CSE 206A: Lattice Algorithms and Applications Fall 2017

Basis Reduction

Instructor: *Daniele Micciancio* UCSD CSE

No efficient algorithm is known to find the shortest vector in a lattice (in arbitrary dimension), or even just computing its length λ_1. A central tool in the algorithmic study of lattices (and their applications) is the LLL algorithm of Lenstra, Lenstra and Lovasz. The LLL algorithm runs in polynomial time and finds an approximate solution $\mathbf{x}$ to the shortest vector problem, in the sense that the length of the solution $\mathbf{x}$ found by the algorithm is at most $\gamma \cdot \lambda_1$, for some approximation factor γ. The approximation factor $\gamma = 2^{O(n)}$ achieved by LLL is exponential in the dimension of the lattice. Later in the course, we will study polynomial time algorithms that achieve (slightly) better factors. Still, the approximate solutions found by LLL are enough in many applications. We design and analyze the LLL algorithm in two steps:

1. We first define a notion of "reduced" basis, and show that the first vector of a reduced basis is an approximately shortest vector in the lattice.

2. Next, we give an efficient algorithm to compute a reduced basis for any lattice.

1 Reduced basis

Remember the Gram-Schmidt orthogonalization process:

$$\mathbf{b}_i^* \quad = \quad \mathbf{b}_i - \textstyle\sum_{j<i} \mu_{i,j} \mathbf{b}_j^* \qquad\qquad \text{where} \qquad \mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}$$

Define the orthogonal projection operations π_i from $\mathbb{R}^m$ onto $\sum_{j \geq i} \mathbb{R}\mathbf{b}_j^*$by

$$\pi_i(\mathbf{x}) = \sum_{j=i}^{n} \frac{\langle \mathbf{x}, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \mathbf{b}_j^*.$$

Notice that the Gram-Schmidt orthogonalized vectors can be expressed as $\mathbf{b}_i^* = \pi_i(\mathbf{b}_i)$.

We can now define LLL-reduced basis. For reasons that will be clear in the running time analysis of the algorithm, we introduce a real parameter $1/4 < \delta < 1$ and define LLL-reduced basis with respect to δ.

Definition 1 *A basis* $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n] \in \mathbb{R}^{m \times n}$ *is* δ*-LLL reduced if:*

- $|\mu_{i,j}| \leq \frac{1}{2}$ *for all* $i > j$

- *for any any pair of consecutive vectors* $\mathbf{b}_i, \mathbf{b}_{i+1}$*, we have*

$$\delta \|\pi_i(\mathbf{b}_i)\|^2 \leq \|\pi_i(\mathbf{b}_{i+1})\|^2.$$

The first condition (usually called "size reduction") is easy to achieve using an integer variant of the Gram-Schmidt orthogonalization procedure, and it is discussed in the next section. In order to understand the second condition it is useful to consider the case when $i = 1$ and $\delta = 1$. For $i = 1$, the projection π_1 is just the identity function (over the linear span of the lattice) and the condition becomes simply $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\|$, i.e., the first two vectors in an LLL reduced basis are sorted in order of nondecreasing length. (Introducing a factor $\delta < 1$ relaxes the nondecreasing condition to allow a small decrease δ in the norm.) For $i > 1$, the LLL reduced basis definition requires a similar condition to hold for the projected basis $\pi_i(\mathbf{B})$.

Another geometric interpretation of the second condition is the following. Notice that

$$\|\pi_i(\mathbf{b}_{i+1})\|^2 = \|\mathbf{b}^*_{i+1} + \mu_{i+1,i}\mathbf{b}^*_i\|^2 = \|\mathbf{b}^*_{i+1}\|^2 + \|\mu_{i+1,i}\mathbf{b}^*_i\|^2 = \|\mathbf{b}^*_{i+1}\|^2 + (\mu_{i+1,i})^2\|\mathbf{b}^*_i\|^2.$$

