Lattice Reduction

Introduction

An Invariant: GSO

LLL Algorithm

Code Reduction

Orthopodality

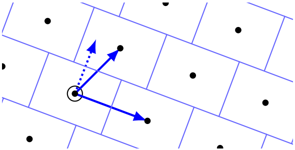
Babai Algorithm for Codes

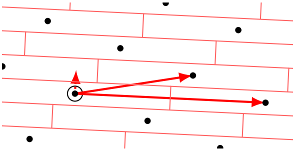
LLL Reduction and LLL Algorithm for Binary Code

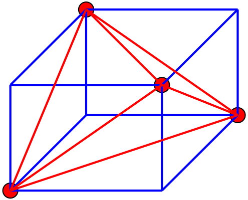
Griesmer's Bound versus LLL

An Algorithmic Reduction Theory for Binary Codes: LLL and more

Thomas Debris-Alazard, Léo Ducas, Wessel P.J. van Woerden







This work

Analogies (definition, proposition, theorem) from Lattices to Codes
via an algorithmic approach (LLL)

We propose a reduction theory for codes (LLL-reduced bases):

1. Proof of bound on codes (Griesmer...)
2. Use to speed-up cryptanalytic algorithms

This work

Analogies (definition, proposition, theorem) from Lattices to Codes
via an algorithmic approach (LLL)

We propose a reduction theory for codes (LLL-reduced bases):

1. Proof of bound on codes (Griesmer...)
2. Use to speed-up cryptanalytic algorithms

A very good reference to learn about lattices
https://homepages.cwi.nl/~dadush/teaching/lattices-2018/

1 Lattice Reduction
- Introduction
- An Invariant: GSO
- LLL Algorithm

2 Code Reduction
- Orthopodality
- Babai Algorithm for Codes
- LLL Reduction and LLL Algorithm for Binary Code
- Griesmer's Bound versus LLL

1 Lattice Reduction
Introduction
An Invariant: GSO
LLL Algorithm

2 Code Reduction
Orthopodality
Babai Algorithm for Codes
LLL Reduction and LLL Algorithm for Binary Code
Griesmer's Bound versus LLL

Lattices

Lattice: $\mathcal{L} \subset \mathbb{R}^n$ discrete subgroup
equipped with Euclidean metric $\|\cdot\|$.

Basis of $\mathcal{L}$ (*full-rank* lattice): $\mathrm{B} \stackrel{\text{def}}{=} (\mathrm{b}_1, \ldots, \mathrm{b}_n)$ such that,

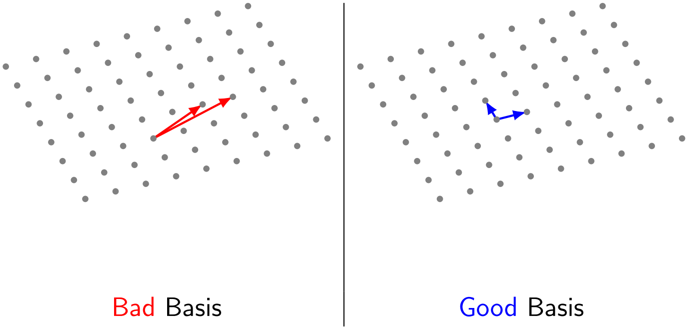
1. Linearly independent (over $\mathbb{R}$),
2. Span $\mathcal{L}$ over $\mathbb{Z}$,

$$\mathcal{L} = \mathrm{Span}_{\mathbb{Z}}(\mathrm{B}) \stackrel{\text{def}}{=} \left\{ \sum_{i=1}^{n} \lambda_i \mathrm{b}_i \ : \ \lambda_i \in \mathbb{Z} \right\}.$$

$$\lambda_1(\mathcal{L}) \stackrel{\text{def}}{=} \min_{\mathrm{x} \in \mathcal{L} \setminus \{0\}} \|\mathrm{x}\|$$

Aim of reduction: find good bases!

Good Versus Bad

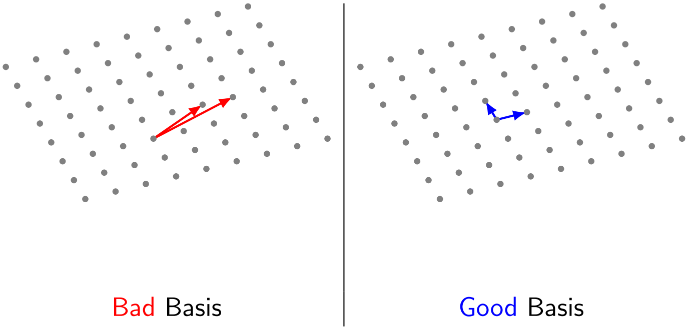


Bad Basis | Good Basis

1. Why the basis is good or not?

2. How to obtain a good basis?

Good Versus Bad



Bad Basis | Good Basis

1. Why the basis is good or not?

   $\rightarrow$ Invariants of a basis, Babai Algorithm...

2. How to get a good basis?

   $\rightarrow$ Lagrange reduction, LLL algorithm...

1 Lattice Reduction
Introduction
An Invariant: GSO
LLL Algorithm

2 Code Reduction
Orthopodality
Babai Algorithm for Codes
LLL Reduction and LLL Algorithm for Binary Code
Griesmer's Bound versus LLL

An Invariant

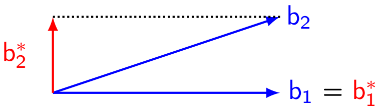
B and B′ are bases of the same lattices *if and only if*,

$$\exists \mathrm{U} \in \mathrm{GL}_n(\mathbb{Z}) \quad : \quad \mathrm{B}' = \mathrm{UB}.$$

$\det(\mathcal{L}) \stackrel{\text{def}}{=} |\det(\mathrm{BB}^{\intercal})|$ is an invariant of $\mathcal{L}$!

