0}, \mathbf{poss\_v\_ind}_{1\times n} := \mathbf{0}$ // store the choices for enumeration and point out which one is the next for search
6: for $x_n = \lceil d_n \rceil, \lfloor d_n \rfloor, 0$ do
7:   $uvec_n := x_n$
8:   $un_{n,i} := x_n \cdot \mu_{n,i}\ \forall i = 1, ..., n-1$
9:   $ylen_n := x_n^2 \|\mathbf{b}^*{}_n\|^2$ // a depth first search starts after recoding these values
10:   for $t = n-1, ..., 1$ do
11:     if $poss\_v\_cnt_t = 0$ then
12:       $(\mathbf{poss\_v}_t, poss\_v\_cnt_t) := \mathbf{CPV}(t, k, d_t, un_{t+1,t})$
13:       $poss\_v\_ind_t := 1$ // poss_v_cnt =0 means CPV procedure has not been called, so update poss_v, poss_v_cnt and poss_v_ind by calling the procedure CPV
14:     else
15:       $poss\_v\_ind_t := poss\_v\_ind_t + 1$ // poss_v_cnt $\neq$ 0 means CPV has been called, poss_v_ind should be increased for the next choice
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:     $ylen_t := ylen_{t+1} + (uvec_t + un_{t+1,t})^2 \|\mathbf{b}^*{}_t\|^2$ // recording to avoid repeated calculation
20:     if $t = 1$ then
21:       if $R_b > ylen_1$ then
22:         if $slen = 0$ or $slen > ylen_1$ then
23:           $slen := ylen_1$
24:           $\mathbf{sv} := \mathbf{uvec}$
25:         end if
26:       end if // when t = 1, the enumeration of a node is done, check if it has a shorter norm, and always store the shortest one in sv and its norm in slen
27:       for $i = t, ..., n-1$ do
```

Continued on next page

Algorithm 1 – continued from previous page

```
28:          if $poss\_v\_ind_i < poss\_v\_cnt_i$ then
29:            $t := i$, break
30:          else
31:            $poss\_v\_cnt_i := 0$
32:          end if
33:        end for // find the first i where $poss\_v\_ind_i < poss\_v\_cnt_i$
           from deep to shallow and reset all poss_v_cnt on the road to
           switch to another branch
34:        $t := t + 1$ // offset the decrease in step 10
35:      end if
36:    end for
37:  end for
38:  $\mathbf{v} = \mathbf{sv}$
39:  return $\mathbf{v}$
```

Procedure 1 – CPV

```
Input: $t, k, d_t, un_{t+1,t}$
Output: a set $\mathbf{c}$ and its cardinality.
 1: $\mathbf{c} := \varnothing$
 2: $\mathbf{c} \leftarrow \mathbf{c} \cup \{\lfloor -un_{t+1,t} \rceil\}$ // add the zero point
 3: if $t \geq n - k + 1$ then
 4:   $\mathbf{c} \leftarrow \mathbf{c} \cup \{\lfloor d_t - un_{t+1,t} \rceil\}$ // add the positive balance point
 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:       $\mathbf{c} \leftarrow \mathbf{c} \cup \{\lfloor d_t - un_{t+1,t} \rceil + 1\}$
 8:     else
 9:       $\mathbf{c} \leftarrow \mathbf{c} \cup \{\lfloor d_t - un_{t+1,t} \rceil - 1\}$
10:     end if
11:   end if// add the second positive balance point if the first one is not
      close enough
12:   $\mathbf{c} \leftarrow \mathbf{c} \cup \{\lfloor -d_t - un_{t+1,t} \rceil\}$ // add the negative balance point
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:       $\mathbf{c} \leftarrow \mathbf{c} \cup \{\lfloor -d_t - un_{t+1,t} \rceil + 1\}$
16:     else
17:       $\mathbf{c} \leftarrow \mathbf{c} \cup \{\lfloor -d_t - un_{t+1,t} \rceil - 1\}$
18:     end if
```

Continued on next page

Procedure 1 – continued from previous page

