Introduction to Lattices

Zacharakis Alexandros

School of Electrical and Computer Engineering, NTUA

May 4, 2017

Contents

1 Lattices in $\mathbb{R}^m$

2 The LLL Algorithm

3 Babai's Nearest Plane Algorithm

4 Complexity Results

Why lattices?

- Algebraic Algorithms

Why lattices?

- Algebraic Algorithms
- Combinatorial Optimization

Why lattices?

- Algebraic Algorithms
- Combinatorial Optimization
- Complexity

Why lattices?

- Algebraic Algorithms
- Combinatorial Optimization
- Complexity
- Cryptanalysis

Why lattices?

- Algebraic Algorithms
- Combinatorial Optimization
- Complexity
- Cryptanalysis
- Cryptography
  - Very efficient
  - Worst case security guarantees
  - No known quantum attacks
  - Exotic constructions

Contents

1 Lattices in $\mathbb{R}^m$

2 The LLL Algorithm

3 Babai's Nearest Plane Algorithm

4 Complexity Results

Definition

Definition

Given n linearly independent vectors $b_1, \ldots, b_n \in \mathbb{R}^m$ the lattice generated by them is the set

$$\mathcal{L}(b_1, \ldots, b_n) = \{\sum_{i=1}^{n} x_i b_i \mid x_i \in \mathbb{Z}\}$$

Denoting $B = [b_1 \; b_2 \; \ldots \; b_n]$ equivalently we have

$$\mathcal{L}(B) = \{Bx \mid x \in \mathbb{Z}^n\}$$

Definition

Definition

Given n linearly independent vectors $b_1, \ldots, b_n \in \mathbb{R}^m$ the lattice generated by them is the set

$$\mathcal{L}(b_1, \ldots, b_n) = \{\sum_{i=1}^{n} x_i b_i \mid x_i \in \mathbb{Z}\}$$

Denoting $B = [b_1\ b_2\ \ldots\ b_n]$ equivalently we have

$$\mathcal{L}(B) = \{Bx \mid x \in \mathbb{Z}^n\}$$

- B is the basis of the lattice.
- m is its dimension.
- n is its rank.
- A lattice is full rank if $m = n$.
- The span of the lattice is the linear span of its basis.

- A lattice has many (infinite) equivalent bases.
- We next define the fundamental parallilepiped with respect to a basis B.

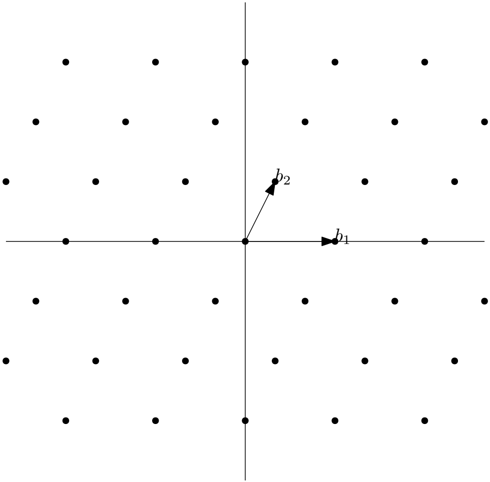
Definition

Fundamental Parallilepiped For a basis B the fundamental parallilepiped is the set

$$\mathcal{P}(B) = \{Bx \mid x \in [0, 1)\}$$

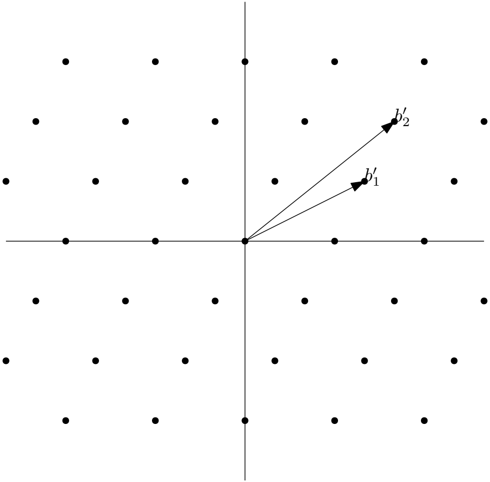
- Placing a copy of $\mathcal{P}(B)$ in every lattice point we partition $\text{span}(B)$.

A Lattice in $\mathbb{R}^2$



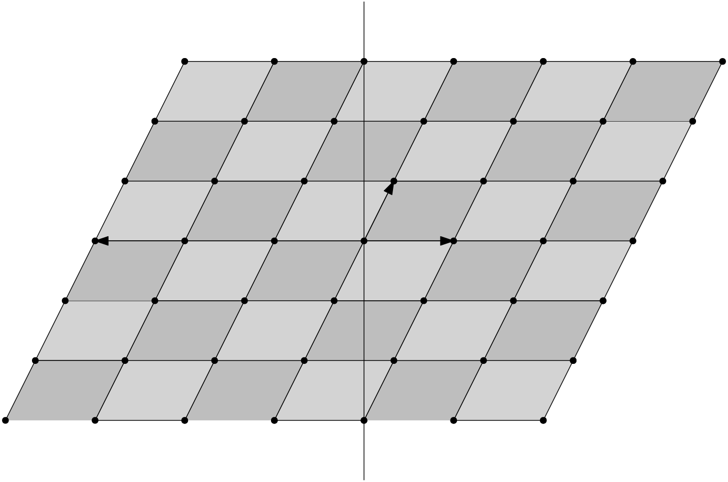
A lattice $\Lambda = \mathcal{L}(b_1, b_2) \subseteq \mathbb{R}^2$.

A different basis for the same lattice

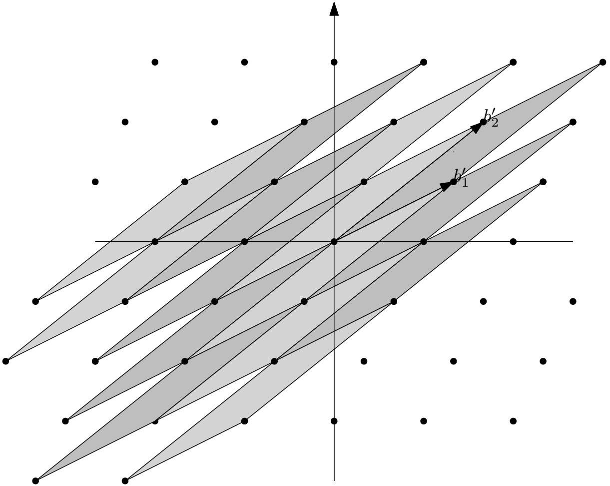


A lattice $\Lambda = \mathcal{L}(b_1, b_2) \subseteq \mathbb{R}^2$.

Fundamental Parallilepiped I



Fundamental Parallilepiped II



Characterizing a Basis

Lemma

Let Λ be a lattice and $b_1, \ldots, b_n \in \Lambda$ be n linearly independent vectors. Then $\Lambda = \mathcal{L}(b_1, \ldots, b_n)$ iff $\mathcal{P}(b_1, \ldots, b_n) \cap \Lambda = \{0\}$.

Characterizing a Basis

Lemma

Let Λ be a lattice and $b_1, \ldots, b_n \in \Lambda$ be n linearly independent vectors. Then $\Lambda = \mathcal{L}(b_1, \ldots, b_n)$ iff $\mathcal{P}(b_1, \ldots, b_n) \cap \Lambda = \{0\}$.

Proof.

Characterizing a Basis

Lemma

Let Λ be a lattice and $b_1, \ldots, b_n \in \Lambda$ be n linearly independent vectors. Then $\Lambda = \mathcal{L}(b_1, \ldots, b_n)$ iff $\mathcal{P}(b_1, \ldots, b_n) \cap \Lambda = \{0\}$.

Proof.

($\Rightarrow$) Suppose $x \in \mathcal{P}(b_1, \ldots, b_n) \cap \Lambda = \{0\}$. Then

$$x = Bz \text{ for } z \in \mathbb{Z}^n$$

$$x = By \text{ for } y \in [0, 1)^n$$

Since $b_1, \ldots, b_n$ are linearly independent we get $z = 0$.

Characterizing a Basis

Lemma

Let Λ be a lattice and $b_1, \ldots, b_n \in \Lambda$ be n linearly independent vectors. Then $\Lambda = \mathcal{L}(b_1, \ldots, b_n)$ iff $\mathcal{P}(b_1, \ldots, b_n) \cap \Lambda = \{0\}$.

Proof.

($\Rightarrow$) Suppose $x \in \mathcal{P}(b_1, \ldots, b_n) \cap \Lambda = \{0\}$. Then

$$x = Bz \text{ for } z \in \mathbb{Z}^n$$

$$x = By \text{ for } y \in [0, 1)^n$$

Since $b_1, \ldots, b_n$ are linearly independent we get $z = 0$.
($\Leftarrow$) Suppose $x \in \Lambda$. Then $x = By$ for $y \in \mathbb{R}^n$. Now $x' = B(y - \lfloor y \rfloor) \in \Lambda$. We get that $y = \lfloor y \rfloor$.

■

Equivalent Basis

Lemma

Suppose $B, D \in \mathbb{R}^{m \times n}$ rank n matrices. Then $\mathcal{L}(B) = \mathcal{L}(D)$ iff $D = BU$ for $U \in \mathbb{Z}^{n \times n}$ unimodular matrix.

