

In this lecture we describe an approximation algorithm to the Shortest Vector Problem (SVP). This algorithm, developed in 1982 by A. K. Lenstra, H. W. Lenstra, Jr. and L. Lovasz, usually called the **LLL** algorithm, gives a  $(\frac{2}{\sqrt{3}})^n$  approximation ratio, where  $n$  is the dimension of the lattice. In many of the applications, this algorithm is applied for a constant  $n$ ; in such cases, we obtain a constant approximation factor.

In 1801, Gauss gave an algorithm that can be viewed as an algorithm for solving SVP in two dimensions. The LLL algorithm is, in some way, a generalization of Gauss's algorithm to higher dimensions. In 1987, Schnorr presented an improved algorithm for the SVP. This improved algorithm obtains an approximation factor that is slightly subexponential, namely  $2^{O(n(\log \log n)^2 / \log n)}$ .

The LLL algorithm has many applications in diverse fields of computer science. Some of these will be described in the following lectures. Here is a brief description of some of these applications.

1. Factoring polynomials over the integers or the rational numbers. For example, given  $x^2 - 1$  factor it into  $x + 1$  and  $x - 1$ .
2. Finding the minimal polynomial of an algebraic number given to a good enough approximation. For example, given 1.414213 output  $x^2 - 2 = 0$  and given 0.645751 output  $x^2 + 4x - 3 = 0$ .
3. Finding integer relations. A set of real numbers  $\{x_1, \dots, x_n\}$  is said to possess an integer relation if there exist integers  $\{a_1, \dots, a_n\}$  such that  $a_1x_1 + \dots + a_nx_n = 0$ , with not all  $a_i = 0$ . As an example, try to find an integer relation among  $\arctan(1) \approx 0.785398$ ,  $\arctan(\frac{1}{5}) \approx 0.197395$ , and  $\arctan(\frac{1}{239}) \approx 0.004184$ . It turns that an integer relation exists:

$$\arctan(1) - 4 \arctan(1/5) + \arctan(1/239) = 0$$

(this equality is known as Machin's formula).

4. Integer Programming. This is a well-known **NP**-complete problem. Using LLL, one can obtain a polynomial time solution to integer programming with a fixed number of variables.
5. Approximation to the Closest Vector Problem (CVP), as well as other lattice problems.
6. Various applications in cryptanalysis (i.e., breaking cryptographic protocols). For example, there are many attacks on knapsack based cryptographic systems. Moreover, there are some more recent attacks on some special cases of RSA such as the low public exponent attack.

For simplicity, we describe the LLL algorithm for full-rank lattices; it is easy to remove this restriction. Moreover, our description only applies to the  $\ell_2$  norm. Extensions to other norms are known.

Let us now turn to describe LLL. The exposition is divided into three stages.

1. Define an LLL reduced basis.
2. Present an algorithm to find such a basis.
3. Analyze its running time.

## 1 Reduced basis

We first recall the Gram-Schmidt orthogonalization process.

**DEFINITION 1** Given  $n$  linearly independent vectors  $b_1, \dots, b_n \in \mathbb{R}^n$ , the **Gram-Schmidt orthogonalization** of  $b_1, \dots, b_n$  is defined by  $\tilde{b}_i = b_i - \sum_{j=1}^{i-1} \mu_{i,j} \tilde{b}_j$ , where  $\mu_{i,j} = \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}$ .

DEFINITION 2 A basis  $B = \{b_1, \dots, b_n\} \in \mathbb{R}^n$  is a  $\delta$ -LLL **Reduced Basis** if the following holds:

1.  $\forall 1 \leq i \leq n, j < i. |\mu_{i,j}| \leq \frac{1}{2}$ ,
2.  $\forall 1 \leq i < n. \delta \|\tilde{b}_i\|^2 \leq \|\mu_{i+1,i}\tilde{b}_i + \tilde{b}_{i+1}\|^2$ .

REMARK 1 It is always possible to transform a basis to a reduced basis. Actually, this is what the LLL algorithm does.

REMARK 2 It is helpful to consider the case  $\delta = \frac{3}{4}$ . The algorithm works with any  $\frac{1}{4} < \delta < 1$ .

