Reducing Lattice Bases with Bergman Exchange

Jingwei Chen, Yong Feng*, Wenyuan Wu
Chongqing Key Laboratory of Automated Reasoning and Cognition, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China
e-mail: {chenjingwei, yongfeng, wuwenyuan}@cigit.ac.cn

***Abstract*—We present a new algorithm to reduce bases for Euclidean lattices. In contrast to the celebrated Lenstra-Lenstra-Lovász (LLL) algorithm that uses the local Lovász exchange rule, the algorithm in this paper utilizes a global exchange strategy that is introduced by Bergman (1980). We show that the algorithm computes an LLL-reduced basis within polynomial time in the size of the input basis.**

Keywords-lattice basis reduction; LLL; Bergman exchange

I. INTRODUCTION

Lattice basis reduction has been frequently used in many communication systems [1], such as GPS, frequency estimation, and particularly data detection and precoding in MIMO wireless communication systems, and has been applied in many other areas as well, such as factoring polynomials, breaking cryptosystems, solving Diophantine approximation problems, or solving subset sum problems; see [2] and references therein. In this paper, we investigate the common used notion: LLL lattice basis reduction. In the LLL algorithm [3], the crucial point is the so-called Lovász exchange rule, which is powerful to deal with lattice bases (each basis consists of a group of linear independent vectors) and makes the algorithm terminate within polynomial time.

In fact, there exists another exchange rule to deal with more general structures. When one wants to deal with a group of linearly dependent vectors $(\boldsymbol{b}_i)_{i\le n}$, the Lovász exchange rule may not work, since there may exist an element in $\mathcal{L}(\boldsymbol{b}_1,\dots,\boldsymbol{b}_n)=\sum_{i=1}^{n}\mathbb{Z}\,\boldsymbol{b}_i$ which could become arbitrarily small but nonzero. To deal with this case, Bergman proposed an exchange rule in [1]. One can use it to separate the discrete component of $\mathcal{L}(\boldsymbol{b}_1,\dots,\boldsymbol{b}_n)$ (e.g., in [4]), and in the dual sense, to find integer relations for a given real vector [5, 6, 7].

We now summarize three already-known exchange strategies as follows. Each of them depends on the norm of Gram-Schmidt vectors of the given basis.

1. The Lovász exchange: choose the smallest index k such that the Lovász condition (see section II for the definition) does not hold and exchange $\boldsymbol{b}_k$ and $\boldsymbol{b}_{k+1}$.

2. The Siegel exchange: choose the smallest index k such that the Siegel condition (see section II for the definition) does not hold and exchange $\boldsymbol{b}_{\mathrm{k}}$ and $\boldsymbol{b}_{\mathrm{k+1}}$.

3. The Bergman exchange: choose the smallest index k $(1\le k\le n)$ that maximizes $\alpha^k \parallel \boldsymbol{b}_k^{\,*} \parallel^2$ and exchange $\boldsymbol{b}_k$ and $\boldsymbol{b}_{k+1}$, where $\boldsymbol{b}_k^{\,*}$ is the k-th Gram-Schmidt vector of the basis $(\boldsymbol{b}_i)_{i\le n}$ and α >4/3 is an appropriate constant.

The Lovász exchange and the Siegel exchange can be alternatively used for LLL-reducing lattice bases, although the latter one is slightly weaker. In addition, both the two exchange rules are local strategies because that only two consecutive vectors are considered. In contrast, the Bergman exchange considers all vectors so that it is global.

When $n=2$, i.e., there are only two input vectors, the Siegel exchange rule is equivalent to the Bergman exchange rule. Naturally, one will ask what is the relationship among them for $n\ge 3$. As indicated above, the local exchange rules are not suitable to deal with linear dependent vectors. However, can we use the Bergman exchange rule to reduce lattice bases? To the author's best knowledge, there seems no existing result for this question.

In this paper, we give an affirmative answer of the above specified question. We present an algorithm, named BLLL, which computes a reduced basis using the Bergman exchange rule. Given a basis $B=(\boldsymbol{b}_1,\dots,\boldsymbol{b}_n)\in\mathbb{Z}^{n\times n}$ with $\log\max \parallel \boldsymbol{b}_i \parallel \le \beta$, BLLL returns a reduced basis within $\mathcal{O}(n^5\beta)$ arithmetic operations on integers with binary length at most $\mathcal{O}(n\beta)$. It is not difficult to reduce the total bit-complexity bound to $\mathcal{O}(n^{3+\frac{1}{5-\omega}+\varepsilon}\beta^{2+\varepsilon})$ by using modular arithmetic and fast matrix multiplication, as in Stojorhann's LLL algorithm [8], where $\varepsilon>0$ is an arbitrary positive number and $\omega\le 2.376$ is the linear algebra exponent.