Equivalent Basis

Lemma

Suppose $B, D \in \mathbb{R}^{m \times n}$ rank n matrices. Then $\mathcal{L}(B) = \mathcal{L}(D)$ iff $D = BU$ for $U \in \mathbb{Z}^{n \times n}$ unimodular matrix.

Proof.

Equivalent Basis

Lemma

Suppose $B, D \in \mathbb{R}^{m\times n}$ rank n matrices. Then $\mathcal{L}(B) = \mathcal{L}(D)$ iff $D = BU$ for $U \in \mathbb{Z}^{n\times n}$ unimodular matrix.

Proof.

($\Rightarrow$) If $\mathcal{L}(B) = \mathcal{L}(D)$ then each column $b_i \in \mathcal{L}(D)$ so $b_i = Du_i$. In matrix form we have $B = DU$. Similarly $D = BV$. Then we have $B^TB = U^TV^TB^TBVU$ so $\det(B^TB) = \det(U^TV^T)\det(B^TB)\det(VU)$ and so $\det(VU)^2 = 1$ and we get $\det(V)\det(U) = \pm 1$.

Equivalent Basis

Lemma

Suppose $B, D \in \mathbb{R}^{m\times n}$ rank n matrices. Then $\mathcal{L}(B) = \mathcal{L}(D)$ iff $D = BU$ for $U \in \mathbb{Z}^{n\times n}$ unimodular matrix.

Proof.

($\Rightarrow$) If $\mathcal{L}(B) = \mathcal{L}(D)$ then each column $b_i \in \mathcal{L}(D)$ so $b_i = Du_i$. In matrix form we have $B = DU$. Similarly $D = BV$. Then we have $B^TB = U^TV^TB^TBVU$ so $\det(B^TB) = \det(U^TV^T)\det(B^TB)\det(VU)$ and so $\det(VU)^2 = 1$ and we get $\det(V)\det(U) = \pm 1$.
($\Leftarrow$) Suppose $D = BU$. Then $D \subseteq \mathcal{L}(B)$. Also $B = DU^{-1}$ so $B \subseteq \mathcal{L}(D)$. We get $\mathcal{L}(B) = \mathcal{L}(D)$. ∎

Equivalent Basis

Another characterizations for equivalent basis is the following: B, D are equivalent basis iff we can constract D from B with the following operations

Equivalent Basis

Another characterizations for equivalent basis is the following: B, D are equivalent basis iff we can constract D from B with the following operations

1. $b_i \leftarrow b_i + kb_j$ for $k \in \mathbb{Z}$

Equivalent Basis

Another characterizations for equivalent basis is the following: B, D are equivalent basis iff we can constract D from B with the following operations

1. $b_i \leftarrow b_i + kb_j$ for $k \in \mathbb{Z}$
2. $b_i \leftrightarrow b_j$

Equivalent Basis

Another characterizations for equivalent basis is the following: B, D are equivalent basis iff we can constract D from B with the following operations

1. $b_i \leftarrow b_i + kb_j$ for $k \in \mathbb{Z}$
2. $b_i \leftrightarrow b_j$
3. $b_i \leftarrow -b_i$

Determinant of a Lattice

Definition

The determinant of a lattice Λ is the n-dimentional volume of a fundamental parralilepiped $\mathcal{P}(B)$, that is $\det(\Lambda) = \sqrt{\det(B^T B)}$.

Determinant of a Lattice

Definition

The determinant of a lattice Λ is the n-dimentional volume of a fundamental parralilepiped $\mathcal{P}(B)$, that is $\det(\Lambda) = \sqrt{\det(B^T B)}$.

- Note that the determinant is a lattice invariant (does not depend on the lattice basis).

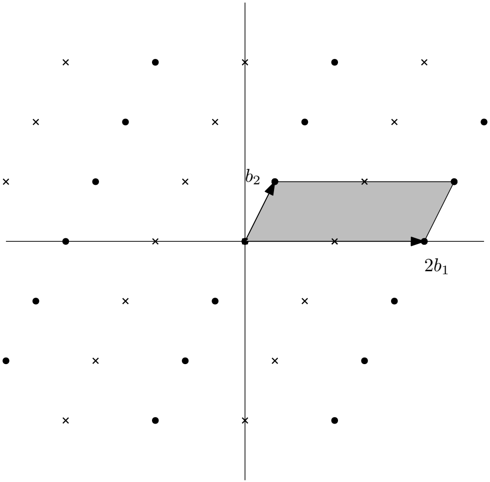
Determinant of a Lattice

Definition

The determinant of a lattice Λ is the n-dimentional volume of a fundamental parralilepiped $\mathcal{P}(B)$, that is $\det(\Lambda) = \sqrt{\det(B^T B)}$.

- Note that the determinant is a lattice invariant (does not depend on the lattice basis).
- It expresses the *density* of a lattice.

A Sublattice



Gram Schmidt Orthogonalization

GSO is an algorithm that takes as input n linear independent vectors and produces n orthogonal vectors.

Gram Schmidt Orthogonalization

GSO is an algorithm that takes as input n linear independent vectors and produces n orthogonal vectors.
It transforms b_i to

$$\tilde{b}_i = b_i - \sum_{j=1}^{i-1} \mu_{i,j} \tilde{b}_j \text{ where } \mu_{i,j} = \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}$$

Gram Schmidt Orthogonalization

GSO is an algorithm that takes as input n linear independent vectors and produces n orthogonal vectors.
It transforms b_i to

$$\tilde{b}_i = b_i - \sum_{j=1}^{i-1} \mu_{i,j} \tilde{b}_j \text{ where } \mu_{i,j} = \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}$$

- For all $i \neq j$ $\langle \tilde{b}_i, \tilde{b}_j \rangle = 0$.

Gram Schmidt Orthogonalization

GSO is an algorithm that takes as input n linear independent vectors and produces n orthogonal vectors.
It transforms b_i to

$$\tilde{b}_i = b_i - \sum_{j=1}^{i-1} \mu_{i,j} \tilde{b}_j \text{ where } \mu_{i,j} = \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}$$

- For all $i \neq j$ $\langle \tilde{b}_i, \tilde{b}_j \rangle = 0$.
- For all i $\text{span}(b_1, \ldots, b_i) = \text{span}(\tilde{b}_1, \ldots, \tilde{b}_i)$

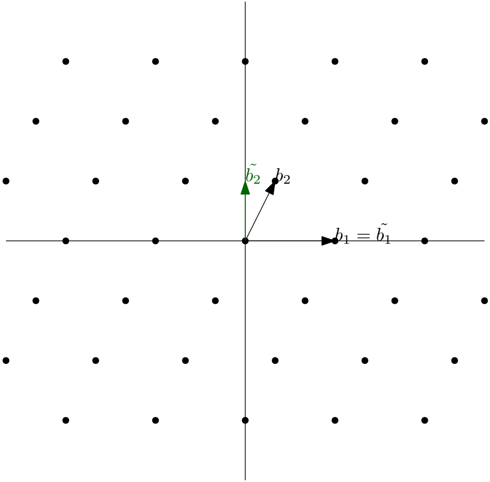
Gram Schmidt Orthogonalization

GSO is an algorithm that takes as input n linear independent vectors and produces n orthogonal vectors.
It transforms b_i to

$$\tilde{b}_i = b_i - \sum_{j=1}^{i-1} \mu_{i,j} \tilde{b}_j \text{ where } \mu_{i,j} = \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}$$

- For all $i \neq j$ $\langle \tilde{b}_i, \tilde{b}_j \rangle = 0$.
- For all i $\text{span}(b_1, \ldots, b_i) = \text{span}(\tilde{b}_1, \ldots, \tilde{b}_i)$
- The order of the input matters.

An example of GSO



Gram Schmidt Orthogonormal Basis

Let $b_1, \ldots, b_n \in \mathbb{R}^m$. If we normalize the GSO vectors we get an orthonormal basis

$$\frac{\tilde{b_1}}{\|\tilde{b_1}\|}, \ldots, \frac{\tilde{b_n}}{\|\tilde{b_n}\|}$$

Gram Schmidt Orthogonormal Basis

Let $b_1, \ldots, b_n \in \mathbb{R}^m$. If we normalize the GSO vectors we get an orthonormal basis

$$\frac{\tilde{b_1}}{||\tilde{b_1}||}, \ldots, \frac{\tilde{b_n}}{||\tilde{b_n}||}$$

In this basis we have

$$B = \begin{bmatrix} ||\tilde{b_1}|| & \mu_{2,1}||\tilde{b_1}|| & \ldots & \mu_{n,1}||\tilde{b_1}|| \\ 0 & ||\tilde{b_2}|| & \ldots & \mu_{n,2}||\tilde{b_2}|| \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & ||\tilde{b_n}|| \\ 0 & \ldots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \ldots & 0 & 0 \end{bmatrix}$$

Gram Schmidt Orthogonormal Basis

Let $b_1, \ldots, b_n \in \mathbb{R}^m$. If we normalize the GSO vectors we get an orthonormal basis

$$\frac{\tilde{b_1}}{||\tilde{b_1}||}, \ldots, \frac{\tilde{b_n}}{||\tilde{b_n}||}$$

