Manipuler les réseaux euclidiens

Damien Stehlé

ÉNS de Lyon – équipe ARIC

SIESTE, Lyon, Octobre 2012

L'équipe ARIC (Arithmetic and Computing)

- 5 enseignants-chercheurs
- 7 chercheurs
- 2 post-doctorants
- 8 doctorants

Objectif : Améliorer les calculs, en se focalisant sur l'arithmétique

- Arithmétique flottante
  (opérateurs, fonctions élémentaires, FPGA, validation)
- Calcul formel (algèbre linéaire flottante et exacte)
- **Réseaux euclidiens (algorithmes et cryptographie)**
- Théorie des nombres

L'équipe ARIC (Arithmetic and Computing)

- 5 enseignants-chercheurs
- 7 chercheurs
- 2 post-doctorants
- 8 doctorants

Objectif : Améliorer les calculs, en se focalisant sur l'arithmétique

- Arithmétique flottante
  (opérateurs, fonctions élémentaires, FPGA, validation)
- Calcul formel (algèbre linéaire flottante et exacte)
- **Réseaux euclidiens (algorithmes et cryptographie)**
- Théorie des nombres

Goals and plan of the talk

Goals:

- An introduction to the computational aspects of lattices
- An example of how floating-point arithmetic can be used to accelerate an algebraic computation

Plan of the talk:

1. **Euclidean lattices**
2. Applications of euclidean lattices
3. The LLL algorithm and its accelerations

Goals and plan of the talk

Goals:

- An introduction to the computational aspects of lattices
- An example of how floating-point arithmetic can be used to accelerate an algebraic computation

Plan of the talk:

1. **Euclidean lattices**
2. Applications of euclidean lattices
3. The LLL algorithm and its accelerations

Euclidean lattices

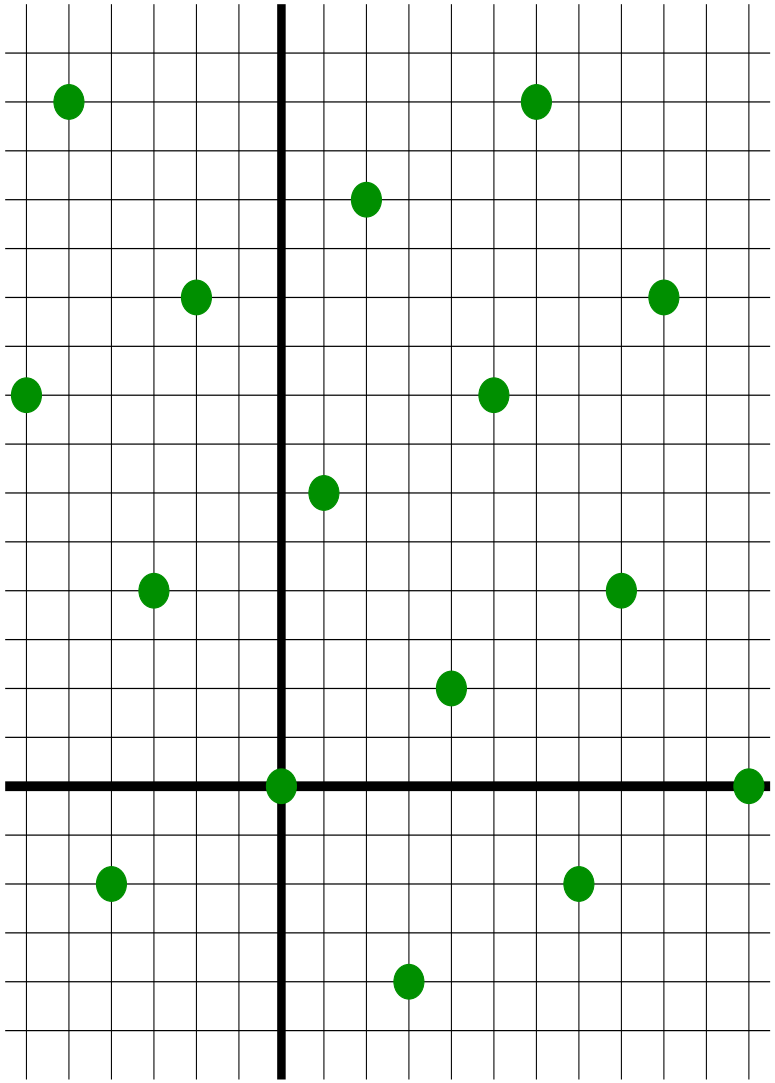
Lattice $\equiv$ discrete subgroup of $\mathbb{R}^n$

$\equiv \{\sum_{i \leq n} x_i \mathbf{b}_i : x_i \in \mathbb{Z}\}$

If the $\mathbf{b}_i$'s are linearly independent, they are called a **basis**.

