

Optimized LLL Algorithm

Zhengjun Cao, Lihua Liu

Abstract. The LLL algorithm involves reductions and swaps, corresponding to projection and ordering conditions. But the transformation $\mathbf{b}'_i \leftarrow \mathbf{b}_i - \lfloor \mu_{i,k} \rfloor \mathbf{b}_k$ is not a general size-reduction, where $\mu_{i,k} = \frac{\langle \mathbf{b}_i, \mathbf{b}_k^* \rangle}{\langle \mathbf{b}_k^*, \mathbf{b}_k^* \rangle}$, $\mathbf{b}_k^*$ is the orthogonalized vector of $\mathbf{b}_k$, which cannot ensure $\|\mathbf{b}'_i\| \leq \|\mathbf{b}_i\|$. The condition $(\delta - \mu_{i+1,i}^2)\|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2$ for $\delta \in (1/4, 1)$, cannot ensure $\|\mathbf{b}_i\| \leq \|\mathbf{b}_{i+1}\|$. The two drawbacks possibly result in: (1) the first vector could be longer than others, (2) some vectors could be further reduced. In this paper, we present an optimized LLL algorithm which is independent of orthogonalization and reduces the complexity from $O(n^4)$ to $O(n^3)$. By a probabilistic argument, we show the new algorithm runs in polynomial time.

Keywords: LLL algorithm, projection condition, ordering condition, size-reduction, Gram-Schmidt orthogonalization.

1 Introduction

For the set of linearly independent vectors $B = \{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n\}$ in $\mathbb{R}^k$ ($n \leq k$), the associated lattice of B is defined as $\mathcal{L} = \{v : v = c_1\mathbf{b}_1 + c_2\mathbf{b}_2 + \dots + c_n\mathbf{b}_n, c_i \in \mathbb{Z}\}$. The set B is the basis of the rank n lattice $\mathcal{L}$. The Gram-Schmidt orthogonalization $B^* = \{\mathbf{b}_1^*, \mathbf{b}_2^*, \dots, \mathbf{b}_n^*\}$ of B is computed as follows:

$$\mathbf{b}_1^* = \mathbf{b}_1, \quad \mathbf{b}_2^* = \mathbf{b}_2 - \frac{\langle \mathbf{b}_2, \mathbf{b}_1^* \rangle}{\langle \mathbf{b}_1^*, \mathbf{b}_1^* \rangle} \mathbf{b}_1^*, \quad \dots, \quad \mathbf{b}_n^* = \mathbf{b}_n - \sum_{j=1}^{n-1} \frac{\langle \mathbf{b}_n, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \mathbf{b}_j^*.$$

Notice that $B \subset \mathcal{L}$. In general, $B^* \not\subset \mathcal{L}$.

Define the projection coefficients

$$\mu_{i,k} = \frac{\langle \mathbf{b}_i, \mathbf{b}_k^* \rangle}{\langle \mathbf{b}_k^*, \mathbf{b}_k^* \rangle}, i > k, \quad \mu_{k,k} = 1, \quad \mu_{i,k} = 0 \text{ if } i < k.$$

Z. Cao, Department of Mathematics, Shanghai University, Shanghai, 200444, China.

L. Liu, Department of Mathematics, Shanghai Maritime University, Shanghai, 201306, China.

Email: liulh@shmtu.edu.cn

The Gram-Schmidt orthogonalization can be expressed in matrix form:

$$B = B^* \begin{bmatrix} 1 & \mu_{2,1} & \mu_{3,1} & \cdots & \mu_{n,1} \\ 0 & 1 & \mu_{3,2} & \cdots & \mu_{n,2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Lemma 1 (Ref. [4]). For $\mathbf{u}, \mathbf{v} \in \mathcal{L}$, define $\mu = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle}$. Then, $\|\mathbf{u} - \lfloor \mu \rfloor \mathbf{v}\| \leq \|\mathbf{u}\|$, where $\lfloor \mu \rfloor$ denotes the nearest integer of μ , and $\|\mathbf{u}\|$ denotes the Euclidean norm of $\mathbf{u}$.