In this basis we have

$$B = \begin{bmatrix} ||\tilde{b_1}|| & \mu_{2,1}||\tilde{b_1}|| & \ldots & \mu_{n,1}||\tilde{b_1}|| \\ 0 & ||\tilde{b_2}|| & \ldots & \mu_{n,2}||\tilde{b_2}|| \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & ||\tilde{b_n}|| \\ 0 & \ldots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \ldots & 0 & 0 \end{bmatrix}$$

From this we can easily get that $\det(\mathcal{L}(B)) = \prod_{i=1}^{n} ||\tilde{b_i}||$

Successive Minima

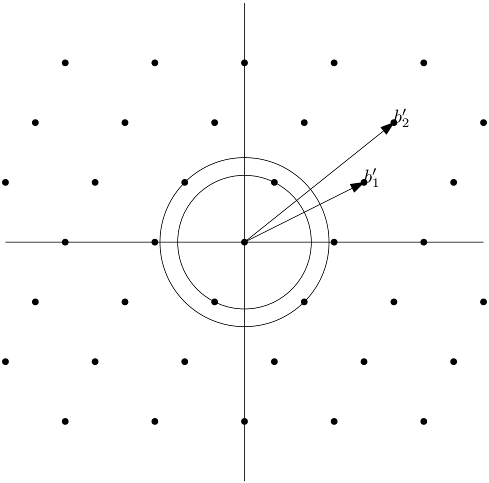
We define the i-th successive minima of Λ as the radius of the smallest ball that contains that contains i linearly independent lattice points. More formally

Definition

The i-th successive minima of Λ is

$$\lambda_i(\Lambda) = \inf\{r \mid \dim(\mathrm{span}(\Lambda \cap \overline{\mathbf{B}}(0, r))) > i\}$$

Successive Minima Example



Successive Minima

Theorem

Let $\mathcal{L}(B)$ be a lattie and $\tilde{B}$ its GSO. Then $\lambda_1(\mathcal{L}(B)) \geq \min_i \|\tilde{b}_i\| > 0$.

Successive Minima

Theorem

Let $\mathcal{L}(B)$ be a lattie and $\tilde{B}$ its GSO. Then $\lambda_1(\mathcal{L}(B)) \geq \min_i \|\tilde{b}_i\| > 0$.

Proof.

- Suppose $x \in \mathbb{Z}^n$ and $Bx \in \mathcal{L}(B)$.

Successive Minima

Theorem

Let $\mathcal{L}(B)$ be a lattie and $\tilde{B}$ its GSO. Then $\lambda_1(\mathcal{L}(B)) \geq \min_i \|\tilde{b}_i\| > 0$.

Proof.

- Suppose $x \in \mathbb{Z}^n$ and $Bx \in \mathcal{L}(B)$.
- Let $j \in [n]$ be the greatest such that $x_j \neq 0$.

Successive Minima

Theorem

Let $\mathcal{L}(B)$ be a lattie and $\tilde{B}$ its GSO. Then $\lambda_1(\mathcal{L}(B)) \geq \min_i \|\tilde{b}_i\| > 0$.

Proof.

- Suppose $x \in \mathbb{Z}^n$ and $Bx \in \mathcal{L}(B)$.
- Let $j \in [n]$ be the greatest such that $x_j \neq 0$.
- $|\langle Bx, \tilde{b}_j \rangle| = |\langle \sum_{i=1}^{j} b_i x_i, \tilde{b}_j \rangle| = |x_j| \|\tilde{b}_j\|^2$

Successive Minima

Theorem

Let $\mathcal{L}(B)$ be a lattie and $\tilde{B}$ its GSO. Then $\lambda_1(\mathcal{L}(B)) \geq \min_i \|\tilde{b}_i\| > 0$.

Proof.

- Suppose $x \in \mathbb{Z}^n$ and $Bx \in \mathcal{L}(B)$.
- Let $j \in [n]$ be the greatest such that $x_j \neq 0$.
- $|\langle Bx, \tilde{b}_j \rangle| = |\langle \sum_{i=1}^{j} b_i x_i, \tilde{b}_j \rangle| = |x_j| \|\tilde{b}_j\|^2$
- We also have $|\langle Bx, \tilde{b}_j \rangle| \leq \|Bx\| \cdot \|\tilde{b}_j\|$.

Successive Minima

Theorem

Let $\mathcal{L}(B)$ be a lattie and $\tilde{B}$ its GSO. Then $\lambda_1(\mathcal{L}(B)) \geq \min_i \|\tilde{b}_i\| > 0$.

Proof.

- Suppose $x \in \mathbb{Z}^n$ and $Bx \in \mathcal{L}(B)$.
- Let $j \in [n]$ be the greatest such that $x_j \neq 0$.
- $|\langle Bx, \tilde{b}_j \rangle| = |\langle \sum_{i=1}^{j} b_i x_i, \tilde{b}_j \rangle| = |x_j| \|\tilde{b}_j\|^2$
- We also have $|\langle Bx, \tilde{b}_j \rangle| \leq \|Bx\| \cdot \|\tilde{b}_j\|$.
- We get that $\|Bx\| \geq |x_j| \|\tilde{b}_j\| \geq \|\tilde{b}_j\| \geq \min_i \|\tilde{b}_i\|$

∎

Successive Minima

Theorem

Let $\mathcal{L}(B)$ be a lattie and $\tilde{B}$ its GSO. Then $\lambda_1(\mathcal{L}(B)) \geq \min_i \|\tilde{b}_i\| > 0$.

Proof.

- Suppose $x \in \mathbb{Z}^n$ and $Bx \in \mathcal{L}(B)$.
- Let $j \in [n]$ be the greatest such that $x_j \neq 0$.
- $|\langle Bx, \tilde{b}_j \rangle| = |\langle \sum_{i=1}^{j} b_i x_i, \tilde{b}_j \rangle| = |x_j| \|\tilde{b}_j\|^2$
- We also have $|\langle Bx, \tilde{b}_j \rangle| \leq \|Bx\| \cdot \|\tilde{b}_j\|$.
- We get that $\|Bx\| \geq |x_j| \|\tilde{b}_j\| \geq \|\tilde{b}_j\| \geq \min_i \|\tilde{b}_i\|$

■

- Lattices are discrete structures.

Blichfeld Theorem

Theorem

For any full rank lattice Λ *and any measurable* $S \subseteq \mathbb{R}^n$ *with* $\mathrm{vol}(S) > \det(\Lambda)$, *there exist* $z_1, z_2 \in S$ *such that* $z_1 - z_2 \in \Lambda$.

Blichfeld Theorem

Theorem

For any full rank lattice Λ and any measurable $S \subseteq \mathbb{R}^n$ with $\mathrm{vol}(S) > \det(\Lambda)$, there exist $z_1, z_2 \in S$ such that $z_1 - z_2 \in \Lambda$.

Proof.

Blichfeld Theorem

Theorem

For any full rank lattice Λ *and any measurable* $S \subseteq \mathbb{R}^n$ *with* $\mathrm{vol}(S) > \det(\Lambda)$, *there exist* $z_1, z_2 \in S$ *such that* $z_1 - z_2 \in \Lambda$.

Proof.

- Consider for each point $x \in \Lambda$ the set $x + \mathcal{P}(B) = \{x + y \mid y \in \mathcal{P}(B)\}$.

Blichfeld Theorem

Theorem

For any full rank lattice Λ *and any measurable* $S \subseteq \mathbb{R}^n$ *with* $\mathrm{vol}(S) > \det(\Lambda)$, *there exist* $z_1, z_2 \in S$ *such that* $z_1 - z_2 \in \Lambda$.

Proof.

- Consider for each point $x \in \Lambda$ the set $x + \mathcal{P}(B) = \{x + y \mid y \in \mathcal{P}(B)\}$.
- These sets partition $\mathbb{R}^n$.

Blichfeld Theorem

Theorem

For any full rank lattice Λ *and any measurable* $S \subseteq \mathbb{R}^n$ *with* $\mathrm{vol}(S) > \det(\Lambda)$, *there exist* $z_1, z_2 \in S$ *such that* $z_1 - z_2 \in \Lambda$.

Proof.

- Consider for each point $x \in \Lambda$ the set $x + \mathcal{P}(B) = \{x + y \mid y \in \mathcal{P}(B)\}$.
- These sets partition $\mathbb{R}^n$.
- We define $S_x = S \cap (x + \mathcal{P}(B))$.

Blichfeld Theorem

Theorem

For any full rank lattice Λ *and any measurable* $S \subseteq \mathbb{R}^n$ *with* $\mathrm{vol}(S) > \det(\Lambda)$, *there exist* $z_1, z_2 \in S$ *such that* $z_1 - z_2 \in \Lambda$.

Proof.

- Consider for each point $x \in \Lambda$ the set $x + \mathcal{P}(B) = \{x + y \mid y \in \mathcal{P}(B)\}$.
- These sets partition $\mathbb{R}^n$.
- We define $S_x = S \cap (x + \mathcal{P}(B))$.
- $S = \bigcup_{x \in \Lambda} S_x$ so $\mathrm{vol}(S) = \sum_{x \in \Lambda} \mathrm{vol}(S_x) > \mathrm{vol}(\mathcal{P}(B))$.