Bases are not unique, but they can be obtained from each other by integer transforms of determinant ± 1:

$$\begin{bmatrix} -2 & 1 \\ 10 & 6 \end{bmatrix} = \begin{bmatrix} 4 & -3 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}.$$



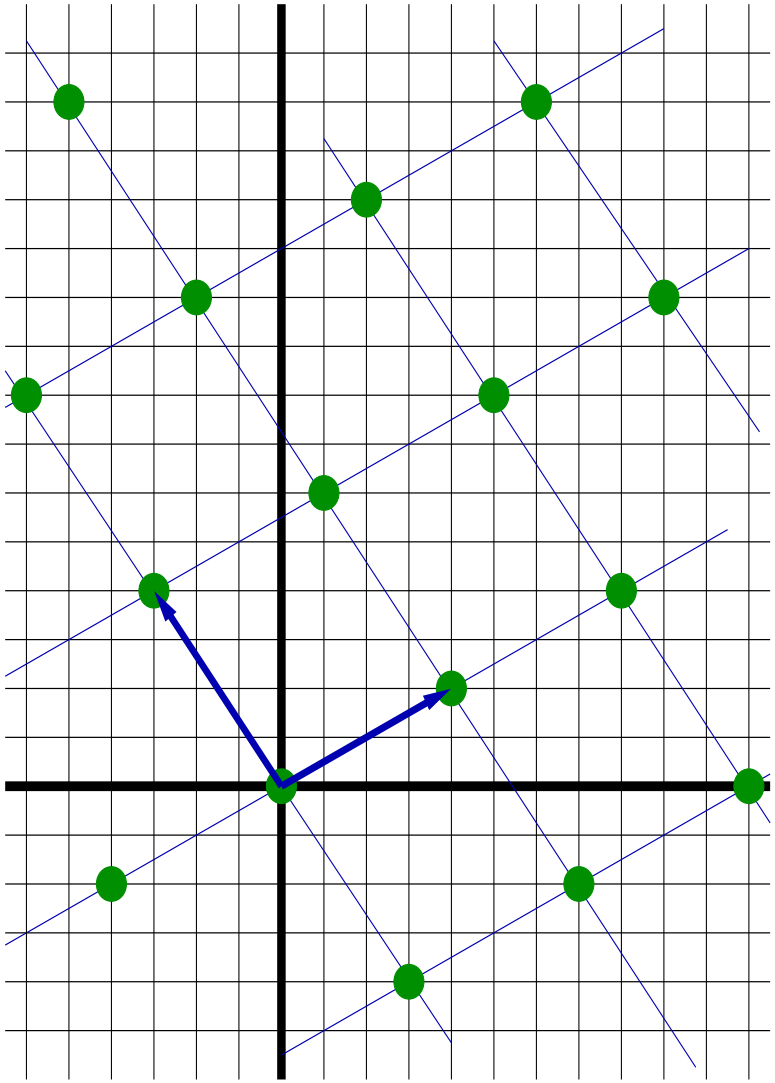
Euclidean lattices

Lattice $\equiv$ discrete subgroup of $\mathbb{R}^n$
$\equiv \{\sum_{i \leq n} x_i \mathbf{b}_i : x_i \in \mathbb{Z}\}$

If the $\mathbf{b}_i$'s are linearly independent, they are called a **basis**.

Bases are not unique, but they can be obtained from each other by integer transforms of determinant ± 1:

$$\begin{bmatrix} -2 & 1 \\ 10 & 6 \end{bmatrix} = \begin{bmatrix} 4 & -3 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}.$$



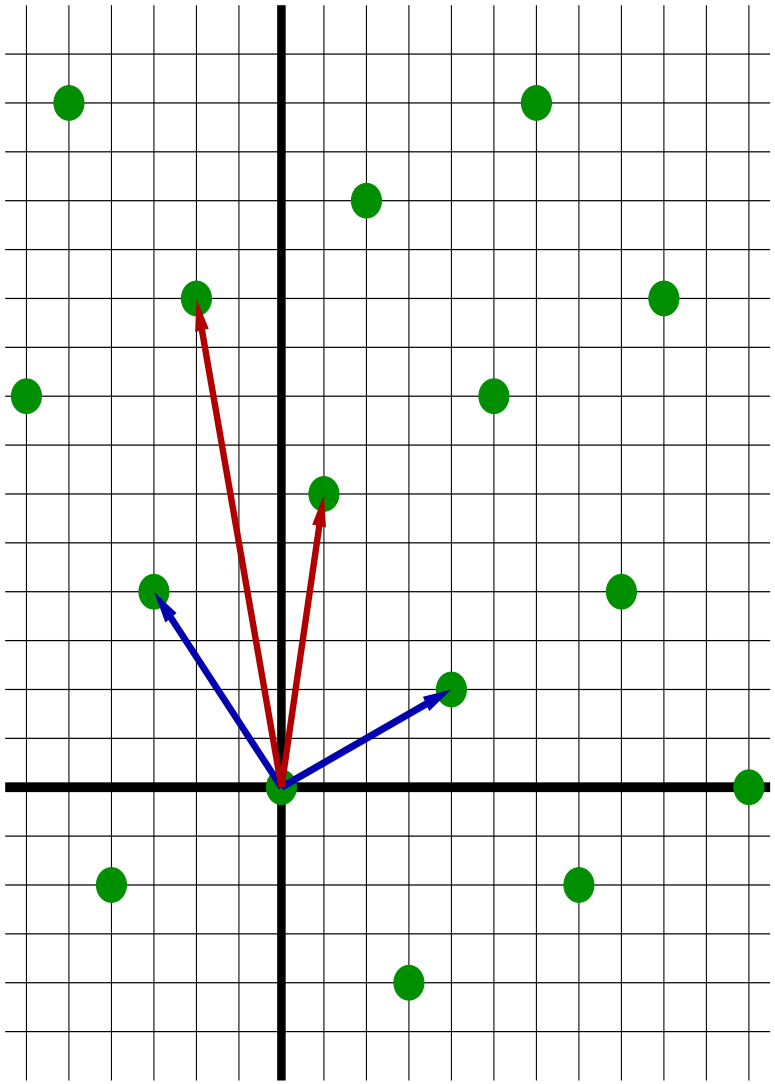
Euclidean lattices

Lattice $\equiv$ discrete subgroup of $\mathbb{R}^n$
$\equiv \{\sum_{i \le n} x_i \mathbf{b}_i : x_i \in \mathbb{Z}\}$

If the $\mathbf{b}_i$'s are linearly independent, they are called a **basis**.