```
19:     end if// add the second negative balance point if the first one is not
        close enough
20: end if// for those where $t < n$-$k$+1, only the zero point is included
21: return c, $card(\mathbf{c})$
```

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 needed to search. As we can see in Algorithm 1 and Procedure 1, enumerations are restricted to $(x_{n-k+1}, ..., x_n)$ while other x_is are directly computed. The expected number of nodes N can be computed as follows. Let $AvgN_i$ be the average number of choices searched for x_i, then $AvgN_i \leq 5$. Therefore

$$N = 3 \cdot \prod_{i=n-k+1}^{n-1} AvgN_i \leq 3 \cdot 5^{k-1}.$$

For a lattice basis $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_n]$ and its Gram-Schimdt orthogonalization $\mathbf{B}^* = [\mathbf{b}^*{}_1, \dots, \mathbf{b}^*{}_n]$, let $\mathbf{x}^* = (x_1^*, ..., x_n^*)$ be the coefficients of a shortest vector $\mathbf{v}$ with respect to $\mathbf{B}^*$ and set $\mathbf{v}_i = x_i^* \mathbf{b}^*{}_i = x_i^* \|\mathbf{b}^*{}_i\| \cdot \mathbf{b}^*{}_i / \|\mathbf{b}^*{}_i\|$, we have

$$\mathbf{v} = \sum_{i=1}^{n} x_i^* \mathbf{b}^*{}_i = \sum_{i=1}^{n} \mathbf{v}_i.$$

We note that $\mathbf{y} = (y_1, ..., y_n) = (\lfloor x_1^* \rceil, ..., \lfloor x_n^* \rceil)$ is the orthogonalized integer representation of $\mathbf{v}$. Under the Heuristic 2 and Heuristic 3, we can assume that $(\|\mathbf{v}_1\|, ..., \|\mathbf{v}_n\|)$ is distributed uniformly. Thus the success probability of orthogonalized enumeration with parameter k in an n-dimensional lattice can be estimated as:

$$P_{succ}(n, k) = \underset{\mathbf{v} \sim Ball_n(\|\mathbf{b_1}\|)}{Pr} \left(\forall j \in [1, n], \lfloor \|\mathbf{v}_j\| / \|\mathbf{b}^*{}_j\| \rceil \in \tau_j \right)$$

where $\tau_j = \{0\}$ for $j \in [1, n-k]$, and for $j \in [n-k+1, n]$, we have:

$$\tau_j = \begin{cases} \{\lfloor d_j \rceil, \lfloor -d_j \rceil, 0\} & \text{if } \max(|d_j - \lfloor d_j \rceil|, |-d_j - \lfloor -d_j \rceil|) \leq 0.4, \\ \{\lfloor d_j \rfloor, \lceil d_j \rceil, \lfloor -d_j \rceil, 0\} & \text{if } |-d_j - \lfloor -d_j \rceil| \leq 0.4 < |d_j - \lfloor d_j \rceil|, \\ \{\lfloor d_j \rceil, \lfloor -d_j \rfloor, \lceil -d_j \rceil, 0\} & \text{if } |d_j - \lfloor d_j \rceil| \leq 0.4 < |-d_j - \lfloor -d_j \rceil|, \\ \{\lfloor d_j \rfloor, \lceil d_j \rceil, \lfloor -d_j \rfloor, \lceil -d_j \rceil, 0\} & \text{if } \min(|d_j - \lfloor d_j \rceil|, |-d_j - \lfloor -d_j \rceil|) > 0.4. \end{cases}$$

By using Monte-Carlo Simulation, we obtain relationships between the lattice dimension n (ranging from 40 to 130) and success probability $P_{succ}(n,k)$ of the orthogonalized enumeration for each $k = 8, 9, \cdots, 20$. The results are displayed in Figure 1.

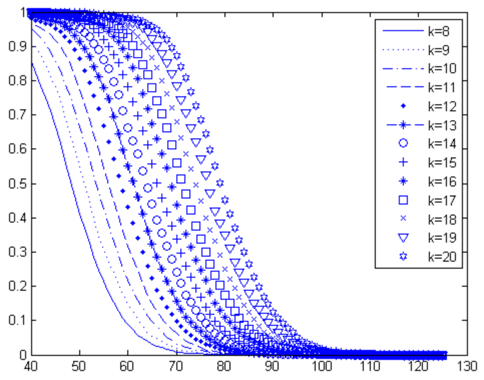

Figure 1. Relationships between n and $P_{succ}(n,k)$ for each $k = 8, 9, \cdots, 20$, the Horizontal Axis is for n and the Vertical Axis for Probability

3.5 A Comparison of Orthogonalized Enumeration and Existing Enumerations

The number N and probability $P_{succ}(n,k)$ for the orthogonalized enumeration obtained in the previous section gives us expected number of nodes needed to search an n dimensional basis using the orthogonalized enumeration, namely $N/P_{succ}(n,k)$. It is remarked that a large k is not always a good choice for maximizing enumeration efficiency. There is a proper range for k that is suitable for enumeration with certain dimension n. For example, $k \leq 10$ when $n = 90$, $k \leq 13$ when $n = 100$ and $k \leq 16$ when $n = 110$. By studying behavior with a proper k we can get the expected number of nodes needed to search an n dimensional basis using the orthogonalized enumeration, denoted as N_{orth}. We also estimate the expected numbers of nodes when using full enumeration, linear pruning enumeration and extreme pruning enumeration, and denote them N_{full}, N_{linear}, $N_{extreme}$. We depict the comparison in Figure 2.

Compared to full enumeration, linear pruning enumeration and extreme pruning enumeration achieve a speedup of 1.189^n and 1.414^n by using a well-chosen strategy, while our experimental data shows that the orthogonalized enumera-

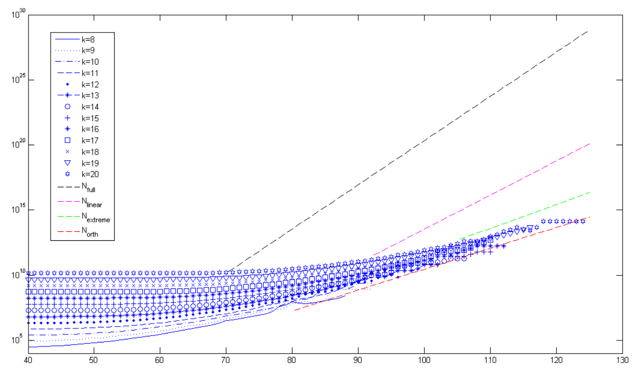

Figure 2. Relationship between Dimension n and Expected Number of Nodes Needed for Different Enumeration Methods, the Horizontal Axis for n and and the Vertical Axis for Expected Number of Nodes

tion can further improve the full enumeration by a factor of 1.512^n. The extreme pruning enumeration uses a nice technique to prune the searching space of $(x_1, ..., x_{n-k})$ to a very small extent and that makes it an extremely effective method. In orthogonalized enumeration, the segment $(x_1, ..., x_{n-k})$ is fixed and needs no work. So the orthogonalized enumeration has a much smaller searching space which is limited by k. Our introduction of the parameter k also provides flexibility 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 the Algorithm

The main idea of Mixed BKZ (MBKZ) is to alternately use orthognalized enumeration and traditional enumeration (full enumeration, linear pruning enumeration, extreme pruning enumeration etc) in solving SVP. In MBKZ 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 2. We also slightly modify the orthognalized enumeration algorithm in Algorithm 3 to make MBKZ more effective.

The design of MBKZ is due to the following reason. According to [7], probability enumeration can speedup the search but the output may not be a shortest vector or even may not return any vector. As a result, in BKZ 2.0, randomizing technique is used to ensure that the enumeration process produces a shorter vector in acceptable time. This is a useful technique, but according to [4], it also brings unavoidable overheads since the bases are not good after being randomized and an extra reduction process needs to be called to reduce the randomized bases before enumeration. Even though no quantitative analysis about the proportion of the extra overheads is given, it is non-negligible in practice. While in MBKZ, we use a new technique to avoid randomizing bases and also ensure enumeration success probability. Experimental data shows that this new technique is more effective and it makes MBKZ a more efficient method compared to the previous ones. We shall explain the main idea of MBKZ in detail next.

In BKZ process, enumeration is called to successively search a better vector $\mathbf{v}$ which is a combination of $(\mathbf{b}_i, ..., \mathbf{b}_j)$ to replace $\mathbf{b}_i$ for all i from 1 to $n-1$, where $j = \min(i+\beta-1, n)$ with β the blocksize. However, the searching of orthognalized enumeration is conducted among the last k dimensions $(\mathbf{b}_{j-k+1}, ..., \mathbf{b}_j)$ and β is set to n, so these mean 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 $\mathbf{b}_{i-1}$ fails, we can reuse the intermediate results to search $\mathbf{b}_i$ to avoid repeating enumeration process. As a result, we can run orthognalized enumeration when $i = 1$, store a best result for each depth and decide which $\mathbf{b}_i$ should be replaced after enumeration. This idea is incorporated in Algorithm 3.

Now we provide some explanations to the terms used in Algorithm 2. The variable $\mathbf{sv}$ is an $n \times n$ matrix to store n shortest vectors that are linear combinations of $n, n-1, ..., 1$ base vectors respectively and the vector $\mathbf{slen}$ records norms of these n vectors. z, jj, kk, h are some indexes for the original BKZ algorithm, z indicates the number of successive enumerations which fails to find a shorter vector, BKZ terminates when $z = n-1$. The variable jj shows the starting index of the block for the next enumeration and kk shows the end index of the block. h is a parameter to bound data size and is used when updating basis by LLL. We introduce a new parameter cnt to switch between traditional enumeration and orthognalized enumeration in MBKZ.

Algorithm 2 – 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 μ, $\|\mathbf{b}^*{}_1\|^2, ..., \|\mathbf{b}^*{}_n\|^2$ and orthognalized enumeration parameter k
Output: A $MBKZ-\beta$ reduced basis $(\mathbf{b}_1, ..., \mathbf{b}_n)$