Proof. When $|\mu| \leq 1/2$, μ rounds to 0. Hence, $\|\mathbf{u} - \lfloor \mu \rfloor \mathbf{v}\| = \|\mathbf{u}\|$. When $|\mu| > 1/2$, rewrite $\mathbf{u} = \mu \mathbf{v} + \mathbf{v}^*$, where $\mathbf{v}^*$ is a vector orthogonal to $\mathbf{v}$. Then

$$\|\mathbf{u} - \lfloor \mu \rfloor \mathbf{v}\| = \|(\mu - \lfloor \mu \rfloor) \mathbf{v} + \mathbf{v}^*\| \leq \left\| \frac{1}{2} \mathbf{v} + \mathbf{v}^* \right\| < \|\mu \mathbf{v} + \mathbf{v}^*\| = \|\mathbf{u}\|.$$

B is called δ -LLL reduced for $\delta \in (1/4, 1)$, if it satisfies the following two conditions [5]:

- (1) $|\mu_{i,k}| \leq 1/2$ whenever $i > k$. (projection condition)
- (2) $(\delta - \mu_{i+1,i}^2) \|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2$ for all $i < n$. (ordering condition)

Given $\mathcal{L}$, the shortest vector problem (SVP) is to find the shortest non-zero vector $\lambda(B)$ contained in the set $\mathcal{L}(B)$. It is an NP-hard problem to find a vector in a lattice with length within a factor of $\sqrt{2}$ of the length of the shortest vector [9]. Some lattices are used as the core of certain encryption algorithms [7]. These algorithms rely on both the ease of constructing a lattice used for encryption, and the difficulty of obtaining the shortest vectors of a given lattice.

In this paper, we revisit the LLL algorithm, and use the general size-reduction $\mathbf{u} - \lfloor \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle} \rfloor \mathbf{v}$ to replace the transformation $\mathbf{u} - \lfloor \frac{\langle \mathbf{u}, \mathbf{v}^* \rangle}{\langle \mathbf{v}^*, \mathbf{v}^* \rangle} \rfloor \mathbf{v}$. It needs no swap calls. These changes save lots of computational cost for Gram-Schmidt orthogonalization. By a probabilistic argument, we show that the new algorithm terminates in polynomial time. The efficient algorithm suffices for most application cases, in which only the shorter vector is retrieved. We also present a hybrid of LLL algorithm which is suitable for the case that all shorter vectors in the reduced basis should be retrieved.

2 LLL algorithm

The Lenstra-Lenstra-Lovász (LLL) algorithm aims to reduce vectors in a basis for a lattice so as to find short vectors. It takes a basis and repeatedly change the vectors in that basis to make them consistently smaller, while retaining them within the lattice. Notice that the reduction operation does not affect the Gram-Schmidt orthogonalized basis B^* . But the swap operation does affect the orthogonalization. For example, in the step $\mathbf{b}_i \leftarrow \mathbf{b}_i - m \mathbf{b}_k, k < i$, the new

orthogonalized vector is

$$\begin{aligned}
(\mathbf{b}_i - m\mathbf{b}_k) - \sum_{j=1}^{i-1} \frac{\langle (\mathbf{b}_i - m\mathbf{b}_k), \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \mathbf{b}_j^* &= \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^* - m\mathbf{b}_k + m \sum_{j=1}^{i-1} \mu_{k,j} \mathbf{b}_j^* \\
&= \mathbf{b}_i^* - m\mathbf{b}_k + m\mathbf{b}_k^* + m \sum_{j=1}^{k-1} \mu_{k,j} \mathbf{b}_j^* = \mathbf{b}_i^*.
\end{aligned}$$

