SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2018, Vol. 61 032115:1–032115:15
https://doi.org/10.1007/s11432-017-9307-0

Orthogonalized lattice enumeration for solving SVP

Zhongxiang ZHENG[1], Xiaoyun WANG[2,3*], Guangwu XU[4] & 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]*Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong Universtiy, Jinan* 250100*, China;*
[4]*Department of Electrical Engineering and Computer Sciences, University of Wisconsin, Milwaukee, WI* 53201*, USA*

Received 6 September 2017/Accepted 30 November 2017/Published online 17 January 2018

Abstract The orthogonalized integer representation was independently proposed by Ding et al. using genetic algorithm and Fukase et al. using sampling technique to solve shortest vector problem (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.

Keywords lattice-based, SVP, sparse representations, enumeration, BKZ

Citation Zheng Z X, Wang X Y, Xu G W, et al. Orthogonalized lattice enumeration for solving SVP. Sci China Inf Sci, 2018, 61(3): 032115, https://doi.org/10.1007/s11432-017-9307-0

1 Introduction

A lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^m$. It is generated by n linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ 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 lattice problems.

(1) Shortest vector problem (SVP). Given a basis of lattice $\mathcal{L}$, find a shortest nonzero vector in the lattice.

(2) 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 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. One of the earliest and most well-known algorithm with exponential complexity is the randomized sieve algorithm proposed in 2001 by Ajtai et al. [1]. The sieve method reduces the upper bound of the time to $2^{O(n)}$ at the cost of

* Corresponding author (email: xiaoyunwang@mail.tsinghua.edu.cn)

$2^{O(n)}$ space compared with Kannan's enumeration method [2]. It has been developed into some improved sieves including heuristic methods in recent years [3–5]. Another important study is the deterministic algorithm with $2^{3.4n+O(n)}$ time and $2^{1.97n+O(n)}$ space given by Micciancio and Voulgaris [6]. The latest progress is the randomized algorithm with $2^{n+o(n)}$ time using the discrete Gaussian sampling method [7], which is the first randomized algorithm (without heuristic assumption) faster than the deterministic algorithm of [6].

For the class of polynomial-space algorithms, two popular techniques are used, one is lattice reduction, including the famous LLL alogrithm [8], HKZ reduction [2] and BKZ (Block-Korkine-Zolotarev) reduction [9]. The other important technique is the 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. One of the most earliest algorithm simultaneously with reasonably time-efficiency and space-efficiency was provided by Kannan [2] in 1980s. This theoretical enumeration algorithm is based on a strong preprocessing 'quasi-HKZ-reduced' basis and achieves worst-case time complexity of $2^{O(n \log n)}$. A more accurate analysis on Kannan's algorithm was given by Helfrich [10], 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 [11] (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 the work of Fincke and Pohst [12] enumeration using LLL-reduced basis, and its enumeration complexity is estimated by Gama et al. [13] 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 ($2^{O(n^2)}$) 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 [14]. It is noted that many security assessments [15–18] and SVP searching methods [19, 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 et al. [13] proposed an improved enumeration using the extreme pruning technique and the speedup is exponential. Chen and Nguyen [21] used the technique to design BKZ 2.0 which improves the BKZ algorithm. Further recent improvements of BKZ include methods based on progressive strategy [22] and result predictions [23].

The main purpose of our paper is to propose a new enumeration method called the orthogonalized 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 orthogonalized basis tend to decrease quickly as the component index gets large. This indicates that for a shortest lattice vector, its coefficients with respect to the othogonalized basis are likely to be zero after rounding 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 [24]. This idea was expanded independently by Ding et al. [25] in a genetic algorithm and by Fukase and Kashiwabara [26] in a sampling algorithm. The genetic algorithm was initiated by Holland [27] in 1975, and it has been used to solve optimization problems such as timetabling, scheduling, and engineering problems [28–30]. 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 and has 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 [26] extended Schnorr'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 $\boldsymbol{y} = (y_1, \ldots, y_n)$ under the Gram-Schmidt basis $\boldsymbol{B^*}$ and the (coefficients) vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ with respect to the lattice basis $\boldsymbol{B}$ of a shortest lattice vector to be searched. By choosing a theoretical estimated threshold k and setting $y_i = 0$ $(1 \leqslant i \leqslant n-k)$ where the components of short vectors are zero with high probability, we are able to cut the searching space for the shortest vector $\boldsymbol{v}$ into $(x_{n-k+1}, \ldots, x_n)$. For every $(x_{n-k+1}, \ldots, x_n)$ and its corresponding $(y_{n-k+1}, \ldots, y_n)$, we can compute the unique values of x_i, $\forall i = 1, \ldots, n-k$. 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). 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 a rather small number depending on 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.

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. Finally a conclusion is given in Section 5.

2 Preliminaries

Lattice. A lattice $\mathcal{L}$ is defined as the set of all integral combinations of n linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ in $\mathbb{R}^m$ $(m \geqslant n)$, these linearly independent vectors are a basis of $\mathcal{L}$:

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

The integer n is the dimension of $\mathcal{L}$ and $\mathrm{vol}(\mathcal{L})$ 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 $\mathrm{vol}(\mathcal{L})$. 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 $\|\boldsymbol{v}\|$ denote the Euclidean norm of a vector $\boldsymbol{v} \in \mathbb{R}^m$, then $\lambda_1(\mathcal{L}) = \min_{\boldsymbol{v} \in \mathcal{L} \setminus \{\boldsymbol{0}\}} \|\boldsymbol{v}\|$.

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 $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$, the Gram-Schmidt process generates an orthogonal set $\boldsymbol{B}^* = [\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*]$ as follows:

$$\boldsymbol{b}_i^* = \boldsymbol{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \boldsymbol{b}_j^*. \tag{1}$$

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

The Gram-Schmidt procedure projects each $\boldsymbol{b}_i$ to the space orthogonal to the space spanned by

$\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_{i-1}^*$, and keeps the determinant unchanged, $\det(\boldsymbol{B}) = \prod_{i=1}^n \|\boldsymbol{b}_i^*\|$.