```
1: $\mathbf{sv}_{n\times n} := \mathbf{0}, \mathbf{slen}_{1\times n} := \mathbf{0}$ // *different from those in algorithm 1, sv and slen
     expand n times to store n vectors with small norm which is respectively a li-*
```

Continued on next page

Algorithm 2 – continued from previous page

```
   near combination of n,n-1,...,1 base vectors
2: $z := 0$, $jj := 0$, $cnt := 0$ // z and jj are two indexes for the original BKZ algorithm and cnt is a new index for MBKZ
3: $LLL(\mathbf{b}_1, ..., \mathbf{b}_n, \mu)$ // LLL is called
4: while $z < n - 1$ do
5:     $jj := jj \bmod (n-1) + 1$ // jj is the index which shows where the block starts and loops among [1,n-1]
6:     if $jj = 1$ then
7:         $cnt := cnt + 1$ // cnt decides whether a Traditonal_Enum or a Orthognalized_Enum should be called and changes when jj=1
8:     end if
9:     if $cnt \bmod 2 = 0$ and $jj = 1$ then
10:        $kk := n$, $h := n$, $\mathbf{v} := (1, 0, ..., 0)$ //in Orthognalized_Enum the blocksize is set to the maximum, v stores the vector obtained from enumeration
11:        $(\mathbf{sv}, \mathbf{slen}) = \mathbf{Orth\_Enum\_for\_MBKZ}(\mu_{[jj,kk]}, \|\mathbf{b}^*{}_{jj}\|^2, ..., \|\mathbf{b}^*{}_{kk}\|^2, k)$ //get n vectors with small norm from the enumeration
12:        for $i = jj, ..., kk$ do
13:            if $slen_i < \|\mathbf{b}^*{}_i\|^2$ then
14:                $\mathbf{v} := \mathbf{sv}_i$, $jj := i$, break;
15:            end if
16:        end for//check successively from 1 to n and find the first vector shorter than the current base vectors
17:    else
18:        $kk := min(jj + \beta - 1, n)$, $h := min(kk + 1, n)$ // Traditonal_Enum demands a normal blocksize, kk and h is reset for Traditonal_Enum
19:        $\mathbf{v} = \mathbf{Traditonal\_Enum}(\mu_{[jj,kk]}, \|\mathbf{b}^*{}_{jj}\|^2, ..., \|\mathbf{b}^*{}_{kk}\|^2)$ // get a short vector in the block
20:    end if
21:    if $\mathbf{v} \neq (1, 0, ..., 0)$ then
22:        $z := 0$
23:        update basis by $LLL(\mathbf{b}_1, ..., \sum_{i=jj}^{kk} v_i \mathbf{b}_i, \mathbf{b}_j, ..., \mathbf{b}_h, \mu)$
24:    else
25:        $z := z + 1$
26:        reduce the next block by $LLL(\mathbf{b}_1, ..., \mathbf{b}_h, \mu)$
27:    end if//z is the index which represents the end condition of BKZ, when a shorter vector is found, z is set to 0, and when no shorter vector can be found for n-1 trials, the algorithm ends
28: end while
```

Algorithm 3 – Orth_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 $\mathbf{d} := [d_1, ..., d_n] = [n^{-0.5}\|\mathbf{b}^*{}_1\|/\|\mathbf{b}^*{}_1\|, ..., n^{-0.5}\|\mathbf{b}^*{}_1\|/\|\mathbf{b}^*{}_n\|]$.
 2: $\underline{\mathbf{sv}_{n\times n} := \mathbf{0}, \mathbf{slen}_{1\times n} := \mathbf{0}}$ //*different from those in algorithm 1, sv and slen expand n times to store n vectors with small norm which is respectively a linear combination of n,n-1,...,1 base vectors*
 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 $x_n = \lceil d_n \rceil, \lfloor d_n \rfloor, 0$ do
 6:   $uvec_n := x_n$
 7:   $un_{n,i} := x_n \cdot \mu_{n,i}\ \forall i = 1, ..., n-1$
 8:   $ylen_n := x_n^2 \|\mathbf{b}^*{}_n\|^2$
 9:   for $t = n-1, ..., 1$ do
10:     if $poss\_v\_cnt_t = 0$ then
11:       $(\mathbf{poss\_v}_t, poss\_v\_cnt_t) := \mathbf{CPV}(t, k, d_t, un_{t+1,t})$
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:     $ylen_t := ylen_{t+1} + (uvec_t + un_{t+1,t})^2 \|\mathbf{b}^*{}_t\|^2$
19:     $\underline{\textbf{if } slen_t = 0 \text{ or } slen_t > ylen_t \textbf{ then}}$
20:       $\underline{slen_t := ylen_t}$
21:       $\underline{\mathbf{sv}_t := (0, ..., 0, uvec_t, ..., uvec_n)}$
22:     $\underline{\textbf{end if}}$//*in algorithm 1, we only concentrate on $slen_1$ and $ylen_1$ because we are aimed to find the shortest vector. While in a MBKZ algorithm, enumeration is called many times to continuously update every base vectors from the first to the last, so recording shortest $ylen_t$ and their norm $slen_t$ is meaningful*
23:     if $t = 1$ then
24:       for $i = t, ..., n-1$ do
25:         if $poss\_v\_ind_i < poss\_v\_cnt_i$ then
26:           $t := i$, *break*
27:         else
28:           $poss\_v\_cnt_i := 0$
29:         end if
30:       end for
31:       $t := t + 1$ // *offset the decrease in step 9*
```