LLL algorithm
Input: the lattice basis $B = \{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n\}$. Repeat the following two steps until finding the LLL reduced basis. Step 1: Reduction compute $B^* = \{\mathbf{b}_1^*, \mathbf{b}_2^*, \dots, \mathbf{b}_n^*\}$ for $i = 2$ to n do for $k = i - 1$ to 1 do if $\lfloor \mu_{i,k} \rfloor \neq 0$, $\mathbf{b}_i \leftarrow \mathbf{b}_i - \lfloor \mu_{i,k} \rfloor \mathbf{b}_k$ end end Step 2: Swap for $i = 1$ to $n - 1$ do if $(\delta - \mu_{i+1,i}^2) \ \mathbf{b}_i^*\ ^2 > \ \mathbf{b}_{i+1}^*\ ^2$ then swap $\mathbf{b}_{i+1}$ and $\mathbf{b}_i$ go to step 1 end end Output: the reduced basis $\{\mathbf{b}'_1, \mathbf{b}'_2, \dots, \mathbf{b}'_n\}$.

Coppersmith [1] have showed that the LLL algorithm can be used to find a small root of a univariate modular equation. In 2009, Nguyen and Stehlé [10] presented a variation of LLL algorithm with quadratic complexity. Hanrot [3] used LLL algorithm for effective Diophantine approximation. May [8] suggested the usage of LLL algorithm for solving RSA and factorization problems. In 2024, Kalbach and Chinburg [4] revisited the LLL algorithm and its complexity argument. Li and Nguyen [6] provided a complete analysis of the BKZ lattice reduction algorithm. Ducas, Pulles, and Stevens [2] investigated a modern LLL implementation.

3 Analysis of LLL algorithm

Since $\mathbf{b}_k^*$ could be out of the lattice, the LLL algorithm uses the transformation for reduction

$$T(i, k) : \mathbf{b}_i \mapsto \mathbf{b}_i - \left\lfloor \frac{\langle \mathbf{b}_i, \mathbf{b}_k^* \rangle}{\langle \mathbf{b}_k^*, \mathbf{b}_k^* \rangle} \right\rfloor \mathbf{b}_k, \text{ where } k < i \quad (1)$$

which is a variation of

$$S(i, k) : \mathbf{b}_i \mapsto \mathbf{b}_i - \left\lfloor \frac{\langle \mathbf{b}_i, \mathbf{b}_k \rangle}{\langle \mathbf{b}_k, \mathbf{b}_k \rangle} \right\rfloor \mathbf{b}_k, \text{ where } k < i \quad (2)$$

By Lemma 1, we know that $S(i, k)$ is a size-reduction. But $T(i, k)$ is not a general size-reduction.

Let $m = \lfloor \mu_{i,k} \rfloor$. Since $i > k$, we have

$$\begin{aligned} \|\mathbf{b}_i - m\mathbf{b}_k\|^2 &= \left\| \mathbf{b}_i - m \left(\mathbf{b}_k^* + \sum_{j=1}^{k-1} \mu_{k,j} \mathbf{b}_j^* \right) \right\|^2 \\ &= \|\mathbf{b}_i - m\mathbf{b}_k^*\|^2 - 2m \langle \mathbf{b}_i - m\mathbf{b}_k^*, \sum_{j=1}^{k-1} \mu_{k,j} \mathbf{b}_j^* \rangle + m^2 \langle \sum_{j=1}^{k-1} \mu_{k,j} \mathbf{b}_j^*, \sum_{j=1}^{k-1} \mu_{k,j} \mathbf{b}_j^* \rangle \\ &= \|\mathbf{b}_i - m\mathbf{b}_k^*\|^2 + m \langle m \sum_{j=1}^{k-1} \mu_{k,j} \mathbf{b}_j^* - 2\mathbf{b}_i, \sum_{j=1}^{k-1} \mu_{k,j} \mathbf{b}_j^* \rangle \\ &\leq \|\mathbf{b}_i\|^2 + m \langle m \sum_{j=1}^{k-1} \mu_{k,j} \mathbf{b}_j^* - 2\mathbf{b}_i, \sum_{j=1}^{k-1} \mu_{k,j} \mathbf{b}_j^* \rangle \end{aligned}$$

If $m \neq 0$, $m \langle m \sum_{j=1}^{k-1} \mu_{k,j} \mathbf{b}_j^* - 2\mathbf{b}_i, \sum_{j=1}^{k-1} \mu_{k,j} \mathbf{b}_j^* \rangle$ could be greater than 0. In this case, one can not ensure $\|\mathbf{b}_i - m\mathbf{b}_k\| \leq \|\mathbf{b}_i\|$.