REMARK 3 The second property in Definition 2 can be written as:

$$\delta \|\tilde{b}_i\|^2 \leq \|\mu_{i+1,i}\tilde{b}_i + \tilde{b}_{i+1}\|^2 = \mu_{i+1,i}^2 \|\tilde{b}_i\|^2 + \|\tilde{b}_{i+1}\|^2$$

where the second equality follows since  $\tilde{b}_i$  and  $\tilde{b}_{i+1}$  are orthogonal. It follows that

$$\|\tilde{b}_{i+1}\|^2 \geq (\delta - \mu_{i+1,i}^2) \|\tilde{b}_i\|^2 \geq (\delta - \frac{1}{4}) \|\tilde{b}_i\|^2$$

Put this way, the second property reads “ $\tilde{b}_{i+1}$  is not much shorter than  $\tilde{b}_i$ ”.

To better understand this definition, consider the orthonormal basis obtained by normalization the Gram-Schmidt vectors  $\tilde{b}_1, \dots, \tilde{b}_n$ . In this basis,  $B$  can be written as

$$\begin{pmatrix} \|\tilde{b}_1\| & * & \cdots & * \\ 0 & \|\tilde{b}_2\| & \cdots & * \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & \|\tilde{b}_n\| \end{pmatrix}$$

where column  $i$  shows the coordinates of  $b_i$  in this orthonormal basis. The first condition in the definition of an LLL-reduced basis guarantees that the absolute value of any off-diagonal element is at most half the value written in the diagonal element on the same row. This can be written as

$$\begin{pmatrix} \|\tilde{b}_1\| & \leq \frac{1}{2} \|\tilde{b}_1\| & \cdots & \leq \frac{1}{2} \|\tilde{b}_1\| \\ 0 & \|\tilde{b}_2\| & \cdots & \leq \frac{1}{2} \|\tilde{b}_2\| \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & \leq \frac{1}{2} \|\tilde{b}_{n-1}\| \\ & & & \|\tilde{b}_n\| \end{pmatrix}$$

where  $\leq \frac{1}{2} \|\tilde{b}_j\|$  indicates that the absolute value of this coordinate is at most  $\frac{1}{2} \|\tilde{b}_j\|$ . For the second property, consider the  $2 \times 2$  submatrix of the above matrix, with the upper left entry indexed at  $(i, i)$ .

$$\begin{pmatrix} \|\tilde{b}_i\| & \mu_{i+1,i} \|\tilde{b}_i\| \\ 0 & \|\tilde{b}_{i+1}\| \end{pmatrix}$$

Then the second property requires that the second column of this matrix is almost as long as its first column. Let us mention that in Schnorr’s improvement to the LLL algorithm, this second property is replaced with some condition on  $k \times k$  submatrices for some  $k > 2$ .

One important property of LLL-reduced basis is that its first vector is relatively short, as shown in the next claim.

CLAIM 1 Let  $b_1, \dots, b_n \in \mathbb{R}^n$  be a  $\delta$ -LLL-reduced basis. Then  $\|b_1\| \leq \left(\frac{2}{\sqrt{4\delta-1}}\right)^{n-1} \lambda_1(\mathcal{L})$ .

REMARK 4 For  $\delta = \frac{3}{4}$  this gives  $\|b_1\| \leq 2^{(n-1)/2} \lambda_1(\mathcal{L})$ .

PROOF: Since for any basis  $b_1, \dots, b_n$ ,  $\lambda_1(\mathcal{L}) \geq \min_i \|\tilde{b}_i\|$ , we get that

$$\|\tilde{b}_n\|^2 \geq \left(\delta - \frac{1}{4}\right) \|\tilde{b}_{n-1}\|^2 \geq \dots \geq \left(\delta - \frac{1}{4}\right)^{n-1} \|\tilde{b}_1\|^2 = \left(\delta - \frac{1}{4}\right)^{n-1} \|b_1\|^2$$

where the last equality follows by the definition  $\tilde{b}_1 = b_1$ . Then, for any  $i$ ,

$$\|b_1\| \leq \left(\delta - \frac{1}{4}\right)^{-(i-1)/2} \|\tilde{b}_i\| \leq \left(\delta - \frac{1}{4}\right)^{-(n-1)/2} \|\tilde{b}_i\|.$$

Hence,

$$\|b_1\| \leq \left(\delta - \frac{1}{4}\right)^{-(n-1)/2} \min_i \|\tilde{b}_i\| \leq \left(\delta - \frac{1}{4}\right)^{-(n-1)/2} \cdot \lambda_1(\mathcal{L})$$

□

REMARK 5 LLL-reduced bases have many other good properties; some are mentioned in the homework.