Continued on next page

Algorithm 3 – continued from previous page

```
        end if
    end for
end for
return sv, slen
```

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

Based on our previous discussion, 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 $P_{succ}(j,k) \neq 0$. To be more precise, $P_{succ\_MBKZ}(m,k)$ is the probability of successfully finding a better vector to replace $\mathbf{b}_{n-m+1}$ in the process of orthognalized enumeration. Without using the orthognalized enumeration, this task requires executing a traditional enumeration on an m dimensional lattice. Figure 3 shows graphs of $P_{succ\_MBKZ}(m,k)$ for each $k = 8, 9, \cdots, 20$.

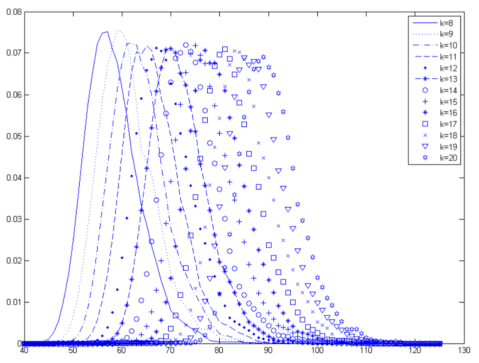

Figure 3. Relationship between m and $P_{succ\_MBKZ}(m,k)$ with $k = 8, 9, ..., 20$, the Horizontal Axis for m and the Vertical Axis for Probability

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 compare orthognalized enumeration and other enumeration methods, we compute expected numbers of nodes needed for different methods inside

MBKZ. Let $\phi_{full}(m)$, $\phi_{linear}(m)$ and $\phi_{extreme}(m)$ denote expected numbers of nodes for finding a better vector in m dimension using full enumeration, linear pruning enumeration and extreme pruning enumeration respectively, then the expected numbers of nodes needed by the three traditional enumerations (to substitute the orthognalized enumeration with the parameter k), denoted as $N_{full,k}$, $N_{linear,k}$ and $N_{extreme,k}$, are given by

$$\begin{aligned}
N_{full,k} &= \sum_m \left(P_{succ\_MBKZ}(m,k) \cdot \phi_{full}(m)\right) \\
N_{linear,k} &= \sum_m \left(P_{succ\_MBKZ}(m,k) \cdot \phi_{linear}(m)\right) \\
N_{extreme,k} &= \sum_m \left(P_{succ\_MBKZ}(m,k) \cdot \phi_{extreme}(m)\right).
\end{aligned}$$

See Figure 4 for the graphs for each $k = 8, 9, \cdots, 20$.

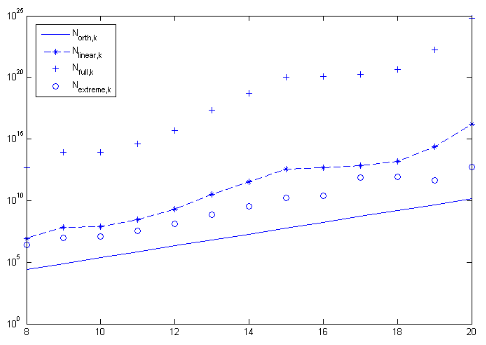

Figure 4. Number of Nodes Needed for Different Enumeration Methods, the Horizontal Axis for k and the Vertical Axis for Expected Number of Nodes

According to the description and discussion given earlier in this section, the design for the orthognalized enumeration in MBKZ brings another speedup of $O(n)$ compared to the original orthognalized enumeration. We have conducted experiments with bases from the SVP challenge site [29] for dimensions up to 140, the results are consistent with our analysis.

Based on our observation through experiments, we have several remarks to make:

1. Orthognalized enumeration can exponentially speedup traditional enumeration, however it is uncertain about which $\mathbf{b}_i$ will be replaced. That is why combining orthognalized enumeration and traditional enumeration method works better and MBKZ can improve previous BKZ methods sharply. It

is remarked that when using orthognalized enumeration independently as an enumeration process in BKZ algorithm, results may not be good enough since k should be set large enough in this situation, in order to keep the probability of updating $\mathbf{b}_1$ non-negligible. This may introduce extra overhead in enumeration.

2. The output of MBKZ generally has better quality compared to that of BKZ or BKZ 2.0 with the same blocksize (this is the blocksize of traditional enumeration used in MBKZ and is different from that of orthognalized enumeration used in MBKZ, the latter is always n), a shortest vector for dimensions $100 - 120$ can be directly found by MBKZ with the blocksize about $40 - 42$. However, BKZ or BKZ 2.0 require a much larger blocksize to work, for example, the blocksize in BKZ 2.0 should be set to 75 to solve challenges with dimensions $90 - 112$ according to [7].
3. When we choose linear pruning or extreme pruning as the traditional enumeration method in MBKZ, randomizing technique is not as necessary as that in BKZ 2.0, because the orthognalized enumeration and traditional enumeration methods have different searching spaces and are continuously updating them independently. Though it is hard to make quantitative analysis, this is thought to be an effective way to reduce duplicate searching and improve the effectiveness further.