BKZ. BKZ is a lattice reduction technique with blockwise algorithms [9]. 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 $\beta \geqslant 2$ and a basis $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ of a lattice, it firstly applies LLL to $\boldsymbol{B}$ and then applies enumeration to each lattice $L_{[j,\min(j+\beta-1,n)]}$ generated by the block $\boldsymbol{B}_{[j,\min(j+\beta-1,n)]} = [\pi_j(\boldsymbol{b}_j), \pi_j(\boldsymbol{b}_{j+1}), \ldots, \pi_j(\boldsymbol{b}_{\min(j+\beta-1,n)})]$, where $\pi_j(\boldsymbol{x}) = \sum_{i=j}^n \frac{\langle \boldsymbol{x}, \boldsymbol{b}_i^* \rangle}{\langle \boldsymbol{b}_i^*, \boldsymbol{b}_i^* \rangle} \boldsymbol{b}_i^*$ is the orthogonal projection on $\mathrm{span}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{j-1})^\perp$. As a result, an integer vector $\boldsymbol{v} = (v_j, \ldots, v_{\min(j+\beta-1,n)})$ is found such that $\|\pi_j(\sum_{i=j}^{\min(j+\beta-1,n)} v_i \boldsymbol{b}_i)\| = \lambda_1(L_{[j,\min(j+\beta-1,n)]})$. 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 $\|\boldsymbol{b}_i^*\|/\|\boldsymbol{b}_{i+1}^*\| \approx q$ with q depending on the quality of BKZ, see also [13]. All of the lattice bases $\boldsymbol{B}$ 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 [13]. These heuristics are listed as follows.

(1) 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 $\mathrm{vol}(\mathcal{S})/\mathrm{vol}(\mathcal{L})$.

(2) Heuristic 2. The distribution of the coordinates of the shortest vector $\boldsymbol{v}$, when written in the normalized Gram-Schmidt basis $(\boldsymbol{b}_1^*/\|\boldsymbol{b}_1^*\|, \ldots, \boldsymbol{b}_n^*/\|\boldsymbol{b}_n^*\|)$ of the input basis, has the same distribution as that of a uniformly distributed vector of norm $\|\boldsymbol{v}\|$.

(3) Heuristic 3. The distribution of the normalized Gram-Schmidt orthogonalization $(\boldsymbol{b}_1^*/\|\boldsymbol{b}_1^*\|, \ldots, \boldsymbol{b}_n^*/\|\boldsymbol{b}_n^*\|)$ of a random reduced basis $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ has the same distribution as that of a uniformly distributed orthogonal matrix.

Orthogonalized integer representations. A lattice vector $\boldsymbol{v}$ can be represented as a combination of basis vectors $\boldsymbol{v} = \boldsymbol{B}\boldsymbol{x}$. According to the orthogonalized integer representation [25, 26], $\boldsymbol{x}$ can be transformed into an integer vector $\boldsymbol{y}$ with respect to $\boldsymbol{B}^*$ through the following manner: the basis $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$ and its Gram-Schimdt orthogonalization $\boldsymbol{B}^* = [\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*]$ are related by $\boldsymbol{B} = \boldsymbol{B}^* \boldsymbol{R}^{\mathrm{T}}$ where $\boldsymbol{R} = [R_{ij}]$ with

$$R_{ij} = \begin{cases} \mu_{ij}, & \text{if } 1 \leqslant j < i \leqslant n, \\ 1, & \text{if } 1 \leqslant i = j \leqslant n, \\ 0, & \text{if } 1 \leqslant i < j \leqslant n. \end{cases}$$

For any vector $\boldsymbol{v} \in \mathcal{L}(\boldsymbol{B})$, assume that $\boldsymbol{v} = \boldsymbol{B}\boldsymbol{x}$ with $= (x_1, \ldots, x_n)$. We define $\boldsymbol{y}$ to be the rounding integer vector of $\boldsymbol{R}^{\mathrm{T}}\boldsymbol{x}$. More precisely, we first define a vector $\boldsymbol{t} = (t_1, \ldots, t_n) \in \mathbb{R}^n$ as

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

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

$$y_i = \lfloor x_i^* \rceil = \lfloor x_i + t_i \rceil = x_i + \lfloor t_i \rceil, \quad \text{for } 1 \leqslant i \leqslant n.$$

Since $x_i \in \mathbb{Z}$, we have established a one-to-one correspondence between $\boldsymbol{x}$ and $\boldsymbol{y}$, and also a one-to-one correspondence between $\boldsymbol{v}$ and $\boldsymbol{y}$:

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

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

3 Enumeration

In this section, we first recall full enumeration and extreme pruning 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 $\boldsymbol{B}^*$ and an upper bound R, the full enumeration method [9] enumerates $x_n, x_{n-1} \ldots, x_1$ of $\boldsymbol{x}$ successively under the following constraints:

$$\begin{aligned}
&x_n^2\|\boldsymbol{b}_n^*\|^2 \leqslant R^2,\\
&(x_{n-1}+\mu_{n,n-1}x_n)^2\|\boldsymbol{b}_{n-1}^*\|^2 \leqslant R^2-(x_n)^2\|\boldsymbol{b}_n^*\|^2,\\
&\left(x_i+\sum_{j=i+1}^{n}\mu_{j,i}x_j\right)^2\|\boldsymbol{b}_i^*\|^2 \leqslant R^2-\sum_{j=i+1}^{n}l_j.
\end{aligned}$$

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

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 $N_e \approx \sum_{l=1}^{n} H_l$ [13], with the summand H_l being the estimated number of nodes at level l:

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

where $V_l(R)=R^l\cdot\frac{\pi^{l/2}}{\Gamma(l/2+1)}$ and $\|\boldsymbol{b}_i^*\|/\|\boldsymbol{b}_{i+1}^*\|\approx q$ according to the heuristic of BKZ output. It is noted that H_l gets its maximum value $q^{n^2/8}2^{O(n)}$ when $l=n/2$.