In fact, there are several difficulties to use the global Bergman exchange rule. The first one is that if the exchange position k decided by the Bergman exchange is equal to n, then there is no $\boldsymbol{b}_{n+1}$ to be exchanged. It is obvious that only considering the first $n-1$ vectors is not enough. To circumvent this obstacle, we introduce another index ℓ (see the BLLL algorithm) to control the index. The second difficulty is that LLL features a sub-reduced structure, i.e., if the exchange position is k, then the previous $k-1$ vectors are already reduced, however, Bergman-based algorithm does not, because the Bergman exchange rule considers all indices instead two consecutive indices. This leads that it is difficult to control the bit size during the while loop. We use full size reduction instead of the partial size reduction to cope with this problem.

A. Related Work

For a detailed material of LLL and its application, we refer to the book [2] and references therein. We here only focus on those related to the present paper. In [9], Kaltofen presented a lattice reduction algorithm which uses modular arithmetic for the **SizeReduce** step and showed that the

978-1-5090-3820-6/17/$31.00 ©2017 IEEE

algorithm computes a reduced basis (see section II for definition) with bit-complexity bound $O(n^5\beta^2(n+\beta)^{\varepsilon})$. Schnorr [10] proved a bound of $O(n^4\beta^2(n+\beta)^{1+\varepsilon})$ bit operations by employing floating-point arithmetic. Schnorr and Euchner [11] gave efficient implementations. Storjohann [8] achieved $\mathcal{O}(n^{3+\frac{1}{5-\omega}+\varepsilon}\beta^{2+\varepsilon})$ bit operations by employing fraction-free Gaussian elimination, a modular approach and fast matrix multiplication. Koy and Schnorr's segment reduction [12] uses $\mathcal{O}(n^3 \log n)$ arithmetic operations on n-dimensional lattices with basis vector of Euclidean norm at most 2^n. Nguyen and Stehlés algorithm L^2 [13] computes a reduced basis within $O(n^{4+\varepsilon}\beta(n+\beta))$ by employing floating-point arithmetic. Morel, Stehlé and Villard presented H-LLL [14] which has the same complexity bound with L^2 but with a simplified proof. The $\tilde{L}^1$ algorithm [15] presented by Novocin, Stehlé and Villard also employs floating-point arithmetic and achieves complexity bound of $O(n^{5+\varepsilon}\beta^{\varepsilon} + n^{4+\varepsilon}\beta^{1+\varepsilon})$. Saruchi, Morel, Stehlé and Villard [16] proposed a potential speed-up strategy by keeping only the most significant bits of the input basis. Fontein, Schneider, Wagner [17] presented a polynomial time version of LLL with deep insertions. For the moment, the best complexity result is due to Neumaier and Stehlé [18], whose algorithm terminates within $O(n^{4+\varepsilon}\beta^{1+\varepsilon})$ bit operations.

Note that all algorithms appeared in the above work is based on the Lovász exchange rule. Schnorr [19, Section 2] gave a comparison on the Lovász exchange and the Bergman exchange rule. Just [20] uses the Lovász exchange rule and the Bergman exchange rule alternatively for the Diophantine approximation problem. Nevertheless, there seems no algorithm in literature that reduces lattice bases with the Bergman exchange rule. In this sense, the BLLL algorithm proposed in this paper is novel.

However, we should note that the theoretical bit-complexity bound of the naïve version of our Bergman-based reducing algorithm is even not as good as the original LLL algorithm. The reason is that we have to do full size reduction to control the bit size during the while loop of the algorithm. Anyway, it is hopeful to improve it further. For instance, all techniques in [8] apply to our algorithm, and hence resulting in an algorithm with bit complexity bound $\mathcal{O}(n^{3.382}\beta^{2+\varepsilon})$.

B. Notations

For $x \in \mathbb{R}$, $\lceil x \rfloor$ is the nearest integer to x. All vectors are in column and denoted in bold. Let $\| \boldsymbol{x} \|$ and $\| \boldsymbol{x} \|_{\infty}$ be the Euclidean norm and ∞-norm of a vector $\boldsymbol{x}$, respectively.