Bases are not unique, but they can be obtained from each other by integer transforms of determinant ± 1:

$$\begin{bmatrix} -2 & 1 \\ 10 & 6 \end{bmatrix} = \begin{bmatrix} 4 & -3 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}.$$



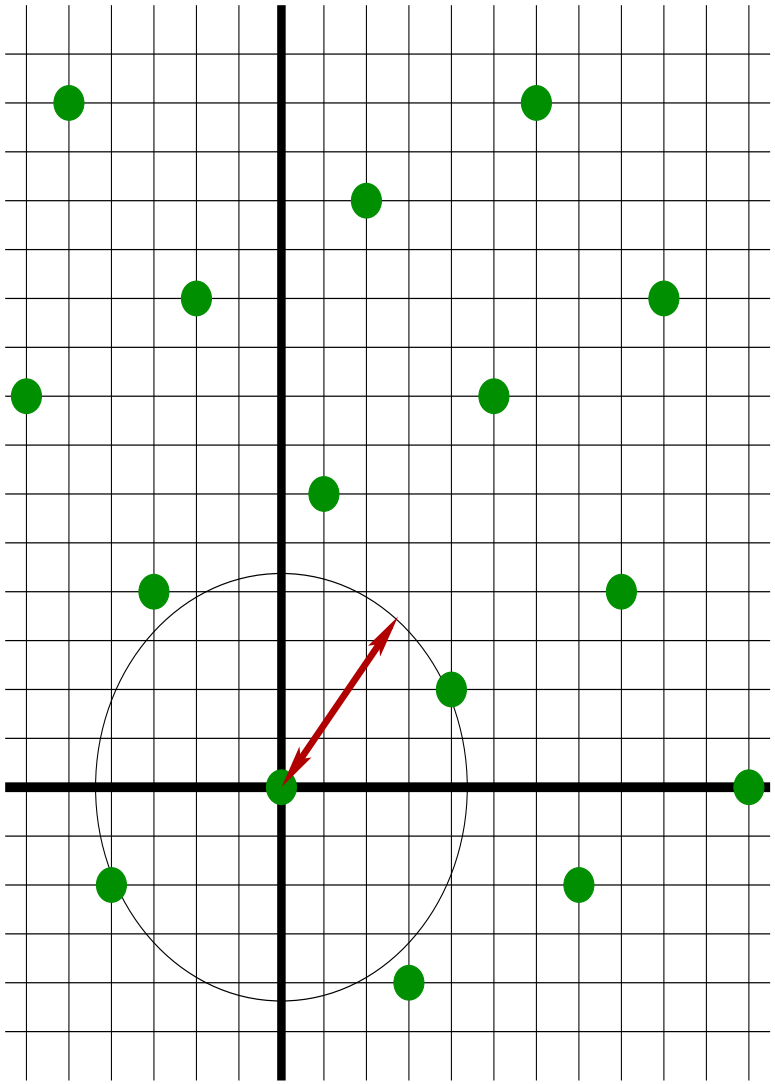
Lattice invariants

First minimum:
$\lambda(L) = \min(\|\mathbf{b}\| : \mathbf{b} \in L \setminus \mathbf{0})$.

Determinant:
$\det L = |\det(\mathbf{b}_i)_i|$, for any basis

Minkowski theorem:
$\lambda(L) \leq \sqrt{n} \cdot (\det L)^{1/n}$.

Algorithmic approach: lattice reduction
Start from a basis, and progressively improve its norm/orthogonality.