So, the second condition in the definition of LLL-reduced basis can be equivalently rewritten as

$$(\delta - \mu^2_{i+1,i})\|\mathbf{b}^*_i\|^2 \leq \|\mathbf{b}^*_{i+1}\|^2.$$

So, although the Gram-Schmidt vectors $\mathbf{b}^*_i$ can get shorter and shorter, their length cannot decrease too quickly. Specifically, for any $1/4 < \delta < 1$, if we set $\alpha = \frac{1}{\delta - \frac{1}{4}}$, then

$$\|\mathbf{b}^*_i\|^2 \leq \alpha\|\mathbf{b}^*_{i+1}\|^2. \tag{1}$$

For example, if $\delta = 3/4$, then $\alpha = 2$ and each $\|\mathbf{b}^*_{i+1}\|^2$ is least $\frac{1}{2}\|\mathbf{b}^*_i\|^2$. Using (1) repeatedly, we get

$$\|\mathbf{b}^*_1\|^2 \leq \alpha^{i-1}\|\mathbf{b}^*_i\|^2 \leq \alpha^{n-1}\|\mathbf{b}^*_i\|^2. \tag{2}$$

Since this is true for all $i = 1, \ldots, n$, the first vector in an LLL reduced basis satisfies

$$\|\mathbf{b}_1\| \leq \alpha^{(n-1)/2} \min \|\mathbf{b}^*_i\| \leq \alpha^{(n-1)/2}\lambda_1$$

where we have used the lower bound $\lambda_1 \geq \min_i \|\mathbf{b}^*_i\|$ on the length of the shortest vector in a lattice. In particular, if $\delta = 3/4$, the first vector in an LLL reduced basis is a $\gamma = 2^{(n-1)/2}$ approximate solution to SVP. Simiarly, one can also show that the set of vectors in an LLL reduced basis are a solution to the approximate SIVP.

Exercise 1 *Prove that if* $\mathbf{b}$ *is a* δ*-LLL reduced basis, then* $\max_i \|\mathbf{b}_i\| \leq \alpha^{(n-1)/2}\lambda_n$ *where* $\alpha = (1 - 1/\delta)^{-1}$.

In many applications, the length of the shortest lattice vector λ_1 is not known, but it can be estimated using Minkowski's theorem $\lambda_1 \leq \sqrt{n}\det(\mathbf{B})^{1/n}$. Combining the bound $\|\mathbf{b}_1\| \leq \alpha^{(n-1)/2}\lambda_1$ with Minkowski's theorem we get $\|\mathbf{b}_1\| \leq \sqrt{n}\alpha^{(n-1)/2}\det(\mathbf{B})^{1/n}$. A stronger bound can be obtained relating the length of $\mathbf{b}_1$ in an LLL reduced basis directly to the determinant of the lattice as follows. Take the product of (2) for $i = 1, \ldots, n$, to get

$$\|\mathbf{b}_1\|^n \leq \prod_i \alpha^{(i-1)/2}\|\mathbf{b}^*_i\| = \alpha^{n(n-1)/4}\det(\mathbf{B}).$$

So, we have

$$\|\mathbf{b}_1\| \leq \alpha^{(n-1)/4} \det(\mathbf{B}^{1/n}),$$

which can be interpreted as a weak (but algorithmic) version of Minkowski's theorem.

Remark 2 *Even if Minkowski's bound can be efficiently computed given a lattice basis, no efficient algorithm is known to find lattice vectors achieving the bound, even approximately for approximation factors that are significantly better than exponental in the dimension of the lattice.*

Our analysis of LLL reduced basis is summarized in the following theorem.

Theorem 3 *For any $1/4 < \delta \leq 1$, if* $\mathbf{b}$ *is a δ-LLL reduced basis, then*

- $\|\mathbf{b}_1\| \leq \alpha^{(n-1)/2}\lambda_1$
- $\max_i \|\mathbf{b}_i\| \leq \alpha^{(n-1)/2}\lambda_n$
- $\|\mathbf{b}_1\| \leq \alpha^{(n-1)/4} \det(\mathbf{B})^{1/n}$

where $\alpha = 1/(\delta - 1/4) \geq 4/3$.

In the following sections we give an algorithm to compute a δ-LLL reduced basis for any lattice in time polynomial in the input size and $(1-\delta)^{-1}$. It is not known whether the LLL algorithm runs in polynomial time when $\delta = 1$, which is the value that gives the best results. Still, we can achieve essentially the same result by setting $\delta = 1 - 1/n^c = 1 - o(1)$ and still maintain polynomial running time. This gives polynomial time solutions to SVP and SIVP for exponential approximation factors.