Claim 1 provides us with an approximation to the SVP problem. Assuming we can generate a  $\delta$ -LLL-reduced basis from our input basis, we can then return  $b_1$  as our answer. For  $\delta = 3/4$  we obtain a  $2^{(n-1)/2}$  approximation. In what follows, we describe how to transform an arbitrary basis into a  $\delta$ -LLL-reduced one.

## 2 The LLL Algorithm

INPUT: Lattice basis  $b_1, \dots, b_n \in \mathbb{Z}^n$

OUTPUT:  $\delta$ -LLL-reduced basis for  $\mathcal{L}(B)$

Start: compute  $\tilde{b}_1, \dots, \tilde{b}_n$

Reduction Step:

**for**  $i = 2$  to  $n$  **do**

**for**  $j = i - 1$  to  $1$  **do**

$$b_i \leftarrow b_i - c_{i,j} b_j \text{ where } c_{i,j} = \lceil \langle b_i, \tilde{b}_j \rangle / \langle \tilde{b}_j, \tilde{b}_j \rangle \rceil$$

Swap Step:

**if**  $\exists i$  s.t.  $\delta \|\tilde{b}_i\|^2 > \|\mu_{i+1,i} \tilde{b}_i + \tilde{b}_{i+1}\|^2$  **then**

$$b_i \leftrightarrow b_{i+1}$$

**goto** start

Output  $b_1, \dots, b_n$

REMARK 6 We use  $\lceil \cdot \rceil$  to denote rounding to the nearest integer, e.g.,  $\lceil 3.3 \rceil = 3$ ,  $\lceil 3.8 \rceil = 4$ .

Let us make some important observations on this procedure. It is easy to see that the swap step takes care of the second property of an LLL-reduced basis. Indeed, if the algorithm ever terminates, then its output must satisfy the second property. The reduction step takes care of the first property. In order to see this, first notice that throughout the reduction step, the Gram-Schmidt basis does not change (hence the vectors  $\tilde{b}_1, \dots, \tilde{b}_n$  need not be recomputed). This holds since we only perform column operations of the form

$b_i \leftarrow b_i + ab_j$  for  $i > j$  and  $a \in \mathbb{Z}$ . Such operations do not change the Gram-Schmidt orthogonalization. In the  $i$ th iteration of the outer loop, the reduction step makes sure that the projection of  $b_i$  on  $\tilde{b}_j$  for any  $j < i$  is at most  $\frac{1}{2}\|\tilde{b}_j\|$ . It does so by subtracting from column  $i$  the right integer multiple of column  $j$  such that the  $j$ th coordinate becomes at most  $\frac{1}{2}\|\tilde{b}_j\|$  in absolute value. Notice that it is crucial that the inner loop goes from  $i - 1$  down to 1.

To demonstrate the reduction step, let us write  $B$  in the orthonormal basis obtained by normalizing the Gram-Schmidt vectors. Consider, for example, the  $i$ th iteration of the outer loop and the  $j = 2$  iteration of the inner loop. Then at this point, the matrix  $B$  looks like

$$\begin{pmatrix} \|\tilde{b}_1\| & \leq \frac{1}{2}\|\tilde{b}_1\| & \leq \frac{1}{2}\|\tilde{b}_1\| & \cdots & * & * & \cdots \\ 0 & \|\tilde{b}_2\| & \leq \frac{1}{2}\|\tilde{b}_2\| & \cdots & * & * & \cdots \\ 0 & & \|\tilde{b}_3\| & \cdots & \leq \frac{1}{2}\|\tilde{b}_3\| & * & \cdots \\ \vdots & & \ddots & & & \vdots & \\ & & & & \leq \frac{1}{2}\|\tilde{b}_{i-1}\| & * & \\ 0 & \cdots & & & \|\tilde{b}_i\| & * & \cdots \\ & & & & 0 & \|\tilde{b}_{i+1}\| & \cdots \\ \vdots & & & & \vdots & & \ddots \end{pmatrix}$$

At this iteration, we subtract some integer multiple of the second column from column  $i$  to make the second entry in the  $i$ th column at most  $\frac{1}{2}\|\tilde{b}_2\|$  in absolute value. Similarly, in the last iteration of the inner loop, we subtract some integer multiple of the first column from column  $i$ .