Gram-Schmidt Ortholpo. (GSO)

$b_1, \ldots, b_n$ basis of $\mathcal{L}$.



- $b_1^* \stackrel{\text{def}}{=} b_1$
- Projection orthogonal to $\text{Span}_{\mathbb{R}}(b_1^*, \ldots, b_{i-1}^*)$,

$$b_i^* \stackrel{\text{def}}{=} \pi_i(b_i) \quad \text{where} \quad \pi_i(b_i) \stackrel{\text{def}}{=} b_i - \sum_{j<i} \frac{\langle b_i, b_j \rangle}{\|b_j\|^2} b_j^*$$

$(b_1^*, \ldots, b_n^*)$ is not a basis of $\mathcal{L}$... but:

$$\det(\mathcal{L}) = \prod_i \|b_i^*\| \quad \text{and} \quad \text{Span}_{\mathbb{R}}(\mathcal{L}) = \text{Span}_{\mathbb{R}}(b_1^*, \ldots, b_n^*).$$

Decrease First Length Vector

$$\det(\mathcal{L}) = \|b_1\| \times \|b_2^*\| \times \cdots \times \|b_n^*\|$$

$$\|b_2^*\| \times \cdots \times \|b_n^*\| \nearrow \quad \rightsquigarrow \quad \|b_1\| \searrow$$

$\rightarrow$ Increase $\|b_2^*\|, \ldots, \|b_n^*\|$ to find a short lattice point!

Admittedly, but...

Quality of a basis $\iff$ What can we do algorithmically with it?

Tiling of the Space

$$\mathcal{P}(\mathbf{B}^*) \overset{\text{def}}{=} \left\{ \sum_i \lambda_i \mathbf{b}_i^* \ : \ \lambda_i \in [0, 1/2) \right\} \quad \text{(Babai's Fundamental Domain)}$$

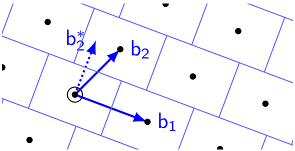
$\mathcal{P}(\mathbf{B}^*)$ tiles the space according to $\mathcal{L}$

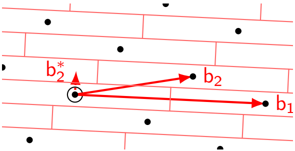
1. $\mathcal{L}$-packing,

$$\forall \mathbf{x}, \mathbf{y} \in \mathcal{L}, \quad (\mathbf{x} + \mathcal{P}(\mathbf{B}^*)) \cap (\mathbf{y} + \mathcal{P}(\mathbf{B}^*)) = \emptyset$$

2. $\mathcal{L}$-covering,

$$\mathcal{L} + \mathcal{P}(\mathbf{B}^*) = \mathbb{R}^n$$





And?
$\rightarrow$ Babai Algorithm!

Babai Algorithm

Algorithm 1: Babai Nearest Plan algorithm

Input : B basis of $\mathcal{L}$ and $y \in \mathbb{R}^n$ (word to "decode")

Output: $e \in \mathcal{P}(B^*)$ and $x \in \mathcal{L}$: $y = x + e$.

$e := y$

$x := 0$

for $i = n$ down to 1 **do**

  $k := \left\lfloor \frac{\langle e, b_i^* \rangle}{\|b_i^*\|} \right\rceil$

  $e := e - k b_i$

  $x := x + k b_i$

"If $i < j$ then $e \leftarrow e - k b_i$ doesn't modify $\langle e, b_j^* \rangle$"

Balance GSO's lengths

$$\mathbf{y} \xrightarrow{\text{Babai}(\mathbf{B})} (\mathbf{x}, \mathbf{e}) : \mathbf{y} = \mathbf{x} + \mathbf{e},\ \mathbf{x} \in \mathcal{L} \text{ and } \mathbf{e} \in \mathcal{P}(\mathbf{B}^*)$$

$$\mathcal{P}(\mathbf{B}^*) = \left\{ \sum_i \lambda_i \mathbf{b}_i^* \ : \ \lambda_i \in (-1/2, 1/2) \right\}$$

$\|\mathbf{e}\|$ small: minimize $1/4 \sum_i \|\mathbf{b}_i^*\|^2$ with constraint $\prod_i \|\mathbf{b}_i^*\| = \det(\mathcal{L})$

$\rightarrow$ Balance the lengths $\|\mathbf{b}_1^*\| \approx \cdots \approx \|\mathbf{b}_n^*\|$

Aim of LLL: Balance the $\|\mathbf{b}_i^*\|$'s

1. **Lattice Reduction**
   - Introduction
   - An Invariant: GSO
   - LLL Algorithm
2. **Code Reduction**
   - Orthopodality
   - Babai Algorithm for Codes
   - LLL Reduction and LLL Algorithm for Binary Code
   - Griesmer's Bound versus LLL

Aim of LLL

Balance GSO lengths $\|\mathbf{b}_i^*\|$'s

$\rightarrow$ Let us start with lattices of dimension 2

Wristwatch lemma

Theorem (Wristwatch lemma)

Let $\mathcal{L}$ be a lattice of dimension 2. It exists a basis $(\mathbf{b}_1, \mathbf{b}_2)$ such that:

- $\mathbf{b}_1$ *is a shortest vector of $\mathcal{L}$,*
- $|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \leq 1/2 \|\mathbf{b}_1\|^2$ *(will be useful for Hermite constant).*

$\rightarrow$ Proof of this theorem by an algorithm!

Lagrange Reduction.