II. PRELIMINARIES

In this section, we recall some definitions and notations that are helpful for the rest of this paper.

A. Gram-Schmidt Orthogonalization

Let $B = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_n) \in \mathbb{Z}^{n\times n}$ be a full rank matrix and $B^* = (\boldsymbol{b_1}^*, \dots, \boldsymbol{b}_n^*)$ its Gram-Schmidt orthogonalization (GSO) matrix with a unique upper triangular $n \times n$ transform matrix $U = (\mu_{i,j})$ such that $B = B^*U$, where $\mu_{i,i} = 1$ for $1 \le i \le n$ and $\mu_{i,j} = \frac{\langle \boldsymbol{b}_j, \boldsymbol{b}_i^* \rangle}{\langle \boldsymbol{b}_i^*, \boldsymbol{b}_i^* \rangle}$ for $1 \le i < j \le n$. The matrix B^* has rational entries if B has, and then computing the GSO requires $O(n^3)$ operations over $\mathbb{Z}$.

B. Lattices

Let $(\boldsymbol{b}_i)_{i\le n}$ be linearly independent vectors in $\mathbb{R}^m$. A lattice Λ generated by $(\boldsymbol{b}_i)_{i\le n}$ is the set $\mathcal{L}(\boldsymbol{b}_1, \dots, \boldsymbol{b}_n) = \sum_{i=1}^{n} \mathbb{Z}\,\boldsymbol{b}_i$ which consists of all integer linear combinations of $(\boldsymbol{b}_i)_{i\le n}$. We call $(\boldsymbol{b}_i)_{i\le n}$ a basis of Λ. A lattice may have infinitely many bases when $n \ge 2$, but all of them are associated by unimodular matrices (square integer matrices having determinant ± 1), however, each basis has a same number of vectors, which is called the dimension of the lattice. Without loss of generality, we assume $m = n$ in this paper. Let $B = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_n)$. The determinant of Λ, denoted by $\det \Lambda$, is defined as $\sqrt{\det(BB^T)}$.

C. Reduced Basis

The goal of lattice reduction is to find a good basis of a given lattice, in the sense that the basis vectors are as short and orthogonal to each other as possible. Among many existing reduction methods, the LLL algorithm [9] is the most common used one. Given a basis of a lattice, the algorithm halts with an LLL-reduced basis.

Let $1/4 < \delta < 1$, $\alpha > 4/3$ and $B = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_n) \in \mathbb{Z}^{n\times n}$ with GSO $B = B^*U$. Then (1) B is *size-reduced* if $|\mu_{i,j}| \le 1/2$ for all $1 \le i < j \le n$; (2) B is said to satisfy *the Lovász condition* if $\delta \| \boldsymbol{b}_i^* \|^2 \le \| \boldsymbol{b}_{i+1}^* \|^2 + \mu_{i+1,i}^2 \| \boldsymbol{b}_i^* \|^2$ for all i; (3) B is said to satisfy *the Siegel condition* if $\| \boldsymbol{b}_i^* \|^2 \le \alpha \| \boldsymbol{b}_{i+1}^* \|^2$ for all i. Furthermore, B is called *LLL-reduced* if condition (1) and (2) hold; B is called *α-reduced* if condition (1) and (3) hold.

If B is LLL-reduced, then B satisfies the Siegel condition with $\alpha = 4/(4\delta - 1)$, but the converse may not hold. In this sense, the α-reduced notion is slightly weaker than the LLL-reduced notion. However, both conditions (2) and (3) lead to the following conclusion: for every nonzero $\boldsymbol{x} \in \mathcal{L}(\boldsymbol{b}_1, \dots, \boldsymbol{b}_n)$, $\| \boldsymbol{b}_1 \|^2 \le \alpha^{n-1} \| \boldsymbol{x} \|^2$.

D. The LLL Algorithm

We recall the original LLL algorithm as Fig. 1.

Input: A basis $(\boldsymbol{b}_i)_{i\le n}$ of a lattice $\Lambda \subseteq \mathbb{Z}^n$.
Output: An LLL-reduced basis of Λ.