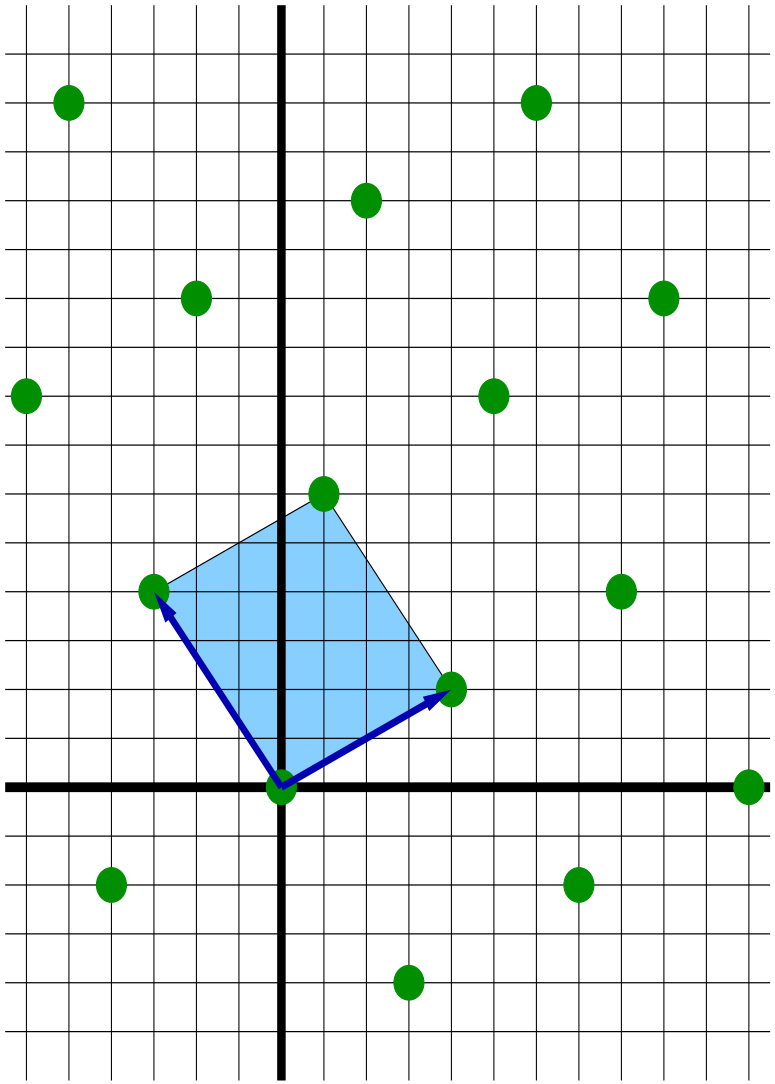
Lattice invariants

First minimum:
$\lambda(L) = \min(\|\mathbf{b}\| : \mathbf{b} \in L \setminus \mathbf{0})$.

Determinant:
$\det L = |\det(\mathbf{b}_i)_i|$, for any basis

Minkowski theorem:
$\lambda(L) \leq \sqrt{n} \cdot (\det L)^{1/n}$.

Algorithmic approach: lattice reduction
Start from a basis, and progressively improve its norm/orthogonality.



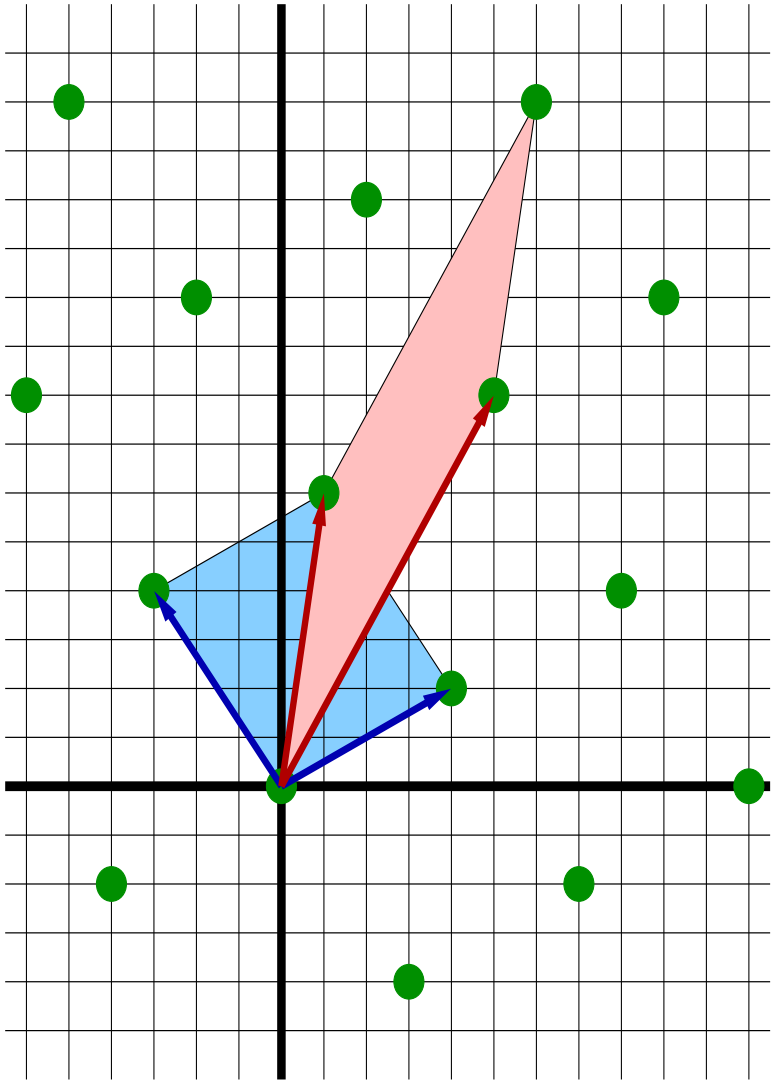
Lattice invariants

First minimum:
$\lambda(L) = \min(\|\mathbf{b}\| : \mathbf{b} \in L \setminus \mathbf{0})$.

Determinant:
$\det L = |\det(\mathbf{b}_i)_i|$, for any basis

Minkowski theorem:
$\lambda(L) \leq \sqrt{n} \cdot (\det L)^{1/n}$.

Algorithmic approach: lattice reduction
Start from a basis, and progressively improve its norm/orthogonality.