4.3 Experiments

It should be noted that MBKZ is a deterministic method, if given the same starting basis and the same set of parameters, the same results will be obtained eventually. We make program codes for MBKZ and all starting lattice bases used for the following experiments publicly available. These experiments about MBKZ can be repeated [1] .

4.3.1 Comparison between Orthognalized Enumeration and Traditional Enumeration during MBKZ

MBKZ runs by alternately using orthognalized enumeration and traditional enumeration, so an important question is that which enumeration plays the biggest role to find a shorter vector, the following experiment result (conducted on a 121-dimensional basis with seed 0) shows the updated base vector with the smallest index after an orthognalized enumeration or after n times traditional enumerations. Results are illustrated in Figure 5.

[1] Program Codes for MBKZ and Experiment Data are available at: https://github.com/zhengzx/MBKZ

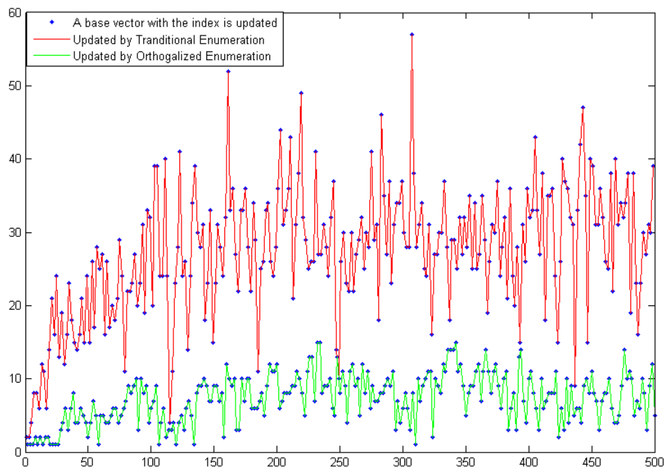

Figure 5. Comparison between Orthognalized Enumeration and Traditional Enumeration during MBKZ, the Horizontal Axis for the Number of Enumerations are called and the Vertical Axis for the Updated Base Vectors' Smallest Index

4.3.2 SVP Challenge

SVP Challenge [29] provides sample lattices for testing algorithms that solve SVP in Euclidean lattices. Many algorithms have been used for solving SVP of the sample lattices. For examples, Kashiwabara and Teruya solved challenges with dimension up to 150 by RSR algorithm [10] using more than 1000 cores and 394 cpu-days, Aono and Nguyen finished challenges with dimension up to 130 by BKZ 2.0 algorithm [7], Wang and Aono et al. achieved challenges with dimension up to 123 by progressive BKZ algorithm [4]. We also conduct experiments in SVP Challenge to test MBKZ and solve several challenges including dimension $99, 105, 113, 121$, see Table 1 for detail. It is remarked that our computational resource is quite limited.

4.3.3 Comparison of MBKZ with Other Methods

We also conduct experiments on different methods including BKZ, BKZ 2.0 and MBKZ under the following conditions.

Basis. All methods start with the same 121-dimension BKZ-10 reduced basis (separately conducted on basis with seed $1, 2, 3$ to avoid accidental circumstances).

Table 1. MBKZ's Results in Solving SVP Challenge

Dimension	Previous Norm	Our Results	CPU Used	Cpu Frequency
99	2642 (seed 0)	2635 (seed 997)	3 CPUs respectively running in seed 997,998 and 999	2.5 GHz
		2606 (seed 998)		
		2604 (seed 999)		
105	2659 (seed 0)	2655 (seed 997)	3 CPUs respectively running in seed 997,998 and 999	2.5 GHz
		2643 (seed 997)		
113	2804 (seed 0)	2739 (seed 999)	3 CPUs respectively running in seed 997,998 and 999	2.5 GHz
121	—	2921 (seed 72)	100 CPUs respectively running in seed 0,1,...,99	2.93 GHz
		2910 (seed 62)		

Blocksize. The blocksize of MBKZ is set to $\beta = 40$. And for BKZ and BKZ 2.0, one of the most efficient progressive strategies, the step-by-step progressive strategy, is used where BKZ (BKZ 2.0) with $\beta = 20, 21, ..., n-1$ is called successively.

Other Parameters and Implementations. The parameter k in MBKZ is set to 12, the pruning parameter for progressive BKZ is 0.15 and the pruning parameter for progressive BKZ 2.0 is set to 20% according to [7]. All methods are based on the C++ NTL library [33]. Progressive BKZ is implemented based on the function BKZ_FP combing with a step-by-step progressive strategy. For progressive BKZ 2.0, since an source code of BKZ 2.0 is not publicly available, we implement it by consulting the pseudo-code in [11, 7] and source codes in NTL library [33] and Progressive BKZ library [5].

Operating System. Linux version 2.6.18 with CPU frequency 2.93 GHz.

Results. As shown in Figure 6,7,8, BKZ and BKZ 2.0 fail to find a shorter vector than MBKZ in all three experiments, even combined with the step-by-step progressive strategy. BKZ and BKZ 2.0 have similar trends, faster in the middle and slower at the start and at the end, while MBKZ's has a relatively uniform speed from start to the end. To obtain a short enough vector, a large blocksize and a good basis are both necessary for BKZ and BKZ 2.0, and that is what progressive strategy is used for. However, an enumeration in a high dimensional lattice is expensive and demands for a low-probability pruning strategy. BKZ and BKZ 2.0 using traditional enumeration methods with a low-probability pruning strategy may get trapped in a local optimum easily. Therefore there is a need to to change search space by either increasing blocksize or randomizing blocks so that the process of finding a shorter vector can be continued. The increasing blocksize strategy has been studied in [4]. The randomizing blocks strategy is likely to introduce extra overhead because the results are uncertain. In contrast, the orthognalized enumeration in MBKZ can be somehow regarded as a ‘positive randomizing blocks strategy’, it changes search space for traditional enumerations and ensures the output basis is a better one at the same time. Combining with the advantages of orthognalized enumeration over traditional enumerations, it is seen that MBKZ is a more efficient method compared to the existing ones.

