

COMP8760 Lecture

Lattice

Sanjay Bhattacharjee

University of Kent

Outline

Euclidean n -Space

Lattices

- Gram-Schmidt Orthogonalisation
- Shortest Vector Problem

Post-Quantum Cryptography

Lattice Reduction

- LLL
- DeepLLL
- Pot-DeepLLL and SS-DeepLLL

A Greedy Global Framework for Lattice Reduction

- Pot-GGLLL and SS-GGLLL

Outline

Euclidean n -Space

Lattices

- Gram-Schmidt Orthogonalisation
- Shortest Vector Problem

Post-Quantum Cryptography

Lattice Reduction

- LLL
- DeepLLL
- Pot-DeepLLL and SS-DeepLLL

A Greedy Global Framework for Lattice Reduction

- Pot-GGLLL and SS-GGLLL

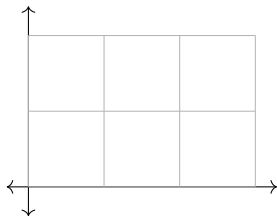
Euclidean n -Space

2-space vectors:

$$\mathbf{b}_1 = (1, 2)$$

$$\mathbf{b}_2 = (2, 0)$$

$$\mathbf{b}_1 + \mathbf{b}_2 = (3, 2)$$



n -space vector: $\mathbf{b} = (b_1, b_2, \dots, b_n)$

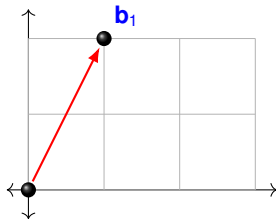
Euclidean n -Space

2-space vectors:

$$\mathbf{b}_1 = (1, 2)$$

$$\mathbf{b}_2 = (2, 0)$$

$$\mathbf{b}_1 + \mathbf{b}_2 = (3, 2)$$



n -space vector: $\mathbf{b} = (b_1, b_2, \dots, b_n)$

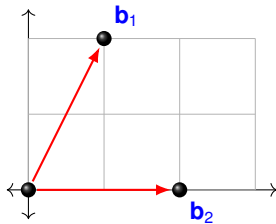
Euclidean n -Space

2-space vectors:

$$\mathbf{b}_1 = (1, 2)$$

$$\mathbf{b}_2 = (2, 0)$$

$$\mathbf{b}_1 + \mathbf{b}_2 = (3, 2)$$



n -space vector: $\mathbf{b} = (b_1, b_2, \dots, b_n)$

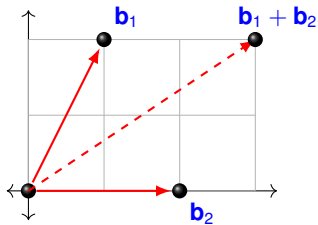
Euclidean n -Space

2-space vectors:

$$\mathbf{b}_1 = (1, 2)$$

$$\mathbf{b}_2 = (2, 0)$$

$$\mathbf{b}_1 + \mathbf{b}_2 = (3, 2)$$



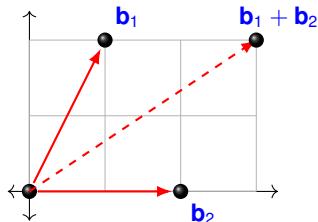
n -space vector: $\mathbf{b} = (b_1, b_2, \dots, b_n)$

Euclidean n -Space

Length of vector $\mathbf{b} = (b_1, \dots, b_n)$

$$\|\mathbf{b}\| = \sqrt{\sum_{i=1}^n b_i^2} = \sqrt{b_1^2 + \dots + b_n^2}$$

Examples:



$$\begin{aligned}\|\mathbf{b}_1\| &= \sqrt{1^2 + 2^2} = \sqrt{5} \\ \|\mathbf{b}_2\| &= \sqrt{2^2 + 0^2} = 2 \\ \|\mathbf{b}_1 + \mathbf{b}_2\| &= \sqrt{3^2 + 2^2} = \sqrt{13}\end{aligned}$$

Outline

Euclidean n -Space

Lattices

Gram-Schmidt Orthogonalisation
Shortest Vector Problem

Post-Quantum Cryptography

Lattice Reduction

LLL
DeepLLL
Pot-DeepLLL and SS-DeepLLL

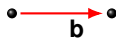
A Greedy Global Framework for Lattice Reduction

Pot-GGLLL and SS-GGLLL

Lattice: Single Dimension

$$\mathcal{L} = \{x\mathbf{b}, x \in \mathbb{Z}\}$$

All **integer** multiples of **b**



Lattice: Single Dimension

$$\mathcal{L} = \{x\mathbf{b}, x \in \mathbb{Z}\}$$

All **integer** multiples of **b**



Lattice: Single Dimension

$$\mathcal{L} = \{x\mathbf{b}, x \in \mathbb{Z}\}$$

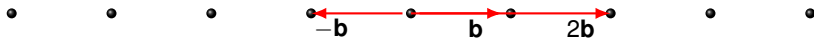
All **integer** multiples of **b**



Lattice: Single Dimension

$$\mathcal{L} = \{x\mathbf{b}, x \in \mathbb{Z}\}$$

All **integer** multiples of \mathbf{b}



Span: Single Dimension

$$\text{span}(\mathbf{b}) = \{x\mathbf{b}, x \in \mathbb{R}\}$$

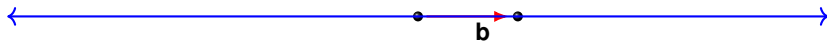
All **real** multiples of **b**



Span: Single Dimension

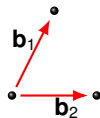
$$\text{span}(\mathbf{b}) = \{x\mathbf{b}, x \in \mathbb{R}\}$$

All **real** multiples of **b**



Linearly Independent Vectors

$\mathbf{b}_1 \notin \text{span}(\mathbf{b}_2)$ and vice versa: they are linearly independent

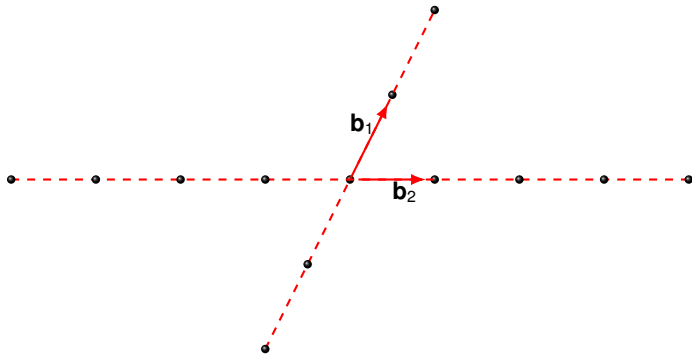


In general (for $n > 2$), $(\mathbf{b}_1, \dots, \mathbf{b}_n)$ are linearly independent vectors if and only if

$$\mathbf{b}_i \notin \text{span}(\mathbf{b}_1, \dots, \mathbf{b}_{i-1}, \mathbf{b}_{i+1}, \mathbf{b}_n)$$

Linearly Independent Vectors

$\mathbf{b}_1 \notin \text{span}(\mathbf{b}_2)$ and vice versa: they are linearly independent

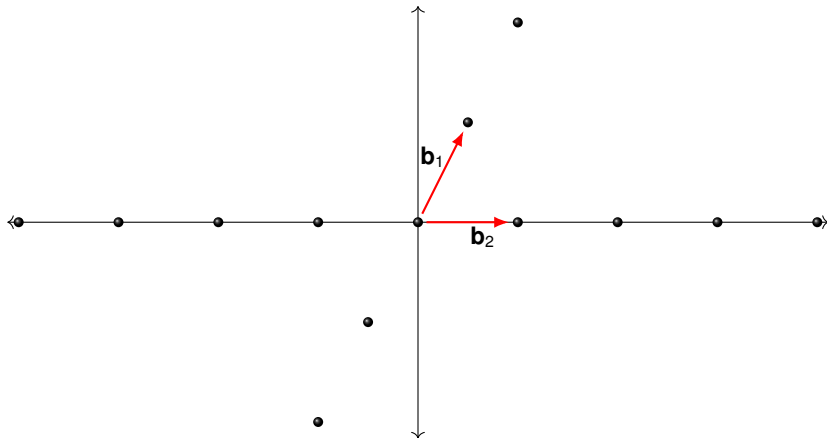


In general (for $n > 2$), $(\mathbf{b}_1, \dots, \mathbf{b}_n)$ are linearly independent vectors if and only if

$$\mathbf{b}_i \notin \text{span}(\mathbf{b}_1, \dots, \mathbf{b}_{i-1}, \mathbf{b}_{i+1}, \mathbf{b}_n)$$

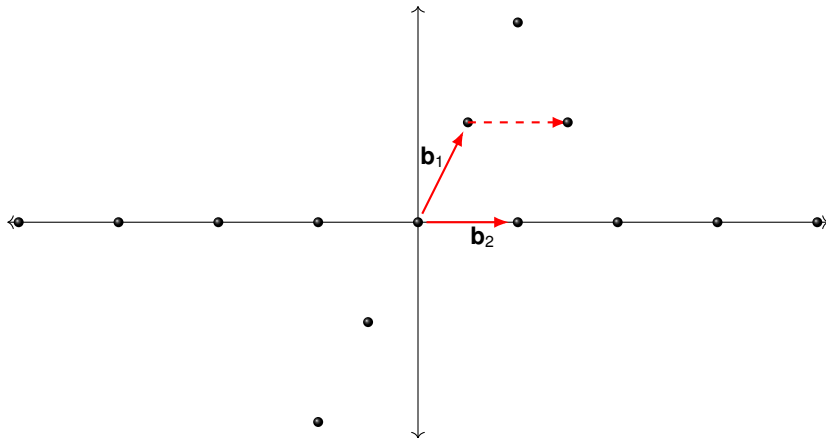
Adding Vectors

$$\mathbf{b}_1 + x\mathbf{b}_2, x \in \mathbb{Z}$$



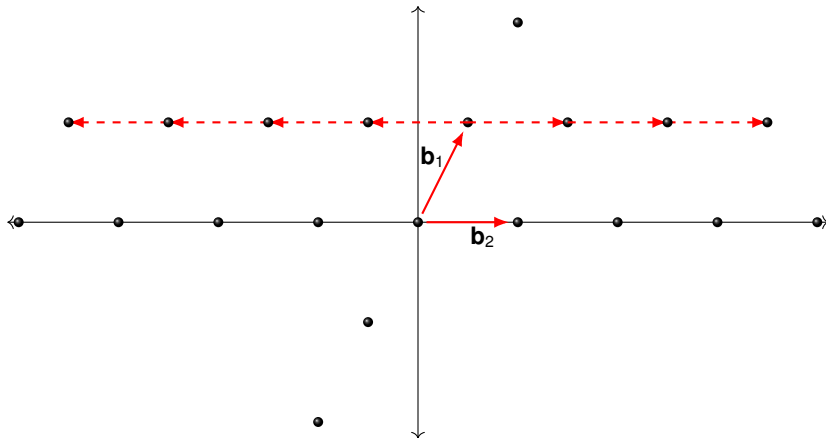
Adding Vectors

$$\mathbf{b}_1 + x\mathbf{b}_2, x \in \mathbb{Z}$$



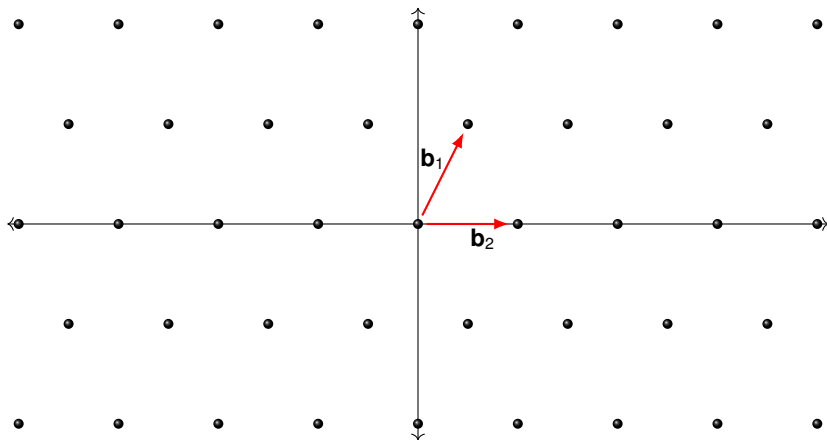
Adding Vectors

$$\mathbf{b}_1 + x\mathbf{b}_2, x \in \mathbb{Z}$$



Lattice: All integer linear combinations of $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$

$$\mathcal{L} = \{x_1 \mathbf{b}_1 + x_2 \mathbf{b}_2, x_1, x_2 \in \mathbb{Z}\}$$