1. Let $k := 1$.
2. While $k \le n - 1$ do
   (a) **SizeReduce**$(k+1, k)$.
   (b) If the Lovász condition holds for k, then **SizeReduce**$(k+1, k)$ for $j = k-1, \dots, 1$ and set $k := k + 1$.
   (c) Else **Exchange** (k); set $k := \max\{k - 1, 1\}$.
3. Return the current basis $(\boldsymbol{b}_i)_{i\le n}$.

Figure 1. The LLL algorithm.

Exchange(k)**:**

(a) Set $\mu = \mu_{k,k+1}$; $\| \boldsymbol{b}_k^* \|^2_{new} \coloneqq \| \boldsymbol{b}_{k+1}^* \|^2 + \mu^2 \|\boldsymbol{b}_k^*\|^2$; $\| \boldsymbol{b}_{k+1}^* \|^2 \coloneqq \frac{\|\boldsymbol{b}_k^*\|^2 \|\boldsymbol{b}_{k+1}^*\|^2}{\|\boldsymbol{b}_k^*\|^2_{new}}$; $\mu_{k,k+1} \coloneqq \mu \frac{\|\boldsymbol{b}_k^*\|^2}{\|\boldsymbol{b}_k^*\|^2_{new}}$; $\| \boldsymbol{b}_k^* \|^2 \coloneqq \| \boldsymbol{b}_k^* \|^2_{new}$.

(b) $\boldsymbol{b}_k \leftrightarrow \boldsymbol{b}_{k+1}$.

(c) For $j = 1,2, \dots, k-1$ do: $\mu_{j,k} \leftrightarrow \mu_{j,k+1}$.

(d) For $j = k+2, \dots, n$ set

$$\begin{pmatrix} \mu_{k,j} \\ \mu_{k+1,j} \end{pmatrix} \coloneqq \begin{pmatrix} 1 & \mu_{k,k+1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -\mu \end{pmatrix} \begin{pmatrix} \mu_{k,j} \\ \mu_{k+1,j} \end{pmatrix}.$$

SizeReduce(k, i)**:**

1. If $|\mu_{i,k}| > 1/2$ then do the following:
   (a) Set $\boldsymbol{b}_k \coloneqq \boldsymbol{b}_k - \lceil \mu_{i,k} \rfloor \boldsymbol{b}_i$.
   (b) For $j = 1,2, \dots, i-1$ do: $\mu_{j,k} \coloneqq \mu_{j,k} - \lceil \mu_{i,k} \rfloor \mu_{j,i}$.
   (c) Set $\mu_{i,k} \coloneqq \mu_{i,k} - \lceil \mu_{i,k} \rfloor$.

In [9], $\delta = 3/4$, so that $\alpha = 2$. Assuming the fast arithmetic is employed, the algorithm correctly computes an LLL reduced basis within $O(n^{5+\varepsilon}\beta^{2+\varepsilon})$ bit operations.

III. The BLLL Algorithm

In this section, we present our main algorithm BLLL (see Fig. 2) and prove its correctness.

Input: A basis $(\boldsymbol{b}_i)_{i \le n}$ of a lattice $\Lambda \subseteq \mathbb{Z}^n$.
Output: A α-reduced basis of Λ.

1. Compute the GSO $B = B^*U$ and set $\ell \coloneqq n$.
2. While $\ell \geq 1$ do
   (a) The Bergman exchange rule determines the smallest k that maximizes $\alpha^k \| \boldsymbol{b}_k^* \|^2$.
   (b) If $k < \ell$ then **SizeReduce**(i,j) for $i = k+1, k, \dots, 1$ and $j = i-1, i-2, \dots, 1$, and then **Exchange**(k).
   (c) If $k = \ell = n$ then set $\ell \coloneqq \ell - 1$.
   (d) If $k = \ell < n$ **SizeReduce**(i,j) for $i = k+1, k, \dots, 1$ and $j = i-1, i-2, \dots, 1$, and then do the following:
      i. If $\| \boldsymbol{b}_k^* \|^2 \leq \alpha \| \boldsymbol{b}_{k+1}^* \|^2$, then $\ell \coloneqq \ell - 1$.
      ii. Else **Exchange**(k) and set $\ell \coloneqq \ell + 1$.
3. Return the current basis $(\boldsymbol{b}_i)_{i \le n}$.

Figure 2. The BLLL algorithm.

If the Bergman exchange rule decides the exchange position $k = n$, then there is no $\boldsymbol{b}_{n+1}$ to be exchanged. We introduce another index ℓ to deal with this case. The index ℓ is helpful for the correctness of the algorithm; see Lemma 1 for details.