**LEMMA 3 (CORRECTNESS)** *If the LLL procedure described above ever terminates, then its output is a  $\delta$ -LLL-reduced basis for the lattice spanned by the input basis  $b_1, \dots, b_n$ .*

**PROOF:** We need to prove that the output of the LLL algorithm is a basis for  $\mathcal{L}(B)$  that satisfies both properties of a  $\delta$ -LLL-reduced basis. The second property of a  $\delta$ -LLL-reduced basis is enforced by the check during the swap step. The reason that the output of the algorithm is indeed a basis for  $\mathcal{L}(B)$ , is that we only perform column operations of the form  $b_i \leftarrow b_i + ab_j$  for  $i \neq j$ , and  $a \in \mathbb{Z}$ .

We next show that after the reduction step,  $b_1, \dots, b_n$  satisfy  $|\mu_{i,j}| \leq \frac{1}{2}$ , for all  $i > j$ . First, notice that throughout the reduction step, the Gram-Schmidt basis does not change. Now, consider some  $i > j$ , and consider the  $j$ th iteration of the inner loop in the  $i$ th iteration of the outer loop. Then  $|\mu_{i,j}|$  can be written as

$$|\mu_{i,j}| = \left| \frac{\langle b_i - c_{i,j} \cdot b_j, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle} \right| = \left| \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle} - \left\lceil \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle} \right\rceil \cdot \frac{\langle b_j, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle} \right| \leq \frac{1}{2}$$

where the first equality follows from the definition of the reduction step and the last inequality follows from the fact that  $\langle b_j, \tilde{b}_j \rangle = \langle \tilde{b}_j, \tilde{b}_j \rangle$ .  $\square$

### 3 Analyzing the Running Time

Our analysis consists of two steps. First, we bound the number of iterations. Second, we bound the running time of a single iteration.

We show that the overall running time of the algorithm is polynomial in the input size. A rough lower bound on the latter is given by  $M \doteq \max\{n, \log(\max_i \|b_i\|)\}$  (because each of the  $n$  vectors requires at least one bit to represent and a vector of norm  $r$  requires at least  $r$  bits to represent). In the following, we show that the running time of the algorithm is polynomial in  $M$ .

LEMMA 4 *The number of iterations is polynomial in  $M$ .*

PROOF: Our first step is to define a function mapping a lattice basis to some positive number. This function can be thought of as a ‘potential function’.

DEFINITION 5 *Let  $B = \{b_1, \dots, b_n\}$  be a lattice basis. The potential of  $B$ , denoted  $\mathcal{D}_B$ , is defined by*

$$\prod_{i=1}^n \|\tilde{b}_i\|^{n-i+1} = \prod_{i=1}^n \|\tilde{b}_1\| \|\tilde{b}_2\| \dots \|\tilde{b}_i\| = \prod_{i=1}^n \mathcal{D}_{B,i}$$

where  $\mathcal{D}_{B,i} \doteq \det \Lambda_i$  and  $\Lambda_i$  is defined as the lattice spanned by  $b_1, \dots, b_i$

REMARK 7 Notice that in this definition, a higher weight is given to the first vectors.

Our aim is to show that the initial value of  $\mathcal{D}_B$  is not too large, and that it decays quickly. Since  $\|\tilde{b}_i\| \leq \|b_i\|$ , the initial value of  $\mathcal{D}_B$  can be bounded from above by  $(\max_i \|b_i\|)^{n(n-1)/2}$ . Note that the logarithm of this value is polynomial in  $M$ .