Blichfeld Theorem

Theorem

For any full rank lattice Λ *and any measurable* $S \subseteq \mathbb{R}^n$ *with* $\mathrm{vol}(S) > \det(\Lambda)$, *there exist* $z_1, z_2 \in S$ *such that* $z_1 - z_2 \in \Lambda$.

Proof.

- Consider for each point $x \in \Lambda$ the set $x + \mathcal{P}(B) = \{x + y \mid y \in \mathcal{P}(B)\}$.
- These sets partition $\mathbb{R}^n$.
- We define $S_x = S \cap (x + \mathcal{P}(B))$.
- $S = \bigcup_{x \in \Lambda} S_x$ so $\mathrm{vol}(S) = \sum_{x \in \Lambda} \mathrm{vol}(S_x) > \mathrm{vol}(\mathcal{P}(B))$.
- Define $\overline{S_x} = S_x - x$ (we move them to the origin).

Blichfeld Theorem

Theorem

For any full rank lattice Λ *and any measurable* $S \subseteq \mathbb{R}^n$ *with* $\mathrm{vol}(S) > \det(\Lambda)$, *there exist* $z_1, z_2 \in S$ *such that* $z_1 - z_2 \in \Lambda$.

Proof.

- Consider for each point $x \in \Lambda$ the set $x + \mathcal{P}(B) = \{x + y \mid y \in \mathcal{P}(B)\}$.
- These sets partition $\mathbb{R}^n$.
- We define $S_x = S \cap (x + \mathcal{P}(B))$.
- $S = \bigcup_{x \in \Lambda} S_x$ so $\mathrm{vol}(S) = \sum_{x \in \Lambda} \mathrm{vol}(S_x) > \mathrm{vol}(\mathcal{P}(B))$.
- Define $\overline{S_x} = S_x - x$ (we move them to the origin).
- There must be $y \neq x$ such that $\overline{S_x} \cap \overline{S_y} \neq \emptyset$. Suppose z is in both.

Blichfeld Theorem

Theorem

For any full rank lattice Λ *and any measurable* $S \subseteq \mathbb{R}^n$ *with* $\mathrm{vol}(S) > \det(\Lambda)$, *there exist* $z_1, z_2 \in S$ *such that* $z_1 - z_2 \in \Lambda$.

Proof.

- Consider for each point $x \in \Lambda$ the set $x + \mathcal{P}(B) = \{x + y \mid y \in \mathcal{P}(B)\}$.
- These sets partition $\mathbb{R}^n$.
- We define $S_x = S \cap (x + \mathcal{P}(B))$.
- $S = \bigcup_{x \in \Lambda} S_x$ so $\mathrm{vol}(S) = \sum_{x \in \Lambda} \mathrm{vol}(S_x) > \mathrm{vol}(\mathcal{P}(B))$.
- Define $\overline{S_x} = S_x - x$ (we move them to the origin).
- There must be $y \neq x$ such that $\overline{S_x} \cap \overline{S_y} \neq \emptyset$. Suppose z is in both.
- We have $(z + x), (z + y) \in S$.

Blichfeld Theorem

Theorem

For any full rank lattice Λ *and any measurable* $S \subseteq \mathbb{R}^n$ *with* $\mathrm{vol}(S) > \det(\Lambda)$, *there exist* $z_1, z_2 \in S$ *such that* $z_1 - z_2 \in \Lambda$.

Proof.

- Consider for each point $x \in \Lambda$ the set $x + \mathcal{P}(B) = \{x + y \mid y \in \mathcal{P}(B)\}$.
- These sets partition $\mathbb{R}^n$.
- We define $S_x = S \cap (x + \mathcal{P}(B))$.
- $S = \bigcup_{x \in \Lambda} S_x$ so $\mathrm{vol}(S) = \sum_{x \in \Lambda} \mathrm{vol}(S_x) > \mathrm{vol}(\mathcal{P}(B))$.
- Define $\overline{S_x} = S_x - x$ (we move them to the origin).
- There must be $y \neq x$ such that $\overline{S_x} \cap \overline{S_y} \neq \emptyset$. Suppose z is in both.
- We have $(z + x), (z + y) \in S$.
- $(z + x) - (z + y) = x - y \in \Lambda$.

∎

Minkowski's Convex Body Theorem

Theorem

For any full rank lattice Λ *and any centrally symmetric, convex set* $S \subseteq \mathbb{R}^n$ *if* $\text{vol}(S) > 2^n \det(\Lambda)$*, then* S *contains a non zero lattice point.*

Minkowski's Convex Body Theorem

Theorem

For any full rank lattice Λ *and any centrally symmetric, convex set* $S \subseteq \mathbb{R}^n$ *if* $\mathrm{vol}(S) > 2^n \det(\Lambda)$*, then* S *contains a non zero lattice point.*

Proof.

Minkowski's Convex Body Theorem

Theorem

For any full rank lattice Λ *and any centrally symmetric, convex set* $S \subseteq \mathbb{R}^n$ *if* $\mathrm{vol}(S) > 2^n \det(\Lambda)$, *then* S *contains a non zero lattice point.*

Proof.

- Define $S' = \frac{1}{2}S = \{x \mid 2x \in S\}$.

Minkowski's Convex Body Theorem

Theorem

For any full rank lattice Λ *and any centrally symmetric, convex set* $S \subseteq \mathbb{R}^n$ *if* $\mathrm{vol}(S) > 2^n \det(\Lambda)$, *then* S *contains a non zero lattice point.*

Proof.

- Define $S' = \frac{1}{2}S = \{x \mid 2x \in S\}$.
- We have $\mathrm{vol}(S') = 2^{-n}\mathrm{vol}(S) > \det(\Lambda)$.

Minkowski's Convex Body Theorem

Theorem

For any full rank lattice Λ and any centrally symmetric, convex set $S \subseteq \mathbb{R}^n$ if $\text{vol}(S) > 2^n \det(\Lambda)$, then S contains a non zero lattice point.

Proof.

- Define $S' = \frac{1}{2}S = \{x \mid 2x \in S\}$.
- We have $\text{vol}(S') = 2^{-n}\text{vol}(S) > \det(\Lambda)$.
- By the previous theorem, there exist $z_1, z_2 \in S'$ such that $z_1 - z_2 \in \Lambda$.

Minkowski's Convex Body Theorem

Theorem

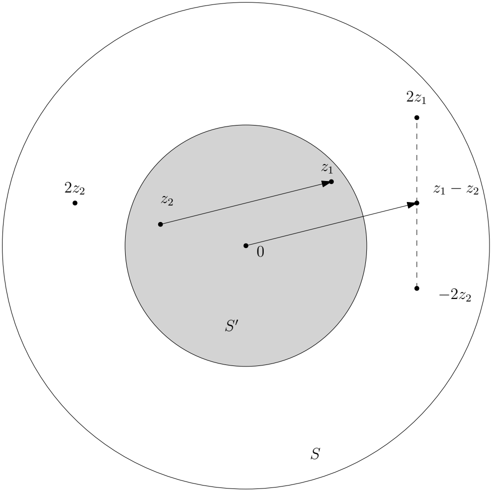
For any full rank lattice Λ *and any centrally symmetric, convex set* $S \subseteq \mathbb{R}^n$ *if* $\text{vol}(S) > 2^n \det(\Lambda)$*, then* S *contains a non zero lattice point.*

Proof.

- Define $S' = \frac{1}{2}S = \{x \mid 2x \in S\}$.
- We have $\text{vol}(S') = 2^{-n}\text{vol}(S) > \det(\Lambda)$.
- By the previous theorem, there exist $z_1, z_2 \in S'$ such that $z_1 - z_2 \in \Lambda$.
- We have $2z_1, -2z_2 \in S$ and we get $z_1 - z_2 \in S$

■

Minkowski's Convex Body Theorem



Bounding Successive Minima

By selecting appropriate sets (ball and ellipsoid respectively) we can deduce the following upper bounds for the succesive minima using the previous theorem.

$$\lambda_1(\Lambda) \leq \sqrt{n} \cdot \det(\Lambda)^{\frac{1}{n}}$$

$$(\prod_{i=1}^{n} \lambda_i(\Lambda))^{\frac{1}{n}} \leq \sqrt{n} \cdot \det(\Lambda)^{\frac{1}{n}}$$

Computational Problems

Algebraic lattice points are easy:

Computational Problems

Algebraic lattice points are easy:

- **Membership**: Given a matrix B and a point x decide wheather $x \in \mathcal{L}(B)$.

Computational Problems

Algebraic lattice points are easy:

- **Membership**: Given a matrix B and a point x decide wheather $x \in \mathcal{L}(B)$.
- **Equivalence**: Given matrices B, D decide wheather $\mathcal{L}(B) = \mathcal{L}(D)$.

Computational Problems

Algebraic lattice points are easy:

- **Membership**: Given a matrix B and a point x decide wheather $x \in \mathcal{L}(B)$.
- **Equivalence**: Given matrices B, D decide wheather $\mathcal{L}(B) = \mathcal{L}(D)$.

Things get harder when geometry comes to play.

Shortest Vector Problem

- **SearchSVP$_\gamma$**: Given $B \in \mathbb{Z}^{m \times n}$ find $v \in \mathcal{L}(B)$ such that $v \neq 0$ and $\|v\| \leq \gamma \cdot \lambda_1(\mathcal{L}(B))$.