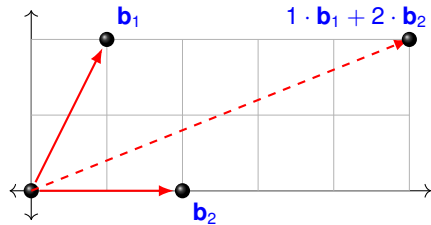
In general (for $n > 2$), $\mathcal{L} = \{\mathbf{B}\mathbf{x} : \mathbf{x} \in \mathbb{Z}^n\}$, where $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$, $\mathbf{b}_i \in \mathbb{R}^m$

Lattice: $\mathcal{L} = \{\mathbf{B}\mathbf{x} : \mathbf{x} \in \mathbb{Z}^n\}$, where $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$, $\mathbf{b}_i \in \mathbb{R}^m$

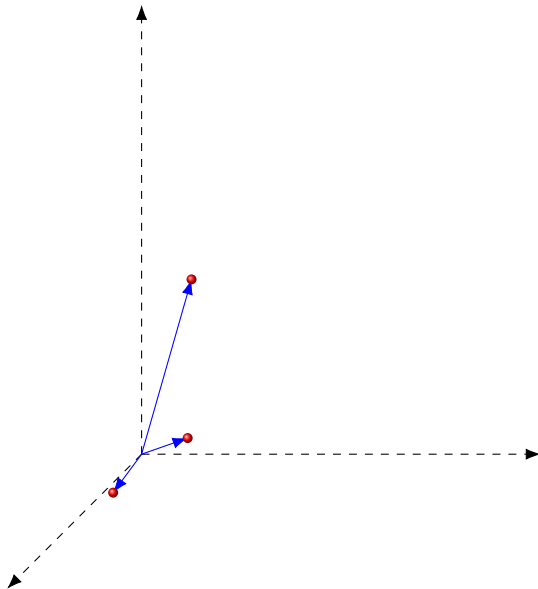
Integer linear combinations of $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$:

$$\mathbf{B}\mathbf{x} = \underbrace{\begin{pmatrix} | & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \dots & \mathbf{b}_n \\ | & | & & | \end{pmatrix}}_{\mathbf{B}} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{pmatrix} | & & & | \\ x_1\mathbf{b}_1 & + & x_2\mathbf{b}_2 & + & \dots & + & x_n\mathbf{b}_n \\ | & & & | \end{pmatrix}}_{\mathbf{B}\mathbf{x}}$$

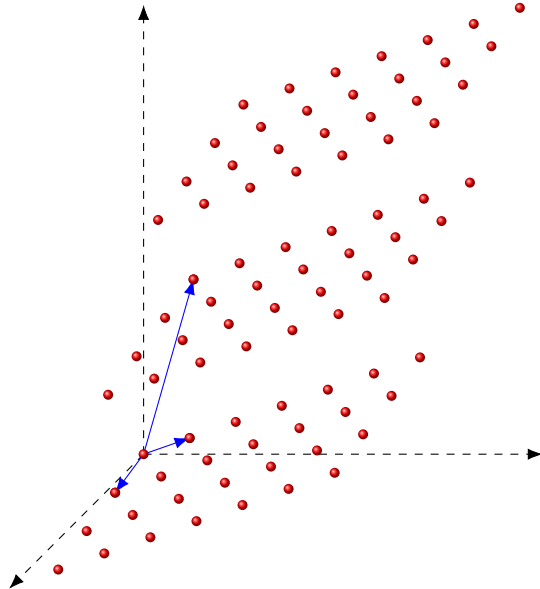
$$\underbrace{\begin{pmatrix} 1 & 2 \\ 2 & 0 \end{pmatrix}}_{\mathbf{B}} \underbrace{\begin{pmatrix} 1 \\ 2 \end{pmatrix}}_{\mathbf{x}} = \begin{pmatrix} 1 \times 1 & + & 2 \times 2 \\ 1 \times 2 & + & 2 \times 0 \end{pmatrix} = \underbrace{\begin{pmatrix} 5 \\ 2 \end{pmatrix}}_{\mathbf{B}\mathbf{x}}$$



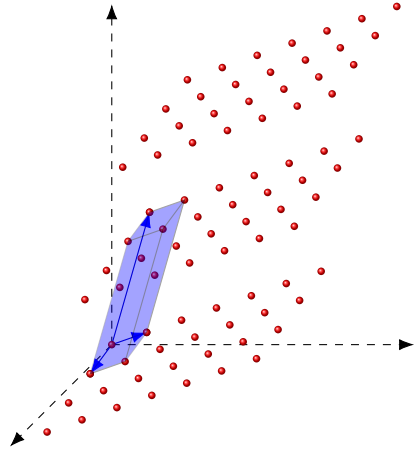
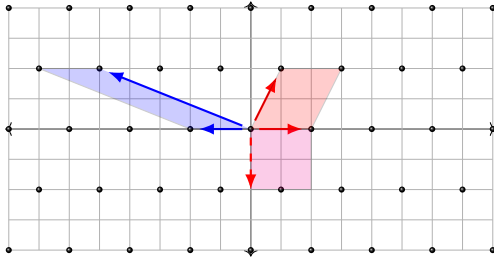
Lattice \mathcal{L} : 3 Dimensions



Lattice \mathcal{L} : 3 Dimensions



Lattice Volume: $\text{Vol}(\mathcal{L})$



Lattice: Summary

Lattice

$$\mathcal{L} = \{\mathbf{B}\mathbf{x} : \mathbf{x} \in \mathbb{Z}^n\}, \mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n), \mathbf{b}_i \in \mathbb{R}^m$$

- ▶ Only $\mathbf{b}_i \in \mathbb{Z}^m$, instead of $\mathbf{b}_i \in \mathbb{R}^m$
- ▶ Infinite set of points (vectors) in \mathbb{Z}^m

Basis \mathbf{B} for a Lattice \mathcal{L}

- ▶ $(\mathbf{b}_1, \dots, \mathbf{b}_n)$ are linearly independent vectors

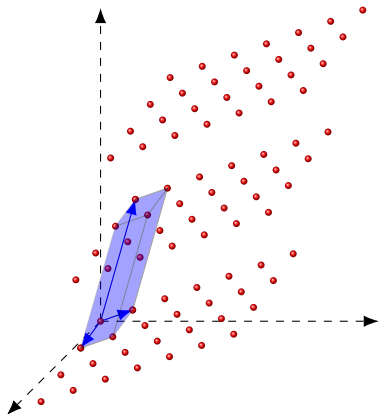
$$\mathbf{b}_i \notin \text{span}(\mathbf{b}_1, \dots, \mathbf{b}_{i-1}, \mathbf{b}_{i+1}, \mathbf{b}_n)$$

- ▶ For $n \geq 2$, there are infinitely many bases for \mathcal{L}

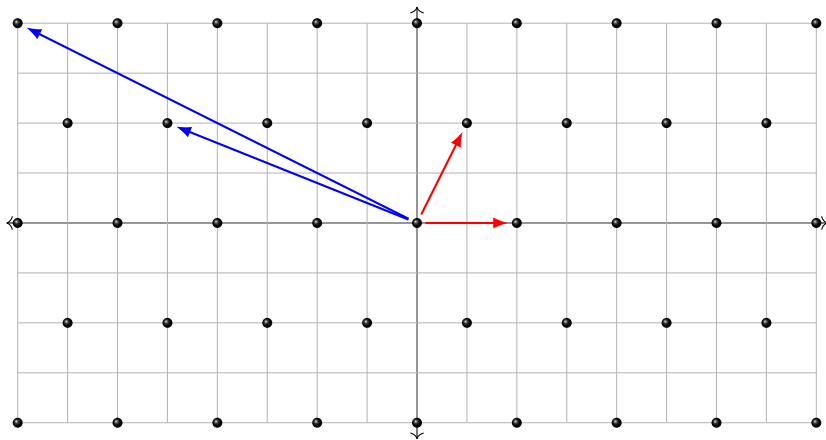
$$\text{span}(\mathbf{B}) = \{\mathbf{B}\mathbf{x} : \mathbf{x} \in \mathbb{R}^n\}$$

For $n = 1$, it is the **line** spanned by \mathbf{b}_1 .

For $n = 2$, it is the **plane** spanned by $(\mathbf{b}_1, \mathbf{b}_2)$.