During the reduction step,  $\mathcal{D}_B$  does not change, because the Gram-Schmidt basis does not change. Now consider the swap step. Suppose that  $b_i$  is swapped with  $b_{i+1}$ . For all  $k \neq i$ ,  $\Lambda_k$  does not change, and so  $\mathcal{D}_{B,k}$  does not change; only  $\mathcal{D}_{B,i}$  changes. Let  $\Lambda'_i, \mathcal{D}'_{B,i}$  denote the new values of  $\Lambda_i$  and  $\mathcal{D}_{B,i}$ , respectively. We have that

$$\begin{aligned} \frac{\mathcal{D}'_{B,i}}{\mathcal{D}_{B,i}} &= \frac{\det \Lambda'_i}{\det \Lambda_i} \\ &= \frac{\det \mathcal{L}(b_1, \dots, b_{i-1}, b_{i+1})}{\det \mathcal{L}(b_1, \dots, b_i)} \\ &= \frac{(\prod_{j=1}^{i-1} \|\tilde{b}_j\|) \|\mu_{i+1,i} \tilde{b}_i + \tilde{b}_{i+1}\|}{\prod_{j=1}^i \|\tilde{b}_j\|} \\ &= \frac{\|\mu_{i+1,i} \tilde{b}_i + \tilde{b}_{i+1}\|}{\|\tilde{b}_i\|} < \sqrt{\delta} \end{aligned}$$

where the last inequality follows from the condition in the swap step.

As shown above, in each iteration,  $\mathcal{D}_B$  decreases by a multiplicative factor,  $\sqrt{\delta}$ . Let  $\mathcal{D}_{B,0}$  be the initial value of  $\mathcal{D}_B$ . Since  $\mathcal{D}_B$  is a nonzero integer, and in particular at least 1, this means that we can bound from above the number of iterations by

$$\log_{\frac{1}{\sqrt{\delta}}} \mathcal{D}_{B,0} = \frac{\log \mathcal{D}_{B,0}}{\log \frac{1}{\sqrt{\delta}}} \leq \frac{1}{\log \frac{1}{\sqrt{\delta}}} \cdot \frac{n(n-1)}{2} \log(\max_i \|b_i\|).$$

For any constant  $\delta < 1$ , this is polynomial in  $M$ .  $\square$

REMARK 8 A somewhat tedious calculation shows that even for  $\delta = \frac{1}{4} + (\frac{3}{4})^{\frac{n}{n-1}}$ , which is closer to 1 than any constant, the running time is polynomial. For such  $\delta$  the approximation factor is  $(\frac{2}{\sqrt{3}})^n$ . This approximation factor is essentially the best one can obtain with the LLL algorithm. For better approximation factors, one needs to apply Schnorr’s algorithm.

LEMMA 6 *The running time of each iteration is polynomial in  $M$ .*

PROOF: It is not difficult to see that in each iteration we perform only a polynomial number of arithmetic operations (i.e., additions, multiplications, etc.). Hence, in the rest of the proof, it is enough to show that the numbers that arise in each iteration can be represented using a polynomial number of bits.

To demonstrate why this is necessary, consider a repeated squaring algorithm that given a number  $x$ , squares it  $n$  times. Even though the number of arithmetic operations is only  $n$ , the number of bits required to represent the resulting numbers quickly grows to  $2^{O(n)}$ . Hence, the actual running time of the algorithm (measured in bit operations) is *exponential* in  $n$ .

We establish the bound on numbers arising during an iteration using two claims. The first concerns the Gram-Schmidt vectors  $\tilde{b}_1, \dots, \tilde{b}_n$ , which are somewhat simpler to bound, as they do not change during the reduction step. The second concerns the basis vectors  $b_1, \dots, b_n$ .

CLAIM 2 *The Gram-Schmidt vectors  $\tilde{b}_1, \dots, \tilde{b}_n$  can be computed in polynomial time in  $M$ . Moreover, for every  $1 \leq i \leq n$ , we have that  $\mathcal{D}_B^2 \tilde{b}_i \in \mathbb{Z}^n$  and that  $\|\tilde{b}_i\| \leq \mathcal{D}_B^2$ .*

REMARK 9 Notice that these two properties of the Gram-Schmidt vectors imply that they can be represented in space polynomial in  $M$ . Indeed, the bound on the norm implies that each coordinate of  $\tilde{b}_i$  contains a number of absolute value at most  $\mathcal{D}_B^2$ . Moreover, since  $\mathcal{D}_B^2 \tilde{b}_i \in \mathbb{Z}^n$  we know that the denominators cannot be larger than  $\mathcal{D}_B^2$ . Hence, each coordinate requires at most  $O(\log \mathcal{D}_B)$  bits to represent and there are  $n^2$  of them. Since the initial value of  $\log \mathcal{D}_B$  is polynomial in  $M$  and later on it can only decrease, we obtain that the Gram-Schmidt vectors can be represented in space polynomial in  $M$ .