Corollary 4 *There is a polynomial time algorithm that solves SVP and SIVP within approximation factor $\gamma(n) = (2/\sqrt{3})^n$. The algorithm also produces nonzero lattice vectors of length at most $(2/\sqrt{3})^{n/2} \cdot \det(\mathcal{L}(\mathbf{B}))^{1/n}$.*

The LLL algorithm is designed to work in the Euclidean norm. Still, since all norms are within a factor n from the Euclidean norm, is also provides solutions to lattice problems in other norms within essentially the same approximation factor.

2 The Nearest Plane Algorithm

The size reduction condition ($|\mu_{i,j}| \leq 1/2$) in the definition of LLL reduced basis can be achieved using an integer variant of the Gram-Schmidt orthogonalization procedure. Both the size reduction condition and the associated algorithm have nice geometric interpretations which we are going to explain first.

It is easy to see that any point in the linear span of a lattice can be written as the sum of a lattice point $\mathbf{x} \in \mathcal{L}(\mathbf{B})$ plus a vector in the fundamental parallelepiped

$$\mathbf{y} \in \mathcal{P}(\mathbf{B}) = \{\mathbf{Bx} : 0 \leq \mathbf{x} < 1\}.$$

Moreover, such a decomposition is unique. In other words, the sets $\mathbf{x} + \mathcal{P}(\mathbf{B})$ (indexed by $\mathbf{x} \in \mathcal{L}(\mathbf{B})$) form a partition of span($\mathbf{b}$). A subset $S \subseteq$ span($\mathbf{b}$) such that $\{\mathbf{x} + S : \mathbf{x} \in \mathcal{L}(\mathbf{B})\}$ form a partition of span($\mathbf{b}$) is called a *fundamental region* for the lattice, and $\mathcal{P}(\mathbf{B})$ is an example of fundamental region. There are many other examples of interesting fundamental regions. For example, one can consider the *centered* half open parallelepiped

$$\mathcal{C}(\mathbf{B}) = \left\{\mathbf{Bx} : -\frac{1}{2} \leq \mathbf{x} < +\frac{1}{2}\right\}.$$

Another important fundamental region is the Voronoi cell of the lattice $\mathcal{V}(\mathbf{B})$, i.e., the set of all points that are closer to the origin than to any other lattice point.[1]

Notice that the partition associated to $\mathcal{P}(\mathbf{B})$ can be easily computed, in the sense that given a target point $\mathbf{t} \in$ span($\mathbf{b}$), one can efficiently find the lattice point $\mathbf{Bx}$ such that $\mathbf{t} \in \mathbf{Bx} + \mathcal{P}(\mathbf{B})$. (Just solve $\mathbf{By} = \mathbf{t}$ and round the solution to the lattice point $\mathbf{B}\lfloor\mathbf{y}\rfloor$.) The partition associated to the centered parallelepiped $\mathcal{C}(\mathbf{B})$ can also be computed similarly, rounding to the closest integers $\mathbf{B}\lfloor\mathbf{y}\rceil$. On the other hand, the partition associated to the Voronoi cell seems hard to compute: by definition, finding which Voronoi cell $\mathbf{Bx} + \mathcal{V}(\mathbf{B})$ contains a given target point $\mathbf{t}$ is equivalent to finding the lattice point $\mathbf{Bx}$ closest to $\mathbf{t}$, and instance of the CVP.

The size reduction condition in the definition of LLL reduced basis can be easily interpreted as partitioning the space according to still another fundamental region: the orthogonalized centered parallelepiped $\mathcal{C}(\mathbf{B}^*)$, where $\mathbf{B}^*$ is the Gram-Schmidt matrix of $\mathbf{b}$.

Exercise 2 *Prove that $\mathcal{C}(\mathbf{B}^*)$ is a fundamental region for lattice $\mathcal{L}(\mathbf{B})$.*

The cell $\mathbf{Bx} + \mathcal{C}(\mathbf{B}^*)$ containing a given target $\mathbf{t}$ can be easily found using the Nearest Plane algorithm, a simple variant of the Gram-Schmidt algorithm given as Algorithm 1.