Lattice: (Good and Bad) Bases



Outline

Euclidean n -Space

Lattices

Gram-Schmidt Orthogonalisation

Shortest Vector Problem

Post-Quantum Cryptography

Lattice Reduction

LLL

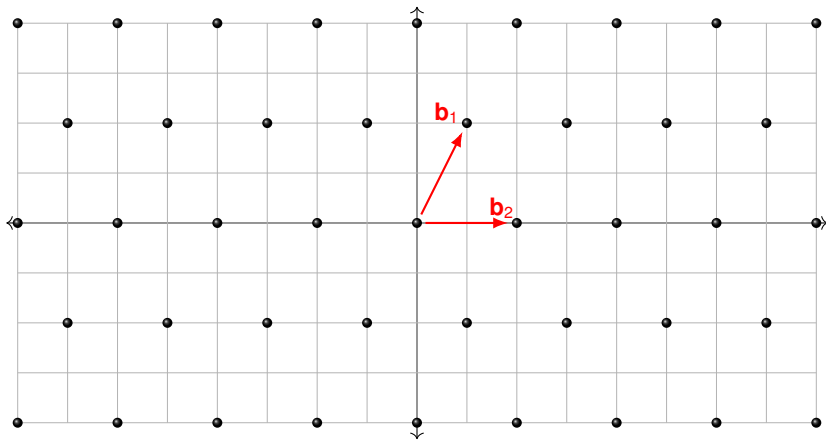
DeepLLL

Pot-DeepLLL and SS-DeepLLL

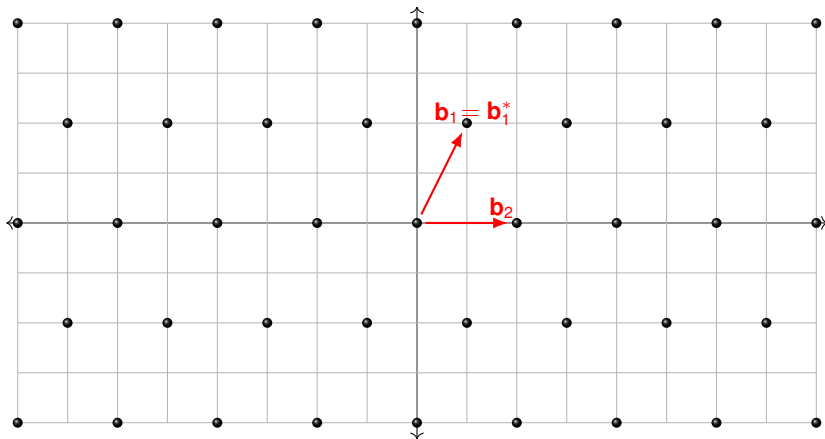
A Greedy Global Framework for Lattice Reduction

Pot-GGLLL and SS-GGLLL

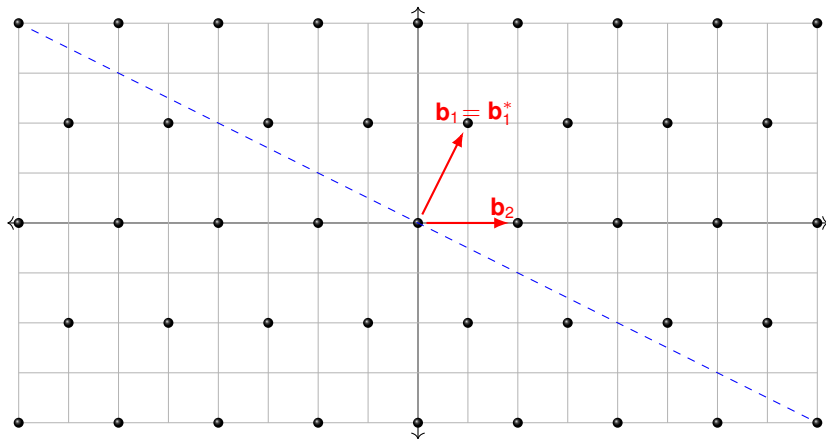
The Gram-Schmidt Orthogonalised (GSO) Basis



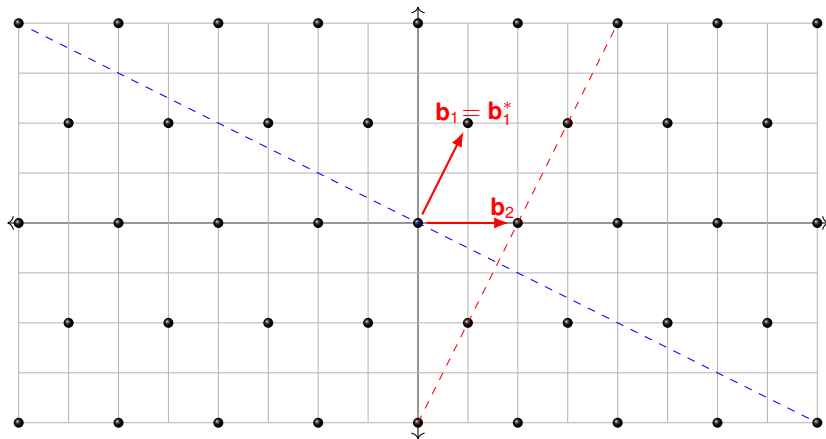
The Gram-Schmidt Orthogonalised (GSO) Basis



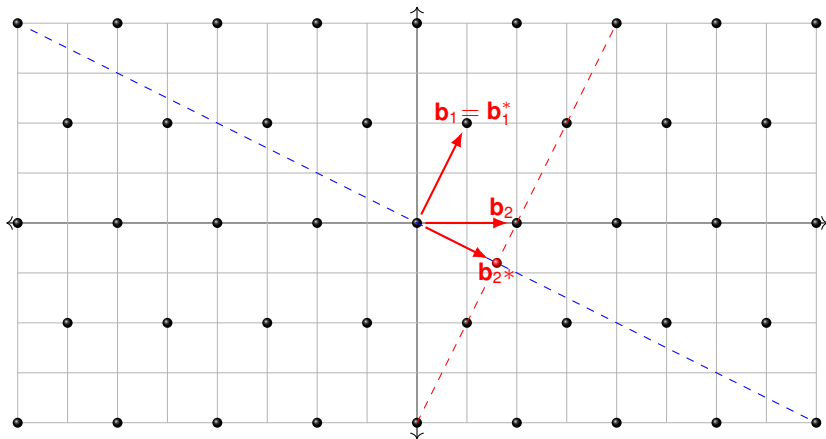
The Gram-Schmidt Orthogonalised (GSO) Basis



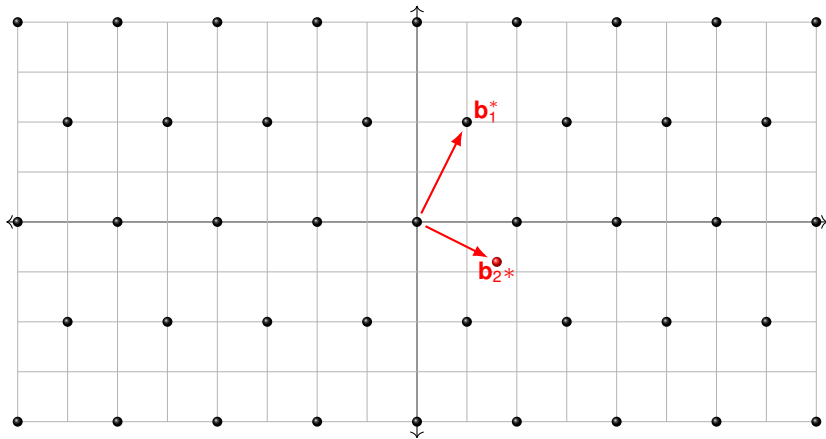
The Gram-Schmidt Orthogonalised (GSO) Basis



The Gram-Schmidt Orthogonalised (GSO) Basis

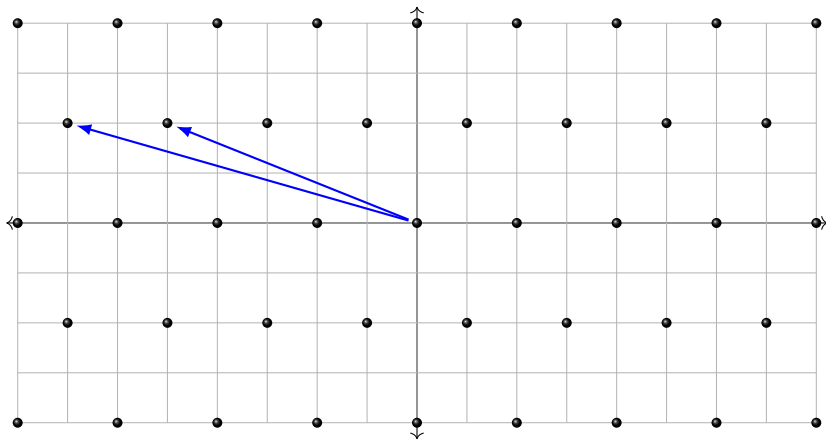


The Gram-Schmidt Orthogonalised (GSO) Basis

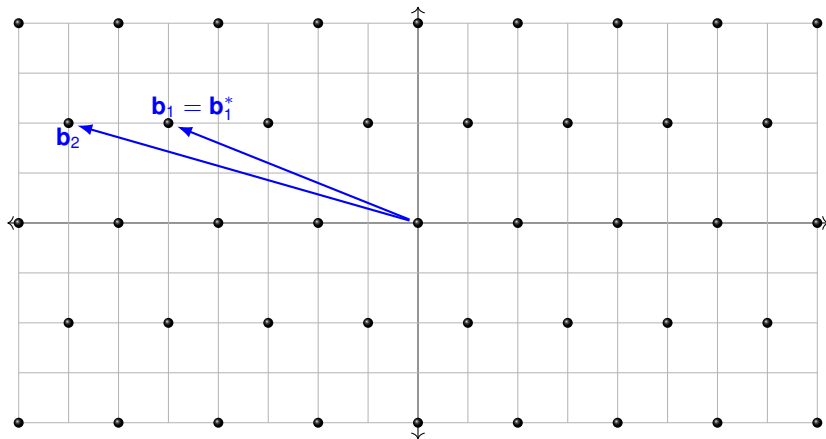


Note: $\mathbf{b}_2^* \notin \mathcal{L} \implies \mathbf{B}^* = (\mathbf{b}_1^*, \dots, \mathbf{b}_n^*)$ is not a basis for \mathcal{L}

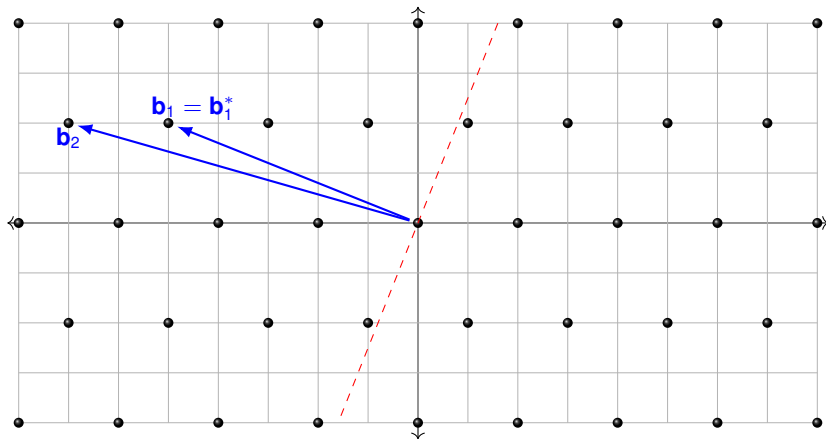
The Gram-Schmidt Orthogonalised (GSO) Basis



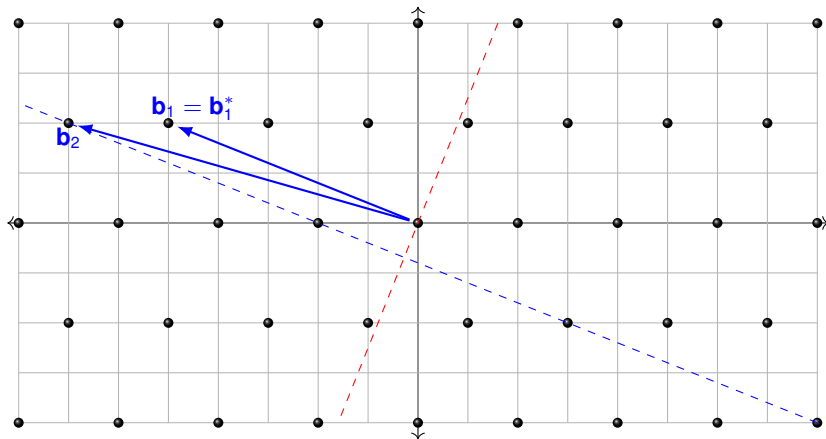
The Gram-Schmidt Orthogonalised (GSO) Basis