3.2 Extreme pruning enumeration

Extreme pruning enumeration improves full enumeration by replacing the bound R by a serial of bounding functions $R_1,\ldots,R_n$. 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

$$N_{\text{ext}}=\frac{1}{2}\sum_{t=1}^{n}\frac{V_{R_1,\ldots,R_t}}{\prod_{i=n+1-t}^{n}\|\boldsymbol{b}_i^*\|},$$

where $V_{R_1,\ldots,R_t}=V_t(R_t)\cdot\mathbf{Pr}_{\boldsymbol{u}\sim\mathbf{Ball}_t}(\forall j\in[1,t],\sum_{i=1}^{j}u_i^2\leqslant\frac{R_j^2}{R_t^2})$, $V_t(R_t)=R_t^t\cdot\frac{\pi^{t/2}}{\Gamma(t/2+1)}$ and $\mathbf{Ball}_t$ denotes a t-dimensional Euclidean ball.

Analysis under the aforementioned heuristics given by [13] 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 [26] and genetic algorithm [25]. 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 parameter 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 $y_i = 0$ $(1 \leqslant i \leqslant n-k)$, an enumeration is performed among $(x_{n-k+1}, \ldots, x_n)$. For every $(x_{n-k+1}, \ldots, x_n)$ and its corresponding $(y_{n-k+1}, \ldots, y_n)$, we can compute the unique values of $x_i, \forall i = 1, \ldots, n-k$ under the condition that $y_i = 0, \forall i = 1, \ldots, n-k$ by using the method of Babai's nearest hyperplane algorithm. For more detail, one can find a pseudo-code of orthogonalized enumeration in Algorithm 1.

Algorithm 1 Orthogonalized enumeration algorithm

Input: BKZ-reduced basis: $\boldsymbol{B}$, an upper bound of $\|\boldsymbol{v}\|^2$: R_b, k.
Output: The shortest vector $\boldsymbol{v}$ with $\|\boldsymbol{v}\|^2 < R_b$.

For the input basis $\boldsymbol{B}$, compute Gram-Schmidt orthogonalization of it as $\boldsymbol{B}^*$ and $\mu_{i,j}$ as the elements of the lower-triangular matrix where $\boldsymbol{b}_i = \boldsymbol{b}_i^* + \sum_{j=1}^{i-1} \mu_{i,j} \boldsymbol{b}_j^*$;
Compute the $\boldsymbol{d} \leftarrow [d_1, \ldots, d_n] = [R_b^{0.5} n^{-0.5} / \|\boldsymbol{b}_1^*\|, \ldots, R_b^{0.5} n^{-0.5} / \|\boldsymbol{b}_n^*\|]$; // the average values
$\mathbf{sv}_{1\times n} \leftarrow \mathbf{0}$, slen $\leftarrow 0$; // **sv** stores the shortest vector and slen stores its norm
$\mathbf{un}_{n\times n} \leftarrow \mathbf{0}$, $\mathbf{ylen}_{1\times n} \leftarrow \mathbf{0}$, $\mathbf{uvec}_{1\times n} \leftarrow \mathbf{0}$; // store intermediate results
$\mathbf{poss\_v}_{n\times 5} \leftarrow \mathbf{0}$, $\mathbf{poss\_v\_cnt}_{1\times n} \leftarrow \mathbf{0}$, $\mathbf{poss\_v\_ind}_{1\times n} \leftarrow \mathbf{0}$; // store the choices for enumeration and point out which one is the next for search
for $x_n = \lceil d_n \rceil, \lfloor d_n \rfloor, 0$ **do**
  $\mathrm{uvec}_n \leftarrow x_n$;
  $\mathrm{un}_{n,i} \leftarrow x_n \cdot \mu_{n,i}$, $\forall i = 1, \ldots, n-1$;
  $\mathrm{ylen}_n \leftarrow x_n^2 \|\boldsymbol{b}_n^*\|^2$; // a depth first search starts after recording these values
  for $t = n-1, \ldots, 1$ **do**
    if $\mathrm{poss\_v\_cnt}_t = 0$ **then**
      $(\mathbf{poss\_v}_t, \mathrm{poss\_v\_cnt}_t) \leftarrow \mathrm{CPV}(t, k, d_t, \mathrm{un}_{t+1,t})$;
      $\mathrm{poss\_v\_ind}_t \leftarrow 1$; // $\mathrm{poss\_v\_cnt} = 0$ means CPV procedure has not been called, so update poss_v, poss_v_cnt and poss_v_ind by calling the procedure CPV
    else
      $\mathrm{poss\_v\_ind}_t \leftarrow \mathrm{poss\_v\_ind}_t + 1$; // $\mathrm{poss\_v\_cnt} \neq 0$ means CPV has been called, poss_v_ind should be increased for the next choice
    end if
    $\mathrm{uvec}_t \leftarrow \mathrm{poss\_v}_{t,\mathrm{poss\_v\_ind}_t}$;
    $\mathrm{un}_{t,i} \leftarrow \mathrm{un}_{t+1,i} + \mathrm{uvec}_t \cdot \mu_{t,i}$, $\forall i = 1, \ldots, t-1$;
    $\mathrm{ylen}_t \leftarrow \mathrm{ylen}_{t+1} + (\mathrm{uvec}_t + \mathrm{un}_{t+1,t})^2 \|\boldsymbol{b}_t^*\|^2$; // recording to avoid repeated calculation
    if $t = 1$ **then**
      if $R_b > \mathrm{ylen}_1$ **then**
        if slen $= 0$ or slen $> \mathrm{ylen}_1$ **then**
          slen $\leftarrow \mathrm{ylen}_1$;
          sv $\leftarrow$ **uvec**;
        end if
      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
      for $i = t, \ldots, n-1$ **do**
        if $\mathrm{poss\_v\_ind}_i < \mathrm{poss\_v\_cnt}_i$ **then**
          $t \leftarrow i$;
          break
        else
          $\mathrm{poss\_v\_cnt}_i \leftarrow 0$;
        end if
      end for // find the first i where $\mathrm{poss\_v\_ind}_i < \mathrm{poss\_v\_cnt}_i$ from deep to shallow and reset all poss_v_cnt on the road to switch to another branch
      $t \leftarrow t + 1$; // offset the decrease in step 10
    end if
  end for
end for
$\boldsymbol{v} = \mathbf{sv}$;
return $\boldsymbol{v}$.