Algorithm 1, on input a rank $n > 0$ lattice $\mathbf{b}$ and a target $\mathbf{t}$ proceeds as follows. Let $\mathbf{B}' = [\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}]$ be the sublattice generated by the first $n-1$ basis vectors. The lattice $\mathcal{L}(\mathbf{B})$ can be decomposed into hyperplanes of the form

$$\mathcal{L}(\mathbf{B}) = c\mathbf{b}_k + \mathcal{L}(\mathbf{B}') \subset c\mathbf{b}_k^* + \text{span}(\mathbf{B}').$$

The algorithm selects the hyperplane $c = \lfloor\langle\mathbf{t}, \mathbf{b}_k^*\rangle / \|\mathbf{b}_k^*\|^2\rceil$ closest to the target, and recursively search for a lattice point in $c\mathbf{b}_k + \mathcal{L}(\mathbf{B}')$ close to $\mathbf{t}$, or equivalently, a lattice point in the lower dimensional sublattice $\mathcal{L}(\mathbf{B}')$ close to $\mathbf{t} - c\mathbf{b}_k$. The base case of the algorithm is when the rank of the lattice is reduced to 0 and the only possible output is the origin $\mathbf{0}$.

[1] In order to get a partition, one needs also to include boundary points, with some care to avoid including the same point in multiple regions. For example, the norm relation $\|\mathbf{x}\| \leq \|\mathbf{y}\|$ can be extended to a total order by defining $\mathbf{x} < \mathbf{y}$ if and only if $\|\mathbf{x}\| < \|\mathbf{y}\|$ or $\|\mathbf{x}\| = \|\mathbf{y}\|$ and the first nonzero coordinate of $\mathbf{x} - \mathbf{y}$ is negative. Then, the *half-open* Voronoi cell can be defined as the sef of all points $\mathbf{x}$ such that $\mathbf{x} \leq (\mathbf{x} - \mathbf{y})$ for any lattice point $\mathbf{y} \in \mathcal{L}(\mathbf{B})$.

```
NearestPlane($\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ ,$\mathbf{t}$):
 if $n = 0$ then
   return $\mathbf{0}$
 else
   $\mathbf{b}^* \leftarrow$ GramSchmidt($\mathbf{B}$)
   $c \leftarrow \lfloor \langle \mathbf{t}, \mathbf{b}_n^* \rangle / \|\mathbf{b}_n^*\|^2 \rceil$
   return $c \cdot \mathbf{b}_n +$ NearestPlane($[\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}], \mathbf{t} - c \cdot \mathbf{b}_n$)
```

Figure 1: Nearest Plane Algorithm. On input a lattice basis $\mathbf{b}$ and a target vector $\mathbf{t}$, output a lattice point $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ such that $\langle \mathbf{t} - \mathbf{v}, \mathbf{b}_i^* \rangle / \|\mathbf{b}_i^*\|^2 \in [-1/2, 1/2)$ for all $i = 1, \ldots, n$.

Lemma 5 *On input a lattice basis and a target vector* $\mathbf{t}$*, Algorithm 1 outputs a lattice vector* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *such that* $\langle \mathbf{t} - \mathbf{v}, \mathbf{b}_i^* \rangle / \|\mathbf{b}_i^*\|^2 \in [-1/2, 1/2)$ *for all* $i = 1, \ldots, n$*. In particular, if* $\mathbf{t} \in \mathit{span}(\mathbf{b})$*, then* $\mathbf{t} \in \mathbf{v} + \mathcal{C}(\mathbf{B}^*)$*.*

Proof. For the base case, the property is vacuously true. So assume $n > 0$ and that the lemma holds for lower rank lattices. Let $\mathbf{B} = [\mathbf{C}|\mathbf{b}]$ where $\mathbf{C} = [\mathbf{b}_1, \ldots, \mathbf{b_{n-1}}]$, and notice that $\mathbf{b}^* = [\mathbf{C}^*|\mathbf{b}_n^*]$. By inductive hypothesis, the recursive call returns a lattice point $\mathbf{v} \in \mathcal{L}(\mathbf{C})$ such that $(\mathbf{t} - c\mathbf{b}_n) - \mathbf{v} = \mathbf{C}^*\mathbf{z}$ for some $\mathbf{z}$ such that $z_i \in [-1/2, +1/2]$. The output of the algorithm $\mathbf{v} + c\mathbf{b}_n$ satisfies