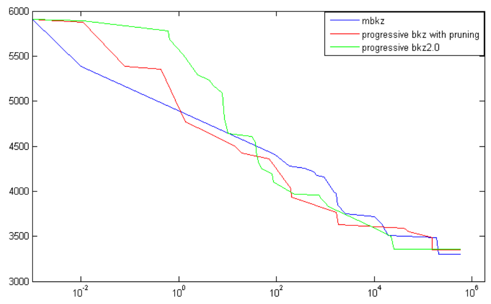

Figure 6. Comparison between Progressive BKZ, Progressive BKZ 2.0 and MBKZ on Basis with Seed 1, the Horizontal Axis for Time and the Vertical Axis for $\|\mathbf{b}_1\|$

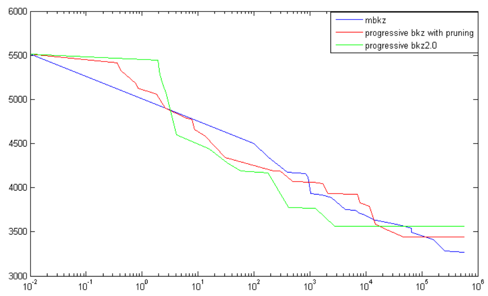

Figure 7. Comparison between Progressive BKZ, Progressive BKZ 2.0 and MBKZ on Basis with Seed 2, the Horizontal Axis for Time and the Vertical Axis for $\|\mathbf{b}_1\|$

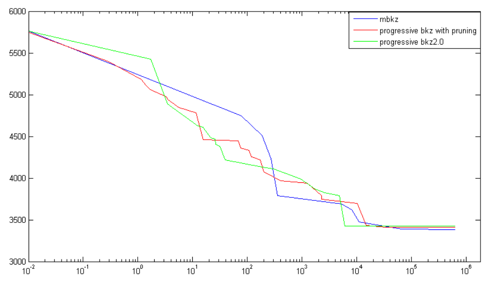

Figure 8. Comparison between Progressive BKZ, Progressive BKZ 2.0 and MBKZ on Basis with Seed 3, the Horizontal Axis for Time and the Vertical Axis for $\|\mathbf{b}_1\|$

5 Further Improvements

In this section, we describe an interesting technique to reduce the searching space of enumeration with non-negligible probability. The main ingredient is to use some special linear transformations to compute some of the coefficients of a possible shortest vector, thus enumerations can be performed in a reduced space to archive greater speedup in all enumeration methods. Even though an exact quantitative success probability analysis for the method is not currently available, experimental results are promissing.

5.1 Description of Method

The main idea of the method is to utilize the property of basis transformation. Let $\mathbf{v} = \sum x_i \mathbf{b}_i$ represent the shortest vector in $\mathcal{L}(\mathbf{B})$. Let

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

and for a new basis $\mathbf{B'} = \mathbf{UB}$. Write $\mathbf{B'} = [\mathbf{b'}_1, ..., \mathbf{b'}_n] = [\mathbf{b}_1 + a_1\mathbf{b}_n, ..., \mathbf{b}_i + a_i\mathbf{b}_n, ..., \mathbf{b}_n]$, we see that

$$\mathbf{v} = \sum x'_i \mathbf{b'}_i = \sum_{i=1}^{n-1} x'_i(\mathbf{b}_i + a_i\mathbf{b}_n) + x'_n\mathbf{b}_n = \sum_{i=1}^{n-1} x'_i\mathbf{b}_i + (\sum_{i=1}^{n-1} a_i x'_i + x'_n)\mathbf{b}_n$$

and hence

$$x'_i = x_i, \qquad \text{for } i = 1, 2, \cdots, n-1,$$
$$x'_n = -\sum_{i=1}^{n-1} a_i x_i + x_n$$

Let us make two extremely strong assumptions first for recovering $(x_1, ..., x_{n-1})$:
Extremely Strong Assumption 1. We know x'_n exactly.
Extremely Strong Assumption 2. $a_i = \omega^i$ and $\omega > 2|x_i|$ for $i = 1, 2, \cdots, n$.
Based on these two extremely strong assumptions, the following Algorithm 4 calculates $(x_1, ..., x_{n-1})$:

Algorithm 4 – The x_i Recovery Algorithm

Input: x'_n, $a_i = \omega^i$
Continued on next page

Algorithm 4 – continued from previous page

```
Output: $(x_1, ..., x_{n-1})$
1: $s := x'_n$
2: for $i = n-1, ..., 1$ 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 $i = 1, ..., n-2$ 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 neat and interesting, but problems are that an exact x'_n may not be available and a_i is too big. So we revisit our assumptions to make them more practical:

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

Assumption 2. $a_i = 0$ for $i = 1, \cdots, m-1$, and $a_i = \omega^{i-m+1}$ for $\omega > 2|x_i|$ and $i = m, \cdots, n$.

From the heuristic 2 and heuristic 3, we assume the coordinates $\mathbf{x}$ of the target vector $\mathbf{v}$ with respect to the orthogonal basis $(\mathbf{b}^*{}_n/\|\mathbf{b}^*{}_n\|, ..., \mathbf{b}^*{}_1/\|\mathbf{b}^*{}_1\|)$ distributes like a uniform vector where $\|\mathbf{x}\| = \|\mathbf{v}\|$. We also assume the same for $\mathbf{x}^*$ under the basis $(\mathbf{b'}^*{}_n/\|\mathbf{b'}^*{}_n\|, ..., \mathbf{b'}^*{}_1/\|\mathbf{b'}^*{}_1\|)$.

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 whose entries on diagonal are all 1, looks like that of a uniformly distributed orthogonal matrix.

We estimate $|x'_n|$ by $n^{-0.5}\|\mathbf{b'}_1\|/\|\mathbf{b'}^*{}_n\|$ according to the heuristic, let $E_{|x'_n|}$ denote the error of the estimation and $P_{error}(\theta)$ denote the probability that

$E_{|x'_n|} \leq \theta$:

$$E_{|x'_n|} = \frac{||x'_n| - n^{-0.5}\|\mathbf{b}'_1\|/\|\mathbf{b}'^*{}_n\||}{|x'_n|}$$
$$P_{error}(\theta) = Pr(E_{|x'_n|} \leq \theta).$$

The probability distribution obtained by Monte-Carlo simulation is depicted in Figure 9.

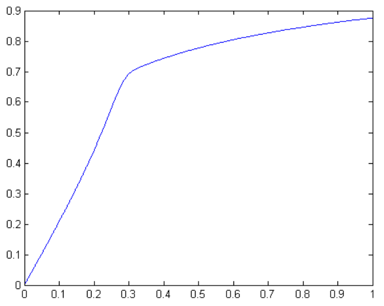

Figure 9. Relationship between θ and P_{error}, the Horizontal Axis for θ and the Vertical Axis for Probability

Errors in the estimation of $|x'_n|$ causes error of computing x_i (note that we use x'_n instead of $|x'_n|$ 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 affected by the error θ, denoted as $th(\omega, i)$, is calculated as $\omega^{-(n-1-i)} - \omega^{-(n-i)}$ (the maximum and minimum of $th(\omega, i)$ are $\omega^{-(n-1-i)}$ and $\omega^{-(n-1-i)} - 2\omega^{-(n-i)}$ respectively and we take the average). For $\omega = 3, 5$ and 7, some values of $th(\omega, i)$ are given in Table 2.

Table 2. $th(\omega, i)$ 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 the probability $P_{nf}(\omega, i)$ of the computation of $(x_i, ..., x_{n-1})$ without being affected by an error caused by an estimation of $|x'_n|$, namely, $P_{nf}(\omega, i) = P_{error}(th(\omega, i))$. Table 3 shows some values of P_{nf} for $\omega = 3, 5$ and 7.

Table 3. $P_{nf}(\omega, i)$ 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 values of $P_{nf}(\omega, i)$, we see that the larger ω is, the harder it is to get more x_is due to the error caused by the estimation of $|x'_n|$.

Now we have $P_{nf}(\omega, i)$ which indicates the influence of the error generated in the estimation of $|x'_n|$. From the analysis earlier, we know that the larger ω is, the smaller $P_{nf}(\omega, i)$ becomes. Let $P_{|x|}(\omega, m)$ be the probability that the assumption 2 holds, where $|x_i| < \omega/2$ for $m \leq i \leq n$. 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 are not able to have an analysis about $P_{|x|}(\omega, m)$, but qualitative analysis can be done. Many experiments based on orthogonalized integer representations [10, 8] have revealed that the orthogonalized integer representation $\mathbf{y} = (y_1, ..., y_n)$ of a shortest vector usually has small components y_i where $y_i = x_i + \lfloor \sum_{j=i+1}^{n} \mu_{j,i} x_j \rceil$. Notice that for a BKZ-reduced basis we generally have $|\mu| < 0.5$, so when i is close to n, one gets x_i from a small y_i subtracting a small number of terms $\mu_{j,i} x_j$ (usually having small absolute value). However, when i decreases, $\lfloor \sum_{j=i+1}^{n} \mu_{j,i} x_j \rceil$ tends to be large which implies $|x_i|$ to be large as well. As a result, $P_{|x|}(\omega, m)$ tends to have positive correlation with m with a fixed ω and tends to be larger when conducting with a better reduced basis.

Though an exact quantitative success probability analysis for the method is not currently available, our experiment shows that the method is practical in a not-very-well reduced basis and the details of the experiment is discussed in the next subsection.

5.3 Experiment

An experiment has been conducted to show how this method works, we study on a 127-dimension reduced basis with $seed = 0, \|\mathbf{b_1}\| = 3344.88, 1.05GH(\mathcal{L}) = 2993.78$ and a number of different $\mathbf{U}$s with various $\omega \leq 5$ are generated. Each of these $\mathbf{U}$s is used in the method described above. In our experiment, we take $m = n - 3$ and concentrate on the combinations of $(x_{n-3}, x_{n-2}, x_{n-1})$ of the shortest vector. The experimental result is shown in Figure 10. It is interesting to see that only limited number of combinations of $(x_{n-3}, x_{n-2}, x_{n-1})$ can be obtained with an uneven distribution, and one of the combination, $(1, -1, 1)$, occurs much more frequently than the others. This is of great help when conducting enumerations on these specific combinations with a high priority to search for the shortest vector. With these x_is possible combinations, enumerations can be conducted in a reduced space and hence achieve a greater efficiency in all enumeration methods including orthogonalized enumeration in this paper and other existing ones.

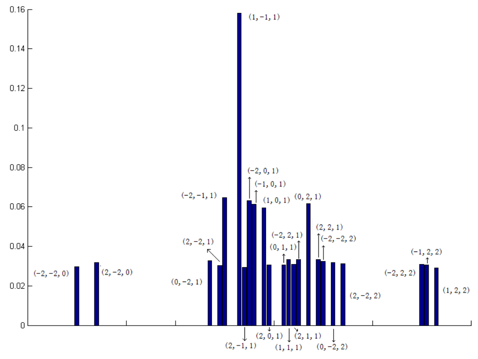

Figure 10. Combinations of $x_{n-3}, x_{n-2}, x_{n-1}$ and Probability, the Horizontal Axis for Different Combinations of $x_{n-3}, x_{n-2}, x_{n-1}$ and the Vertical Axis for Probability