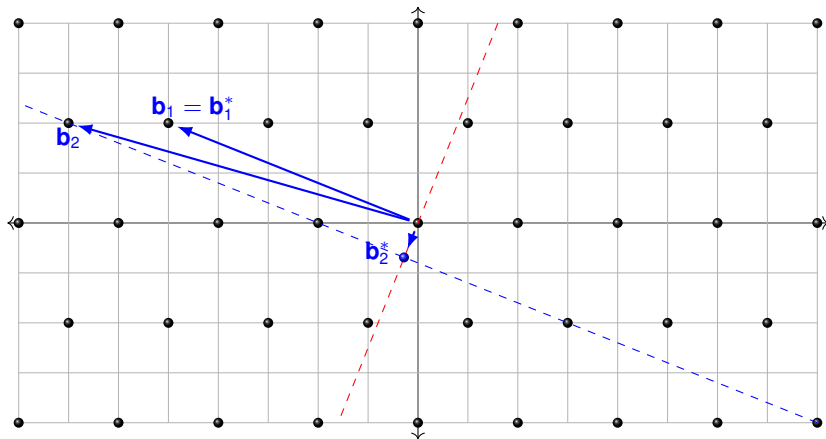
The Gram-Schmidt Orthogonalised (GSO) Basis



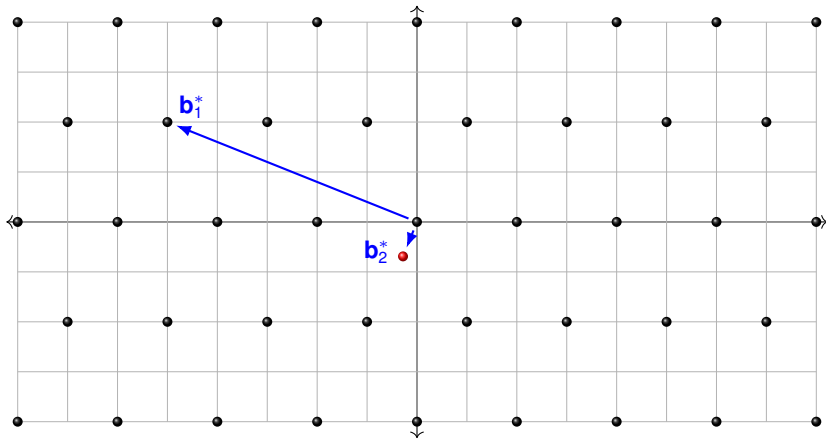
The Gram-Schmidt Orthogonalised (GSO) Basis



The Gram-Schmidt Orthogonalised (GSO) Basis



The Gram-Schmidt Orthogonalised (GSO) Basis



Note: $\mathbf{b}_2^* \notin \mathcal{L} \implies \mathbf{B}^* = (\mathbf{b}_1^*, \dots, \mathbf{b}_n^*)$ is not a basis for \mathcal{L}

Generalised Projection $\pi_i(\mathbf{b}_k)$

Consider an ordered set of linearly independent vectors:

$$\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4, \mathbf{b}_5, \mathbf{b}_6, \mathbf{b}_7, \mathbf{b}_8, \mathbf{b}_9, \mathbf{b}_{10})$$

Projection of \mathbf{b}_k orthogonal to $\text{span}(\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_{i-1})$

is denoted as

$$\pi_i(\mathbf{b}_k)$$

Examples: Considering projections orthogonal to $\text{span}(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4)$

- ▶ $\pi_5(\mathbf{b}_5)$ = Projection of \mathbf{b}_5 orthogonal to $\text{span}(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4)$
- ▶ $\pi_8(\mathbf{b}_8)$ = Projection of \mathbf{b}_8 orthogonal to $\text{span}(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4)$

GSO Basis vectors \mathbf{b}_i^* as projections of \mathbf{b}_i

$$\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4, \mathbf{b}_5, \mathbf{b}_6, \mathbf{b}_7, \mathbf{b}_8, \mathbf{b}_9, \mathbf{b}_{10})$$

$$\mathbf{B}^* = (\mathbf{b}_1^*, \mathbf{b}_2^*, \mathbf{b}_3^*, \mathbf{b}_4^*, \mathbf{b}_5^*, \mathbf{b}_6^*, \mathbf{b}_7^*, \mathbf{b}_8^*, \mathbf{b}_9^*, \mathbf{b}_{10}^*)$$

GSO vectors \mathbf{b}_i^*

The i^{th} GSO vector \mathbf{b}_i^* is the projection of \mathbf{b}_i in a direction orthogonal to $\text{span}(\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_{i-1})$

$$\mathbf{b}_i^* = \pi_i(\mathbf{b}_i)$$

Projection of \mathbf{b}_k orthogonal to $\text{span}(\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_{i-1})$

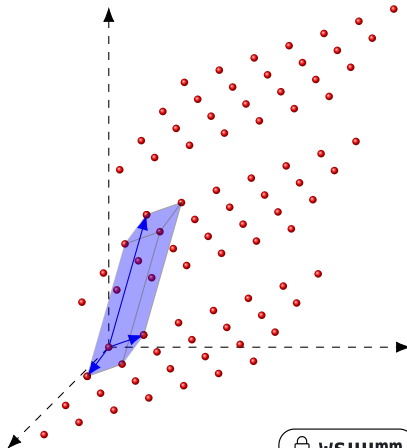
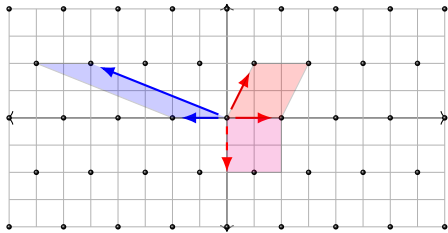
is computed as

$$\pi_i(\mathbf{b}_k) = \mathbf{b}_k^* + \sum_{\ell=i}^{k-1} \mu_{k,\ell} \mathbf{b}_\ell^* = \mathbf{b}_k - \sum_{\ell=1}^{i-1} \mu_{k,\ell} \mathbf{b}_\ell^*$$

where $\mu_{k,\ell} = \frac{\langle \mathbf{b}_k, \mathbf{b}_\ell^* \rangle}{\|\mathbf{b}_\ell^*\|^2}.$

Lattice Volume: $\text{Vol}(\mathcal{L})$

$$\text{Vol}(\mathcal{L}) = \prod_{i=1}^n \|\mathbf{b}_i^*\| = \|\mathbf{b}_1^*\| \times \|\mathbf{b}_2^*\| \times \cdots \times \|\mathbf{b}_n^*\|$$



Outline

Euclidean n -Space

Lattices

Gram-Schmidt Orthogonalisation

Shortest Vector Problem

Post-Quantum Cryptography

Lattice Reduction

LLL

DeepLLL

Pot-DeepLLL and SS-DeepLLL

A Greedy Global Framework for Lattice Reduction

Pot-GGLLL and SS-GGLLL

Shortest Vector Problem (SVP): Given a basis \mathbf{B} for a lattice \mathcal{L} , find a non-zero vector $\mathbf{x} \in \mathcal{L}$ such that

$$\|\mathbf{x}\| \leq \|\mathbf{y}\|$$

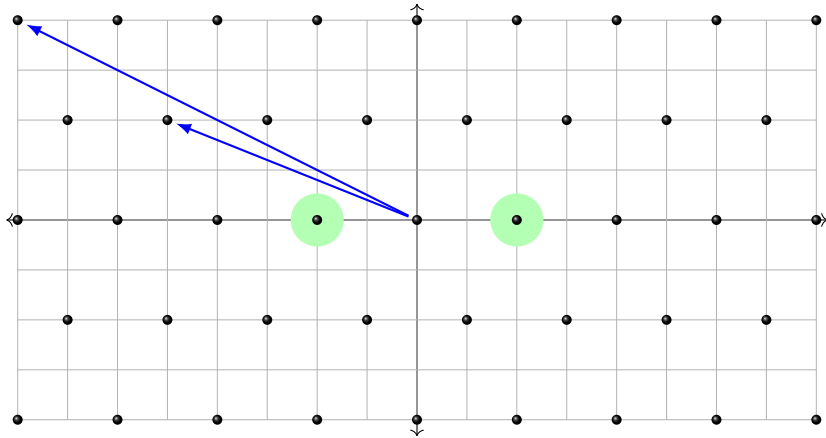
for all non-zero $\mathbf{y} \in \mathcal{L}$.

γ -Approximate SVP (SVP_γ): Given a basis \mathbf{B} for a lattice \mathcal{L} , find a non-zero vector $\mathbf{x} \in \mathcal{L}$ such that

$$\|\mathbf{x}\| \leq \gamma \cdot \lambda_1(\mathcal{L})$$

where $\lambda_1(\mathcal{L})$ denotes the length of the shortest vector in \mathcal{L} .

Shortest Vector Problem



Outline

Euclidean n -Space

Lattices

Gram-Schmidt Orthogonalisation

Shortest Vector Problem

Post-Quantum Cryptography

Lattice Reduction

LLL

DeepLLL

Pot-DeepLLL and SS-DeepLLL

A Greedy Global Framework for Lattice Reduction

Pot-GGLLL and SS-GGLLL

Classical Public-Key Cryptography

- ▶ Classical public key cryptosystems are considered secure by assuming
 - ▶ Integer factorisation problem, and
 - ▶ Discrete logarithm problemare computationally hard.
- ▶ Shor's *quantum* algorithm¹ solves both of these problems in polynomial time
- ▶ With large-scale quantum computers, classical public-key cryptosystems will no longer be secure

¹Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", 1997.

Definition (Post-Quantum Cryptography, PQC)

Cryptographic algorithms which can be implemented on classical computers which are believed to be secure against classical *and* quantum attack.