$$\langle \mathbf{t} - (\mathbf{v} + c\mathbf{b}_n), \mathbf{b}_i^* \rangle = \langle (\mathbf{t} - c\mathbf{b}_n) - \mathbf{v}, \mathbf{b}_i^* \rangle \in [-1/2, 1/2) \cdot \|\mathbf{b}_n^*\|^2$$

for all $i = 1, \ldots, n-1$ and

$$\langle \mathbf{t} - (\mathbf{v} + c\mathbf{b}_n), \mathbf{b}_n^* \rangle = \langle \mathbf{t}, \mathbf{b}_n^* \rangle - c\langle \mathbf{b}_n, \mathbf{b}_n^* \rangle \in [-1/2, 1/2) \cdot \|\mathbf{b}_n^*\|^2$$

where we have used the fact that $\langle \mathbf{b}_n, \mathbf{b}_n^* \rangle = \|\mathbf{b}_n^*\|^2$. □

Remark 6 *The fundamental region* $\mathcal{C}(\mathbf{B}^*)$ *contains a sphere centered in* $\mathbf{0}$ *of radius* $\min_i \|\mathbf{b}_i^*\|/2 \leq \lambda(\mathcal{L}(\mathbf{B}))/2$*. Since NearestPlane maps all points in* $\mathbf{v} + \mathcal{C}(\mathbf{B}^*)$ *to* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$*, if* $\mathbf{t}$ *is within distance* $\min_i \|\mathbf{b}_i^*\|/2$ *from the lattice, then NearestPlane(*$\mathbf{b}, \mathbf{t}$*) returns the lattice point* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *closest to* $\mathbf{t}$*.*

Observe that if $\mathbf{b}_i - \mathbf{b}_i^* \in \mathbf{Bx} + \mathcal{C}(\mathbf{B}^*)$, then $-1/2 \leq \mu_{i,j} < 1/2$ for all $j < i$. So, given a lattice basis $\mathbf{b}$, a size reduced basis for the same lattice can be easily obtained using Algorithm 2.

It is clear that the final $\mathbf{b}$ is a basis for the original lattice because we only executed elementary integer column operations in step 3. Moreover, it is easy to verify that after iteration i, the size reduction condition holds for all $j < i$. Finally, at iteration i, the Gram-Schmidt coefficients $\mu_{i',j}$ with $j < i' < i$ do not change. So, upon termination, the basis is size reduced.

```
SizeReduce($\mathbf{B}$):
for $i$ = 2 to $n$
  $\mathbf{x} \leftarrow$ NearestPlane($\mathbf{B}, \mathbf{b}_i - \mathbf{b}_i^*$)
  $\mathbf{b}_i \leftarrow \mathbf{b}_i - \mathbf{B}\mathbf{x}$
output $\mathbf{b}$
```

Figure 2: Size Reduce

```
LLL($\mathbf{B}$,$\delta$):
 SizeReduce($\mathbf{B}$)
 if $\delta\|\pi_i(\mathbf{b}_i)\|^2 > \|\pi_i(\mathbf{b}_{i+1})\|^2$ for some $i$ then
   swap($\mathbf{b}_i$,$\mathbf{b}_{i+1}$); return LLL($\mathbf{B}$,$\delta$)
 else
   return $\mathbf{B}$
```

Figure 3: The LLL basis reduction algorithm

3 The LLL algorithm

The LLL algorithm alternates two steps, aimed at achieving the two properties of an LLL reduced basis. Once we have size-reduced the input basis $\mathbf{b}$, there is only one way $\mathbf{b}$ can fail to be LLL reduced: violate the second condition, i.e., $\delta\|\pi_i(\mathbf{b}_i)\|^2 > \|\pi_i(\mathbf{b}_{i+1})\|^2$ for some index i. If this happens, the algorithm swaps $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$. Several pairs might violate the second property. Which one is selected for the swapping does not matter. In the original LLL algorithm i was chosen to be the smallest unordered pair, but any selection is equally good. In fact, one can even swap several disjoint pairs at the same time, leading to a parallel variant of the LLL algorithm. After the swap, the basis is not necessarily size reduced anymore. So, one must repeat the whole process from the reduction step. The LLL algorithm is summarized as Algorithm 3.