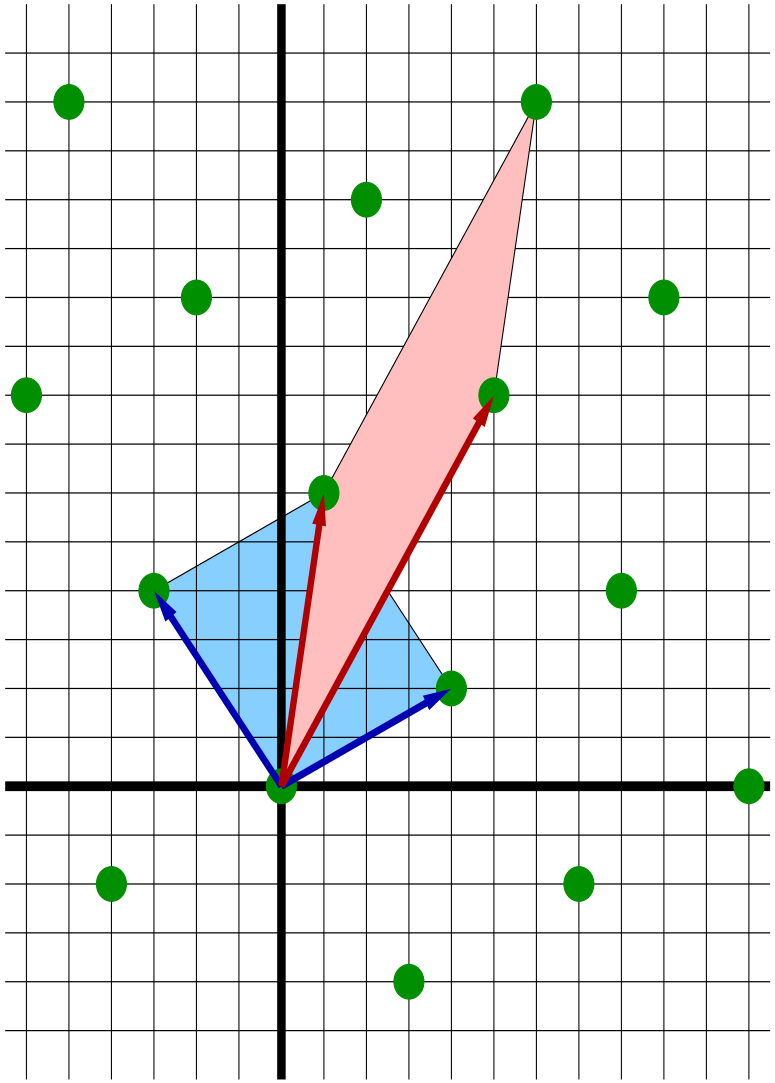
Lattice invariants

First minimum:
$\lambda(L) = \min(\|\mathbf{b}\| : \mathbf{b} \in L \setminus \mathbf{0})$.

Determinant:
$\det L = |\det(\mathbf{b}_i)_i|$, for any basis

Minkowski theorem:
$\lambda(L) \leq \sqrt{n} \cdot (\det L)^{1/n}$.

Algorithmic approach: lattice reduction
Start from a basis, and progressively improve its norm/orthogonality.



Why do we care about lattices?

- Computer algebra: factorisation of rational polynomials.
- Cryptanalysis of variants of RSA.
- Lattice-based cryptography.
- Communications theory: MIMO, GPS.
- Combinatorial optimisation, algorithmic group theory, algorithmic number theory, computer arithmetic, etc.

Lattices tend to pop out every time one wants to use linear algebra but is restricted to discrete transformations.

Why do we care about lattices?

- Computer algebra: factorisation of rational polynomials.
- Cryptanalysis of variants of RSA.
- Lattice-based cryptography.
- Communications theory: MIMO, GPS.
- Combinatorial optimisation, algorithmic group theory, algorithmic number theory, computer arithmetic, etc.

Lattices tend to pop out every time one wants to use linear algebra but is restricted to discrete transformations.

Main computational problem: SVP

- SVP_γ: Given a basis of L, find $\mathbf{b} \in L$ with

$$0 < \|\mathbf{b}\| \leq \gamma \cdot \lambda(L).$$

- $\mathrm{Dec\text{-}SVP}_\gamma$: Given a basis of L, and $t > 0$, reply:

$$\text{YES if } \lambda(L) \leq t \text{ and NO if } \lambda(L) > \gamma \cdot t.$$

- NP-hard for any $\gamma \leq \mathcal{O}(1)$, under randomized reductions.
- Most likely not NP-hard for $\gamma \geq \sqrt{n}$.
- In P for $\gamma \geq 2^{\frac{n \log\log n}{\log n}}$.

Main computational problem: SVP

- SVP_γ: Given a basis of L, find $\mathbf{b} \in L$ with

$$0 < \|\mathbf{b}\| \leq \gamma \cdot \lambda(L).$$

- $\mathrm{Dec\text{-}SVP}_\gamma$: Given a basis of L, and $t > 0$, reply:

$$\text{YES if } \lambda(L) \leq t \quad \text{and NO if} \quad \lambda(L) > \gamma \cdot t.$$

- NP-hard for any $\gamma \leq \mathcal{O}(1)$, under randomized reductions.
- Most likely not NP-hard for $\gamma \geq \sqrt{n}$.
- In P for $\gamma \geq 2^{\frac{n \log\log n}{\log n}}$.