Besides the advantage of using a smaller 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 a small number of 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 $|x_i^*| \|\boldsymbol{b}_i^*\|$ smallest for given values of $x_{i+1}, \ldots, x_n$. This is because x_i^* is of the form $x_i + \sum_{j=i+1}^{n} \mu_{j,i} x_j$ and the summation part has been

caluclated. Balance points are values that make $|x_i^*|\|\boldsymbol{b}_i^*\|$ closest to the average value $\frac{\sqrt{R}}{\sqrt{n}\|\boldsymbol{b}_i^*\|}$ of the i-th coefficient of shortest vectors in terms of $\boldsymbol{B}^*$, where R is the radius of searching. Note that these average values are computed based on Heuristic 2 and 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 $x_i^*\|\boldsymbol{b}_i^*\|$ and the average value has an upper bound 0.5). If a balance point cannot make $x_i^*\|\boldsymbol{b}_i^*\|$ close enough (according to the tolerance bound) to the average value, we extend the balance point to the values that make $x_i^*\|\boldsymbol{b}_i^*\|$ 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 orthogonalized enumeration algorithm, and more detail can be found in Procedure 1.

Procedure 1 CPV

Input: $t, k, d_t, \mathrm{un}_{t+1,t}$.
Output: A set $\boldsymbol{c}$ and its cardinality.
$\boldsymbol{c} \leftarrow \{\lfloor -\mathrm{un}_{t+1,t}\rceil\}$; // initialize with the zero point
if $t \geqslant n-k+1$ **then**
  if $|d_t - \mathrm{un}_{t+1,t} - \lfloor d_t - \mathrm{un}_{t+1,t}\rceil| > 0.4$ **then**
    $\boldsymbol{c} \leftarrow \boldsymbol{c} \cup \{\lfloor d_t - \mathrm{un}_{t+1,t}\rfloor, \lceil d_t - \mathrm{un}_{t+1,t}\rceil\}$;
  else
    $\boldsymbol{c} \leftarrow \boldsymbol{c} \cup \{\lfloor d_t - \mathrm{un}_{t+1,t}\rceil\}$;
  end if // add the second positive balance point if the first one is not close enough
  if $|-d_t - \mathrm{un}_{t+1,t} - \lfloor -d_t - \mathrm{un}_{t+1,t}\rceil| > 0.4$ **then**
    $\boldsymbol{c} \leftarrow \boldsymbol{c} \cup \{\lfloor -d_t - \mathrm{un}_{t+1,t}\rfloor, \lceil -d_t - \mathrm{un}_{t+1,t}\rceil\}$;
  else
    $\boldsymbol{c} \leftarrow \boldsymbol{c} \cup \{\lfloor -d_t - \mathrm{un}_{t+1,t}\rceil\}$;
  end if // add the second negative balance point if the first one is not close enough
end if // for those where $t < n-k+1$, only the zero point is included
return $\boldsymbol{c}$, card($\boldsymbol{c}$).

3.4 Running time and success probability analysis

The running time of enumeration algorithm is given by:

$$T_{\mathrm{node}} \cdot N,$$

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 $(x_{n-k+1}, \ldots, x_n)$ while other x_is are directly computed. The expected number of nodes N can be computed as follows. Let $\mathrm{Avg}N_i$ be the average number of choices searched for x_i, then $\mathrm{Avg}N_i \leqslant 5$. Therefore

$$N = 3 \cdot \prod_{i=n-k+1}^{n-1} \mathrm{Avg}N_i \leqslant 3 \cdot 5^{k-1}.$$

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

$$\boldsymbol{v} = \sum_{i=1}^{n} x_i^*\boldsymbol{b}_i^* = \sum_{i=1}^{n} \boldsymbol{v}_i. \tag{2}$$

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

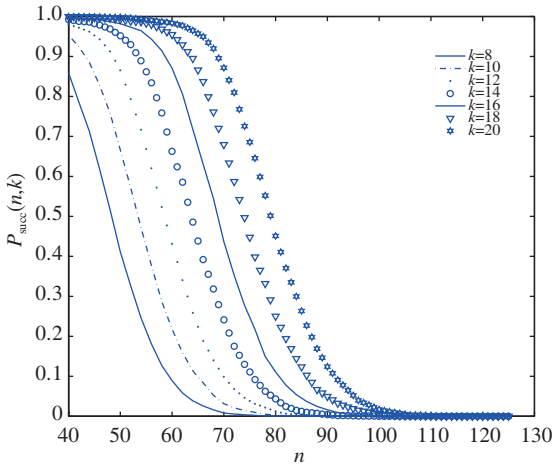

Figure 1 (Color online) Relationships between n and $P_{\text{succ}}(n,k)$ for $k = 8, 10, 12, \ldots, 20$. The horizontal axis is for n and the vertical axis for probability.