Shortest Vector Problem

- **SearchSVP**$_\gamma$: Given $B \in \mathbb{Z}^{m \times n}$ find $v \in \mathcal{L}(B)$ such that $v \neq 0$ and $\|v\| \leq \gamma \cdot \lambda_1(\mathcal{L}(B))$.
- **OptimizationSVP**$_\gamma$: Given $B \in \mathbb{Z}^{m \times n}$ find d such that $d \leq \lambda_1(\mathcal{L}(B)) \leq \gamma \cdot d$.

Shortest Vector Problem

- **SearchSVP$_\gamma$**: Given $B \in \mathbb{Z}^{m \times n}$ find $v \in \mathcal{L}(B)$ such that $v \neq 0$ and $\|v\| \leq \gamma \cdot \lambda_1(\mathcal{L}(B))$.
- **OptimizationSVP$_\gamma$**: Given $B \in \mathbb{Z}^{m \times n}$ find d such that $d \leq \lambda_1(\mathcal{L}(B)) \leq \gamma \cdot d$.
- **PromiseSVP$_\gamma$**: Given (B, r) with $B \in \mathbb{Z}^{m \times n}$ and $r \in \mathbb{Q}$ decide whether $\lambda_1(\mathcal{L}(B)) \leq r$ or $\lambda_1(\mathcal{L}(B)) > \gamma \cdot r$ given the promise that one of these is true

Shortest Vector Problem

- **SearchSVP$_\gamma$**: Given $B \in \mathbb{Z}^{m \times n}$ find $v \in \mathcal{L}(B)$ such that $v \neq 0$ and $\|v\| \leq \gamma \cdot \lambda_1(\mathcal{L}(B))$.
- **OptimizationSVP$_\gamma$**: Given $B \in \mathbb{Z}^{m \times n}$ find d such that $d \leq \lambda_1(\mathcal{L}(B)) \leq \gamma \cdot d$.
- **PromiseSVP$_\gamma$**: Given (B, r) with $B \in \mathbb{Z}^{m \times n}$ and $r \in \mathbb{Q}$ decide whether $\lambda_1(\mathcal{L}(B)) \leq r$ or $\lambda_1(\mathcal{L}(B)) > \gamma \cdot r$ given the promise that one of these is true

Shortest Vector Problem

- **SearchSVP**$_\gamma$: Given $B \in \mathbb{Z}^{m \times n}$ find $v \in \mathcal{L}(B)$ such that $v \neq 0$ and $\|v\| \leq \gamma \cdot \lambda_1(\mathcal{L}(B))$.
- **OptimizationSVP**$_\gamma$: Given $B \in \mathbb{Z}^{m \times n}$ find d such that $d \leq \lambda_1(\mathcal{L}(B)) \leq \gamma \cdot d$.
- **PromiseSVP**$_\gamma$: Given (B, r) with $B \in \mathbb{Z}^{m \times n}$ and $r \in \mathbb{Q}$ decide whether $\lambda_1(\mathcal{L}(B)) \leq r$ or $\lambda_1(\mathcal{L}(B)) > \gamma \cdot r$ given the promise that one of these is true

For $\gamma = 1$ we get the exact versions of these problems. These are computationally equivalent.

Shortest Vector Problem

- **SearchSVP**$_\gamma$: Given $B \in \mathbb{Z}^{m \times n}$ find $v \in \mathcal{L}(B)$ such that $v \neq 0$ and $\|v\| \leq \gamma \cdot \lambda_1(\mathcal{L}(B))$.
- **OptimizationSVP**$_\gamma$: Given $B \in \mathbb{Z}^{m \times n}$ find d such that $d \leq \lambda_1(\mathcal{L}(B)) \leq \gamma \cdot d$.
- **PromiseSVP**$_\gamma$: Given (B, r) with $B \in \mathbb{Z}^{m \times n}$ and $r \in \mathbb{Q}$ decide whether $\lambda_1(\mathcal{L}(B)) \leq r$ or $\lambda_1(\mathcal{L}(B)) > \gamma \cdot r$ given the promise that one of these is true

For $\gamma = 1$ we get the exact versions of these problems. These are computationally equivalent.
For the general case it is an open problem if this holds.

Closest Vector Problem

- **SearchCVP**$_\gamma$: Given $B \in \mathbb{Z}^{m \times n}$ and $t \in \mathbb{Z}^m$ find $v \in \mathcal{L}(B)$ such that $\|v - t\| \leq \gamma \cdot \text{dist}(t, \mathcal{L}(B))$.

Closest Vector Problem

- **SearchCVP$_\gamma$**: Given $B \in \mathbb{Z}^{m \times n}$ and $t \in \mathbb{Z}^m$ find $v \in \mathcal{L}(B)$ such that $\|v - t\| \leq \gamma \cdot \mathrm{dist}(t, \mathcal{L}(B))$.
- **OptimizationCVP$_\gamma$**: Given $B \in \mathbb{Z}^{m \times n}$ and $t \in \mathbb{Z}^m$ find d such that $d \leq \mathrm{dist}(t, \mathcal{L}(B)) \leq \gamma \cdot d$.

Closest Vector Problem

- **SearchCVP**$_\gamma$: Given $B \in \mathbb{Z}^{m \times n}$ and $t \in \mathbb{Z}^m$ find $v \in \mathcal{L}(B)$ such that $\|v - t\| \leq \gamma \cdot \text{dist}(t, \mathcal{L}(B))$.
- **OptimizationCVP**$_\gamma$: Given $B \in \mathbb{Z}^{m \times n}$ and $t \in \mathbb{Z}^m$ find d such that $d \leq \text{dist}(t, \mathcal{L}(B)) \leq \gamma \cdot d$.
- **PromiseCVP**$_\gamma$: Given (B, t, r) with $B \in \mathbb{Z}^{m \times n}$, $t \in \mathbb{Z}^m$ and $r \in \mathbb{Q}$ decide whether $\text{dist}(t, \mathcal{L}(B)) \leq r$ or $\text{dist}(t, \mathcal{L}(B)) > \gamma \cdot r$ given the promise that one of these is true

Closest Vector Problem

- **SearchCVP**$_\gamma$: Given $B \in \mathbb{Z}^{m \times n}$ and $t \in \mathbb{Z}^m$ find $v \in \mathcal{L}(B)$ such that $\|v - t\| \leq \gamma \cdot \text{dist}(t, \mathcal{L}(B))$.
- **OptimizationCVP**$_\gamma$: Given $B \in \mathbb{Z}^{m \times n}$ and $t \in \mathbb{Z}^m$ find d such that $d \leq \text{dist}(t, \mathcal{L}(B)) \leq \gamma \cdot d$.
- **PromiseCVP**$_\gamma$: Given (B, t, r) with $B \in \mathbb{Z}^{m \times n}$, $t \in \mathbb{Z}^m$ and $r \in \mathbb{Q}$ decide whether $\text{dist}(t, \mathcal{L}(B)) \leq r$ or $\text{dist}(t, \mathcal{L}(B)) > \gamma \cdot r$ given the promise that one of these is true

Contents

1 Lattices in $\mathbb{R}^m$

2 The LLL Algorithm

3 Babai's Nearest Plane Algorithm

4 Complexity Results

The LLL Algorithm

- Developed by A. Lenstra, H. Lenstra, L. Lovasz in 1982.

The LLL Algorithm

- Developed by A. Lenstra, H. Lenstra, L. Lovasz in 1982.
- Solves **SVP** for $\gamma = \frac{2}{\sqrt{3}}^{n}$.

The LLL Algorithm

- Developed by A. Lenstra, H. Lenstra, L. Lovasz in 1982.
- Solves **SVP** for $\gamma = \frac{2}{\sqrt{3}}^n$.
- Exponential approximation ratio in the dimention of the lattice.

The LLL Algorithm

- Developed by A. Lenstra, H. Lenstra, L. Lovasz in 1982.
- Solves **SVP** for $\gamma = \frac{2}{\sqrt{3}}^{n}$.
- Exponential approximation ratio in the dimention of the lattice.
- Improved by Schnorr for $\gamma = 2^{\mathcal{O}(n(\log\log n)^2 / \log n)}$.

The LLL Algorithm

- Developed by A. Lenstra, H. Lenstra, L. Lovasz in 1982.
- Solves **SVP** for $\gamma = \frac{2}{\sqrt{3}}^{n}$.
- Exponential approximation ratio in the dimention of the lattice.
- Improved by Schnorr for $\gamma = 2^{\mathcal{O}(n(\log\log n)^2/\log n)}$.
- It is also used for approximating **CVP**.

The LLL Algorithm

- Developed by A. Lenstra, H. Lenstra, L. Lovasz in 1982.
- Solves **SVP** for $\gamma = \frac{2}{\sqrt{3}}^n$.
- Exponential approximation ratio in the dimention of the lattice.
- Improved by Schnorr for $\gamma = 2^{\mathcal{O}(n(\log\log n)^2/\log n)}$.
- It is also used for approximating **CVP**.
- Used for many problems, namely many algebraic problems, combinatorial optimization, cryptanalisis.