The original reduction needs to check the projection condition, and is not a general size-reduction. The original swap operation needs to check the ordering condition. Both operations are dependent on the Gram-Schmidt orthogonalization, which should be updated after each swap, and incurs lots of calculations.

Let $V(B) = \prod_{i=1}^n \|\mathbf{b}_i\|^2$ be the volume of the basis B . Then, the output of LLL algorithm could be not idea:

- the first vector in the output could be longer than others;
- the volume of LLL basis could be further decreased.

Example 1. $B = \{\{11, 23, 75, 301\}, \{102, 273, 13, 51\}, \{-912, -123, 2, 0\}\}$. Its LLL basis is

$$B_1 = \{\{11, 23, 75, 301\}, \{102, 273, 13, 51\}, \{-708, 423, 28, 102\}\}.$$

But $\|\{11, 23, 75, 301\}\|^2 = 96876$, $\|\{102, 273, 13, 51\}\|^2 = 87703$. Its first vector is longer than the second vector. Actually, the basis

$$B_2 = \{\{102, 273, 13, 51\}, \{11, 23, 75, 301\}, \{-810, 150, 15, 51\}\}$$

is more desirable because

$$\|\{102, 273, 13, 51\}\| < \|\{11, 23, 75, 301\}\|, \quad \|\{-810, 150, 15, 51\}\| < \|\{-708, 423, 28, 102\}\|.$$

The Mathematica codes (Fig. 1) can be used to check the correctness.

```

(*-- External function --*)
r[u_List, v_List] := Round[Dot[u, v] / Dot[v, v]]; RD[u_, v_] := (u - r[u, v] * v);
LF[B_List, k_Integer] := Module[{len, i, j, s, t, A, u, v, w, X, Y},
  len = Length[B]; t = 0; A = Sort[B, Norm[#1]^2 < Norm[#2]^2 &];
  Label[begin]; t += 1; X = A;
  For[j = 1, j ≤ len, j++, Y = {}];
    For[i = 1, i ≤ len, i++, u = A[[i]]; v = A[[j]];
      If[i ≠ j, w = RD[u, v]; Y = Append[Y, w], Y = Append[Y, u]]];
  A = Sort[Y, Norm[#1]^2 < Norm[#2]^2 &];
  s = Length[Union[X, A]];
  If[s == len, Print["repetitions=", t]; Goto[end]];
  If[t < k, Goto[begin]]; Label[end]; Return[A];

(*-- Old basis --*)
B = {{11, 23, 75, 301}, {102, 273, 13, 51}, {-912, -123, 2, 0}};
k = 5; S = LF[B, k]; Print["Optimized basis: ", S];
(*-- Internal function: LatticeReduce --*)
T = LatticeReduce[B]; Print["LLL basis: ", T]

repetitions=2

Optimized basis: {{102, 273, 13, 51}, {11, 23, 75, 301}, {-810, 150, 15, 51}}
LLL basis: {{11, 23, 75, 301}, {102, 273, 13, 51}, {-708, 423, 28, 102}}

```

Figure 1: An example of LLL algorithm (Mathematica codes)

4 An optimized LLL algorithm

To improve the original LLL algorithm, one can use the general size-reduction

$$\mathbf{u} \leftarrow \mathbf{u} - \left\lfloor \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle} \right\rfloor \mathbf{v}. \quad (3)$$

If $\|\mathbf{v}\| < \|\mathbf{u}\|$, then $\left| \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle} \right| > \left| \frac{\langle \mathbf{v}, \mathbf{u} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle} \right|$, which means $\left\lfloor \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle} \right\rfloor$ is more likely to unequal to zero. Thus, the probability of event that $\|\mathbf{u} - \left\lfloor \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle} \right\rfloor \mathbf{v}\| < \|\mathbf{u}\|$ is greater than that of $\|\mathbf{v} - \left\lfloor \frac{\langle \mathbf{v}, \mathbf{u} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle} \right\rfloor \mathbf{u}\| < \|\mathbf{v}\|$.