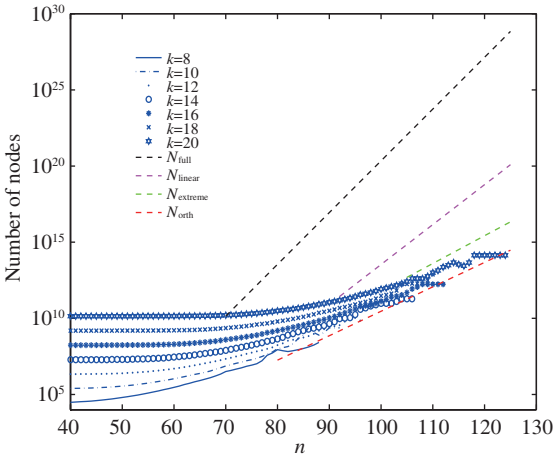

Figure 2 (Color online) Number of nodes needed for different enumeration methods. The horizontal axis for n and the vertical axis for number of nodes.

estimated as

$$P_{\text{succ}}(n,k) = \underset{\boldsymbol{v}\sim\mathbf{Ball}_n(\|\boldsymbol{b}_1\|)}{\mathbf{Pr}}(\forall j \in [1,n], \lfloor \|\boldsymbol{v}_j\|/\|\boldsymbol{b}_j^*\| \rceil \in \tau_j),$$

where $\tau_j = \{0\}$ for $j \in [1, n-k]$, $\mathbf{Ball}_n(\|\boldsymbol{b}_1\|)$ denotes a t-dimensional Euclidean ball of radius $\|\boldsymbol{b}_1\|$. Let d_j denote the expected value of x_j^*, then for $j \in [n-k+1, n]$, we have

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

By using Monte-Carlo simulation, we obtain relationships between the lattice dimension n (ranging from 40 to 130) and success probability $P_{\text{succ}}(n,k)$ of the orthogonalized enumeration for $k = 8, 10, 12, \ldots, 20$. The results are displayed in Figure 1.

3.5 A comparison of orthogonalized enumeration and existing enumerations

The number N and probability $P_{\text{succ}}(n,k)$ for the orthogonalized enumeration obtained in Subsection 3.4 gives us expected number of nodes needed to search an n dimensional basis using the orthogonalized enumeration, namely $N/P_{\text{succ}}(n,k)$. 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, $k \leqslant 10$ when $n = 90$, $k \leqslant 13$ when $n = 100$ and $k \leqslant 16$ 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 enumeration can 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 $(x_1, \ldots, x_{n-k})$ to a very small extent and that makes it an extremely effective method. In orthogonalized enumeration, the segment $(x_1, \ldots, x_{n-k})$ is fixed and does not need to be enumerated. So the orthogonalized enumeration has a

much smaller searching space which is limited by k.

When solving a high-dimensional problem, BKZ method that combines reduction and enumeration is always used. Some time and space efficient enumeration methods including extreme pruning enumeration and orthogonalized enumeration are often with low successful probability. These enumerations may fail to return a desired vector in most cases, and that introduces a greater extra overhead in traditional BKZ method. However, the design of orthogonalized enumeration brings another benefit which allows us to avoid such overhead by reusing intermediate results, in other words, we can always update one of the base vectors after conducting an orthogonalized enumeration. The detail of this benefit is discussed in Section 4. In addition, our introduction of the parameter k also provides flexibility to control the searching process, i.e, the larger k is, the more nodes to be searched and the larger probability to find the shortest vectors. These features make the orthogonalized enumeration a more efficient method than previous methods and they are among our main innovations of this paper.

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) 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 design of MBKZ is due to the following reason. According to [21], 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 [22], 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 $\boldsymbol{v}$ which is a combination of $(\boldsymbol{b}_i, \ldots, \boldsymbol{b}_j)$ to replace $\boldsymbol{b}_i$ 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 $(\boldsymbol{b}_{j-k+1}, \ldots, \boldsymbol{b}_j)$ and β is set to n, so these mean that we are always searching the shortest vector $\boldsymbol{v}$ to replace $\boldsymbol{b}_i$ among the same space $(\boldsymbol{b}_{n-k+1}, \ldots, \boldsymbol{b}_n)$ for all i. All nodes needed to enumerate for replacing $\boldsymbol{b}_i$ are usually included by those enumerated for replacing $\boldsymbol{b}_{i-1}$ if no changes have been made to the basis after the enumeration for $\boldsymbol{b}_{i-1}$. Therefore if the enumeration for $\boldsymbol{b}_{i-1}$ fails, we can reuse the intermediate results to search $\boldsymbol{b}_i$ 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 $\boldsymbol{b}_i$ should be replaced after enumeration. The pseudo code of MBKZ can be found in Algorithm 2 and also the pseudo code of a slightly modified version of orthognalized enumeration algorithm for MBKZ in Algorithm 3.

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 $P_{\text{succ\_MBKZ}}(m, k)$, should be computed as follow:

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

Algorithm 2 The mixed block Korkin-Zolotarev algorithm

Input: A basis $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$, a blocksize $\beta \in \{2, \ldots, n\}$.
Output: A MBKZ$-\beta$ reduced basis $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$.
1: For the input basis $\boldsymbol{B}$, compute Gram-Schmidt orthogonalization $\boldsymbol{B}^*$ and the Gram-Schmidt triangular matrix μ;
2: $\mathbf{sv}_{n\times n} \leftarrow \mathbf{0}$, $\mathbf{slen}_{1\times n} \leftarrow \mathbf{0}$; // different from those in Algorithm 1, **sv** and **slen** expand n times to store n vectors with small norm which are respectively a linear combination of $n, \ldots, 1$ base vectors
3: $z \leftarrow 0$, jj $\leftarrow 0$, cnt $\leftarrow 0$;
4: LLL$(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n, \mu)$; // LLL is called
5: **while** $z < n - 1$ **do**
6: jj $\leftarrow$ (jj (mod $(n-1)$)) $+ 1$;
7: **if** jj $= 1$ **then**
8: cnt $\leftarrow$ cnt $+ 1$; // cnt decides which enum should be called and changes when jj $= 1$
9: **end if**
10: **if** cnt mod $2 = 0$ and jj $= 1$ **then**
11: kk $\leftarrow n$, $\boldsymbol{v} \leftarrow (1, 0, \ldots, 0)$; // in orthognalized_enum the blocksize is set to the maximum
12: $(\mathbf{sv}, \mathbf{slen})$ = Orth_Enum_for_MBKZ$(\mu_{[\text{jj,kk}]}, \|\boldsymbol{b}^*_{\text{jj}}\|^2, \ldots, \|\boldsymbol{b}^*_{\text{kk}}\|^2, k)$; // get n vectors with small norm from the enumeration
13: **for** $i =$ jj, ..., kk **do**
14: **if** $\text{slen}_i < \|\boldsymbol{b}^*_i\|^2$ **then**
15: $\boldsymbol{v} \leftarrow \mathbf{sv}_i$, jj $\leftarrow i$, break;
16: **end if**
17: **end for**// check successively from 1 to n and find the first vector shorter than a basis vector
18: **else**
19: kk $\leftarrow \min(\text{jj} + \beta - 1, n)$;
20: $\boldsymbol{v} =$ Traditonal_Enum$(\mu_{[\text{jj,kk}]}, \|\boldsymbol{b}^*_{\text{jj}}\|^2, \ldots, \|\boldsymbol{b}^*_{\text{kk}}\|^2)$; // get a short vector in the block
21: **end if**
22: **if** $\boldsymbol{v} \neq (1, 0, \ldots, 0)$ **then**
23: $z \leftarrow 0$;
24: insert $\boldsymbol{v}$ into the basis and update it by LLL;
25: **else**
26: $z \leftarrow z + 1$;
27: reduce the next block by LLL;
28: **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
29: **end while**

Algorithm 3 Orth_Enum_for_MBKZ

Input: μ, $\|\boldsymbol{b}^*_1\|^2, \ldots, \|\boldsymbol{b}^*_n\|^2$, k.
Output: $\mathbf{sv}_{n\times n}, \mathbf{slen}_{1\times n}$.
1: Compute the $\boldsymbol{d} \leftarrow [d_1, \ldots, d_n] = [n^{-0.5}\|\boldsymbol{b}^*_1\|/\|\boldsymbol{b}^*_1\|, \ldots, n^{-0.5}\|\boldsymbol{b}^*_1\|/\|\boldsymbol{b}^*_n\|]$;
2: $\mathbf{sv}_{n\times n} \leftarrow \mathbf{0}, \mathbf{slen}_{1\times n} \leftarrow \mathbf{0}$;
Lines from 3 to 18 are the same with the lines from 4 to 19 in Algorithm 1.
19: **if** $\text{slen}_t = 0$ or $\text{slen}_t > \text{ylen}_t$ **then**
20: $\text{slen}_t \leftarrow \text{ylen}_t$;
21: $\mathbf{sv}_t \leftarrow (0, \ldots, 0, \text{uvec}_t, \ldots, \text{uvec}_n)$;
22: **end if**
23: **if** $t = 1$ **then**
Lines from 24 to 35 are the same with the lines from 27 to 38 in Algorithm 1.
36: **return sv**, **slen**.

To be more precise, $P_{\text{succ\_MBKZ}}(m, k)$ is the probability of successfully finding a better vector to replace $\boldsymbol{b}_{n-m+1}$ 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 $P_{\text{succ\_MBKZ}}(m, k)$ for $k = 8, 10, 12, \ldots, 20$.

We compute an expected number of nodes that orthognalized enumeration needs with different k, denoted as $N_{\text{orth},k}$, by using the method described in Subsection 3.4. To compare orthognalized enumeration and other enumeration methods, we compute expected numbers of nodes needed for different methods inside MBKZ. Let $\phi_{\text{full}}(m)$, $\phi_{\text{linear}}(m)$ and $\phi_{\text{extreme}}(m)$ 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 $N_{\text{full},k}$, $N_{\text{linear},k}$ and $N_{\text{extreme},k}$, are given by

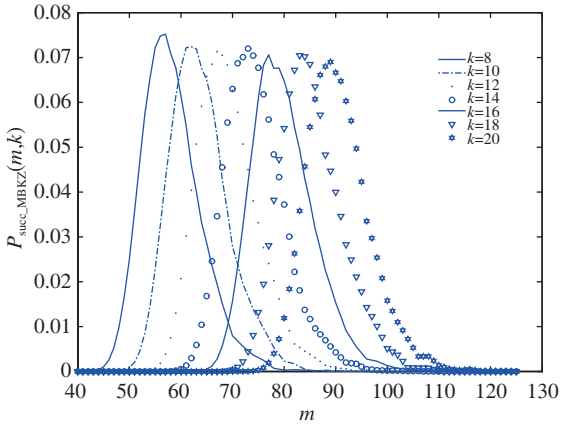

Figure 3 (Color online) Relationship between m and $P_{\text{succ\_MBKZ}}(m,k)$ with $k = 8, 10, 12, \ldots, 20$. The horizontal axis for m and the vertical axis for probability.

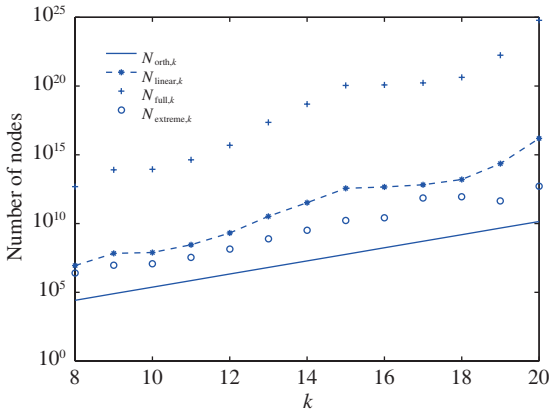

Figure 4 (Color online) Number of nodes needed for different enumeration methods. The horizontal axis for k and the vertical axis for expected number of nodes

$$N_{\text{full},k} = \sum_m (P_{\text{succ\_MBKZ}}(m,k) \cdot \phi_{\text{full}}(m)),$$
$$N_{\text{linear},k} = \sum_m (P_{\text{succ\_MBKZ}}(m,k) \cdot \phi_{\text{linear}}(m)),$$
$$N_{\text{extreme},k} = \sum_m (P_{\text{succ\_MBKZ}}(m,k) \cdot \phi_{\text{extreme}}(m)).$$

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

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 [31] 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 $\boldsymbol{b}_i$ 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 $\boldsymbol{b}_1$ 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 [21].