In addition, for LLL, if the Lovász exchange decides that the exchange position is k, then the previous $k-1$ vectors are already LLL-reduced. Particularly, the previous $k-1$ vectors are size reduced, which is very helpful to control the bit size during the LLL algorithm. Unfortunately, this may not hold for BLLL. To circumvent this obstacle, we adopt full size reduction in step 2b and 2d instead of the partial size reduction, which is useful in section IV.

To prove the correctness, one should prove that the output basis satisfies the Siegel condition and that the output basis is size-reduced. The former one follows from the following observation.

Lemma 1 *At each execution of step 2(d)i,* $\| \boldsymbol{b}_i^* \|^2 \leq \alpha \|\boldsymbol{b}_{i+1}^*\|^2$ *for all* $i = \ell + 1, \dots, n-1$.

Proof. From the procedure, we have $\ell \leq n-2$ and $\|\boldsymbol{b}_{\ell+1}^*\|^2 \leq \alpha \| \boldsymbol{b}_{\ell+2}^* \|^2$ after execution of step 2(d)i. Particularly, after the first execution of step 2(d)i, we have $\ell = n-2$ and $\| \boldsymbol{b}_{n-1}^* \|^2 \leq \alpha \| \boldsymbol{b}_n^* \|^2$.

Furthermore, the other steps in the while loop will not change $\| \boldsymbol{b}_i^* \|^2$ for $i = \ell + 1, \dots, n$. In fact, $\| \boldsymbol{b}_\ell^* \|^2$ and $\|\boldsymbol{b}_{\ell+1}^*\|^2$ only changes possibly in step 2(d)ii, however, after execution of step 2(d)ii, ℓ is updated to $\ell + 1$.

Proposition 2 *If BLLL terminates, then it returns an α-reduced basis.*

Proof. If the while loop of BLLL terminates, it must terminate at the time of $\ell = 0$ after an execution of step 2(d)i. By Lemma 1, the output basis satisfies the Siegel condition for all $i = 1, \dots, n-1$. Furthermore, step 2d makes the output basis size-reduced.

IV. Analysing BLLL

The main goal of this section is to prove a bit complexity bound of BLLL.

A. Termination

Firstly, we show that the while loop terminates after at most polynomial exchanges. As in analyzing the classical LLL algorithm, we also measure the number of exchanges in the loop. On the one hand, ℓ increases only if step 2(d)ii is executed, i.e., the exchange happens for $k = \ell < n$. On the other hand, at first glance it seems possibly that the algorithm executes step 2b forever. Therefore, the termination is proved at once if the number of exchanges can be bounded.

The idea of the estimate on the number of exchanges is as follows. The first step is to define a function $\mathcal{D}$ (Definition 1) mapping a lattice basis to some positive number. This function is usually called as "potential function". Our aim is to show that $\mathcal{D}$ has the following properties: It is not too large at the beginning, and does not change in the algorithm except that at each exchange step it decreases (at least) by a factor of $\sqrt{\delta}(<1)$. Therefore, only few exchanges can happen.

Definition 1 Let $B = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_n)$ be a lattice basis. The potential of B, denoted $\mathcal{D}$, is defined by

$$\mathcal{D} = \prod_{k=1}^{n} \| \boldsymbol{b}_k^* \|^{n-k+1} = \prod_{k=1}^{n} \left(\prod_{j=1}^{k} \| \boldsymbol{b}_j^* \| \right) = \prod_{k=1}^{n} \mathcal{D}_k$$

where $\mathcal{D}_k \coloneqq \det \Lambda_k$ and the lattice $\Lambda_k = \mathcal{L}(\boldsymbol{b}_1, \dots, \boldsymbol{b}_k)$.

Since $\| \boldsymbol{b}_i^* \| \leq \| \boldsymbol{b}_i \|$, the initial value of $\mathcal{D}$ can be bounded from above by $\mathcal{D} \leq 2^{\frac{\beta n(n+1)}{2}}$. During the size-reduction, $\mathcal{D}$ does not change, since the Gram-Schmidt vectors remain. Now we investigate the exchange step. Suppose that **Exchange**(k) happens, i.e., $\boldsymbol{b}_k$ is exchanged

with $\boldsymbol{b}_{k+1}$. For all $i \neq k$, Λ_i does not change, and so $\mathcal{D}_i$ does not change; only $\mathcal{D}_k$ changes. We have that

$$\frac{D_k^{(new)}}{D_k} = \frac{\|\mu_{k,k+1}\boldsymbol{b}_k^* + \boldsymbol{b}_{k+1}^*\|}{\|\boldsymbol{b}_k^*\|} \leq \sqrt{\frac{1}{4} + \frac{1}{\alpha}} = \sqrt{\delta}.$$