Clearly, upon termination the basis is LLL-reduced because it is size reduced and no pairs need to be swapped. So, if the algorithm terminates, then it is correct. We now prove that the algorithm terminates and it is actually polynomial time.

In order to show that the algorithm is polynomial time, we have to prove that the number of iterations is polynomial in the input size, and each iteration takes polynomial time. We first bound the number of iterations.

3.1 Bounding number of iterations

We now bound the number of iterations performed by the algorithm, i.e., we analyze the maximum number of swaps that can occur. This is accomplished by associating a positive integer to the basis $\mathbf{b}$, and showing that each time we swap two vectors this integer decreases by at least a constant factor.

Remember the definition of determinant

$$\det(\mathbf{B}) = \prod \|\mathbf{b}_i^*\| = \sqrt{\det(\mathbf{B}^\top \mathbf{B})}.$$

From the second formula it is clear that if $\mathbf{b}$ is an integer matrix, then the square of the determinant is an integer.

Lemma 7 *For every integer basis* $\mathbf{B} \in \mathbb{Z}^{m\times n}$*, we have* $\det(\mathcal{L}(\mathbf{B}))^2 \in \mathbb{Z}$

Therefore, we can associate to the basis $\mathbf{b}$ the following positive integer

$$\mathcal{D} = \prod_{k=1}^{n} \det(\mathcal{L}(\mathbf{b}_1, \cdots, \mathbf{b}_k))^2 \in \mathbb{Z}$$

We want to show that $\mathcal{D}$ decreases at least by a factor δ at each iteration. First we will show that Size Reduction does not change $\mathcal{D}$. Then we will show that each swap decreases $\mathcal{D}$ at least by δ.

To prove that Size Reduction doesn't change $\mathcal{D}$ we remember that SizeReduce does not affect the $\mathbf{b}_i^*$'s. Since $\mathcal{D}$ can be expressed as a function of the $\mathbf{b}_i^*$'s, the potential $\mathcal{D}$ is unchanged by the size reduction operation.

At this point we need to look at the effect that a swap has on $\mathcal{D}$. Let us look at the effect of a single swap say between $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$. Let $\mathcal{D}$ be the integer associated to the basis B before a swap, and $\mathcal{D}'$ the corresponding integer after the swap.

Notice that for all $j \neq i$ the lattice $\mathcal{L}(\mathbf{b}_1, \cdots, \mathbf{b}_j)$ is not changed by the swap. To prove this look at the two cases $j < i$ and $j > i$. When $j < i$ then there is no change in the basis $[\mathbf{b}_1, \cdots, \mathbf{b}_j]$, so the value of $\det(\mathcal{L}(\mathbf{b}_1, \cdots, \mathbf{b}_j))$ remains the same. On the other hand, if $j > i$ the only change is that two basis vectors in $[\mathbf{b}_1, \cdots, \mathbf{b}_j]$ are swapped, so the lattice $\mathcal{L}(\mathbf{b}_1, \cdots, \mathbf{b}_j)$ does not change and the determinant $\det(\mathcal{L}(\mathbf{b}_1, \cdots, \mathbf{b}_j))$ stays also the same.

So, the only factor in $\mathcal{D}$ that is affected by the swap is the one corresponding to lattice $\mathcal{L}(\mathbf{b}_1, \cdots, \mathbf{b}_i)$. Here we are replacing the last vector $\mathbf{b}_i$ by $\mathbf{b}_{i+1}$. Therefore

$$\frac{\mathcal{D}}{\mathcal{D}'} = \frac{\prod_{j\leq i} \|\mathbf{b}_j^*\|^2}{(\prod_{j<i} \|\mathbf{b}_j^*\|^2) \cdot \|\pi_i(\mathbf{b}_{i+1})\|^2} = \frac{\|\pi_i(\mathbf{b}_i)\|^2}{\|\pi_i(\mathbf{b}_{i+1})\|^2} \geq \frac{1}{\delta}$$

because swaps are performed only when $\|\pi_i(\mathbf{b}_{i+1})\|^2 < \delta \|\pi_i(\mathbf{b}_i)\|^2$.