(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.

4.3.2 *SVP challenge*

SVP Challenge [31] provides sample lattices (indexed by the dimension and seed) for testing algorithms that solve SVP in Euclidean lattices. Many algorithms have been used for solving SVP of the sample lattices. For examples, Fukase and Kashiwabara [26] solved challenges with dimension up to 150 by RSR algorithm using more than 1000 cores and 394 cpu-days, Chen and Nguyen [21] finished challenges with dimension up to 130 by BKZ 2.0 algorithm, Aono et al. [22] achieved challenges with dimension up to 123 by progressive BKZ algorithm. 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 computation resources are 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).

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 $\beta = 20, 21, \ldots, n-1$ is called successively. Besides, we also run a MBKZ with the progressive strategy to make further comparison.

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 [21]. All methods are based on the C++ NTL library [14]. 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 [13, 21] and source codes in NTL library [14] and Progressive BKZ library [32].

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

Results. As shown in Figures 6–8, 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 [22]. 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

1) Program codes for MBKZ and experiment data are available at: https://github.com/zhengzx/MBKZ.

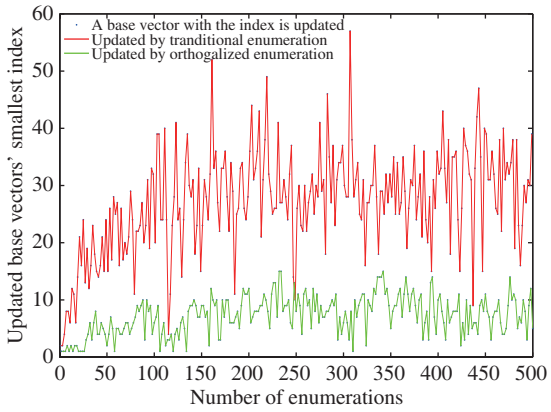

Figure 5 (Color online) Comparison between orthognalized enumeration and traditional enumeration during MBKZ. The horizontal axis for the number of enumerations are called and the vertical axis for the updated base vectors' smallest index.

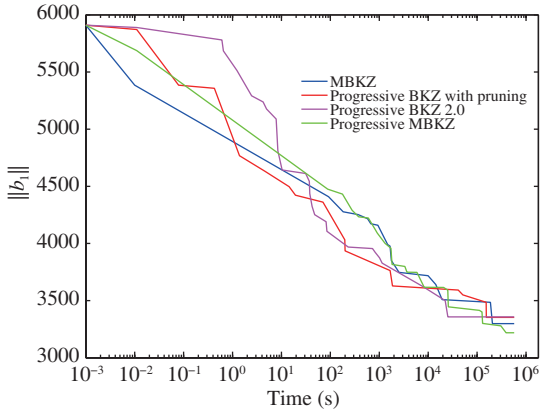

Figure 6 (Color online) Comparison between progressive BKZ, progressive BKZ 2.0, MBKZ and progressive MBKZ on basis with seed 1. The horizontal axis for time and the vertical axis for $\|\boldsymbol{b}_1\|$.

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) 2606 (seed 998) 2604 (seed 999)	3 CPUs respectively running in seed 997998 and 999	2.5 GHz
105	2659 (seed 0)	2655 (seed 997) 2643 (seed 997)	3 CPUs respectively running in seed 997998 and 999	2.5 GHz
113	2804 (seed 0)	2739 (seed 999)	3 CPUs respectively running in seed 997998 and 999	2.5 GHz
121	–	2921 (seed 72) 2910 (seed 62)	100 CPUs respectively running in seed 0,1,...,99	2.93 GHz

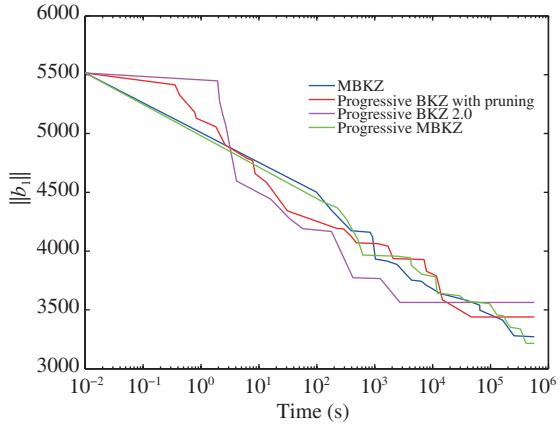

Figure 7 (Color online) Comparison between progressive BKZ, progressive BKZ 2.0, MBKZ and progressive MBKZ on basis with seed 2. The horizontal axis for time and the vertical axis for $\|\boldsymbol{b}_1\|$.

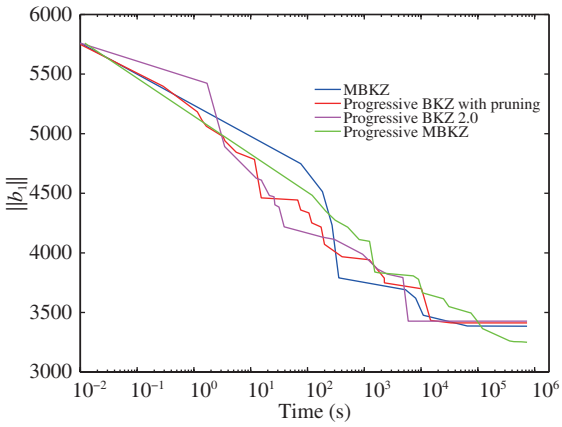

Figure 8 (Color online) Comparison between progressive BKZ, progressive BKZ 2.0, MBKZ and progressive MBKZ on basis with seed 3. The horizontal axis for time and the vertical axis for $\|\boldsymbol{b}_1\|$.