PROOF: The calculation of the Gram-Schmidt basis may be performed as follows. Since  $\tilde{b}_i - b_i \in \text{span}(b_1, \dots, b_{i-1})$ , we can write  $\tilde{b}_i = b_i + \sum_{j=1}^{i-1} a_j b_j$ , for some  $a_1, \dots, a_{i-1} \in \mathbb{R}$ . We are looking for  $a_1, \dots, a_{i-1}$  such that  $\tilde{b}_i$  is orthogonal to each of  $b_1, \dots, b_{i-1}$ . For any  $1 \leq l \leq i-1$ ,  $\langle \tilde{b}_i, b_l \rangle = 0$  can be written as

$$\langle \tilde{b}_i, b_l \rangle = \langle b_i + \sum_{j=1}^{i-1} a_j b_j, b_l \rangle = \langle b_i, b_l \rangle + a_1 \langle b_1, b_l \rangle + a_2 \langle b_2, b_l \rangle + \dots + a_{i-1} \langle b_{i-1}, b_l \rangle = 0.$$

Hence, we obtain the following system of  $i-1$  linear equations in  $i-1$  variables:

$$\begin{aligned} a_1 \langle b_1, b_1 \rangle + a_2 \langle b_2, b_1 \rangle + \dots + a_{i-1} \langle b_{i-1}, b_1 \rangle &= -\langle b_i, b_1 \rangle \\ a_1 \langle b_1, b_2 \rangle + a_2 \langle b_2, b_2 \rangle + \dots + a_{i-1} \langle b_{i-1}, b_2 \rangle &= -\langle b_i, b_2 \rangle \\ &\vdots \\ a_1 \langle b_1, b_{i-1} \rangle + a_2 \langle b_2, b_{i-1} \rangle + \dots + a_{i-1} \langle b_{i-1}, b_{i-1} \rangle &= -\langle b_i, b_{i-1} \rangle. \end{aligned}$$

It is possible to solve such a system in polynomial time.

For the second part of the claim, notice that using Cramer's rule we can write

$$a_j = \frac{\det(\text{some integer matrix})}{\det \begin{pmatrix} \langle b_1, b_1 \rangle & \dots & \langle b_{i-1}, b_1 \rangle \\ \vdots & \ddots & \vdots \\ \langle b_1, b_{i-1} \rangle & \dots & \langle b_{i-1}, b_{i-1} \rangle \end{pmatrix}} = \frac{\text{some integer}}{\det B_{i-1}^T B_{i-1}} = \frac{\text{some integer}}{(\det \Lambda_{i-1})^2}.$$

Hence  $\tilde{b}_i = b_i + \sum_{j=1}^{i-1} a_j b_j$  for some rational numbers  $a_j$  whose denominator is  $(\det \Lambda_{i-1})^2$ . This implies that  $\mathcal{D}_B^2 \tilde{b}_i$  and in particular also  $\mathcal{D}_B^2 \tilde{b}_i$  are integer vectors.

Now we show that the norm of the  $\tilde{b}_i$ 's is not too large. By Definition 5,

$$\mathcal{D}_{B,i} = \left( \prod_{j=1}^{i-1} \|\tilde{b}_j\| \right) \cdot \|\tilde{b}_i\|$$

and so

$$\|\tilde{b}_i\| = \frac{\mathcal{D}_{B,i}}{\prod_{j=1}^{i-1} \|\tilde{b}_j\|} \leq \mathcal{D}_{B,i} \prod_{j=1}^{i-1} \mathcal{D}_{B,j}^2 \leq \mathcal{D}_B^2$$

where the first inequality follows since  $\|\tilde{b}_j\| \geq \frac{1}{\mathcal{D}_{B,j}^2}$ .  $\square$

In the next claim we show that the basis vectors  $b_i$  do not become too large. This is necessary since these basis vectors change during the reduction step (and in fact, it is possible for vectors to become longer by the reduction step). We first bound the length of each  $b_i$  after the  $i$ th iteration of the outer loop is done (i.e., once vector  $b_i$  is reduced). We then bound the length of  $b_i$  *during* the  $i$ th iteration of the outer loop. For this we use the observation that to vector  $b_i$  we only add vectors  $b_j$  for  $j < i$ ; these vectors are already reduced and hence our first bound applies.