- ▶ PQC does not utilise any quantum phenomena like entanglement or superposition.
- ▶ Security of PQC is based on problems which are believed to be hard to solve classically *and* quantumly.
- ▶ Many families of post-quantum algorithms:
 - ▶ Lattice-based
 - ▶ Code-based
 - ▶ Isogeny-based
 - ▶ Hash-based
 - ▶ Multivariate cryptography

NIST Post-Quantum Standardisation

- ▶ January 2017: US National Institute of Standards and Technology (NIST) issued a call for proposals for post-quantum cryptosystems
 - ▶ Round 1 (December 2017): 69 algorithms met the minimum criteria to be accepted
 - ▶ 49 encryption schemes, 20 signature schemes
 - ▶ Round 4 (July 2022): 4 algorithms selected for standardisation
 - ▶ 24 August 2023: NIST published initial drafts of Federal Information Processing Standards (FIPS) link
 - ▶ **CRYSTALS-Kyber** - [FIPS 203 link](#)
 - ▶ **CRYSTALS-Dilithium** - [FIPS 204 link](#)
 - ▶ **FALCON** - draft standard to follow in a few months
 - ▶ **SPHINCS⁺** - [FIPS 205 link](#)
 - ▶ 23 November 2023: Last date for comments on the drafts
- ▶ July 2023: NIST called for additional PQC signatures - received 40 submissions
 - ▶ September 2023: [Oxford Summit \(presentation slides and videos\) link](#)

Outline

Euclidean n -Space

Lattices

Gram-Schmidt Orthogonalisation

Shortest Vector Problem

Post-Quantum Cryptography

Lattice Reduction

LLL

DeepLLL

Pot-DeepLLL and SS-DeepLLL

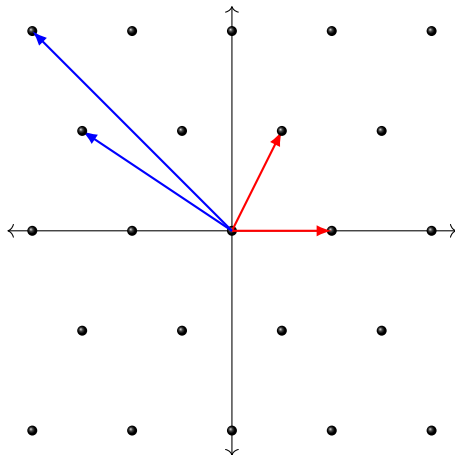
A Greedy Global Framework for Lattice Reduction

Pot-GGLLL and SS-GGLLL

Lattice Basis Reduction

Lattice Basis Reduction

- ▶ Transforms a “bad basis” into a better basis
- ▶ A good basis has shorter, more orthogonal vectors
- ▶ Lattice reduction algorithms:
 - ▶ LLL
 - ▶ BKZ
 - ▶ Slide Reduction
 - ▶ ...



Output Quality parameters: $\|\mathbf{b}_1\|$ and RHF

Definition

Hermite factor (HF) of a basis $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ for a lattice \mathcal{L} is

$$\gamma = \frac{\|\mathbf{b}_1\|}{\text{Vol}(\mathcal{L})^{1/n}}.$$

Root Hermite factor (RHF) of a basis $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ for a lattice \mathcal{L} is

$$\gamma^{1/n} = \left(\frac{\|\mathbf{b}_1\|}{\text{Vol}(\mathcal{L})^{1/n}} \right)^{1/n}$$

- ▶ The HF and RHF relate \mathbf{b}_1 in \mathbf{B} with the volume of the lattice (an invariant of the lattice)
- ▶ The smaller the HF/RHF, the more reduced the basis

Outline

Euclidean n -Space

Lattices

Gram-Schmidt Orthogonalisation

Shortest Vector Problem

Post-Quantum Cryptography

Lattice Reduction

LLL

DeepLLL

Pot-DeepLLL and SS-DeepLLL

A Greedy Global Framework for Lattice Reduction

Pot-GGLLL and SS-GGLLL

The LLL Algorithm

- ▶ Introduced by Lenstra, Lenstra and Lovász in 1982²
- ▶ The first lattice basis reduction algorithm
- ▶ Wide applications in cryptology, algorithmic number theory, factoring polynomials, Diophantine approximation, etc.

LLL solves γ -approximate SVP in polynomial time for γ exponential in n .

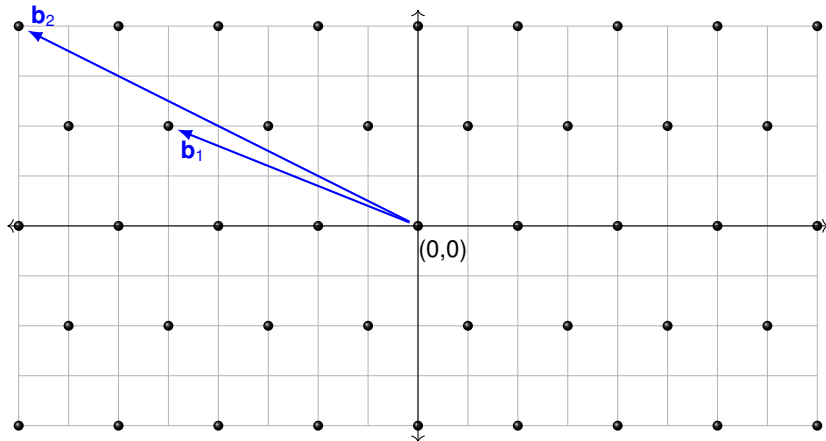
Two steps are applied iteratively to improve the basis:

- ▶ Size reduction of a vector
- ▶ Basis reordering



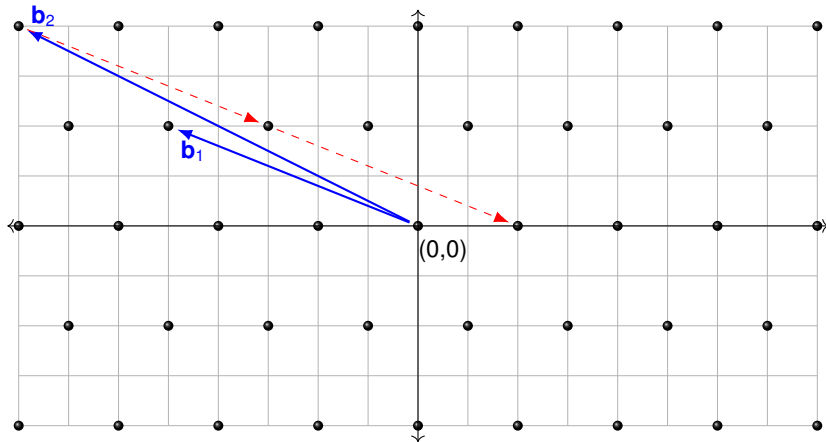
²A. K. Lenstra, H. Lenstra, and Lovász, "Factoring polynomials with rational coefficients", 1982.

Size Reduction: $\mathbf{b}_i \leftarrow \mathbf{b}_i - \lfloor \mu_{i,j} \rfloor \mathbf{b}_j$ where $\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2}$



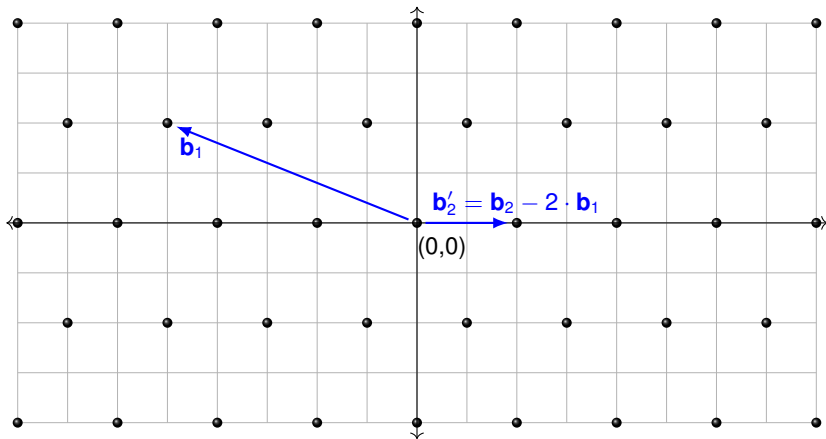
Intuitively, reduce the length of \mathbf{b}_i as much as possible by $\pm \mathbf{b}_j$

Size Reduction: $\mathbf{b}_i \leftarrow \mathbf{b}_i - \lfloor \mu_{i,j} \rfloor \mathbf{b}_j$ where $\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2}$



Intuitively, reduce the length of \mathbf{b}_i as much as possible by $\pm \mathbf{b}_j$

Size Reduction: $\mathbf{b}_i \leftarrow \mathbf{b}_i - \lfloor \mu_{i,j} \rfloor \mathbf{b}_j$ where $\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2}$



Intuitively, reduce the length of \mathbf{b}_i as much as possible by $\pm \mathbf{b}_j$

Basis Reordering - The Lovász Condition

Lovász condition

is a check to see if consecutive vectors should be swapped

$$(\delta - \mu_{k,k-1}^2) \cdot \|\mathbf{b}_{k-1}^*\|^2 \leq \|\mathbf{b}_k^*\|^2$$

for $1/4 < \delta \leq 1$.

Swap $\sigma_{k-1,k}(\mathbf{B})$ if Lovász condition fails

i.e. if:

$$(\delta - \mu_{k,k-1}^2) \cdot \|\mathbf{b}_{k-1}^*\|^2 > \|\mathbf{b}_k^*\|^2$$

The Lovász Condition: Intuition

$$\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_{k-2}, \mathbf{b}_{k-1}, \mathbf{b}_k, \mathbf{b}_{k+1}, \dots, \mathbf{b}_n)$$

Lovász condition

is a check to see if consecutive vectors should be swapped

$$\delta \cdot \left\| \underbrace{\pi_{k-1}(\mathbf{b}_{k-1})}_{\text{Projection of } \mathbf{b}_{k-1} \text{ orthogonal to } \text{span}(\mathbf{b}_1, \dots, \mathbf{b}_{k-2})} \right\|^2 \leq \left\| \underbrace{\pi_{k-1}(\mathbf{b}_k)}_{\text{Projection of } \mathbf{b}_k \text{ orthogonal to } \text{span}(\mathbf{b}_1, \dots, \mathbf{b}_{k-2})} \right\|^2$$

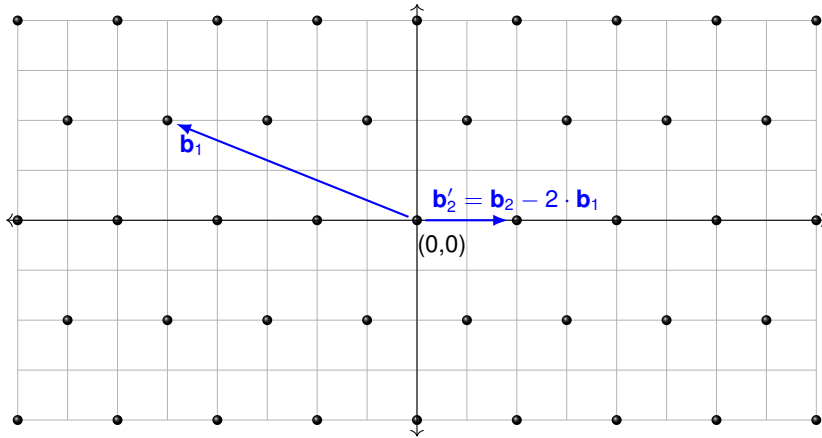
for $1/4 < \delta \leq 1$.