Lagrange Reduction

Algorithm 2: Lagrange reduction algorithm

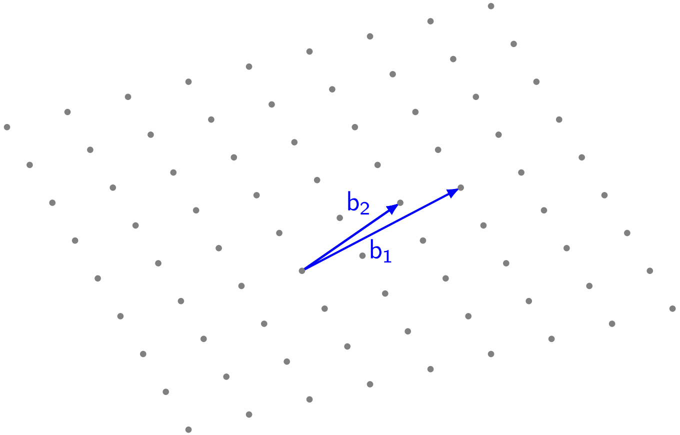
Input : A basis $(\mathbf{b}_1; \mathbf{b}_2)$ of a lattice
Output: A basis $(\mathbf{b}_1; \mathbf{b}_2)$ as in the Wristwatch lemma.

repeat
- Swap $\mathbf{b}_1 \leftrightarrow \mathbf{b}_2$
- $k \leftarrow \left\lfloor \frac{\langle \mathbf{b}_1, \mathbf{b}_2 \rangle}{\|\mathbf{b}_1\|^2} \right\rceil$
- $\mathbf{b}_2 \leftarrow \mathbf{b}_2 - k\mathbf{b}_1$

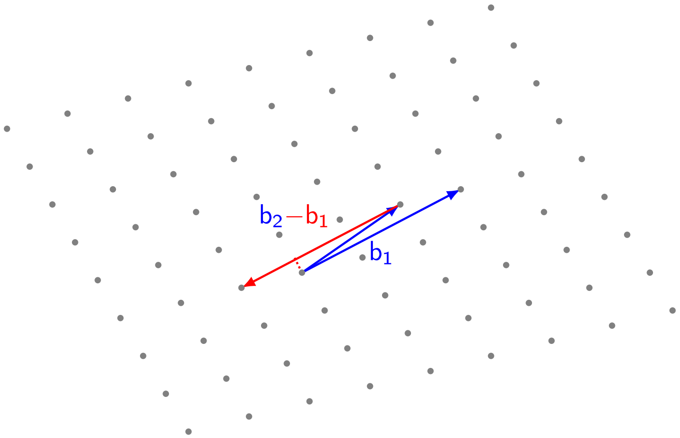
until $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\|$

Algorithm terminates after $O\left(\log_2 \frac{\|\mathbf{b}_1\|}{\sqrt{\det \mathcal{L}}}\right)$ steps!