Fact 1. For random binary vectors $\mathbf{u}, \mathbf{v} \in \mathcal{L}$, $\mu = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle}$, define the size-reduction ratio

$$r = \|\mathbf{u} - \lfloor \mu \rfloor \mathbf{v}\|^2 / \|\mathbf{u}\|^2.$$

Then $r \approx 0.95$. For random positive integer vectors, the ratio $r \approx 0.503$. For random integer vectors, the ratio $r \approx 1$. This is verified by the software Mathematica.

Theorem 1. *The optimized LLL algorithm for binary vectors terminates in polynomial time.*

Proof. Let B be the original basis, and $B^{(s)}$ be the basis after s size-reductions. Since

Optimized LLL algorithm
Input: the lattice basis $B = \{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n\}$.
Step 1: Reduction
$\mu_{i,k} \leftarrow \lfloor \frac{\langle \mathbf{b}_i, \mathbf{b}_k \rangle}{\langle \mathbf{b}_k, \mathbf{b}_k \rangle} \rfloor, i \neq k$
$\mathbf{b}_i \leftarrow \mathbf{b}_i - \mu_{i,k} \mathbf{b}_k$
Repeat Step 1 until $\mu_{i,k} = 0$ for all $i \neq k$
Output: the reduced basis B .

$V(B) = \prod_{i=1}^n \|\mathbf{b}_i\|^2$, we have $1 \leq V(B^{(s)}) \approx 0.95^s V(B)$. At worst, we have

$$s = O(n \log \|\mathbf{b}_t\| / |\log(0.95)|)$$

where $\|\mathbf{b}_t\| = \max_i \|\mathbf{b}_i\|$.

The new algorithm is free of swap calls and Gram-Schmidt orthogonalization. These changes save much computational cost. On average, it terminates after $O(n)$ repetitions. The complexity is $O(n^3)$, less than that of LLL algorithm which needs $O(n^4)$ reductions [4].

In experiments, we find the new algorithm always finds the same shorter vector as LLL algorithm. In the case involving only shorter vectors, the new algorithm is more desirable. But the advantage of smaller volume of reduced basis cannot be kept in some cases. Here is an example.