Swap $\sigma_{k-1,k}(\mathbf{B})$ if Lovász condition fails

i.e. if:

$$\delta \cdot \|\pi_{k-1}(\mathbf{b}_{k-1})\|^2 > \|\pi_{k-1}(\mathbf{b}_k)\|^2$$

Swapping Basis Vectors: $\sigma_{k-1,k}(\mathbf{B})$



We would swap the vectors \mathbf{b}_1 and \mathbf{b}'_2 in this case
as there is scope to reduce \mathbf{b}_1 further

The LLL Algorithm

Algorithm 1: The LLL Algorithm

Input: A basis $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$, a value $1/4 < \delta \leq 1$

Output: A basis $\mathbf{B}' = (\mathbf{b}'_1, \dots, \mathbf{b}'_n)$ which is δ -LLL reduced

Find the GSO basis \mathbf{B}^* and initialise the values of $\mu_{i,j}$

$k \leftarrow 2$

while $k \leq n$ **do**

 Size reduce \mathbf{b}_k

if $\delta \cdot \|\pi_{k-1}(\mathbf{b}_{k-1})\|^2 > \|\pi_{k-1}(\mathbf{b}_k)\|^2$ **then**

 Swap vectors \mathbf{b}_k and \mathbf{b}_{k-1}

 Update \mathbf{b}_{k-1}^* , \mathbf{b}_k^*

$k \leftarrow \max(k-1, 2)$

else

$k \leftarrow k+1$

return \mathbf{B}' , a δ -LLL reduced basis

²A. K. Lenstra, H. Lenstra, and Lovász, “Factoring polynomials with rational coefficients”, 1982.

Tracking the index k in LLL

\mathbf{b}_1 \mathbf{b}_2 \mathbf{b}_3 \mathbf{b}_4 \mathbf{b}_5 \mathbf{b}_6 \mathbf{b}_7 \mathbf{b}_8 \mathbf{b}_9 \mathbf{b}_{10}

- ▶ Basis for a lattice
- ▶ Dimension $n = 10$

Tracking the index k in LLL



- ▶ Size reduce \mathbf{b}_2 with (\mathbf{b}_1)
- ▶ Check to see if \mathbf{b}_1 and \mathbf{b}_2 should be swapped
 - ▶ Let

$$\delta \cdot \|\pi_1(\mathbf{b}_1)\|^2 \leq \|\pi_1(\mathbf{b}_2)\|^2$$

- ▶ So swap $\sigma_{1,2}(\mathbf{B})$ does not happen

Tracking the index k in LLL



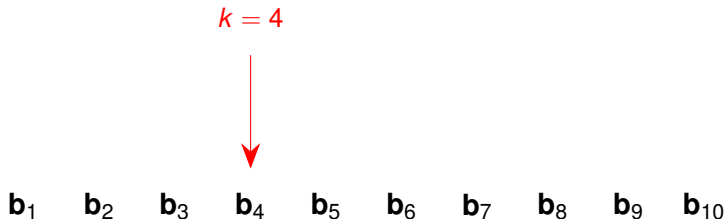
- ▶ Size reduce \mathbf{b}_3 with $(\mathbf{b}_2, \mathbf{b}_1)$
- ▶ Check to see if \mathbf{b}_2 and \mathbf{b}_3 should be swapped

- ▶ Let

$$\delta \cdot \|\pi_2(\mathbf{b}_2)\|^2 \leq \|\pi_2(\mathbf{b}_3)\|^2$$

- ▶ So swap $\sigma_{2,3}(\mathbf{B})$ does not happen

Tracking the index k in LLL

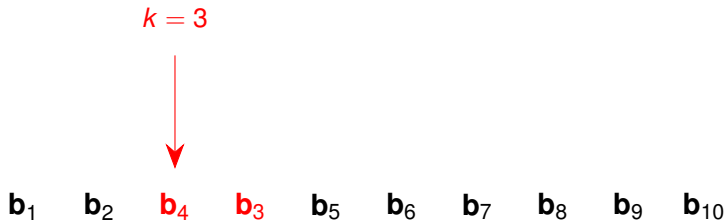


- ▶ Size reduce \mathbf{b}_4 with $(\mathbf{b}_3, \mathbf{b}_2, \mathbf{b}_1)$
- ▶ Check to see if \mathbf{b}_3 and \mathbf{b}_4 should be swapped
 - ▶ Let

$$\delta \cdot \|\pi_3(\mathbf{b}_3)\|^2 > \|\pi_3(\mathbf{b}_4)\|^2$$

- ▶ So swap $\sigma_{3,4}(\mathbf{B})$ DOES happen
because it will reduce the length of the GSO vector at position 3
(the new $\|\mathbf{b}_3^*\|$ after the swap will be less than the old $\|\mathbf{b}_3^*\|$ before the swap)

Tracking the index k in LLL



- ▶ Size reduce \mathbf{b}_4 (now in position 3) with \mathbf{b}_2 and \mathbf{b}_1
- ▶ Check to see whether \mathbf{b}_4 (in position 3) should be swapped with \mathbf{b}_2

Continue the process until you reach $k = n + 1 = 11$, at which point it is a:

δ -LLL reduced basis:

- ▶ all \mathbf{b}_k are size reduced, and
- ▶ all pairs $\mathbf{b}_{k-1}, \mathbf{b}_k$ satisfy the Lovász condition.

$$\delta \cdot \|\pi_{k-1}(\mathbf{b}_{k-1})\|^2 \leq \|\pi_{k-1}(\mathbf{b}_k)\|^2.$$

Outline

Euclidean n -Space

Lattices

Gram-Schmidt Orthogonalisation

Shortest Vector Problem

Post-Quantum Cryptography

Lattice Reduction

LLL

DeepLLL

Pot-DeepLLL and SS-DeepLLL

A Greedy Global Framework for Lattice Reduction

Pot-GGLLL and SS-GGLLL

Using Deep Insertions instead of (Many) Swaps

Schnorr and Euchner³ introduced **DeepLLL** and **BKZ** in the same 1994 paper

DeepLLL: LLL with Deep Insertions

- ▶ At index k , DeepLLL tries to
insert \mathbf{b}_k in the earliest position i where the following fails
(insertion will reduce $\|\mathbf{b}_i^\|$)*

$$\delta \cdot \|\pi_i(\mathbf{b}_i)\|^2 \leq \|\pi_i(\mathbf{b}_k)\|^2$$

- ▶ $\sigma_{i,k}(\mathbf{B})$ denotes the deep insertion of \mathbf{b}_k at position i

N.B. **BKZ** is the current state-of-the-art lattice reduction algorithm

³Schnorr and Euchner, "Lattice basis reduction: improved practical algorithms and solving subset sum problems", 1994.

The Generalised Lovász Condition: Novel Interpretation (Local Measure of Quality)

$$\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_{i-1}, \mathbf{b}_i, \mathbf{b}_{i+1}, \dots, \mathbf{b}_{k-1}, \mathbf{b}_k, \mathbf{b}_{k+1}, \dots, \mathbf{b}_n)$$

Lovász condition

is a check to see if \mathbf{b}_k should be (deep) inserted before \mathbf{b}_i

$$\delta \cdot \left\| \underbrace{\pi_i(\mathbf{b}_i)}_{\text{Projection of } \mathbf{b}_i \text{ orthogonal to } \text{span}(\mathbf{b}_1, \dots, \mathbf{b}_{i-1})} \right\|^2 \leq \left\| \underbrace{\pi_i(\mathbf{b}_k)}_{\text{Projection of } \mathbf{b}_k \text{ orthogonal to } \text{span}(\mathbf{b}_1, \dots, \mathbf{b}_{i-1})} \right\|^2$$

for $1/4 < \delta \leq 1$.

Deep insert \mathbf{b}_k before \mathbf{b}_i if the generalised Lovász condition fails

i.e. if:

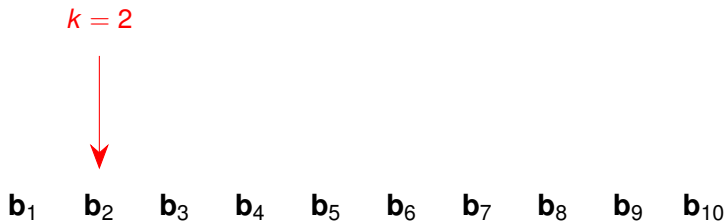
$$\delta \cdot \|\pi_i(\mathbf{b}_i)\|^2 > \|\pi_i(\mathbf{b}_k)\|^2$$

Tracking the index k in DeepLLL

\mathbf{b}_1 \mathbf{b}_2 \mathbf{b}_3 \mathbf{b}_4 \mathbf{b}_5 \mathbf{b}_6 \mathbf{b}_7 \mathbf{b}_8 \mathbf{b}_9 \mathbf{b}_{10}

- ▶ Basis for a lattice
- ▶ Dimension $n = 10$

Tracking the index k in DeepLLL

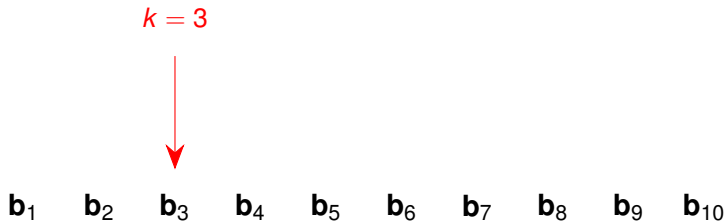


- ▶ Size reduce \mathbf{b}_2 with (\mathbf{b}_1)
- ▶ Check to see if an insertion of \mathbf{b}_2 in position 1 may occur
 - ▶ Assume insertion $\sigma_{1,2}(\mathbf{B})$ does not happen

$$\delta \cdot \|\pi_i(\mathbf{b}_i)\|^2 \leq \|\pi_i(\mathbf{b}_k)\|^2$$

for all $i < 2$

Tracking the index k in DeepLLL

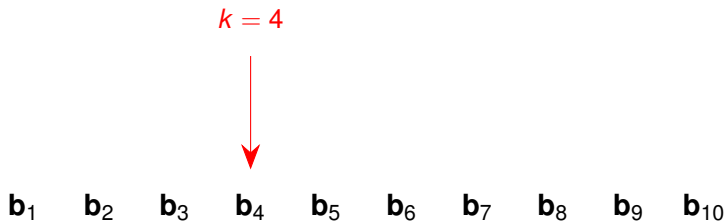


- ▶ Size reduce \mathbf{b}_3 with $(\mathbf{b}_2, \mathbf{b}_1)$
- ▶ Check to see whether inserting \mathbf{b}_3 in position 1, then in position 2, may occur
 - ▶ Assume neither insertions $\sigma_{1,3}(\mathbf{B})$ or $\sigma_{2,3}(\mathbf{B})$ happen

$$\delta \cdot \|\pi_i(\mathbf{b}_i)\|^2 \leq \|\pi_i(\mathbf{b}_k)\|^2$$

for all $i < 3$

Tracking the index k in DeepLLL



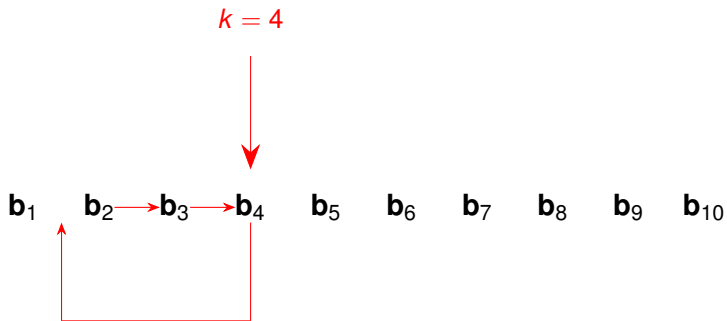
- ▶ Size reduce \mathbf{b}_4 with $(\mathbf{b}_3, \mathbf{b}_2, \mathbf{b}_1)$
- ▶ Check to see whether inserting \mathbf{b}_4 in position 1, then 2, then 3 may occur
 - ▶ Assume earliest possible insertion of \mathbf{b}_4 is in position 2

$$\delta \cdot \|\pi_i(\mathbf{b}_i)\|^2 > \|\pi_i(\mathbf{b}_k)\|^2$$

first for $i = 2$

- ▶ $\sigma_{2,4}(\mathbf{B})$: Deep insert \mathbf{b}_4 at position 2

Tracking the index k in DeepLLL



- ▶ Size reduce \mathbf{b}_4 with $(\mathbf{b}_3, \mathbf{b}_2, \mathbf{b}_1)$
- ▶ Check to see whether inserting \mathbf{b}_4 in position 1, then 2, then 3 may occur
 - ▶ Assume earliest possible insertion of \mathbf{b}_4 is in position 2

$$\delta \cdot \|\pi_i(\mathbf{b}_i)\|^2 > \|\pi_i(\mathbf{b}_k)\|^2$$

first for $i = 2$

- ▶ $\sigma_{2,4}(\mathbf{B})$: Deep insert \mathbf{b}_4 at position 2

Tracking the index k in DeepLLL



Continue the process until $k = n + 1 = 11$ at which point, it is a:

δ -DeepLLL reduced basis:

- ▶ all \mathbf{b}_k are size reduced, and
- ▶ all pairs $\mathbf{b}_i, \mathbf{b}_k$ satisfy the generalised Lovász condition.

$$\delta \cdot \|\pi_i(\mathbf{b}_i)\|^2 \leq \|\pi_i(\mathbf{b}_k)\|^2.$$

DeepLLL returns a basis which is of much better quality than LLL
 \Rightarrow The vectors are generally shorter and more orthogonal.