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. What's more, we are glad to find

that progressive strategy works well on MBKZ and helps to find a shorter vector than MBKZ without it in all three test cases. Setting up a proper progressive strategy can further improve the effectiveness of our method.

5 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. These features make it a practical tool in the research of lattice problems.

Acknowledgements This work was supported by National Basic Research Program of China (973 Program) (Grant No. 2013CB834205), and National Natural Science Foundation of China (Grant No. 61133013). The authors would like to thank the anonymous reviewers for thorough and useful comments which have helped to improve the presentation of the paper.

References

1 Ajtai M, Kumar R, Sivakumar D. A sieve algorithm for the shortest lattice vector problem. In: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, Hersonissos, 2001. 601–610
2 Kannan R. Improved algorithms for integer programming and related lattice problems. In: Proceedings of the 15th Annual ACM Symposium on Theory of Computing. New York: ACM, 1983. 193–206
3 Nguyen P Q, Vidick T. Sieve algorithms for the shortest vector problem are practical. J Math Cryptol, 2008, 2: 181–207
4 Pujol X, Stehlé D. Solving the Shortest Lattice Vector Problem in Time $2^{2.465n}$. IACR Cryptology ePrint Archive, Report 2009/605. http://perso.ens-lyon.fr/damien.stehle/downloads/2465.pdf
5 Wang X, Liu M, Tian C, et al. Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, Hong Kong, 2011. 1–9
6 Micciancio D, Voulgaris P. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. SIAM J Comput, 2013, 42: 1364–1391
7 Aggarwal D, Dadush D, Regev O, et al. Solving the shortest vector problem in 2^n time using discrete Gaussian sampling. In: Proceedings of the 47th Annual ACM Symposium on Theory of Computing, Portland, 2015. 733–742
8 Lenstra A K, Lenstra H W, Lovász L. Factoring polynomials with rational coefficients. Math Ann, 1982, 261: 515–534
9 Schnorr C P, Euchner M. Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math Program, 1994, 66: 181–199
10 Helfrich B. Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. Theor Comput Sci, 1985, 41: 125–139
11 Hanrot G, Stehlé D. Improved analysis of Kannan's shortest lattice vector algorithm. In: Proceedings of the 27th Annual International Cryptology Conference, Santa Barbara, 2007. 170–186
12 Fincke U, Pohst M. A procedure for determining algebraic integers of given norm. In: Proceedings of the European Computer Algebra Conference on Computer Algebra. Berlin: Springer, 1983. 194–202
13 Gama N, Nguyen P Q, Regev O. Lattice enumeration using extreme pruning. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco and Nice, 2010. 257–278
14 Shoup V. Number theory c++ library (ntl). http://www.shoup.net/ntl/
15 Lindner R, Peikert C. Better key sizes (and attacks) for LWE-based encryption. In: Proceedings of Cryptographers' Track at the RSA Conference, San Francisco, 2011. 319–339
16 Micciancio D, Regev O. Lattice-based cryptography. In: Post-Quantum Cryptography. Berlin: Springer, 2009. 147–191
17 Rückert M, Schneider M. Estimating the Security of Lattice-based Cryptosystems. IACR Cryptology ePrint Archive, Report 2010/137. https://pdfs.semanticscholar.org/72de/2153c2f5fbe5769a739dfe7a4eb7cc9271de.pdf
18 Schnorr C P, Hörner H H. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In: Proceedings of the 14th Annual International Conference on Theory and Application of Cryptographic Techniques, Saint-Malo, 1995. 1–12

19 Haque M, Rahman M O, Pieprzyk J. Analysing progressive-BKZ lattice reduction algorithm. In: Proceedings of the 1st National Conference on Intelligent Computing and Information Technology, Chittagong, 2013. 73–80
20 Kuo P C, Schneider M, Dagdelen Ö, et al. Extreme enumeration on GPU and in clouds. In: Proceedings of the 13th International Workshop on Cryptographic Hardware and Embedded Systems, Nara, 2011. 176–191
21 Chen Y M, Nguyen P Q. BKZ 2.0: better lattice security estimates. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security, Seoul, 2011. 1–20
22 Aono Y, Wang Y T, Hayashi T, et al. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, 2016. 789–819
23 Micciancio D, Walter M. Practical, predictable lattice basis reduction. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, 2016. 820–849
24 Schnorr C P. Lattice reduction by random sampling and birthday methods. In: Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science. Berlin: Springer, 2003. 145–156
25 Ding D, Zhu G Z, Wang X Y. A genetic algorithm for searching the shortest lattice vector of SVP challenge. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, Madrid, 2015, 823–830
26 Fukase M, Kashiwabara K. An accelerated algorithm for solving SVP based on statistical analysis. J Inf Process, 2015, 23: 67–80
27 Holland J H. Adaptation in Natural and Artificial Systems. Ann Arbor: The University of Michigan Press, 1975
28 Booker L B, Goldberg D E, Holland J H. Classifier systems and genetic algorithms. Artif Intell, 1989, 40: 235–282
29 Eiben A E, Aarts E H, Van Hee K M. Global convergence of genetic algorithms: a Markov chain analysis. In: Proceedings of International Conference on Parallel Problem Solving from Nature, Dortmund, 1990. 4–12
30 Goldberg D E, Holland J H. Genetic algorithms and machine learning. Mach Learn, 1988, 3: 95–99
31 Schneider M, Gama N. SVP CHALLENGE. http://www.latticechallenge.org/svp-challenge/
32 Aono Y, Wang Y, Hayashi T, et al. Progressive BKZ library. http://www2.nict.go.jp/security/pbkzcode/index.html