Example 2. Let the old basis is

$$B = \begin{pmatrix} 2 & 1 & -2 & 2 & 1 & -2 & -1 & -1 & 2 & -1 & 1 & 0 & -2 & 1 & -1 \\ -1 & 0 & 2 & -1 & -2 & 1 & -1 & 2 & -1 & 1 & 2 & -2 & 2 & 2 & -1 \\ 2 & 0 & 1 & -1 & -1 & -2 & 0 & -1 & 1 & -2 & 0 & -2 & 1 & 1 & -1 \\ -1 & -2 & 0 & -1 & 1 & -1 & 2 & 0 & 2 & -2 & 0 & 0 & 1 & 2 & 1 \\ 0 & -1 & 2 & 0 & 2 & -2 & 2 & 0 & 1 & -1 & -2 & 0 & 2 & -2 & -2 \\ 1 & 2 & 0 & -1 & 0 & 2 & 1 & 1 & -2 & -2 & 2 & -2 & 1 & 2 & 2 \\ 1 & 1 & 2 & 1 & 1 & -2 & 0 & 0 & 0 & 2 & 0 & 1 & -1 & -1 & -1 \\ 2 & 2 & 1 & -1 & -2 & -2 & 0 & -2 & -2 & 0 & -2 & 2 & -2 & 2 & 2 \\ 0 & -1 & 0 & -2 & -2 & 1 & 2 & 1 & 0 & -1 & -1 & 1 & 1 & 2 & -1 \\ -2 & 0 & -1 & 2 & -2 & 0 & 0 & -1 & -1 & 2 & -2 & -2 & -2 & 2 & 2 \\ 2 & -1 & 0 & -2 & -1 & 2 & 0 & 0 & 1 & 2 & 1 & -1 & 2 & -1 & 2 \\ -1 & 1 & -2 & 1 & -1 & 1 & 2 & -1 & -2 & 0 & 2 & 2 & -2 & 0 & -2 \\ 0 & -2 & -2 & 1 & 0 & 1 & 0 & -2 & 2 & -2 & -2 & 2 & -2 & -2 & 2 \\ -1 & 2 & 0 & 2 & 1 & 1 & 1 & -2 & -1 & 2 & -2 & -2 & 2 & 1 & -1 \\ -1 & 0 & 2 & 0 & 2 & -1 & 0 & -2 & 0 & 1 & -2 & -1 & -2 & 1 & -1 \end{pmatrix}$$

Its LLL basis is

$$B_1 = \begin{pmatrix} -1 & 3 & -1 & 0 & -1 & 0 & -1 & -1 & -1 & 0 & 2 & 0 & -1 & 1 & 0 \\ 1 & -1 & 0 & -1 & 0 & -2 & 1 & 1 & -1 & 1 & -1 & 1 & 0 & 0 & 2 \\ 0 & 1 & 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & -2 & -2 & 1 & -2 & -3 \\ 2 & 0 & -1 & 1 & 0 & 0 & 0 & -2 & 2 & 1 & 1 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 1 & 0 & 2 & 1 & 2 & 2 & 0 & 1 & 0 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & 3 & 1 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 & 0 & 1 & 1 & 3 & 0 & -2 & 1 & 2 & 2 & 0 & 0 \\ -1 & 0 & 2 & 0 & 3 & 0 & -1 & 2 & 0 & 0 & 1 & -4 & -1 & 1 & 0 \\ 1 & -1 & 1 & 0 & 1 & -1 & 0 & -1 & 2 & -1 & -1 & 1 & 0 & -2 & 1 \\ -1 & -1 & -1 & 3 & 1 & -3 & 1 & 0 & 0 & 2 & 0 & -2 & 0 & 0 & 1 \\ 0 & 0 & 2 & -1 & 0 & 1 & 1 & 0 & 0 & -2 & -2 & 0 & 2 & -1 & 1 \\ -1 & -1 & 0 & 1 & 1 & -1 & -1 & -1 & 2 & -1 & -1 & 0 & -2 & 3 & 0 \\ -1 & 0 & 0 & 2 & -1 & 0 & 1 & -1 & 0 & -2 & 1 & 0 & 0 & 1 & -3 \\ 0 & 1 & -2 & -1 & -1 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & -2 & -1 & 0 \\ 1 & 0 & -1 & 1 & 1 & 1 & 2 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

with the volume 9462866845440000000.