Main computational problem: SVP

- SVP_γ: Given a basis of L, find $\mathbf{b} \in L$ with

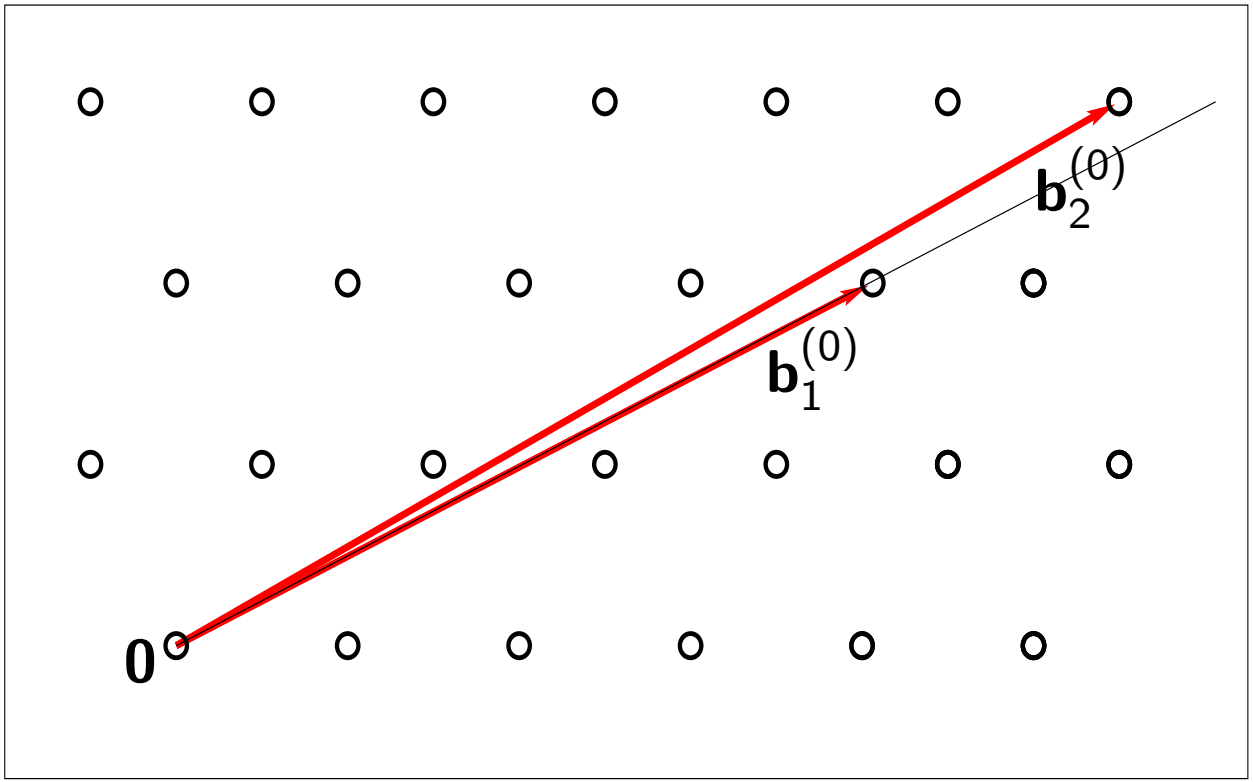
$$0 < \|\mathbf{b}\| \leq \gamma \cdot \lambda(L).$$

- $\mathrm{Dec\text{-}SVP}_\gamma$: Given a basis of L, and $t > 0$, reply:

$$\text{YES if } \lambda(L) \leq t \text{ and NO if } \lambda(L) > \gamma \cdot t.$$

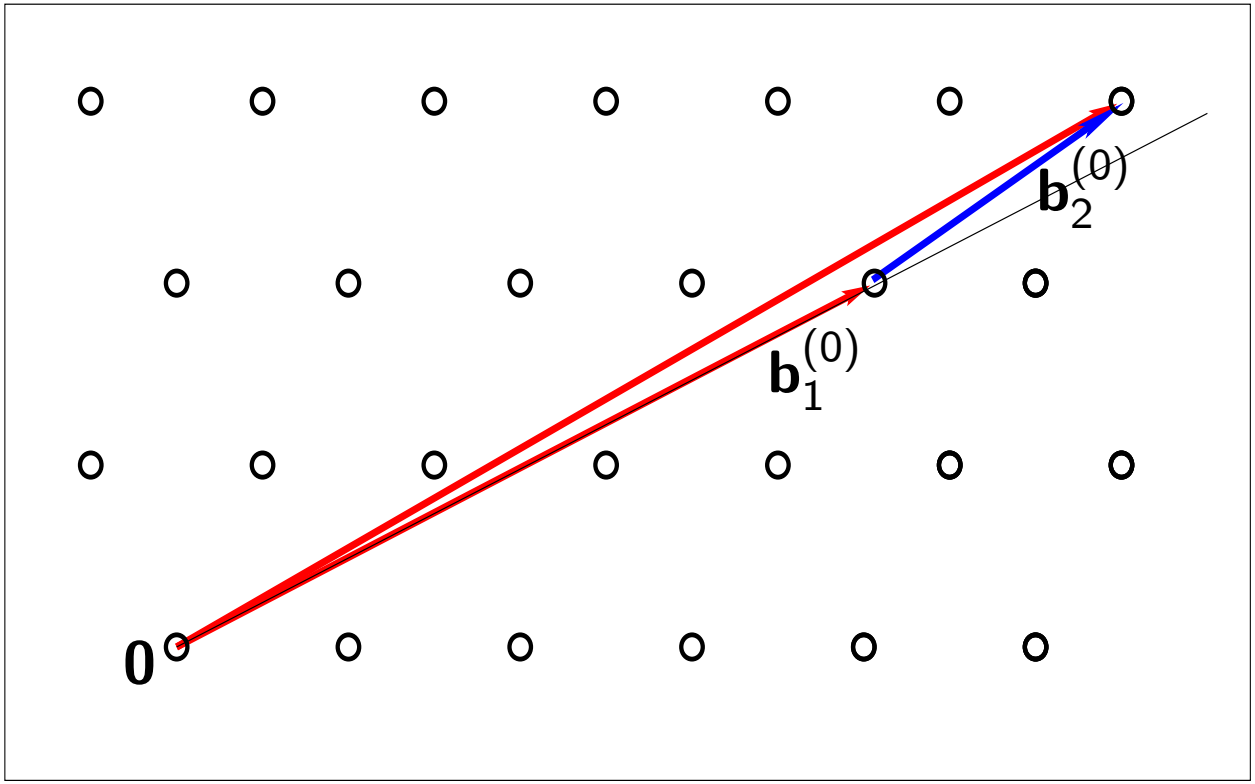
- NP-hard for any $\gamma \leq \mathcal{O}(1)$, under randomized reductions.
- Most likely not NP-hard for $\gamma \geq \sqrt{n}$.
- In P for $\gamma \geq 2^{\frac{n \log\log n}{\log n}}$.