$\delta-$LLL Reduced Basis

Definition

A basis $B = [b_1 \ b_2 \ \ldots \ b_n]$ is a $\delta-$LLL reduced basis if

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

$\delta-$LLL Reduced Basis

Definition

A basis $B = [b_1 \ b_2 \ \dots \ b_n]$ is a $\delta-$LLL reduced basis if

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

We must have $\frac{1}{4} < \delta < 1$. Consider $\delta = \frac{3}{4}$.

$\delta-$LLL Reduced Basis

Definition

A basis $B = [b_1 \; b_2 \; \dots \; b_n]$ is a $\delta-$LLL reduced basis if

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

We must have $\frac{1}{4} < \delta < 1$. Consider $\delta = \frac{3}{4}$.
The second condition can be written as

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

That is $\tilde{b}_{i+1}$ is not much shorter than $\tilde{b}_i$

$\delta-$LLL Reduced Basis

Definition

A basis $B = [b_1 \ b_2 \ \ldots \ b_n]$ is a $\delta-$LLL reduced basis if

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

We must have $\frac{1}{4} < \delta < 1$. Consider $\delta = \frac{3}{4}$.
The second condition can be written as

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

That is $\tilde{b}_{i+1}$ is not much shorter than $\tilde{b}_i$

- LLL produces a $\delta-$LLL reduced basis.

$\delta-$LLL Reduced Basis

Consider the orthonormal basis produced by GSO. A $\delta-$LLL reduced basis looks like this

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

Approximating **SVP** with LLL

Theorem

Suppose $b_1, \ldots, b_n$ is a $\delta-$LLL reduced basis. Then

$$\|b_1\| \leq \left(\frac{2}{\sqrt{4\delta - 1}}\right)^{n-1} \lambda_1(\mathcal{L}(B))$$

Approximating **SVP** with LLL

Theorem

Suppose $b_1, \ldots, b_n$ is a $\delta-$LLL reduced basis. Then

$$\|b_1\| \leq \left(\frac{2}{\sqrt{4\delta - 1}}\right)^{n-1} \lambda_1(\mathcal{L}(B))$$

Proof.

We have

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

Approximating **SVP** with LLL

Theorem

Suppose $b_1, \ldots, b_n$ is a $\delta-$LLL reduced basis. Then

$$\|b_1\| \leq \left(\frac{2}{\sqrt{4\delta - 1}}\right)^{n-1} \lambda_1(\mathcal{L}(B))$$

Proof.

We have

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

After a few calculations we get that forall i

$$\|\tilde{b}_1\| \leq \left(\delta - \frac{1}{4}\right)^{\frac{n-1}{2}}\|\tilde{b}_i\|$$

and since $\lambda_1(\mathcal{L}(B)) \geq \min_i \|\tilde{b}_i\|$ we get the result. ∎

Approximating **SVP** with LLL

Theorem

Suppose $b_1, \dots, b_n$ is a $\delta-$LLL reduced basis. Then

$$\|b_1\| \leq \left(\frac{2}{\sqrt{4\delta - 1}}\right)^{n-1} \lambda_1(\mathcal{L}(B))$$

Proof.

We have

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

After a few calculations we get that forall i

$$\|\tilde{b}_1\| \leq \left(\delta - \frac{1}{4}\right)^{\frac{n-1}{2}}\|\tilde{b}_i\|$$

and since $\lambda_1(\mathcal{L}(B)) \geq \min_i \|\tilde{b}_i\|$ we get the result. ∎

For $\delta = \frac{3}{4}$ this gives a $2^{\frac{n-1}{2}}$ approximation ratio.

The LLL Algorithm

1. **Start** Compute $\tilde{b}_1, \ldots, \tilde{b}_n$
2. **Reduction**
   for $i = 2$ to n
   for $j = i - 1$ to 1
   $c_{i,j} = \lfloor \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle} \rceil$
   $b_i \leftarrow b_i - c_{i,j} b_j$
3. **Swap**
   if there exists i s.t $\delta \|\tilde{b}_i\|^2 > \|\mu_{i+1,i} \tilde{b}_i + \tilde{b}_{i+1}\|^2$
   $b_i \leftrightarrow b_{i+1}$
   goto start

Correctness of the algorithm

- Suppose that the LLL algorithm terminates.

Correctness of the algorithm

- Suppose that the LLL algorithm terminates.
- A simple calculations shows that GSO does not change during the reduction step.

Correctness of the algorithm

- Suppose that the LLL algorithm terminates.
- A simple calculations shows that GSO does not change during the reduction step.
- Condition 2 is enforced by the swap step.

Correctness of the algorithm

- Suppose that the LLL algorithm terminates.
- A simple calculations shows that GSO does not change during the reduction step.
- Condition 2 is enforced by the swap step.
- Condition 1 is achieved by the reduction step. Namely

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

Correctness of the algorithm

- Suppose that the LLL algorithm terminates.
- A simple calculations shows that GSO does not change during the reduction step.
- Condition 2 is enforced by the swap step.
- Condition 1 is achieved by the reduction step. Namely

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

- Each iteration runs in polynomial time with respect to the input (not so simple).

Correctness of the algorithm

- Suppose that the LLL algorithm terminates.
- A simple calculations shows that GSO does not change during the reduction step.
- Condition 2 is enforced by the swap step.
- Condition 1 is achieved by the reduction step. Namely

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

- Each iteration runs in polynomial time with respect to the input (not so simple).
- We need to show that the number of iterations is polynomial.

Bounding iterations of LLL

- We define $\mathcal{D}_{B,i} = \det\Lambda_i = \prod_{j=1}^{i} \|\tilde{b_1}\| \cdots \|\tilde{b_i}\|$

Bounding iterations of LLL

- We define $\mathcal{D}_{B,i} = \det\Lambda_i = \prod_{j=1}^{i} \|\tilde{b_1}\| \cdots \|\tilde{b_i}\|$
- We define the potential function $\mathcal{D}_B = \prod_{i=1}^{n} \mathcal{D}_{B,i}$.

Bounding iterations of LLL

- We define $\mathcal{D}_{B,i} = \det\Lambda_i = \prod_{j=1}^{i} \|\tilde{b_1}\| \cdots \|\tilde{b_i}\|$
- We define the potential function $\mathcal{D}_B = \prod_{i=1}^{n} \mathcal{D}_{B,i}$.
- Initially the size of $\mathcal{D}_B$ is polynomial. This is because

$$\mathcal{D}_B = \prod_{i=1}^{n} \|\tilde{b_1}\| \cdots \|\tilde{b_i}\| = \|b_1\|^n \|b_2\|^{n-1} \cdots \|b_n\| \leq \max_i \|b_i\|^{\frac{n(n+1)}{2}}$$

Bounding iterations of LLL

- We define $\mathcal{D}_{B,i} = \det\Lambda_i = \prod_{j=1}^{i} \|\tilde{b_1}\| \cdots \|\tilde{b_i}\|$
- We define the potential function $\mathcal{D}_B = \prod_{i=1}^{n} \mathcal{D}_{B,i}$.
- Initially the size of $\mathcal{D}_B$ is polynomial. This is because

$$\mathcal{D}_B = \prod_{i=1}^{n} \|\tilde{b_1}\| \cdots \|\tilde{b_i}\| = \|b_1\|^n \|b_2\|^{n-1} \cdots \|b_n\| \leq \max_i \|b_i\|^{\frac{n(n+1)}{2}}$$

- We will show that $\mathcal{D}_B$ decreases by a constant factor in each iteration.

Bounding iterations of LLL

- During the reduction step $\mathcal{D}_B$ does not change.

Bounding iterations of LLL

- During the reduction step $\mathcal{D}_B$ does not change.
- After swaping $b_i \leftrightarrow b_{i+1}$, the only quantity that changes is $\mathcal{D}_{B,i}$

Bounding iterations of LLL

- During the reduction step $\mathcal{D}_B$ does not change.
- After swaping $b_i \leftrightarrow b_{i+1}$, the only quantity that changes is $\mathcal{D}_{B,i}$
- We have $\frac{\mathcal{D}'_B}{\mathcal{D}_B} = \frac{\mathcal{D}'_{B,i}}{\mathcal{D}_{B,i}}$. In particular

$$\frac{\mathcal{D}'_{B,i}}{\mathcal{D}_{B,i}} = \frac{\det(\mathcal{L}(b_1, \ldots, b_{i-1}, b_{i+1}))}{\det(\mathcal{L}(b_1, \ldots, b_i))}$$

$$= \frac{(\prod_{j=1}^{i-1} \|\tilde{b}_j\|)\|\mu_{i+1,i}\tilde{b}_i + \tilde{b}_{i+1}\|}{\prod_{j=1}^{i} \|\tilde{b}_j\|} = \frac{\|\tilde{b}_j\| \cdot \|\mu_{i+1,i}\tilde{b}_i + \tilde{b}_{i+1}\|}{\|\tilde{b}_i\|} \leq \sqrt{\delta}$$

Bounding iterations of LLL

- During the reduction step $\mathcal{D}_B$ does not change.
- After swaping $b_i \leftrightarrow b_{i+1}$, the only quantity that changes is $\mathcal{D}_{B,i}$
- We have $\frac{\mathcal{D}'_B}{\mathcal{D}_B} = \frac{\mathcal{D}'_{B,i}}{\mathcal{D}_{B,i}}$. In particular

$$\frac{\mathcal{D}'_{B,i}}{\mathcal{D}_{B,i}} = \frac{\det(\mathcal{L}(b_1, \ldots, b_{i-1}, b_{i+1}))}{\det(\mathcal{L}(b_1, \ldots, b_i))}$$

$$= \frac{(\prod_{j=1}^{i-1} \|\tilde{b}_j\|)\|\mu_{i+1,i}\tilde{b}_i + \tilde{b}_{i+1}\|}{\prod_{j=1}^{i} \|\tilde{b}_j\|} = \frac{\|\tilde{b}_j\| \cdot \|\mu_{i+1,i}\tilde{b}_i + \tilde{b}_{i+1}\|}{\|\tilde{b}_i\|} \leq \sqrt{\delta}$$

- So we can have at most $\log_{\frac{1}{\sqrt{\delta}}} \mathcal{D}_B$ iteration which is polynomial.