Lagrange Reduction

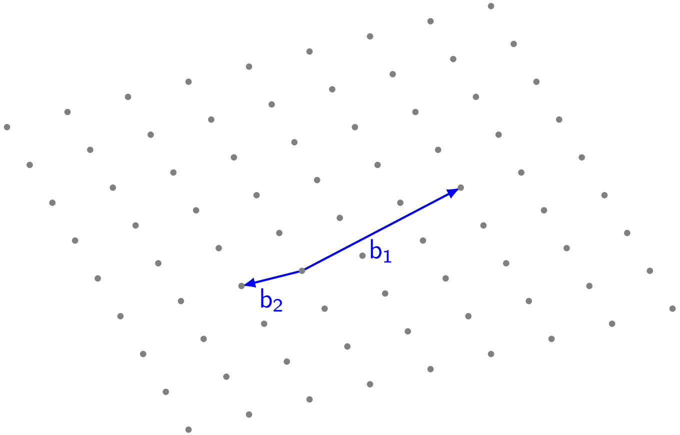


Lagrange Reduction

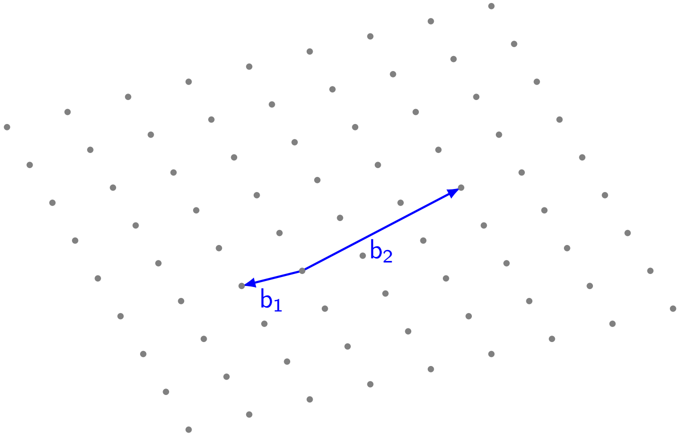


Reduction

Lagrange Reduction

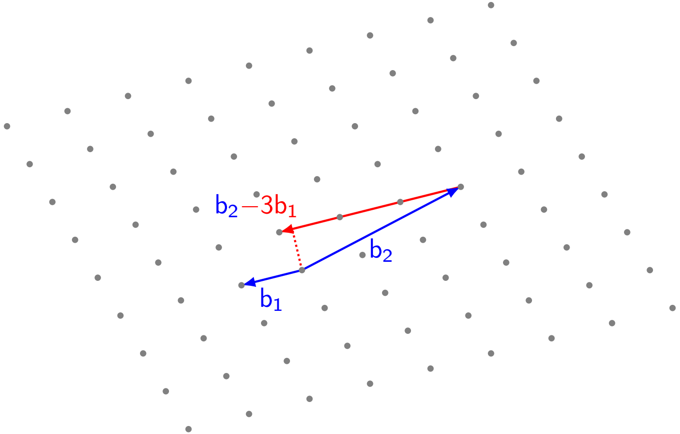


Lagrange Reduction



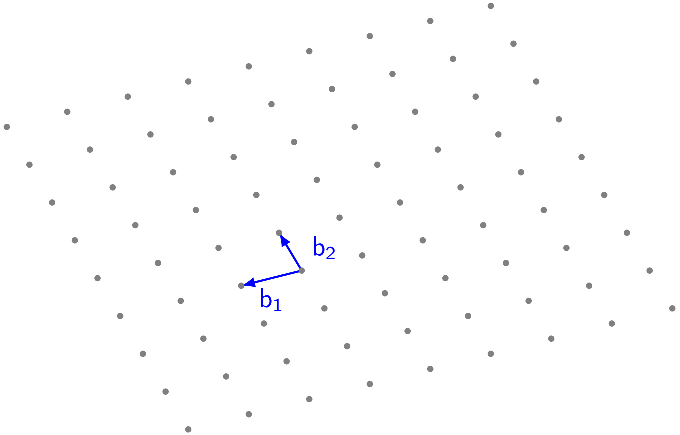
Swap

Lagrange Reduction

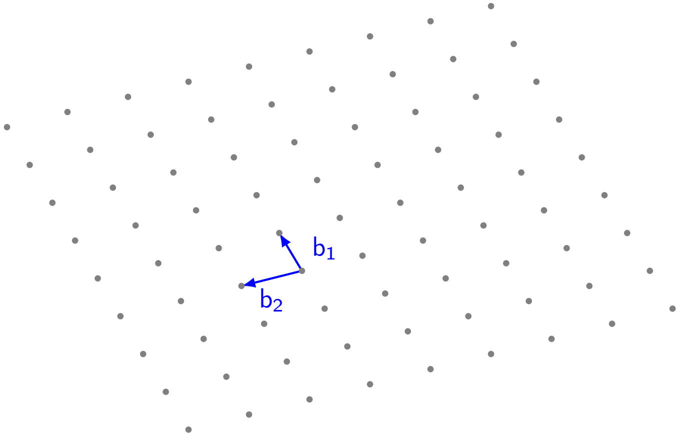


Reduction

Lagrange Reduction



Lagrange Reduction



Swap

Hermite constant

Definition (Hermite constant)

The Hermite constant γ_n is the supremum of over n-dimensional lattices $\mathcal{L}_n$:

$$\gamma_n \stackrel{\text{def}}{=} \sup_{\mathcal{L}_n} \gamma(\mathcal{L}) \quad \text{where} \quad \gamma(\mathcal{L}) \stackrel{\text{def}}{=} \frac{\lambda_1(\mathcal{L})^2}{\det(\mathcal{L})^{n/2}}.$$

For lattices of dimension 2 the Hermite constant is:

$$\gamma_2 = \sqrt{4/3}$$

$\rightarrow$ To obtain this: Lagrange reduction!

(Algorithmic proof of γ_2)

Proof of $\gamma_2 = \sqrt{4/3}$

- $\underline{\gamma_2 \leq \sqrt{4/3}}$: Let $(\mathbf{b}_1, \mathbf{b}_2)$ Lagrange reduced:

  $\mathbf{b}_1$ is a shortest vector of $\mathcal{L}$ and $|\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \leq 1/2$

  Rotating/scaling: $\mathbf{b}_1 = (0,1)$ and $\mathbf{b}_2 = (\alpha, \beta)$:

  $$\lambda_1(\mathcal{L}) / \det \mathcal{L} = 1/|\alpha|$$

  But $\alpha^2 \geq 3/4$ and then $\gamma_2^2 \leq 4/3$,

  $$\left.\begin{array}{l} |\langle \mathbf{b}_1, \mathbf{b}_2 \rangle| \leq 1/2 \iff |\beta| \leq 1/2 \\ \|\mathbf{b}_1\| \leq \|\mathbf{b}_2\| \iff 1 \leq \alpha^2 + \beta^2 \end{array}\right\} \Rightarrow 1 \leq \alpha^2 + \beta^2 \leq \alpha^2 + 1/4.$$

- $\underline{\gamma_2 \geq \sqrt{4/3}}$: Take $\mathbf{b}_1 = (0,1)$ and $\mathbf{b}_2 = (\sqrt{3/4}, 1/2)$.

LLL Reduced

$$\pi_i = \pi_{(\mathbf{b_1},\ldots,\mathbf{b_{i-1}})^\perp}$$

A basis B is LLL-reduced if $(\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}))$ is Lagrange-Reduced for all $i < n$.

$\rightarrow$ Enables to balance the profile, *i.e:* $(\|\mathbf{b}_i^*\|)_i$...

$$\|\mathbf{b}_i^*\| \leq \gamma_2 \times \|\mathbf{b}_{i+1}^*\| = \sqrt{4/3} \times \|\mathbf{b}_{i+1}^*\|$$

LLL Reduced

$$\pi_i = \pi_{(\mathbf{b_1},\ldots,\mathbf{b_{i-1}})^\perp}$$

A basis B is LLL-reduced if $(\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}))$ is Lagrange-Reduced for all $i < n$.

$\rightarrow$ Enables to balance the profile, *i.e:* $(\|\mathbf{b}_i^*\|)_i$...

$$\|\mathbf{b}_i^*\| \leq \gamma_2 \times \|\mathbf{b}_{i+1}^*\| = \sqrt{4/3} \times \|\mathbf{b}_{i+1}^*\|$$

Proof.

Let $\mathcal{L}_i \stackrel{\text{def}}{=} \text{Span}_{\mathbb{Z}}(\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}))$:

$$\frac{\lambda_1(\mathcal{L})^2}{\det(\mathcal{L}_i)} = \frac{\|\pi_i(\mathbf{b}_i)\|^2}{\|\pi_i(\mathbf{b}_i)\| \times \|\underset{\perp \pi_i(\mathbf{b}_i)}{\text{Proj}}(\pi_i(\mathbf{b}_{i+1}))\|} = \frac{\|\pi_i(\mathbf{b}_i)\|}{\|\pi_{i+1}(\mathbf{b}_{i+1})\|} \leq \sqrt{4/3}.$$

□

LLL Algorithm

While $\exists i$ s.t $(\pi_{\mathbf{b}_i}, \pi_i(\mathbf{b}_{i+1}))$ is not Lagrange-reduced, Lagrange reduce it...