SVP is easy in small dimensions!



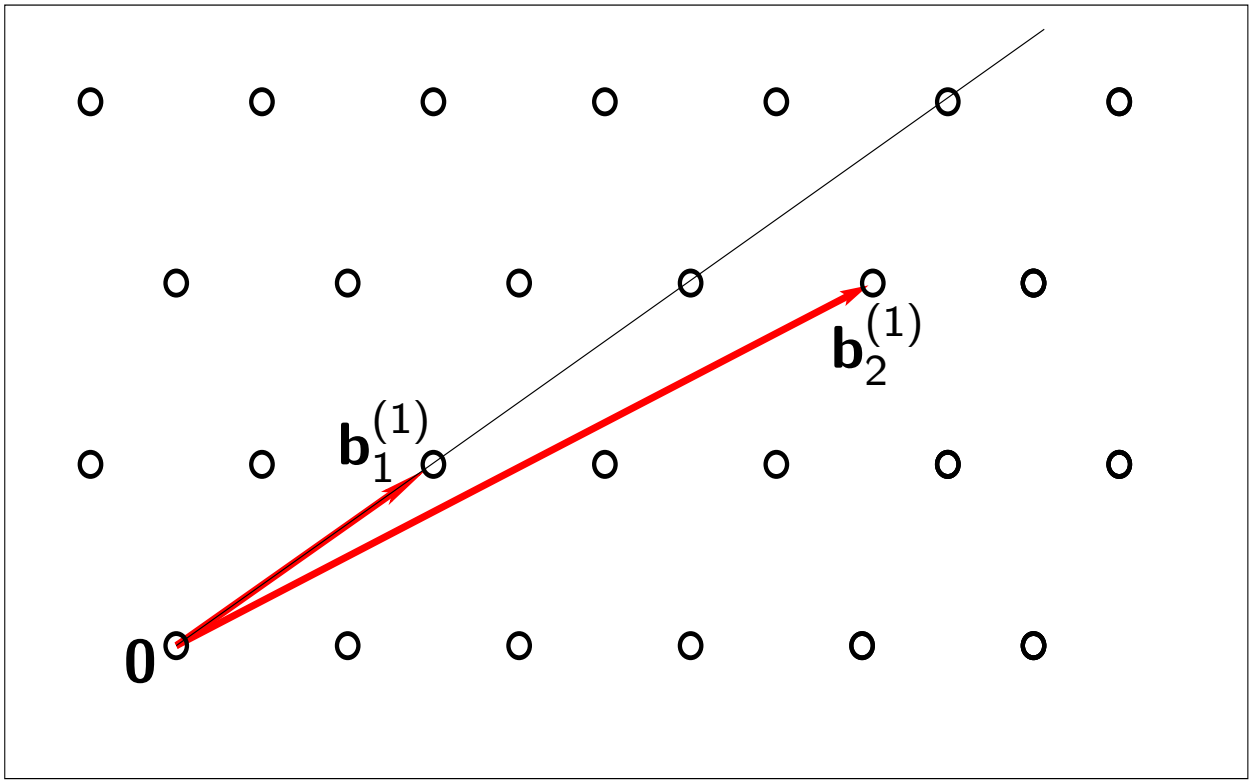
- That's almost Euclid's algorithm!
- Returns a vector reaching $\lambda(L)$.
- Runs in polynomial-time.

SVP is easy in small dimensions!