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. Based on our analysis, we can set a suitable threshold to reduce the enumerated space greatly and achieve an exponential speedup compared to the existing enumeration algorithms based on BKZ reduction. Another contribution of this work is to present a new BKZ method named MBKZ. MBKZ involves less enumeration nodes, it also uses a new technique to reduce the duplicate work caused by probability enumeration and in the meanwhile, to avoid the overheads brought by randomizing technique. In addition, MBKZ generally outputs better basis than other BKZ methods with the same blocksize in practice. Finally, a new technique to reduce enumeration space with non-negligible probability is given, though lack of quantitative analysis about its success probability, effectiveness has been demonstrated by experiments.

Further work will be on a simulation algorithm to predict the performance of MBKZ in terms of running time and output quality, which will be of great help in theoretical analysis and conducting experiments in high dimensions.

References

1. Aggarwal, D., Dadush, D., Regev, O., and Stephens-Davidowitz, N. Solving the shortest vector problem in 2^n time using discrete Gaussian sampling. In *Proc. STOC′15*, pages 733-742, 2015.
2. Agrell, E., Eriksson, T., Vardy, A., and Zeger, K. Closest point search in lattices. *IEEE Trans. Inform. Theory*, 48, pages 2201-2214, Aug. 2002.
3. Ajtai, M., Kumar, R., and Sivakumar, D. A sieve algorithm for the shortest lattice vector problem. In *Proc. STOC′01*, pages 601-610, 2001.
4. Aono, Y., Wang, Y., Hayashi, T., and Takagi, T. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In *Proc. EUROCRYPT′16*, pages 789-819, 2016.
5. Aono, Y., Wang, Y., Hayashi, T. and Takagi, T. Progressive BKZ library, available at http://www2.nict.go.jp/security/pbkzcode/index.html
6. Booker, L. B., Goldberg, D. E., and Holland, J. H. Classifier systems and genetic algorithms. *Artificial intelligence*, 40(1-3), pages 235-282, 1989.
7. Chen, Y., and Nguyen, P. Q. BKZ 2.0: Better lattice security estimates. In *Proc. ASIACRYPT′11*, pages 1-20, 2011.
8. Ding, D., Zhu, G., and Wang, X. A Genetic Algorithm for Searching the Shortest Lattice Vector of SVP Challenge. In *Proc. GECCO′15*, pages 823-830, 2015.
9. Eiben, A. E., Aarts, E. H., and Van Hee, K. M. Global convergence of genetic algorithms: A Markov chain analysis. In *Proc. PPSN′91*, pages 4-12, 1991.
10. Fukase, M., and Kashiwabara, K. An accelerated algorithm for solving SVP based on statistical analysis. In *Proc. JIP′15*, 23(1), pages 67-80, 2015.
11. Gama, N., Nguyen, P. Q., and Regev, O. Lattice enumeration using extreme pruning. In *Proc. EUROCRYPT′10*, pages 257-278, 2010.
12. Goldberg, D. E., and Holland, J. H. Genetic algorithms and machine learning. *Machine learning*, 3(2), pages 95-99, 1988.
13. Hanrot, G., and Stehlé, D. Improved analysis of Kannans shortest lattice vector algorithm. In *Proc. CRYPTO′07*, pages 170-186, 2007.
14. Haque, M., Rahman, M. O., and Pieprzyk, J. Analysing progressive-BKZ lattice reduction algorithm. In *Proc. NCICIT′13*, pages 73-80, 2013.
15. Hastad, J., Just, B., Lagarias, J. C., and Schnorr, C. P. Polynomial time algorithms for finding integer relations among real numbers. *SIAM Journal on Computing*, 18(5), pages 859-881, 1989.
16. Helfrich, B. Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. *Theoretical Computer Science*, 41, pages 125-139, 1985.

17. Holland, J. H. Adaptation in natural and artificial systems, Ann Arbor. *The University of Michigan Press*, 1975.
18. Kannan, R. Improved algorithms for integer programming and related lattice problems. In *Proc. STOC′83*, pages 193-206, 1983.
19. Kannan, R. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3), pages 415-440, 1987.
20. Kuo, P. C., Schneider, M., Dagdelen, Ö., Reichelt, J., Buchmann, J., Cheng, C. M., and Yang, B. Y. Extreme Enumeration on GPU and in Clouds. In *Proc. CHES′11*, pages 176-191, 2011.
21. Lenstra, A. K., Lenstra, H. W., and Lovász, L. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4), pages 515-534, 1982.
22. Lindner, R., and Peikert, C. Better key sizes (and attacks) for LWE-based encryption. In *Proc. CT-RSA′11*, pages 319-339, 2011.
23. Micciancio, D., and Walter, M. Practical, predictable lattice basis reduction. In *Proc. EUROCRYPT′16*, pages 820-849, 2016.
24. Micciancio, D., and Voulgaris, P. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. In *Proc. STOC′10*, pages 351-358. 2010.
25. Micciancio, D., and Regev, O. Lattice-based cryptography. *Post-quantum cryptography*, pages 147-191, 2009.
26. Nguyen, P. Q., and Vidick, T. Sieve algorithms for the shortest vector problem are practical. *Journal of Mathematical Cryptology*, 2(2), pages 181-207, 2008.
27. Pujol, X., and Stehlé, D. Solving the Shortest Lattice Vector Problem in Time $2^{2.465n}$. IACR Cryptology ePrint Archive, Report 2009/605.
28. Rückert, M., and Schneider, M. Estimating the Security of Lattice-based Cryptosystems. IACR Cryptology ePrint Archive, Report 2010/137.
29. Schneider, M., and Gama, N. http://www.latticechallenge.org/svp-challenge/.
30. Schnorr, C. P. Lattice reduction by random sampling and birthday methods. In *Proc. STACS′03*, pages 145-156, 2003.
31. Schnorr, C. P., and Euchner, M. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1-3), pages 181-199, 1994.
32. Schnorr, C. P., and Hörner, H. H. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In *Proc. EUROCRYPT′95*, pages 1-12, 1995.
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., and Bi, J. (2011, March). Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem. In *Proc. ASIACCS′11*, pages 1-9, 2011.

A Appendix

Algorithm 5 – The full enumeration algorithm

```
Input: An integral lattice basis $(\mathbf{b_1}, \cdots, \mathbf{b_d})$, a bound $\mathrm{A} \in \mathbb{Z}$
Output: All vectors in $L(\mathbf{b_1}, \cdots, \mathbf{b_d})$ that are of squared norm $\leq \mathrm{A}$
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
```