**CLAIM 3** *All vectors  $b_i$  appearing during an iteration can be represented using  $\text{poly}(M)$  bits.*

**PROOF:** First, we show that after the reduction step, the length of the  $b_i$ 's is not too large. For each  $1 \leq i \leq n$ ,

$$\|b_i\|^2 = \|\tilde{b}_i\|^2 + \sum_{j=1}^{i-1} \mu_{i,j}^2 \|\tilde{b}_j\|^2 \leq \mathcal{D}_B^4 + \frac{n}{4} \cdot \mathcal{D}_B^4 \leq n\mathcal{D}_B^4$$

The first equality holds because  $\tilde{b}_1, \dots, \tilde{b}_n$  are orthogonal. The first inequality follows from the bound on  $\tilde{b}_1, \dots, \tilde{b}_n$  proven in Claim 2, and using the fact that  $|\mu_{i,j}| \leq \frac{1}{2}$ .

Our bound on the norm implies that each coordinate contains an integer of size at most  $\sqrt{n}\mathcal{D}_B^2$ . For an integer vector, this means that it can be represented in  $\log(\sqrt{n}\mathcal{D}_B^2)$  bits. Our  $b_i$ 's remain integer vectors throughout the procedure – they are such as inputs, and we change their values by adding integers. This means that after the reduction step, we can represent the  $b_i$ 's in  $\text{poly}(M)$  space.

Lastly, we need to show that *during* the reduction step, the  $b_i$ 's are not too large. Consider a vector  $b_i$ , that is manipulated in the inner loop of the reduction step.

$$|c_{i,j}| = \left| \left\lceil \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle} \right\rceil \right| \leq \frac{\|b_i\| \|\tilde{b}_j\|}{\|\tilde{b}_j\|^2} + 1 = \frac{\|b_i\|}{\|\tilde{b}_j\|} + 1 \leq \frac{\|b_i\|}{1/\mathcal{D}_B^2} + 1 \leq 2\mathcal{D}_B^2 \|b_i\|$$

where the first inequality follows by applying Cauchy-Schwartz and using the definition of the rounding operator, and the second inequality uses Claim 2. Therefore,

$$\begin{aligned} \|b_i - c_{i,j}b_j\| &\leq \|b_i\| + |c_{i,j}| \|b_j\| \\ &\leq (1 + 2\mathcal{D}_B^2 \|b_j\|) \|b_i\| \\ &\leq (1 + 2\mathcal{D}_B^2 \sqrt{n}\mathcal{D}_B^2) \|b_i\| \\ &\leq (4n\mathcal{D}_B)^4 \|b_i\| \end{aligned}$$

where the first inequality follows by the triangle inequality, the second inequality by plugging in the bound for  $|c_{i,j}|$ , and the third inequality by plugging in the bound on the length of  $\|b_j\|$  after the reduction step. Indeed, during the reduction step of  $b_i$ , vectors  $b_j$ , for  $j < i$ , have already finished their reduction step, so

we can use this bound. After at most  $n$  iterations of the inner loop, the norm of  $b_i$  has increased by a factor of at most  $(4n\mathcal{D}_B)^{4n}$ . This is of course representable in  $\text{poly}(M)$  size.  $\square$

By Claims 2 and 3 we have, that it is possible to represent the numbers in a polynomial number of bits. This, together with the fact that in each iteration we perform a polynomial number of arithmetic operations, proves the lemma.  $\square$

REMARK 10 The only place where we used that  $|\mu_{i,j}| \leq \frac{1}{2}$  for all  $j < i$  was in the proof of Claim 3. For the rest of the proof, the weaker condition that  $|\mu_{i+1,i}| \leq \frac{1}{2}$  for all  $i$  is enough. This suggests that we might improve the running time by performing the reduction step only on pairs of consecutive vectors so as to obtain the weaker condition. The number of iterations in this modified algorithm is still polynomial, since all of our arguments above hold. However, it is not clear if this modified algorithm still runs in polynomial time because Claim 3 does not seem to hold.

We combine Lemma 4 with Lemma 6 to conclude that the running time of the LLL algorithm is polynomial in the input size. This completes our analysis of LLL.