Contents

1 Lattices in $\mathbb{R}^m$

2 The LLL Algorithm

3 Babai's Nearest Plane Algorithm

4 Complexity Results

Babai's Nearest Plane Algorithm

- Developed by L. Babai in 1986.

Babai's Nearest Plane Algorithm

- Developed by L. Babai in 1986.
- Solves **SearchCVP** for $\gamma = 2 \cdot \frac{2}{\sqrt{3}}^{n}$.

Babai's Nearest Plane Algorithm

- Developed by L. Babai in 1986.
- Solves **SearchCVP** for $\gamma = 2 \cdot \frac{2}{\sqrt{3}}^{n}$.
- Utilizes an LLL basis to solve **SeacrhCVP**.

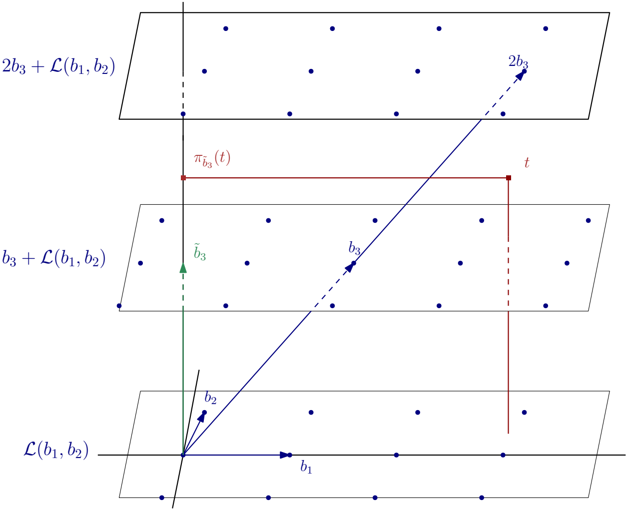
Babai's Nearest Plane Algorithm

- Developed by L. Babai in 1986.
- Solves **SearchCVP** for $\gamma = 2 \cdot \frac{2}{\sqrt{3}}^{n}$.
- Utilizes an LLL basis to solve **SeacrhCVP**.
- We will present the algorithm and omit its analysis.

The Nearest Plane Algorithm

- Compute a $\delta-$LLL reduced basis.
- $b \leftarrow t$
  for $j = n$ to 1
  $c_j = \lfloor \frac{\langle b, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle} \rceil$
  $b \leftarrow b - c_j b_j$
- **return** $t - b$

Geometric Illustration of the Algorithm



Contents

1 Lattices in $\mathbb{R}^m$

2 The LLL Algorithm

3 Babai's Nearest Plane Algorithm

4 Complexity Results

$\textbf{PromiseCVP}_1$ is $\textbf{NP}$ Complete

We show that **SubsetSum** reduces to $\textbf{PromiseCVP}_1$.

$\textbf{PromiseCVP}_1$ is $\textbf{NP}$ Complete

We show that **SubsetSum** reduces to $\textbf{PromiseCVP}_1$. We map an instance of Subset Sum as follows

PromiseCVP_1 is NP Complete

We show that **SubsetSum** reduces to **PromiseCVP**$_1$. We map an instance of Subset Sum as follows

$$\langle \{a_1, \ldots, a_n\}, S \rangle \mapsto \left\langle B = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \\ 2 & 0 & \cdots & 0 \\ 0 & 2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 2 \end{bmatrix}, t = \begin{bmatrix} S \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, r = \sqrt{n} \right\rangle$$

$\textbf{PromiseCVP}_1$ is **NP** Complete

We show that **SubsetSum** reduces to $\textbf{PromiseCVP}_1$. We map an instance of Subset Sum as follows

$$\langle \{a_1, \ldots, a_n\}, S\rangle \mapsto \left\langle B = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \\ 2 & 0 & \cdots & 0 \\ 0 & 2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 2 \end{bmatrix}, \ t = \begin{bmatrix} S \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \ r = \sqrt{n} \right\rangle$$

- If $\langle \{a_1, \ldots, a_n\}, S\rangle \in$ **SubsetSum** then suppose that $\sum_{i \in A} a_i = S$. Now take $y = Bz$ where z is vector with its j-th coordinate equal to 1 if $j \in A$ and 0 otherwise. Then $\|Bz - t\| = \|[0 \ \pm 1 \ \ldots \ \pm 1]^T\| = \sqrt{n}$ and so $\langle B, t, r\rangle \in \textbf{PromiseCVP}_1$.

$\mathbf{PromiseCVP}_1$ is NP Complete

We show that **SubsetSum** reduces to $\mathbf{PromiseCVP}_1$. We map an instance of Subset Sum as follows

$$\langle \{a_1, \ldots, a_n\}, S \rangle \mapsto \left\langle B = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \\ 2 & 0 & \cdots & 0 \\ 0 & 2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 2 \end{bmatrix}, \; t = \begin{bmatrix} S \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \; r = \sqrt{n} \right\rangle$$

- If $\langle \{a_1, \ldots, a_n\}, S \rangle \in$ **SubsetSum** then suppose that $\sum_{i \in A} a_i = S$. Now take $y = Bz$ where z is vector with its j-th coordinate equal to 1 if $j \in A$ and 0 otherwise. Then $\|Bz - t\| = \|[0 \;\; \pm 1 \;\; \ldots \;\; \pm 1]^T\| = \sqrt{n}$ and so $\langle B, t, r \rangle \in \mathbf{PromiseCVP}_1$.
- If $\langle B, t, r \rangle \in \mathbf{PromiseCVP}_1$. Assume $x \in \mathcal{L}(B)$ such that $\|x - t\| \leq \sqrt{n}$ The last n coordinates are even, so subtracting 1 gives at least $\sqrt{n}$. It must be the case that the first coordinate is S.

Decision and Search **CVP**

We will show that we can solve the search problem given an oracle for the decisional problem.

Decision and Search **CVP**

We will show that we can solve the search problem given an oracle for the decisional problem.

- We first find the length of the closest vector r.

Decision and Search **CVP**

We will show that we can solve the search problem given an oracle for the decisional problem.

- We first find the length of the closest vector r.
- We use binary search in $[0, R]$ where $R = \sum_{i=1}^{n} \|b_i\|$.

Decision and Search **CVP**

We will show that we can solve the search problem given an oracle for the decisional problem.

- We first find the length of the closest vector r.
- We use binary search in $[0, R]$ where $R = \sum_{i=1}^{n} \|b_i\|$.
- We have R^2 possibilities since we deal with integers.

Decision and Search **CVP**

We will show that we can solve the search problem given an oracle for the decisional problem.

- We first find the length of the closest vector r.
- We use binary search in $[0, R]$ where $R = \sum_{i=1}^{n} \|b_i\|$.
- We have R^2 possibilities since we deal with integers.
- We need $2 \log R$ queries to the oracle (polynomial).

Decision and Search **CVP**

We will show that we can solve the search problem given an oracle for the decisional problem.

- We first find the length of the closest vector r.
- We use binary search in $[0, R]$ where $R = \sum_{i=1}^{n} \|b_i\|$.
- We have R^2 possibilities since we deal with integers.
- We need $2 \log R$ queries to the oracle (polynomial).
- Note that if we find the closest vector to $t + v$ for $v \in \mathcal{L}$ we are done.

The iterative step of the reduction

We iteratively make sparser the lattice while maintaining three properties

The iterative step of the reduction

We iteratively make sparser the lattice while maintaining three properties

1. $\mathcal{L}(B')$ is a sublattice of $\mathcal{L}(B)$.

The iterative step of the reduction

We iteratively make sparser the lattice while maintaining three properties

1. $\mathcal{L}(B')$ is a sublattice of $\mathcal{L}(B)$.
2. $t' = t + v$ for $v \in \mathcal{L}(B)$.

The iterative step of the reduction

We iteratively make sparser the lattice while maintaining three properties

1. $\mathcal{L}(B')$ is a sublattice of $\mathcal{L}(B)$.
2. $t' = t + v$ for $v \in \mathcal{L}(B)$.
3. $\text{dist}(\mathcal{L}(B'), t') = r$

The iterative step of the reduction

We iteratively make sparser the lattice while maintaining three properties

1. $\mathcal{L}(B')$ is a sublattice of $\mathcal{L}(B)$.
2. $t' = t + v$ for $v \in \mathcal{L}(B)$.
3. $\text{dist}(\mathcal{L}(B'), t') = r$

The iterative step (for b_1) is the following

- Set $B' = [2b_1 \ b_2 \ \ldots b_n]$. We take a sublattice with half the points.