Thus, each exchange decreases $\mathcal{D}$ by a multiplicative factor $\sqrt{\delta} < 1$. Since we always have $1 \leq \mathcal{D} \in \mathbb{Z}$, we can bound from above the number of exchanges by $\log_{1/\sqrt{\delta}} 2^{\frac{\beta n(n+1)}{2}} = \mathcal{O}(n^2\beta)$. Hence we have the following

Proposition 3 *The BLLL algorithm terminates after at most* $\mathcal{O}(n^2\beta)$ exchange steps.

B. Bit-Complexity Bound

It is not difficult to see that in each step we perform only a polynomial number of arithmetic operations (i.e., additions, multiplications, $\lceil\cdot\rfloor$, etc.) over $\mathbb{Z}$. To bound the bit complexity of the algorithm, it suffices to bound the bit sizes of the numbers that arise during and after each step. More precisely, we need to bound the denominators and numerators of the basis vectors $\boldsymbol{b}_i$'s, their Gram-Schmidt vectors $\boldsymbol{b}_i^*$'s, and the rational numbers $\mu_{i,j}$'s. Let's begin with the Gram-Schmidt vectors, since they do not change during the size reduction.

Lemma 4 *Throughout the BLLL algorithm,* $\| \boldsymbol{b}_i^* \| \leq 2^{\beta}$ *for* $1 \leq i \leq n$.

For $\| \boldsymbol{b}_i \|$ and $\mu_{i,j}$, we have the following results.

Lemma 5 *Let* $(\boldsymbol{b}_i^*)_{i\leq n}$ *be the Gram-Schmidt vectors for integer vectors* $(\boldsymbol{b}_i)_{i\leq n}$. *Then we have*

1. $\mathcal{D}_{i-1}^2\boldsymbol{b}_i^* \in \mathbb{Z}$ *for* $1 \leq i \leq n$,
2. $\mathcal{D}_i^2\mu_{i,k} \in \mathbb{Z}$ *for* $1 \leq i < k \leq n$,
3. $|\mu_{i,k}| \leq \mathcal{D}_{i-1} \| \boldsymbol{b}_k \|$ *for* $1 \leq i < k \leq n$.

Lemma 6 *Throughout the algorithm, we have*

1. $|\mu_{i,k}| \leq \sqrt{n}2^{(\beta+1)(n-1)}$ *for* $1 \leq i < k \leq n$,
2. $\| \boldsymbol{b}_i \| \leq n2^{n(\beta+1)}$ *for* $1 \leq i \leq n$.

The standard proof for the bit-complexity bound of the classical LLL algorithm can be adapted for the proof, except that we need full size reduction such that $|\mu_{j,i}| \leq 1/2$ for $1 \leq j < i \leq k$ to prove that during the size reduction,

$$\left|\mu_{j,k} - \lceil\mu_{i,k}\rfloor\mu_{j,i}\right| \leq 2\max_i |\mu_{i,k}|.$$

Putting things together, we get the following main results.

Theorem 7 *The BLLL algorithm computes an* α*-reduced basis of* $\Lambda = \mathcal{L}(\boldsymbol{b}_i) \subset \mathbb{Z}^n$ *and requires at most* $\mathcal{O}(n^5\beta)$ *arithmetic operations on integers with binary length bounded by* $\mathcal{O}(n\beta)$.

Proof. Step 1 takes $\mathcal{O}(n^3)$ integer operations to compute the initial Gram-Schmidt orthogonalization. It is not difficult to conclude that each of either **SizeReduce** or **Exchange** cost $\mathcal{O}(n)$ operations, so that the number of operations used in step 2 is $\mathcal{O}(n^3)$. Thus, it follows from Proposition 3 that the total number of integer arithmetic operations is bounded by $\mathcal{O}(n^5\beta)$.

The denominators $\mathcal{D}_i$ of the rational number computed in the algorithm is bounded from above by $2^{2n\beta}$, whose bit-length is $\mathcal{O}(n\beta)$. The numerators are at most $\max\{\| \boldsymbol{b}_k \|_\infty, \| \mathcal{D}_{k-1}^2\boldsymbol{b}_k^* \|_\infty, \left|\mathcal{D}_i^2\mu_{i,k}\right|\}$.