This proves that

$$\mathcal{D}' \leq \delta \mathcal{D}$$

and by induction on n,

$$\mathcal{D}^{(n)} \leq \delta^n \mathcal{D}$$

where $\mathcal{D}$ is the value associated to the initial basis and $\mathcal{D}^{(n)}$ is the value after n iterations. Since $\mathcal{D}$ is a positive integer, $\mathcal{D} \geq 1$ and $(\frac{1}{\delta})^n \leq \mathcal{D}$ or equivalently

$$n \leq \log_{\frac{1}{\delta}} \mathcal{D} \tag{3}$$

This proves an upper bound on the number of iterations as a function of the initial value of $\mathcal{D}$. Since $\mathcal{D}$ is computable in polynomial time from the input basis, then its size must be polynomial in the input size. An estimate of how big $\mathcal{D}$ is can be easily obtained using Hadamard inequality.

3.2 Bounding the numbers

We proved that the number of iterations is bounded by a polynomial in the input size. In order to bound the running time of the algorithm we still need to show that each iteration also takes polynomial time. The number of arithmetic operations performed at each iteration is clearly polynomial. So, in order to prove a polynomial bound on the running time we only need to show that the size of the numbers involved in the entire computation also is polynomially bounded.

The LLL algorithm uses rational numbers, so we need to bound both the precision required by this number and their magnitude. In the analysis of the Gram-Schmidt algorithm we have already shown that the denominators in the $\mu_{i,j}$ coefficients must divide D_j. Notice that $\mathcal{D} = \prod D_j$ and therefore all entries in $\mu_{i,j}$ and $\mathbf{b}_i^*$ can be written as integers divided by $\mathcal{D}$. By definition, after size reduction, the $\mu_{i,j}$ are at most $1/2$ in absolute value, so their bit-size is bounded by $\log \mathcal{D}$. We now bound the length of the vectors. Using $D_i = \prod_{j=1}^{i} \|\mathbf{b}_i^*\|^2$, we get

$$\|\mathbf{b}_i^*\|^2 = \frac{D_i}{D_{i-1}} \leq D_i \leq \mathcal{D}.$$

Finally,

$$\|\mathbf{b}_i\|^2 = \|\mathbf{b}_i^*\|^2 + \sum_{j<i} \mu_{i,j}^2 \|\mathbf{b}_j^*\|^2 \leq \mathcal{D} + (n/4)\mathcal{D} \leq n\mathcal{D}.$$

So, all numerators and denominators of the numbers occurring in the execution of the algorithm have bit-size polynomial in $\log \mathcal{D}$.

4 A Simple Application in Number Theory

We give a simple application of LLL to algorithmic number theory: showing that we can *efficiently* write any prime $p \equiv 1 \pmod 4$ as the sum of two squares. Remember the proof that any such prime is the sum of two squares: all vectors $[a, b]^\top$ in the lattice

$$\mathbf{B} = \begin{bmatrix} 1 & 0 \\ i & p \end{bmatrix}$$

have the property that $a^2 + b^2$ is a multiple of p. So if we can find a nonzero lattice vector of squared norm less than $2p$, it must be $a^2 + b^2 = p$. Minwoski's theorem assures us that such short vectors exist. The question is: how can we find it? Answer: using LLL!

Run the LLL algorithm on the lattice basis, to obtain a reduced basis $\mathbf{b}_1, \mathbf{b}_2$ for the same lattice. We know, from Theorem 3 that $\|\mathbf{b}_1\| \leq \alpha^{1/4} \det(\mathbf{B})^{1/2}$. Squaring, and using $\det(\mathbf{B}) = p$, we get $\|\mathbf{b}_1\|^2 \leq \sqrt{\alpha} p < 2p$ as desired, provided $\delta > 3/4$.

5 The Closest Vector Problem

The LLL algorithm can be used also to solve the approximate CVP. No new algorithm is needed: we will show that preprocessing the lattice basis using the LLL algorithm and then

```
ApproximateCVP($\mathbf{b}$,$\mathbf{t}$) :
 $\mathbf{B} \leftarrow$ LLL($\mathbf{B}$)
 return NearestPlane($\mathbf{b},\mathbf{t}$)
```

Figure 4: Approximate CVP algorithm

applying the NearestPlane algorithm to the target vector produces approximate solutions to CVP within essentially the same approximation factor as the LLL algorithm for SVP. (See Algorithm 4.)