The iterative step of the reduction

We iteratively make sparser the lattice while maintaining three properties

1. $\mathcal{L}(B')$ is a sublattice of $\mathcal{L}(B)$.
2. $t' = t + v$ for $v \in \mathcal{L}(B)$.
3. $\text{dist}(\mathcal{L}(B'), t') = r$

The iterative step (for b_1) is the following

- Set $B' = [2b_1 \ b_2 \ \ldots b_n]$. We take a sublattice with half the points.
- Using the oracle, if $\text{dist}(\mathcal{L}(B)', t) \leq r$ we set $t' = t$ else $t' = t - b_1$.

The iterative step of the reduction

We iteratively make sparser the lattice while maintaining three properties

1. $\mathcal{L}(B')$ is a sublattice of $\mathcal{L}(B)$.
2. $t' = t + v$ for $v \in \mathcal{L}(B)$.
3. $\text{dist}(\mathcal{L}(B'), t') = r$

The iterative step (for b_1) is the following

- Set $B' = [2b_1\ b_2\ \dots b_n]$. We take a sublattice with half the points.
- Using the oracle, if $\text{dist}(\mathcal{L}(B)', t) \leq r$ we set $t' = t$ else $t' = t - b_1$.

We have that $\mathcal{L}(B) = \mathcal{L}(B') \cup \mathcal{L}(B' + b_1)$. So the distance to from t to the new subblatice or the shifted sublattice is r.

The reduction

- We perform the iterative step $k = n + \log r$ times for each coordinates.

The reduction

- We perform the iterative step $k = n + \log r$ times for each coordinates.
- Finally the basis of the lattice is of the form $B^* = [2^k b_1 \ \ldots \ 2^k b_n]$.

The reduction

- We perform the iterative step $k = n + \log r$ times for each coordinates.
- Finally the basis of the lattice is of the form $B^* = [2^k b_1 \ \dots \ 2^k b_n]$.
- We know that $\text{dist}(B^*, t^*) = r$.

The reduction

- We perform the iterative step $k = n + \log r$ times for each coordinates.
- Finally the basis of the lattice is of the form $B^* = [2^k b_1 \; \ldots \; 2^k b_n]$.
- We know that $\text{dist}(B^*, t^*) = r$.
- Note that $\lambda_1(\mathcal{L}(B^*)) \geq 2^k = 2^n \cdot r$.

The reduction

- We perform the iterative step $k = n + \log r$ times for each coordinates.
- Finally the basis of the lattice is of the form $B^* = [2^k b_1 \ \dots \ 2^k b_n]$.
- We know that $\text{dist}(B^*, t^*) = r$.
- Note that $\lambda_1(\mathcal{L}(B^*)) \geq 2^k = 2^n \cdot r$.
- The second closest vector to t^* is at distance at least $2^n r - r \geq 2^{n-1} \cdot r$.

The reduction

- We perform the iterative step $k = n + \log r$ times for each coordinates.
- Finally the basis of the lattice is of the form $B^* = [2^k b_1 \ \ldots \ 2^k b_n]$.
- We know that $\mathrm{dist}(B^*, t^*) = r$.
- Note that $\lambda_1(\mathcal{L}(B^*)) \geq 2^k = 2^n \cdot r$.
- The second closest vector to t^* is at distance at least $2^n r - r \geq 2^{n-1} \cdot r$.
- So if we run the nearest plane algorithm we get the closest vector.

The reduction

- We perform the iterative step $k = n + \log r$ times for each coordinates.
- Finally the basis of the lattice is of the form $B^* = [2^k b_1 \ \ldots \ 2^k b_n]$.
- We know that $\text{dist}(B^*, t^*) = r$.
- Note that $\lambda_1(\mathcal{L}(B^*)) \geq 2^k = 2^n \cdot r$.
- The second closest vector to t^* is at distance at least $2^n r - r \geq 2^{n-1} \cdot r$.
- So if we run the nearest plane algorithm we get the closest vector.
- From there we can construct t for the initial lattice.

The reduction

- We perform the iterative step $k = n + \log r$ times for each coordinates.
- Finally the basis of the lattice is of the form $B^* = [2^k b_1 \ \ldots \ 2^k b_n]$.
- We know that $\text{dist}(B^*, t^*) = r$.
- Note that $\lambda_1(\mathcal{L}(B^*)) \geq 2^k = 2^n \cdot r$.
- The second closest vector to t^* is at distance at least $2^n r - r \geq 2^{n-1} \cdot r$.
- So if we run the nearest plane algorithm we get the closest vector.
- From there we can construct t for the initial lattice.

We dont know how to generalize this for the gap versions of the problems.

PromiseSVP$_\gamma$ is not harder than **PromiseCVP**$_\gamma$

- SVP looks like CVP with origin as the target vector.

PromiseSVP$_\gamma$ is not harder than **PromiseCVP**$_\gamma$

- SVP looks like CVP with origin as the target vector.
- This cannot be directly utilized to solve SVP with a CVP oracle.

PromiseSVP$_\gamma$ is not harder than PromiseCVP$_\gamma$

- SVP looks like CVP with origin as the target vector.
- This cannot be directly utilized to solve SVP with a CVP oracle.
- The origin is always a lattice point and so the oracle will always return it.

PromiseSVP$_\gamma$ is not harder than **PromiseCVP**$_\gamma$

- SVP looks like CVP with origin as the target vector.
- This cannot be directly utilized to solve SVP with a CVP oracle.
- The origin is always a lattice point and so the oracle will always return it.
- We need to delete it somehow.

PromiseSVP$_\gamma$ is not harder than **PromiseCVP**$_\gamma$

- SVP looks like CVP with origin as the target vector.
- This cannot be directly utilized to solve SVP with a CVP oracle.
- The origin is always a lattice point and so the oracle will always return it.
- We need to delete it somehow.
- If we delete a lattice point we no longer have a lattice.

PromiseSVP$_\gamma$ is not harder than **PromiseCVP**$_\gamma$

- SVP looks like CVP with origin as the target vector.
- This cannot be directly utilized to solve SVP with a CVP oracle.
- The origin is always a lattice point and so the oracle will always return it.
- We need to delete it somehow.
- If we delete a lattice point we no longer have a lattice.
- So in the reduction, we delete a set of lattice points.

The Reduction

- Our input is a basis B and an r.

The Reduction

- Our input is a basis B and an r.
- We try to distinguish between the cases $\lambda_1 \leq r$ and $\lambda_1 > \gamma \cdot r$.

The Reduction

- Our input is a basis B and an r.
- We try to distinguish between the cases $\lambda_1 \leq r$ and $\lambda_1 > \gamma \cdot r$.
- We create n new subblattices with bases
  $B_i = [b_1 \ \ldots \ b_{i-1} \ 2b_i \ b_{i+1} \ \ldots \ b_n]$

The Reduction

- Our input is a basis B and an r.
- We try to distinguish between the cases $\lambda_1 \leq r$ and $\lambda_1 > \gamma \cdot r$.
- We create n new subblattices with bases $B_i = [b_1 \ \ldots \ b_{i-1} \ 2b_i \ b_{i+1} \ \ldots \ b_n]$
- We run the oracle for inputs $\langle B_i, b_i, r \rangle$.

The Reduction

- Our input is a basis B and an r.
- We try to distinguish between the cases $\lambda_1 \leq r$ and $\lambda_1 > \gamma \cdot r$.
- We create n new subblattices with bases
  $B_i = [b_1 \ \ldots \ b_{i-1} \ 2b_i \ b_{i+1} \ \ldots \ b_n]$
- We run the oracle for inputs $\langle B_i, b_i, r \rangle$.
- If any returns YES we return YES otherwise we return NO.

Correctness of the Reduction

We examine the two cases

Correctness of the Reduction

We examine the two cases

- if $(B, r) \notin$ **PromiseSVP**$_\gamma$ then $\lambda_1(\mathcal{L}(B)) > \gamma \cdot r$. So any lattice vector v has length $\|v\| > \gamma \cdot r$. Now suppose that for some i the oracle returned YES. Then there exists $v \in \mathcal{L}(B_i)$ s.t. $\|v - b_i\| \leq r$. But this is a lattice point in $\mathcal{L}(B)$ which is a contradiction.

Correctness of the Reduction

We examine the two cases

- if $(B, r) \notin \mathbf{PromiseSVP}_\gamma$ then $\lambda_1(\mathcal{L}(B)) > \gamma \cdot r$. So any lattice vector v has length $\|v\| > \gamma \cdot r$. Now suppose that for some i the oracle returned YES. Then there exists $v \in \mathcal{L}(B_i)$ s.t. $\|v - b_i\| \leq r$. But this is a lattice point in $\mathcal{L}(B)$ which is a contradiction.
- if $(B, r) \in \mathbf{PromiseSVP}_\gamma$ then $\lambda_1(\mathcal{L}(B)) \leq r$. Let v be the smallest vector. Then $v = a_1 b_1 + \ldots + a_n b_n$ for some a_i odd. Then $b_i + v \in \mathcal{L}(B_i)$ and its distance from b_i is less than r so the oracle must return YES.

The end!

Thank you! Questions?