DeepLLL returns a basis which is of much better quality than LLL
 \Rightarrow The vectors are generally shorter and more orthogonal.

However there is no known bound for the efficiency of DeepLLL
It is thought to be potentially super-exponential time!

Outline

Euclidean n -Space

Lattices

Gram-Schmidt Orthogonalisation

Shortest Vector Problem

Post-Quantum Cryptography

Lattice Reduction

LLL

DeepLLL

Pot-DeepLLL and SS-DeepLLL

A Greedy Global Framework for Lattice Reduction

Pot-GGLLL and SS-GGLLL

Variants of DeepLLL: Using Global Measures

Pot-DeepLLL^a

^aFontein, Schneider, and Wagner, “PotLLL: a polynomial time version of LLL with deep insertions”, 2014.

Insert in the earliest position which reduces the *potential* (Pot) of the basis

$$\text{Pot}(\mathbf{B}) = \prod_{i=1}^n \|\mathbf{b}_i^*\|^{2(n-i+1)}$$

SS-DeepLLL^a

^aYasuda and Yamaguchi, “A new polynomial-time variant of LLL with deep insertions for decreasing the squared-sum of Gram–Schmidt lengths”, 2019.

Insert in the earliest position which reduces the *squared sum* (SS) of the basis

$$\text{SS}(\mathbf{B}) = \sum_{i=1}^n \|\mathbf{b}_i^*\|^2$$

Both of these variants are provably **polynomial time**,
but return bases of lesser quality than DeepLLL

Outline

Euclidean n -Space

Lattices

- Gram-Schmidt Orthogonalisation
- Shortest Vector Problem

Post-Quantum Cryptography

Lattice Reduction

- LLL
- DeepLLL
- Pot-DeepLLL and SS-DeepLLL

A Greedy Global Framework for Lattice Reduction

- Pot-GGLLL and SS-GGLLL

A Greedy Global Framework for LLL

Sanjay Bhattacharjee¹, Julio Hernandez-Castro¹, Jack Moyler^{1*}

^{1*}Institute of Cyber Security for Society and School of Computing,
University of Kent, Canterbury, CT2 7NP, Kent, United Kingdom.

*Corresponding author(s). E-mail(s): jdm58@kent.ac.uk;
Contributing authors: s.bhattacharjee@kent.ac.uk;
j.c.hernandez-castro@kent.ac.uk;

Abstract

LLL-style lattice reduction algorithms iteratively employ size reduction and reordering on ordered basis vectors to find progressively shorter, more orthogonal vectors. These algorithms work with a designated measure of basis quality and perform reordering by inserting a vector in an earlier position depending on the basis quality before and after reordering. **DeepLLL** was introduced alongside the **BKZ** reduction algorithm, however the latter has emerged as the state-of-the-art and has therefore received greater attention. We first show that **LLL**-style algorithms iteratively improve a basis quality measure; specifically that **DeepLLL** improves a sublattice measure based on the generalised Lovász condition. We then introduce a new generic framework for lattice reduction algorithms, working with some quality measure X . We instantiate our framework with two quality measures - basis potential (**Pot**) and squared sum (**SS**) - both of which have corresponding **DeepLLL** algorithms. We prove polynomial runtimes for our X -**GGLL** algorithms and guarantee their output quality. We run two types of experiments (implementations provided publicly) to compare performances of **LLL**, **X-DeepLLL**, **X-GGLL**; with multi-precision arithmetic using overestimated floating point precision for *standalone* comparison with no preprocessing, and with standard datatypes using *LLL-preprocessed* inputs. In preprocessed comparison, we also compare with **BKZ**. In standalone comparison, our **GGLL** algorithms produce better quality bases whilst being much faster than the corresponding **DeepLLL** versions. The runtime of **SS-GGLL** is only second to **LLL** in our standalone comparison. **SS-GGLL** is significantly faster than the **FPLLL** implementation of **BKZ-12** at all dimensions and outputs better quality bases dimensions **100** onward.

Keywords: Lattice reduction, LLL, DeepLLL, greedy global framework, potential, squared sum.

Draft: <https://eprint.iacr.org/2023/261>



A Greedy Global Approach

- ▶ We propose a new framework for DeepLLL-style lattice reduction algorithms
- ▶ We no longer maintain an index k of the vector being considered at each iteration
- ▶ Instead, we

*Globally search over the entire basis for the “best” insertion
to move any one vector to an earlier index*

- ▶ This framework may be instantiated with *a measure X associated with the basis*
E.g. Potential, Squared-Sum
- ▶ The “best” insertion is the one which results in the greatest change in the measure being considered
i.e. the *greedy* choice
- ▶ For a measure X , we call this the X -GGLLL algorithm

The Greedy Global Framework: X-GLLL Algorithm

\mathbf{b}_1 \mathbf{b}_2 \mathbf{b}_3 \mathbf{b}_4 \mathbf{b}_5 \mathbf{b}_6 \mathbf{b}_7 \mathbf{b}_8 \mathbf{b}_9 \mathbf{b}_{10}

- ▶ Basis for a lattice
- ▶ Dimension $n = 10$
- ▶ Size reduce all \mathbf{b}_i

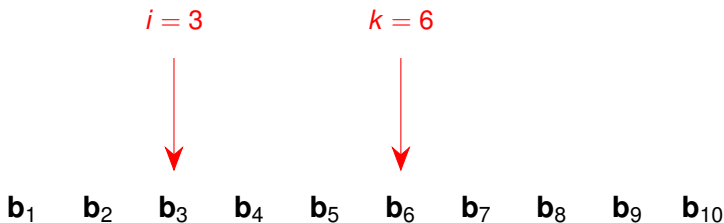
The Greedy Global Framework: X-GGLLL Algorithm

- ▶ Search indices k and i for which the measure X (Pot/SS) reduces most (ΔX is maximised)
- ▶ Found indices $k = 6$ and $i = 3$ for which ΔX is maximum
- ▶ Deep insert \mathbf{b}_6 at position 3, shifting up $\mathbf{b}_3, \mathbf{b}_4, \mathbf{b}_5$ by one place each
- ▶ Size reduce \mathbf{b}'_4, \dots with all their previous vectors starting from position 3 onward

\mathbf{b}_1 \mathbf{b}_2 \mathbf{b}_3 \mathbf{b}_4 \mathbf{b}_5 \mathbf{b}_6 \mathbf{b}_7 \mathbf{b}_8 \mathbf{b}_9 \mathbf{b}_{10}

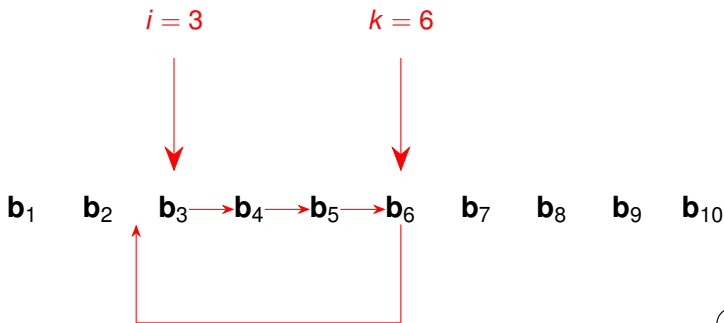
The Greedy Global Framework: X-GGLLL Algorithm

- ▶ Search indices k and i for which the measure X (Pot/SS) reduces most (ΔX is maximised)
- ▶ Found indices $k = 6$ and $i = 3$ for which ΔX is maximum
- ▶ Deep insert \mathbf{b}_6 at position 3, shifting up $\mathbf{b}_3, \mathbf{b}_4, \mathbf{b}_5$ by one place each
- ▶ Size reduce \mathbf{b}'_4, \dots with all their previous vectors starting from position 3 onward



The Greedy Global Framework: X-GGLLL Algorithm

- ▶ Search indices k and i for which the measure X (Pot/SS) reduces most (ΔX is maximised)
- ▶ Found indices $k = 6$ and $i = 3$ for which ΔX is maximum
- ▶ Deep insert \mathbf{b}_6 at position 3, shifting up $\mathbf{b}_3, \mathbf{b}_4, \mathbf{b}_5$ by one place each
- ▶ Size reduce \mathbf{b}'_4, \dots with all their previous vectors starting from position 3 onward



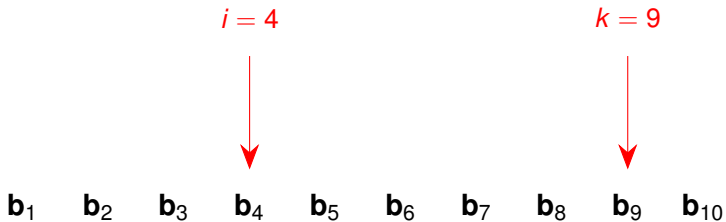
The Greedy Global Framework: X-GGLLL Algorithm

- ▶ Search indices k and i for which the measure X (Pot/SS) reduces most (ΔX is maximised)
- ▶ Found indices $k = 9$ and $i = 4$ for which ΔX is maximum
- ▶ Deep insert \mathbf{b}_9 at position 4, shifting up $\mathbf{b}_4, \dots, \mathbf{b}_8$ by one place each
- ▶ Size reduce \mathbf{b}'_5, \dots with all their previous vectors starting from position 4 onward

\mathbf{b}_1 \mathbf{b}_2 \mathbf{b}_3 \mathbf{b}_4 \mathbf{b}_5 \mathbf{b}_6 \mathbf{b}_7 \mathbf{b}_8 \mathbf{b}_9 \mathbf{b}_{10}