From Lemma 6, we have $\| \boldsymbol{b}_k \|_\infty \leq \| \boldsymbol{b}_k \| \leq n2^{n(\beta+1)}$. From Lemma 5, we have $\| \mathcal{D}_{k-1}^2\boldsymbol{b}_k^* \|_\infty \leq \| \mathcal{D}_{k-1}^2\boldsymbol{b}_k^* \| \leq 2^{2n\beta}$ and $\left|\mathcal{D}_i^2\mu_{i,k}\right| \leq \mathcal{D}_i^2\mathcal{D}_{i-1} \| \boldsymbol{b}_k \| \leq n2^{n(4\beta+1)}$. Therefore, their bit-lengths are all at most $\mathcal{O}(n\beta)$.

V. Further Disscussion and Conclusion

We note that it is enough that $|\mu_{k,k+1}| \leq 1/2$ to guarantee the termination of BLLL. However, to control the bit size during the algorithm, we used $|\mu_{j,i}| \leq 1/2$ for $1 \leq j < i \leq k$. This leads that BLLL needs to do full size reduction to keep the bit size not too large in the loop, and hence the number of operations over $\mathbb{Z}$ for each while loop can only be bounded from above by $\mathcal{O}(n^3)$ rather than $\mathcal{O}(n^2)$ in the classical LLL algorithm. If fast arithmetic is employed, then BLLL returns a reduced basis within $\mathcal{O}(n^{6+\varepsilon}\beta^{2+\varepsilon})$ bit operations.

This result is not even as good as the classical LLL algorithm. However, we note that the techniques used in [8] (including fraction-free Gaussian elimination, modular arithmetic and fast matrix multiplication) apply to BLLL. This can make BLLL has the same bit-complexity as the lattice reduction algorithm in [8], namely $\mathcal{O}(n^{3.382}\beta^{2+\varepsilon})$.

Although the current bit complexity bound of BLLL is not competitive with the best LLL-type algorithms, we believe that there would be some approaches to take the advantages from the new exchange rule and hence resulting in an improving bit complexity bound. For instance, it should be very interesting to design a new size reduction strategy to decrease the number of arithmetic operations. Another intriguing topic is to consider how to use modular arithmetic or floating-point arithmetic to accelerate the algorithm further.

Acknowledgment

The present work was partially supported by NSFC (11501540, 11471307, 11671377), Chongqing Research Program of Basic Research and Frontier Technology (cstc2015jcyjys40001) and "Light of West China" Program of CAS, Key Programs of CAS (QYZDB-SSW-SYS026), and Research Program of Chongqing Municipal Education Commission (KJ1705121).

References

[1] D. Wubben, D. Seethaler, J. Jalden, and G. Matz. "Lattice reduction. " IEEE Signal Processing Magazine, vol. 28, no. 3, pp. 70-91, 2011, doi: 10.1109/MSP.2010.938758.

[2] P. Q. Nguyen and B. Vallée, Eds., The LLL Algorithm: Survey and Applications. Berlin: Springer, 2010, doi:10.1007/978-3-642-02295-1.

[3] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," Mathematische Annalen, vol. 261, no. 4, pp. 515-534, 1982, doi: 10.1007/BF01457454.

[4] J. Chen, D. Stehlé and G. Villard, "A new view on HJLS and PSLQ: Sums and projections of lattices," in Proceedings of ISSAC 2013 (June 26-29, 2013, Boston, MA, USA), M. Kauers, Ed. New York: ACM, 2013, pp. 149-156, doi:10.1145/2465506.2465936.

[5] G. M. Bergman, "Notes on Ferguson and Forcade's generalized euclidean algorithm", University of California at Berkeley, unpublished notes, 1980, available from: http://math.berkeley.edu/~gbergman/papers/unpub/FF_Euc.pdf.

[6] J. Håstad, B. Helfrich, J. C. Lagarias, and C.-P. Schnorr, "Polynomial time algorithms for finding integer relations among real numbers," in Proceedings of STACS '86, ser. Lecture Notes in Computer Science, B. Monien and G. Vidal-Naquet, Eds. Heidelberg: Springer, 1986, vol. 210, pp. 105-118, doi: 10.1007/3-540-16078-7_69.