- Correctness: by definition,
- Termination in poly-time: no details here, need an ε-relaxation,

1 Lattice Reduction
Introduction
An Invariant: GSO
LLL Algorithm

2 Code Reduction
Orthopodality
Babai Algorithm for Codes
LLL Reduction and LLL Algorithm for Binary Code
Griesmer's Bound versus LLL

Codes

Binary Linear Code: $\mathcal{C} \subset \mathbb{F}_2^n$ subspace equipped with Hamming metric $|\cdot|$.

Basis of $\mathcal{C}$ (dimension k code): $\mathrm{B} \stackrel{\text{def}}{=} (\mathrm{b}_1, \ldots, \mathrm{b}_k)$ such that,

1. Linearly independent,
2. Span $\mathcal{C}$ over $\mathbb{F}_2$,

$$\mathcal{L} = \mathcal{C}(\mathrm{B}) \quad \text{where} \quad \mathcal{C}(\mathrm{B}) \stackrel{\text{def}}{=} \left\{ \sum_{i=1}^{n} m_i \mathrm{b}_i \ : \ m_i \in \mathbb{F}_2 \right\}.$$

$$d_{\min}(\mathcal{L}) \stackrel{\text{def}}{=} \min_{\mathrm{c} \in \mathcal{C} \setminus \{0\}} |\mathrm{c}|$$

Once again, aim of reduction: find good bases!

Systematic Form

$$\mathrm{B} = \left(\underbrace{\mathrm{A}}_{n-k} \;\middle|\; \underbrace{\begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix}}_{k} \right)$$

Basis in systematic form is used for:

- Generic decoding, *information set decoding*,
- Finding short codewords, $|\mathrm{b}_i| \approx \frac{n-k}{2}$ when B random.

$\rightarrow$ Can we find better bases in poly-time?
LLL approach?

An LLL Approach for Codes

Use the standard inner product over $\mathbb{F}_2^n$?
Bad idea...

- No information about the weight...

$$\langle x, y \rangle = 0 \not\Rightarrow |x + y| = |x| + |y|.$$

- Invariant associated to it?

1. **Lattice Reduction**
   - Introduction
   - An Invariant: GSO
   - LLL Algorithm
2. **Code Reduction**
   - Orthopodality
   - Babai Algorithm for Codes
   - LLL Reduction and LLL Algorithm for Binary Code
   - Griesmer's Bound versus LLL

Notation

Bitstring Notation

Let $\mathrm{x}, \mathrm{y} \in \mathbb{F}_2^n$,

$$\mathrm{x} \wedge \mathrm{y} = (x_i \wedge y_i)_i \quad \text{and} \quad \mathrm{x} \vee \mathrm{y} = (x_i \vee y_i)$$

Example:

$$\left.\begin{matrix} \boxed{1\,0\,1\,0\,1} \\ \bigwedge \\ \boxed{0\,0\,1\,1\,0} \end{matrix}\right\} = \boxed{0\,0\,1\,0\,0} \quad \Bigg| \quad \left.\begin{matrix} \boxed{1\,0\,1\,0\,1} \\ \bigvee \\ \boxed{0\,0\,1\,1\,0} \end{matrix}\right\} = \boxed{1\,0\,1\,1\,1}$$

Notation

Bitstring Notation

Let $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$,

$$\mathbf{x} \wedge \mathbf{y} = (x_i \wedge y_i)_i \quad \text{and} \quad \mathbf{x} \vee \mathbf{y} = (x_i \vee y_i)$$

Example:

$$\left.\begin{array}{c}\boxed{1\,0\,1\,0\,1}\\ \bigwedge \\ \boxed{0\,0\,1\,1\,0}\end{array}\right\} = \boxed{0\,0\,1\,0\,0} \quad \Bigg| \quad \left.\begin{array}{c}\boxed{1\,0\,1\,0\,1}\\ \bigvee \\ \boxed{0\,0\,1\,1\,0}\end{array}\right\} = \boxed{1\,0\,1\,1\,1}$$

Support

Let $\mathbf{x} \in \mathbb{F}_2^n$, its support is defined as:

$$\mathrm{Supp}(\mathbf{x}) \stackrel{\text{def}}{=} \{i \in [\![1, n]\!] : x_i \neq 0\}.$$

Orthopodality

Fundamental Remark:

$$|x+y| = |x| + |y| - 2|x \wedge y|.$$

Orthopodality

Two vectors $x, y \in \mathbb{F}_2^n$ are said orthopodal:

$$x \perp y \overset{\text{def}}{\Longleftrightarrow} x \wedge y = 0.$$

$$x \perp y \Rightarrow |x| + |y|$$

Orthopodal Projection

$$\pi_y^{\perp} : x \mapsto x \wedge \bar{y}.$$

$\pi_y^{\perp}(x)$ *only keeps coordinates of* x *in* $\mathrm{Sup}(x) \backslash \mathrm{Supp}(y)$
$(\mathrm{Supp}(x) = \{i : x_i \neq 0\})$.

Gram-Schmidt Orthopodalization(I)

For lattices:

$$\pi_i^{\perp} \textit{ orthogonal projection to } (\mathrm{Span}_{\mathbb{R}}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}))^{\perp}$$

For Codes:

$$\pi_i^{\perp} : \mathbf{x} \longmapsto \mathbf{x} \wedge \overline{(\mathbf{b}_1 \vee \cdots \vee \mathbf{b}_{i-1})}$$

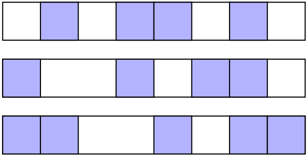
Lattice	*Code*
$\mathrm{Span}_{\mathbb{R}}(\cdot)$	$\mathrm{Supp}(\cdot)$

Gram-Schmidt Orthopodalization(II)