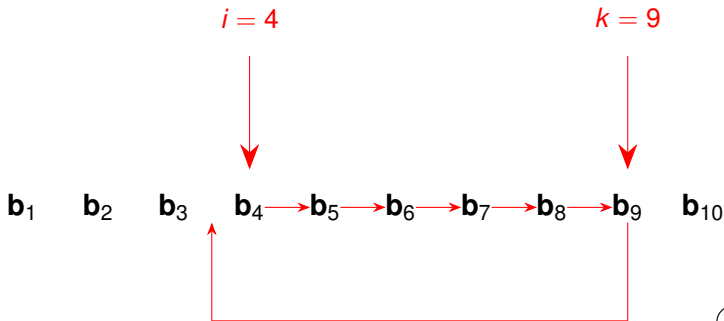
The Greedy Global Framework: X-GGLLL Algorithm

- ▶ Search indices k and i for which the measure X (Pot/SS) reduces most (ΔX is maximised)
- ▶ Found indices $k = 9$ and $i = 4$ for which ΔX is maximum
- ▶ Deep insert \mathbf{b}_9 at position 4, shifting up $\mathbf{b}_4, \dots, \mathbf{b}_8$ by one place each
- ▶ Size reduce \mathbf{b}'_5, \dots with all their previous vectors starting from position 4 onward



The Greedy Global Framework: X-GGLLL Algorithm

- ▶ Search indices k and i for which the measure X (Pot/SS) reduces most (ΔX is maximised)
- ▶ Found indices $k = 9$ and $i = 4$ for which ΔX is maximum
- ▶ Deep insert \mathbf{b}_9 at position 4, shifting up $\mathbf{b}_4, \dots, \mathbf{b}_8$ by one place each
- ▶ Size reduce \mathbf{b}'_5, \dots with all their previous vectors starting from position 4 onward



The Greedy Global Framework: X -GGLL Algorithm

- ▶ Search indices k and i for which the measure X (Pot/SS) reduces most (ΔX is maximised)
- ▶ Found indices $k = 3$ and $i = 1$ for which ΔX is maximum
- ▶ Deep insert \mathbf{b}_3 at position 1, shifting up $\mathbf{b}_1, \mathbf{b}_2$ by one place each
- ▶ Size reduce \mathbf{b}'_2, \dots with all their previous vectors starting from position 1 onward

\mathbf{b}_1 \mathbf{b}_2 \mathbf{b}_3 \mathbf{b}_4 \mathbf{b}_5 \mathbf{b}_6 \mathbf{b}_7 \mathbf{b}_8 \mathbf{b}_9 \mathbf{b}_{10}

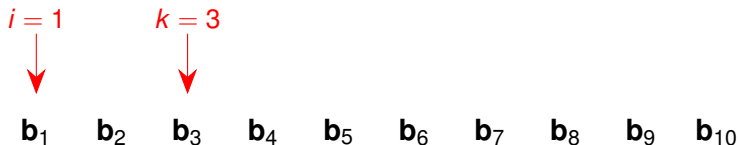
Continue the process until the basis is a:

δ - X -DeepLLL reduced basis:

- ▶ all \mathbf{b}_k are size reduced, and
- ▶ for all $1 \leq i < k \leq n$, $\delta \cdot X(\mathbf{B}) \leq X(\sigma_{i,k}(\mathbf{B}))$

The Greedy Global Framework: X-GGLLL Algorithm

- ▶ Search indices k and i for which the measure X (Pot/SS) reduces most (ΔX is maximised)
- ▶ Found indices $k = 3$ and $i = 1$ for which ΔX is maximum
- ▶ Deep insert \mathbf{b}_3 at position 1, shifting up $\mathbf{b}_1, \mathbf{b}_2$ by one place each
- ▶ Size reduce \mathbf{b}'_2, \dots with all their previous vectors starting from position 1 onward



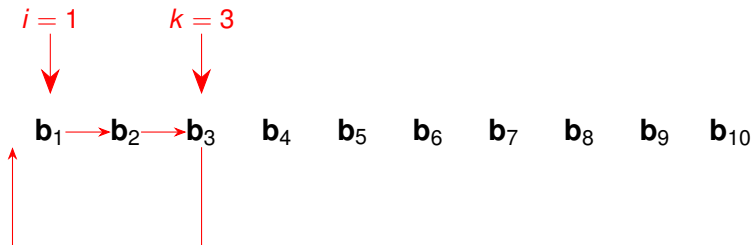
Continue the process until the basis is a:

δ -X-DeepLLL reduced basis:

- ▶ all \mathbf{b}_k are size reduced, and
- ▶ for all $1 \leq i < k \leq n$, $\delta \cdot X(\mathbf{B}) \leq X(\sigma_{i,k}(\mathbf{B}))$

The Greedy Global Framework: X-GGLLL Algorithm

- ▶ Search indices k and i for which the measure X (Pot/SS) reduces most (ΔX is maximised)
- ▶ Found indices $k = 3$ and $i = 1$ for which ΔX is maximum
- ▶ Deep insert \mathbf{b}_3 at position 1, shifting up $\mathbf{b}_1, \mathbf{b}_2$ by one place each
- ▶ Size reduce \mathbf{b}'_2, \dots with all their previous vectors starting from position 1 onward



Continue the process until the basis is a:

δ -X-DeepLLL reduced basis:

- ▶ all \mathbf{b}_k are size reduced, and
- ▶ for all $1 \leq i < k \leq n$, $\delta \cdot X(\mathbf{B}) \leq X(\sigma_{i,k}(\mathbf{B}))$

The X-GGLLL Algorithm

Algorithm 2: The X-GGLLL Algorithm

Input: A basis $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$, a threshold $0 < \delta \leq 1$

Output: $\mathbf{B}' = (\mathbf{b}'_1, \dots, \mathbf{b}'_n)$ which is a δ -X-GGLLL reduced basis

Find the GSO basis \mathbf{B}^* and initialise the values of $\mu_{i,j}$

Size reduce $\mathbf{b}_2, \dots, \mathbf{b}_n$ in this order

Find (i', k') such that $(i', k') = \operatorname{argmax}_{1 \leq i < k \leq n} (\Delta X_{i,k})$ and set $\Delta X = \Delta X_{i',k'}$

while $\Delta X > (1 - \delta) \cdot X(\mathbf{B})$ **do**

 Deep insert $\mathbf{b}_{k'}$ before $\mathbf{b}_{i'}$

 Update \mathbf{B}^* and $\mu_{i,j}$

 Size reduce $\mathbf{b}_{i'+1}, \dots, \mathbf{b}_n$

 Find (i', k') such that $(i', k') = \operatorname{argmax}_{1 \leq i < k \leq n} (\Delta X_{i,k})$ and set $\Delta X = \Delta X_{i',k'}$

end

return \mathbf{B}' , a δ -X-DeepLLL reduced basis.

X-DeepLLL vs X-GGLLL

X-DeepLLL

- ▶ Local choice: Only considers a sublattice \mathcal{L}_k in each iteration
- ▶ Maintains k across iterations
- ▶ First choice: Chooses the earliest i in \mathcal{L}_k
- ▶ Size reduction: only in \mathcal{L}_k in every iteration
- ▶ Needs to search $\mathcal{O}(n)$ indices i for \mathbf{b}_k in each iteration

X-GGLLL

- ▶ Global choice: Considers the whole lattice \mathcal{L} in each iteration
- ▶ Chooses k independently in each iteration
- ▶ Greedy choice: Chooses the best (i, k) in \mathcal{L}_k
 \implies Fewer iterations of the loop
- ▶ Size reduction: whole \mathcal{L} in every iteration
- ▶ Need to search $\mathcal{O}(n^2)$ pairs (i, k) to find the best insertion

Outline

Euclidean n -Space

Lattices

- Gram-Schmidt Orthogonalisation
- Shortest Vector Problem

Post-Quantum Cryptography

Lattice Reduction

- LLL
- DeepLLL
- Pot-DeepLLL and SS-DeepLLL

A Greedy Global Framework for Lattice Reduction

- Pot-GGLLL and SS-GGLLL

Theoretical Results (for Pot-GGLL and SS-GGLL)

The X -GGLL algorithm has a bit complexity of

$$\mathcal{O} \left(\underbrace{\left(\underbrace{n^4 \log^2 C}_{\text{deep insertion}} + \underbrace{mn^4 \log^2 C}_{\text{size reduction}} + \underbrace{f_X(C, m, n)}_{\text{index search}} \right)}_{\text{\#iterations}} \log_{1/\delta} \left(\frac{I_X}{Z_X} \right) \right)$$

where

- ▶ we assume exact \mathbb{Q} arithmetic (but without fast integer arithmetic)
- ▶ n = dimension of lattice
- ▶ m = rank of lattice ($m = n$ in our experiments)
- ▶ C is such that $\|\mathbf{b}_i\|^2 \leq C$ for all i
- ▶ Z_X and I_X denote the lower and upper bounds respectively for the measure $X(\mathbf{B})$
- ▶ $\frac{1}{4} < \delta \leq 1$ - the closer to 1, the stronger the reduction







Theoretical Results (for Pot-GGLLL and SS-GGLLL)

Complexity comparison of X-DeepLLL and X-GGLLL

Algorithm Name	Deep Insertion	Size Reduction	Index Search	Number of Iterations
Pot-DeepLLL	$mn^3 \log^2 C$	$mn^3 \log^2 C$	$n^4 \log^2 C$	$n^3 \log_{1/\delta} C$
Pot-GGLLL	$n^4 \log^2 C$	$mn^4 \log^2 C$	$n^5 \log^2 C$	$n^2 \log_{1/\delta} C$
SS-DeepLLL	$mn^3 \log^2 C$	$mn^3 \log^2 C$	$n^3 \log^2 C$	$n \log_{1/\delta} C$
SS-GGLLL	$n^4 \log^2 C$	$mn^4 \log^2 C$	$n^4 \log^2 C$	$\log_{1/\delta} C$

The X-GGLLL algorithm achieves the same asymptotic bit-complexity as X-DeepLLL

References

-  Fontein, Felix, Michael Schneider, and Urs Wagner. “PotLLL: a polynomial time version of LLL with deep insertions”. In: *Designs, Codes and Cryptography* 73.2 (2014), pp. 355–368. DOI: [10.1007/s10623-014-9918-8](https://doi.org/10.1007/s10623-014-9918-8).
-  Lenstra, Arjen K., Hendrik Lenstra, and László Lovász. “Factoring polynomials with rational coefficients”. In: *Math. Ann.* 261 (1982), pp. 515–534.
-  Schnorr, Claus-Peter and Martin Euchner. “Lattice basis reduction: improved practical algorithms and solving subset sum problems”. In: *Mathematical programming* 66.1 (1994), pp. 181–199.
-  Shor, P. W. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509.
-  The FPLLL development team. “fplll, a lattice reduction library, Version: 5.4.4”. Available at <https://github.com/fplll/fplll>. 2023.
-  Yasuda, Masaya and Junpei Yamaguchi. “A new polynomial-time variant of LLL with deep insertions for decreasing the squared-sum of Gram–Schmidt lengths”. In: *Designs, Codes and Cryptography* 87.11 (2019), pp. 2489–2505. DOI: [10.1007/s10623-019-00634-9](https://doi.org/10.1007/s10623-019-00634-9).



Thank you for your kind attention!