[7] H. R. P. Ferguson and D. H. Bailey, "A polynomial time, numerically stable integer relation algorithm," NASA Ames Research Center, Tech. Rep. RNR-91-032, 1992, available at http://davidhbailey.com/dhbpapers/pslq.pdf.

[8] A. Storjohann, "Faster algorithms for integer lattice basis reduction," ETH, Department of Computer Scicence, Zürich, Switzerland, Tech. Rep. 249, July 1996, available at ftp://ftp.inf.ethz.ch/pub/publications/tech-reports/2xx/249.ps.gz.

[9] E. Kaltofen, "On the complexity of finding short vectors in integer lattices," in Computer Algebra: Proceedings of EUROCAL'83, ser. Lecture Notes in Computer Science, J. A. van Hulzen, Ed. Springer, 1983, vol. 162, pp. 236-244, doi: 10.1007/3-540-12868-9_107.

[10] C.-P. Schnorr, "A more efficient algorithm for lattice basis reduction," Journal of Algorithms, vol. 9, no. 1, pp. 47-62, 1988, doi: 10.1016/0196-6774(88)90004-1.

[11] C.-P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," Mathematical Programming, vol. 66, no. 1-3, pp. 181-199, 1994, doi: 10.1007/BF01581144.

[12] H. Koy and C.-P. Schnorr, "Segment LLL-reduction of lattice bases," in Cryptography and Lattices: Proceedings of CaLC 2001, ser. Lecture Notes in Computer Science, J. H. Silverman, Ed. Springer, 2001, vol. 2146, pp. 67-80, doi: 10.1007/3-540-44670-2_7.

[13] P. Q. Nguyen and D. Stehlé, "Floating-point LLL revisited," in Advances in Cryptology-EUROCRYPT 2005 (Aarhus, Denmark, May 22-26, 2005), ser. Lecture Notes in Computer Science, R. Cramer, Ed. Heidelberg: Springer, 2005, vol. 3494, pp. 215-233, doi: 10.1007/11426639_13.

[14] I. Morel, D. Stehlé, and G. Villard, "H-LLL: Using Householder inside LLL," in Proceedings of the 2009 international symposium on Symbolic and algebraic computation (July 29-31, 2009, Seoul, Republic of Korea), J. R. Johnson, H. Park, and E. Kaltofen, Eds. New York: ACM, 2009, pp. 271-278, doi: 10.1145/1576702.1576740.

[15] A. Novocin, D. Stehlé, and G. Villard, "An LLL-reduction algorithm with quasilinear time complexity: extended abstract," in Proceedings of the 43rd annual ACM symposium on Theory of computing (June 6-8, 2011, San Jose, USA), L. Fortnow and S. P. Vadhan, Eds. New York: ACM, 2011, pp. 403-412, doi: 10.1145/1993636.1993691.

[16] Saruchi, I. Morel, D. Stehlé, and G. Villard, "LLL reducing with the most significant bits," in Proceedings of the 2014 International Symposium on Symbolic and Algebraic Computation (July 23-25, 2014, Kobe, Japan), K. Nabeshima, K. Nagasaka, F. Winkler, and A. Szántó, Eds. New York: ACM, 2014, pp. 367-374, doi: 10.1145/2608628.2608645.

[17] F. Fontein, M. Schneider, and U. Wagner, "PotLLL: a polynomial time version of LLL with deep insertions," Designs, Codes and Cryptography, vol. 73, no. 2, pp. 355-368, 2014, doi: 10.1007/s10623-014-9918-8.

[18] A. Neumaier and D. Stehlé, "Faster LLL-type reduction of lattice bases," in Proceedings of ISSAC '16 (July 20-22, 2016, Waterloo, Ontario, Canada), S. A. Abramov, E. V. Zima, and X.-S. Gao, Eds. New York: ACM, 2016, pp. 373-380, doi: 10.1145/2930889.2930917.

[19] C.-P. Schnorr, "Geometry of numbers and integer programming," in Proceedings of STACS '88, ser. Lecture Notes in Computer Science, R. Cori and M. Wirsing, Eds. Springer, 1988, vol. 294, pp. 1-7, doi: 10.1007/BFb0035826.

[20] B. Just, "Effiziente Kettenbruchalgorithmen in beliebigen Dimensionen," Ph.D. dissertation, Universität Frankfurt, Frankfurt, 1987.