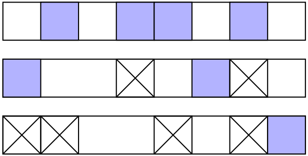
- $b_1^+ \stackrel{\text{def}}{=} b_1$
- Projection orthogonal to $\text{Sup}(b_1, \ldots, b_{i-1})$,

$$b_i^+ \stackrel{\text{def}}{=} \pi_i^\perp(b_i) \quad \text{where} \quad \pi_i^\perp(b_i) = b_i \wedge \overline{(b_1 \vee \cdots \vee b_{i-1})}$$

An example:



b_1, b_2, b_3



b_1^+, b_2^+, b_3^+

$$\pi_i^\perp(x) = x + \sum_{j<i} x \wedge b_j^+$$

Epipodal Matrix

Epipodal Matrix

$\mathrm{B} = (\mathrm{b}_1, \ldots, \mathrm{b}_k)$ be a basis. Its epipodal matrix is defined as

$$\mathrm{B}^+ = (\mathrm{b}_1^+, \ldots, \mathrm{b}_k^+)$$

b_{i+1}^+ support increment from $\mathcal{C}(\mathrm{b}_1, \ldots, \mathrm{b}_{i-1})$ to $\mathcal{C}(\mathrm{b}_1, \ldots, \mathrm{b}_i)$

Lattice Reduction
Introduction
An Invariant: GSO
LLL Algorithm
Code Reduction
Orthopodality
Babai Algorithm for Codes
LLL Reduction and LLL Algorithm for Binary Code
Griesmer's Bound versus LLL

An Invariant

$(\mathbf{b}_1^+, \ldots, \mathbf{b}_k^+)$ is not a basis of $\mathcal{C}$, but...

$$\sum_i |\mathbf{b}_i^+| = \#\mathrm{Supp}(\mathcal{C})$$

where $\mathrm{Supp}(\mathcal{C}) \stackrel{\text{def}}{=} \{i \in [\![1,n]\!], \exists \mathbf{c} \in \mathcal{C}, c_i \neq 0\}$.

$\rightarrow$ Increase $|\mathbf{b}_2^+|, \ldots, |\mathbf{b}_n^+|$ to find a short codeword!

Admittedly, but once again...

Quality of a basis $\iff$ What can we do algorithmically with it?

Thomas Debris-Alazard, Léo Ducas, Wessel P.J. van Woerden

Lattice Reduction
Introduction
An Invariant: GSO
LLL Algorithm
Code Reduction
Orthopodality
Babai Algorithm for Codes
LLL Reduction and LLL Algorithm for Binary Code
Griesmer's Bound versus LLL

1 Lattice Reduction
- Introduction
- An Invariant: GSO
- LLL Algorithm

2 Code Reduction
- Orthopodality
- Babai Algorithm for Codes
- LLL Reduction and LLL Algorithm for Binary Code
- Griesmer's Bound versus LLL

Babai Fundamental Domain

For lattices:

$$\mathcal{P}(\mathbf{B}^*) \stackrel{\text{def}}{=} \left\{ \sum_i \lambda_i \mathbf{b}_i^* \ : \ \lambda_i \in [0, 1/2) \right\} \quad \text{(tiles the space)}$$

Babai Fundamental Domain for Codes

$$\mathcal{F}(\mathbf{B}^+) \stackrel{\text{def}}{=} \left\{ \mathbf{y} \in \mathbb{F}_2^n \ : \ \forall i \in [\![1, k]\!], \ |\mathbf{y} \wedge \mathbf{b}_i^+| + \mathrm{TB}_{\mathbf{b}_i^+}(\mathbf{y}) \leq \frac{|\mathbf{b}_i^+|}{2} \right\}.$$

where (technical):

$$\mathrm{TB}_{\mathbf{p}}(\mathbf{y}) = \begin{cases} 0 & \text{if } |\mathbf{p}| \text{ is odd,} \\ 0 & \text{if } y_j = 0 \text{ where } j = \min(\mathrm{Supp}(\mathbf{p})), . \\ 1/2 & \text{otherwise.} \end{cases}$$

Babai Fundamental Domain

For lattices:

$$\mathcal{P}(\mathrm{B}^*) \stackrel{\text{def}}{=} \left\{ \sum_i \lambda_i \mathrm{b}_i^* \ : \ \lambda_i \in [0, 1/2) \right\} \quad \text{(tiles the space)}$$

Babai Fundamental Domain for Codes

$$\mathcal{F}(\mathrm{B}^+) \stackrel{\text{def}}{=} \left\{ \mathrm{y} \in \mathbb{F}_2^n \ : \ \forall i \in [\![1,k]\!], \ |\mathrm{y} \wedge \mathrm{b}_i^+| + \mathrm{TB}_{\mathrm{b}_i^+}(\mathrm{y}) \leq \frac{|\mathrm{b}_i^+|}{2} \right\}.$$

where (technical):

$$\mathrm{TB}_{\mathrm{p}}(\mathrm{y}) = \begin{cases} 0 & \text{if } |\mathrm{p}| \text{ is odd,} \\ 0 & \text{if } y_j = 0 \text{ where } j = \min(\mathrm{Supp}(\mathrm{p})), . \\ 1/2 & \text{otherwise.} \end{cases}$$

Remark:

$$\text{If } |\mathrm{y} \wedge \mathrm{b}_i^+| \geq \frac{|\mathrm{b}_i^+|}{2}, \text{then} \quad |(\mathrm{y}+\mathrm{b}_i) \wedge \mathrm{b}_i^+| \leq \frac{|\mathrm{b}_i^+|}{2}$$

Babai Fundamental Domain

$\mathcal{F}(\mathrm{B}^+)$ tiles the space

1. $\mathcal{F}(\mathrm{B}^+)$ is $\mathcal{C}$-packing:

$$\forall \mathrm{c} \in \mathcal{C} \backslash \{0\}, \quad (\mathrm{c} + \mathcal{F}(\mathrm{B}^+)) \cap \mathcal{F}(\mathrm{B}^+) = \emptyset,$$