Theorem 8 *Algorithm 4 solves CVP within a factor* $\gamma(n) = (2/\sqrt{3})^n$.

Proof. The algorithm runs in polynomial time because it involves just two polynomial time computations. Proving that the algorithm is correct requires to look again into the details of LLL basis reduction and the Nearest Plane algorithm. We want to prove that if $\mathbf{b}$ is LLL reduced, then NearestPlane solves the approximate CVP. The proof is by induction on the rank of $\mathbf{b}$. The base case when the rank is $n = 0$ is trivial. So, assume $\mathbf{b}$ is an LLL reduced basis of rank $n > 0$. NearestPlane selects a hyperplane index $c = \lfloor \langle \mathbf{t}, \mathbf{b}_n^* \rangle / \|\mathbf{b}_n^*\|^2 \rceil$ and makes a recursive call on input the sublattice $\mathbf{C} = [\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}]$ and target $\mathbf{t} - c \cdot \mathbf{b}_n$. First of all notice that if $\mathbf{b}$ is LLL reduced, then $\mathbf{C}$ is also LLL reduced. So, we can invoke the inductive hypothesis on the recursive call. Let $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ be a lattice vector closest to $\mathbf{t}$. We distinguish two cases:

- If $\mathbf{v} \in c\mathbf{b}_n + \mathcal{L}(\mathbf{C})$, then the correctness of the final output follows by induction. Specifically, on input $\mathbf{C}$ and $\mathbf{t} - c\mathbf{b}_n$, the recursive call to NearestPlane returns a lattice point $\mathbf{w} \in \mathcal{L}(\mathbf{C})$ such that

  $$\|(\mathbf{t} - c\mathbf{b}_n) - \mathbf{w}\| \leq \gamma(n-1) \cdot \|(\mathbf{t} - c\mathbf{b}_n) - (\mathbf{v} - c\mathbf{b}_n)\| \leq \gamma(n) \cdot \|\mathbf{t} - \mathbf{v}\|.$$

  So, Algorithm 4 returns a lattice vector $c\mathbf{b}_n + \mathbf{w}$ at a distance from the target $\mathbf{t}$ which is within a factor $\gamma(n)$ from optimal.

- Otherwise, it must be $\mathbf{v} \in c'\mathbf{b}_n + \mathcal{L}(\mathbf{C})$ for some $c' \neq c$. Let $\mathbf{t} = \mathbf{t}' + \mathbf{t}''$, where $\mathbf{t}' \in \operatorname{span}(\mathbf{b})$ and $\mathbf{t}'' \perp \operatorname{span}(\mathbf{b})$. Then the distance between $\mathbf{t}$ and $\mathbf{v}$ is at least $\sqrt{\|\mathbf{t}''\|^2 + \|\mathbf{b}_n^*\|^2/4}$. On the other hand, NearestPlane returns a lattice vector $\mathbf{w}$ within distance $\sqrt{\|\mathbf{t}''\|^2 + \sum_i \|\mathbf{b}_i^*\|^2/4}$ from $\mathbf{t}$.Using the property (1) of LLL reduces basis we get

  $$\frac{\|\mathbf{t} - \mathbf{w}\|}{\|\mathbf{t} - \mathbf{v}\|} \leq \sqrt{\frac{\|\mathbf{t}''\|^2 + \sum_i \|\mathbf{b}_i^*\|^2/4}{\|\mathbf{t}''\|^2 + \|\mathbf{b}_n^*\|^2/4}} \leq \sqrt{\frac{\sum_i \|\mathbf{b}_i^*\|^2/4}{\|\mathbf{b}_n^*\|^2/4}} \leq \sqrt{\sum_i \alpha^{n-i}}.$$

  Setting $\gamma = \sqrt{\sum_i \alpha^{n-i}} = \sqrt{(\alpha^n - 1)/(\alpha - 1)} \leq \sqrt{3}(2/\sqrt{3})^n$ concludes the proof.

□