The basis generated by the new algorithm is

$$B_2 = \begin{pmatrix} 1 & 3 & -1 & -1 & 0 & 0 & 0 & -1 & -1 & 2 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 0 & -1 & 1 & 0 & 1 & 0 & -2 & 0 & 0 & 1 & 0 & -1 & 1 \\ 0 & 1 & 1 & 0 & 1 & -1 & -1 & 0 & -2 & 1 & -1 & 1 & -2 & -1 & -2 \\ -2 & 0 & -1 & 2 & -1 & 1 & -2 & 0 & -1 & 0 & -1 & -2 & -1 & 1 & -2 \\ 0 & 0 & 0 & 2 & -1 & 0 & 0 & 3 & 2 & -2 & -2 & 2 & 0 & -2 & -1 \\ 0 & 0 & 1 & 1 & 0 & 2 & 1 & 1 & 0 & 1 & 2 & 0 & 0 & 2 & -2 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 & 3 & 2 & -1 & 2 \\ 1 & 0 & 2 & 0 & 0 & 1 & -1 & -1 & -1 & 0 & -2 & 2 & 1 & 2 & 0 \\ -1 & -1 & 1 & 2 & 0 & 0 & 0 & 0 & -2 & -2 & 0 & -1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 0 & -3 & 2 & 0 & 2 & 0 & -1 & -1 & 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & 0 & 1 & -2 & 0 & 1 & -1 & 1 & 2 & 0 & 1 & 2 & 2 \\ 1 & -1 & 0 & 2 & -1 & 1 & 0 & 0 & 1 & 3 & 0 & -1 & 0 & -2 & 0 \\ 1 & 0 & 0 & 0 & -2 & -1 & -1 & 1 & 1 & 0 & 1 & -1 & 1 & 2 & -2 \\ 0 & 1 & -2 & 1 & 1 & 1 & -2 & 1 & 0 & 0 & 2 & 0 & 0 & -1 & 2 \\ -1 & 0 & -1 & 0 & -1 & 2 & -3 & -2 & 1 & 0 & -1 & 1 & 0 & -1 & 1 \end{pmatrix}$$

with the volume 350141110338539520000, which is greater than the previous volume. But both have the same shorter vector $\pm\{1, -1, 0, -2, 0, 0, 1, 1, -1, 1, 0, 1, 1, 0, -1\}$.

In the case that the reduced basis should be of smaller volume, one can use the hybrid of new algorithm and LLL algorithm. Let n be the number of independent vectors of B , B_1 be the LLL basis and B_2 be the optimized basis. Let A be the union of B_1 and B_2 . Sort the set A by the length of each vector. Take the first n shorter vectors to generate a new output.

5 Conclusions

We present an efficient version of LLL algorithm, which is more applicable to the case that only the shorter vector is needed. It reduces the reductions from $O(n^4)$ to $O(n^3)$. In some cases, it can find a reduced basis with smaller volume than LLL algorithm, though it cannot always keep this advantage. The new algorithm can be used to test the security levels of some encryption algorithms based on lattices. We hope the findings could inspire more readers to develop new versions of LLL algorithm and their new applications.

References

- [1] Don Coppersmith. Finding a small root of a univariate modular equation. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 155–165. Springer, 1996.
- [2] Léo Ducas, Ludo N. Pulles, and Marc Stevens. Towards a modern LLL implementation. In Goichiro Hanaoka and Bo-Yin Yang, editors, *Advances in Cryptology - ASIACRYPT 2025 - 31st International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, VIC, Australia, December 8-12, 2025, Proceedings, Part III*, volume 16247 of *Lecture Notes in Computer Science*, pages 65–99. Springer, 2025.
- [3] Guillaume Hanrot. LLL: A tool for effective diophantine approximation. In Phong Q. Nguyen and Brigitte Vallée, editors, *The LLL Algorithm - Survey and Applications*, Information Security and Cryptography, pages 215–263. Springer, 2010.
- [4] Alex Kalbach and Ted Chinburg. Lll algorithm for lattice basis reduction, 2024.
- [5] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.
- [6] Jianwei Li and Phong Q. Nguyen. A complete analysis of the BKZ lattice reduction algorithm. *J. Cryptol.*, 38(1):12, 2025.
- [7] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43:1–43:35, 2013.
- [8] Alexander May. Using lll-reduction for solving RSA and factorization problems. In Phong Q. Nguyen and Brigitte Vallée, editors, *The LLL Algorithm - Survey and Applications*, Information Security and Cryptography, pages 315–348. Springer, 2010.
- [9] Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, 2001.
- [10] Phong Q. Nguyen and Damien Stehlé. An LLL algorithm with quadratic complexity. *SIAM J. Comput.*, 39(3):874–903, 2009.