2. $\mathcal{F}(\mathrm{B}^+)$ is $\mathcal{C}$-covering:

$$\mathcal{C} + \mathcal{F}(\mathrm{B}^+) = \mathbb{F}_2^n.$$

Babai Algorithm for Codes:

$$\mathrm{y} \xmapsto{\text{Babai}(\mathrm{B})} (\mathrm{c}, \mathrm{e}) : \ \mathrm{y} = \mathrm{c} + \mathrm{e}, \mathrm{c} \in \mathcal{C} \text{ and } \mathrm{e} \in \mathcal{F}(\mathrm{B}^+)$$

Babai Algorithm

Input : A basis $\mathbf{B} = (\mathbf{b}_1; \ldots; \mathbf{b}_k) \in \mathbb{F}_2^{k \times n}$ and a target $\mathbf{y} \in \mathbb{F}_2^n$
Output: $\mathbf{e} \in \mathcal{F}(\mathbf{B}^+)$ such that $\mathbf{e} + \mathbf{y} \in \mathcal{C}(\mathbf{B})$

$\mathbf{e} \leftarrow \mathbf{y}$
for $i = k$ **down to** 1 **do**
  if $|\mathbf{e} \wedge \mathbf{b}_i^+| + \mathrm{TB}_{\mathbf{b}_i^+}(\mathbf{e}) > |\mathbf{b}_i^+|/2$ **then**
    $\mathbf{e} \leftarrow \mathbf{e} + \mathbf{b}_i$
return $\mathbf{e}$

"If $i < j$ then $\mathbf{e} \leftarrow \mathbf{e} + \mathbf{b}_i$ doesn't modify $\mathbf{e} \wedge \mathbf{b}_j^+$"

An Example

$$\mathbf{B} = \begin{pmatrix} 1-1 & \big| & 1 & & \\ * & \big| & & \ddots & \\ & \big| & & & 1 \end{pmatrix} \quad \Bigg| \quad \mathbf{B}^+ = \begin{pmatrix} 1-1 & \big| & 1 & & \\ 0 & \big| & & \ddots & \\ & \big| & & & 1 \end{pmatrix}$$

$$\begin{aligned} \mathbf{b}_n^+ &= (0,\dots,0,0,1) \\ \mathbf{b}_{n-1}^+ &= (0,\dots,0,1,0) \\ &\vdots \end{aligned}$$

We have,

$$\forall i \in [\![2,k]\!], \quad |\mathbf{b}_i^+| = 1.$$

We add $\mathbf{b}_i$ $(i > 2)$ to $\mathbf{y}$ if and only if,

$$|\mathbf{y} \wedge \mathbf{b}_i^+| > |\mathbf{b}_i^+|/2 \iff |\mathbf{y} \wedge \mathbf{b}_i^+| > 1/2 \iff y_i = 1.$$

$\rightarrow$ Prange Algorithm!

Consequence

Previous Example:

$$\forall i \in [\![2,k]\!],\ |\mathbf{b}_i^+| = 1 \quad \text{and} \quad |\mathbf{b}_1^+| = n-k+1$$

For Babai to be efficient, we would like:

$$|\mathbf{b}_i^+| > 1 \text{ for as most as possible } i \in [\![2,k]\!]$$

But the invariant...

$$\sum_{i=1}^{k} |\mathbf{b}_i^+| = n.$$

Consequence

Previous Example:

$$\forall i \in [\![2, k]\!],\ |\mathbf{b}_i^+| = 1 \quad \text{and} \quad |\mathbf{b}_1^+| = n - k + 1$$

For Babai to be efficient, we would like:

$$|\mathbf{b}_i^+| > 1 \text{ for as most as possible } i \in [\![2, k]\!]$$

But the invariant...

$$\sum_{i=1}^{k} |\mathbf{b}_i^+| = n.$$

More generally, we can prove that Babai will be the more efficient if:

$$|\mathbf{b}_1^+| \approx \cdots \approx |\mathbf{b}_k^+|$$

$\rightarrow$ The aim of LLL (as for Lattices)

1. Lattice Reduction
   - Introduction
   - An Invariant: GSO
   - LLL Algorithm
2. Code Reduction
   - Orthopodality
   - Babai Algorithm for Codes
   - LLL Reduction and LLL Algorithm for Binary Code
   - Griesmer's Bound versus LLL

Codes of Dimension 2

What is the best "balanced" basis for a code of dimension 2?

Lemma (Lagrange Reduced Basis)

For any code $\mathcal{C}$ of dimension 2, there exists a basis $(\mathrm{b}_1, \mathrm{b}_2)$ such that:

$$|\mathrm{b}_1| = d_{\min}(\mathcal{C}) \quad \textit{and} \quad |\mathrm{b}_1 \wedge \mathrm{b}_2| \leq \frac{1}{2}|\mathrm{b}_1|$$

1. We cannot hope better in the worst case

$$\mathcal{C} = \mathcal{C}((110),(011))$$

2. We have:

$$|\mathrm{b}_1| \leq 2 \times |\mathrm{b}_2^+|$$

Lattice Reduction

Introduction

An Invariant: GSO

LLL Algorithm

Code Reduction

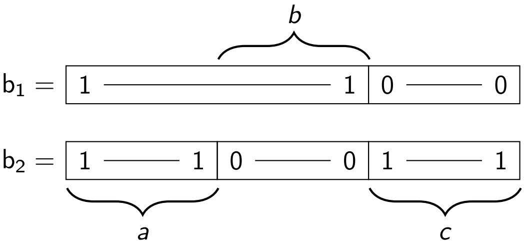
Orthopodality

Babai Algorithm for Codes

LLL Reduction and LLL Algorithm for Binary Code

Griesmer's Bound versus LLL

The Proof



First:

$$a \text{ and } b > \frac{1}{2}(a+b) \ : \ \textit{imposible}$$

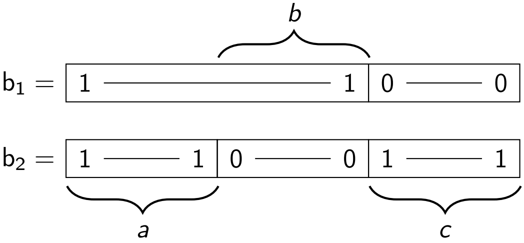
therefore,

$$(|\mathrm{b}_1 \wedge \mathrm{b}_2| = a \quad \text{or} \quad |\mathrm{b}_1 \wedge (\mathrm{b}_1 + \mathrm{b}_2)| = b) \quad \leq \quad \frac{1}{2}(a+b) = \frac{1}{2}|\mathrm{b}_1|.$$

Now, $d_{\min}(\mathcal{C}) = a + b \leq a + c$ and $\leq b + c$. Therefore,

$$|\mathrm{b}_2^+| = 2c \geq a + b = |\mathrm{b}_1|.$$

In the Random Case



For a random code: $a \approx b \approx c$. Therefore,

$$2|b_2^+| \approx |b_1|$$

LLL Reduced

B is said LLL-reduced if $(\pi_i(\mathrm{b}_i), \pi_i(\mathrm{b}_{i+1}))$ is LLL-reduced

Two guarantees:

$$|\mathrm{b}_i^+| \leq 2|\mathrm{b}_{i+1}^+| \quad \text{and} \quad |\mathrm{b}_i^+| \geq 1.$$

Bound on code:

$$n = \sum_{i=1}^{k} |\mathrm{b}_i^+| \geq \sum_i \left\lceil \frac{|\mathrm{b}_1|}{2^i} \right\rceil$$

Therefore,

$$\longrightarrow \quad |\mathrm{b}_1| - \frac{\lceil \log_2(\mathrm{b}_1) \rceil}{2} \leq \frac{n-k}{2} + 1$$

First vector of LLL-reduced of weight $\approx (n-k)/2$ in the worst case.

LLL Algorithm

While $\exists i$ s.t $(\pi_{\mathbf{b}_i}, \pi_i(\mathbf{b}_{i+1}))$ is not Lagrange-reduced, Lagrange reduce it...

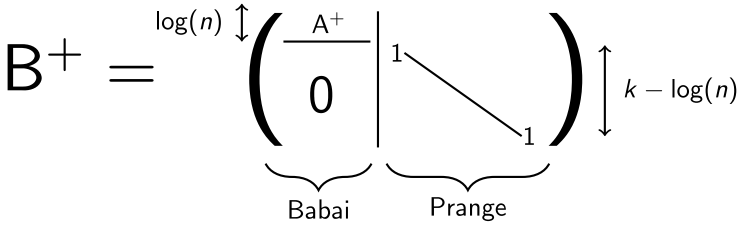
- Correctness: by definition,
- Termination in poly-time: no details here, same argument as the original LLL

$\rightarrow$ It shows the existence of LLL-reduced bases...

Shape of LLL-reduced Bases

We typically expect $|\mathbf{b}_1| = \frac{n-k}{2}$ and $|\mathbf{b}_i^+| = \left\lceil \frac{|\mathbf{b}_1|}{2^i} \right\rceil$, therefore:

$$|\mathbf{b}_i^+| = \Omega(1) \text{ for } i = O(\log_2(n)).$$



A basis of a dimension $\log(n)$-code, we cannot hope typically:

1. to get codewords of weight $\leq (1-\varepsilon)\frac{n-k}{2}$,
2. to improve Prange's algorithm by more than a polynomial factor.

Griesmer's Bound

LLL produces (in poly-time) a basis B of $\mathcal{C}$ verifying:

$$n \geq \sum_{i=1}^{k} \left\lceil \frac{|\mathrm{b}_1|}{2^i} \right\rceil$$

But $|\mathrm{b}_1| \geq d_{\min}(\mathcal{C})$...

$$\to n \geq \sum_{i} \left\lceil \frac{d_{\min}(\mathcal{C})}{2^i} \right\rceil \quad \text{(Griesmer Bound!)}$$

- LLL $\to$ algorithmic proof of Griesmer,
- Systematic form $\to$ proves Singleton ($d \leq n - k + 1$)

Griesmer Reduced Bases

How works the proof of Griesmer?

$\rightarrow$ With existential arguments:

Lemma

Let $\mathcal{C}$ be an $[n,k]$-code and $\mathrm{c} \in \mathcal{C}$ with $|\mathrm{c}| = d_{\min}(\mathcal{C})$. Then $\mathcal{C}' \stackrel{def}{=} \pi_{\mathrm{c}}^{\perp}(\mathcal{C}) = \mathcal{C} \wedge \overline{\mathrm{c}}$ satisfies:

1. *$|\mathcal{C}'| = n - d_{\min}(\mathcal{C})$ and its dimension is $k-1$,*
2. *$d_{\min}\mathcal{C}' \geq \lceil d_{\min}(\mathcal{C})/2 \rceil$.*

Proof of 2. as Lagrange-reduced basis!

HKZ Bases for Codes

In fact Griesmer proves the existence of bases:

Definition (Griesmer-reduced basis)

A basis B is said Griesmer-reduced if b_i^+ is a shortest non-zero codeword of the projected subcode $\pi_i(\mathcal{C}(\mathrm{b}_i; \ldots; \mathrm{b}_k))$ for all $i \in [\![1, k]\!]$.

$\rightarrow$ Direct analogue HKZ-bases for lattice bases!

Conclusion, what Else?

In the paper:

- Study of the Babai's fundamental domain $\mathcal{F}(B)$,
- An hybrid Babai + Lee-Brickell algorithm,
- Implementations and experiments.

Open questions:

- Duality,
- More bounds (generalized Hamming weight...)
- More algorithms (BKZ,...)
- ...

Thank You!