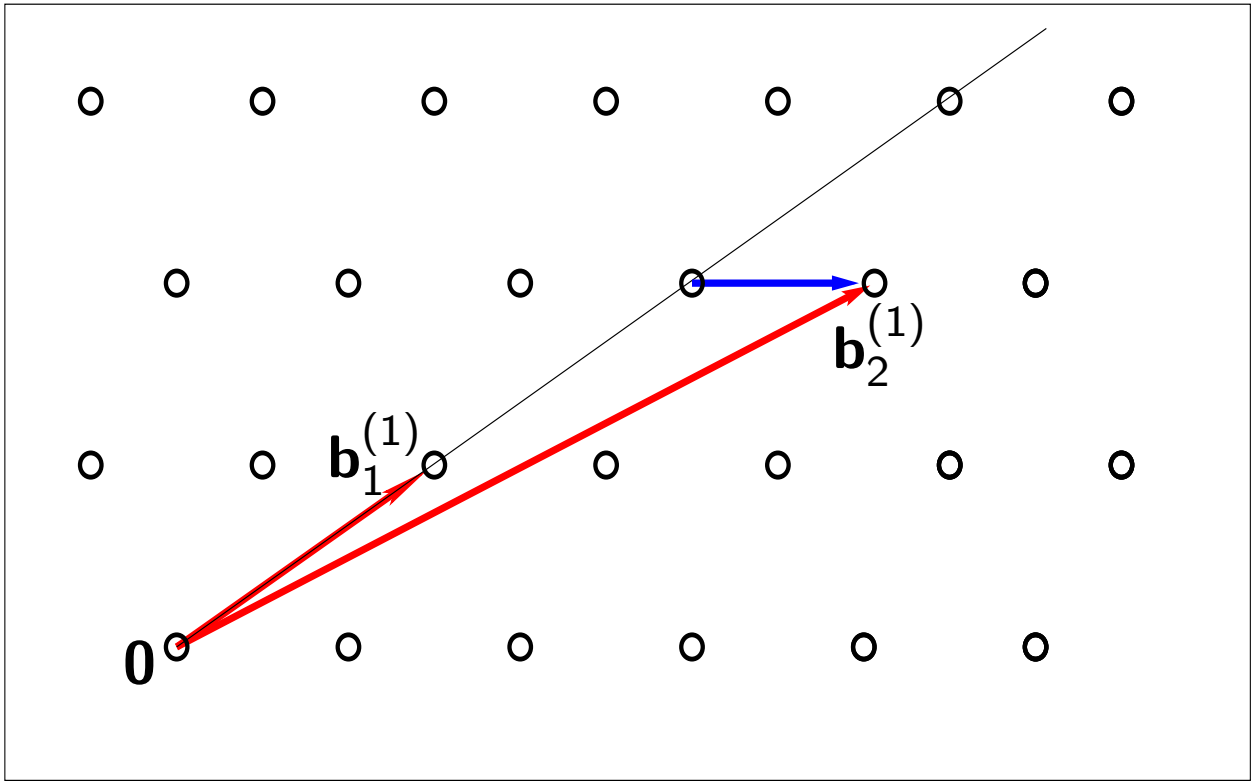
- That's almost Euclid's algorithm!
- Returns a vector reaching $\lambda(L)$.
- Runs in polynomial-time.

SVP is easy in small dimensions!



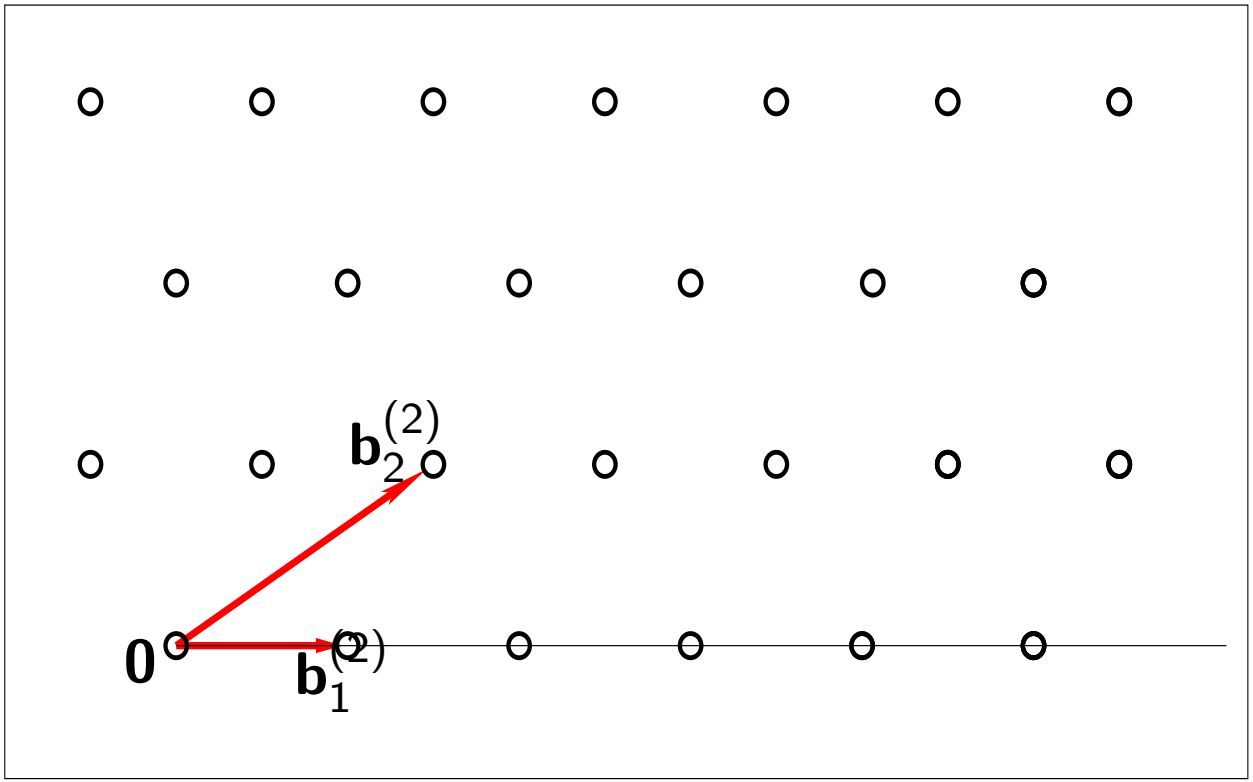
- That's almost Euclid's algorithm!
- Returns a vector reaching $\lambda(L)$.
- Runs in polynomial-time.

SVP is easy in small dimensions!



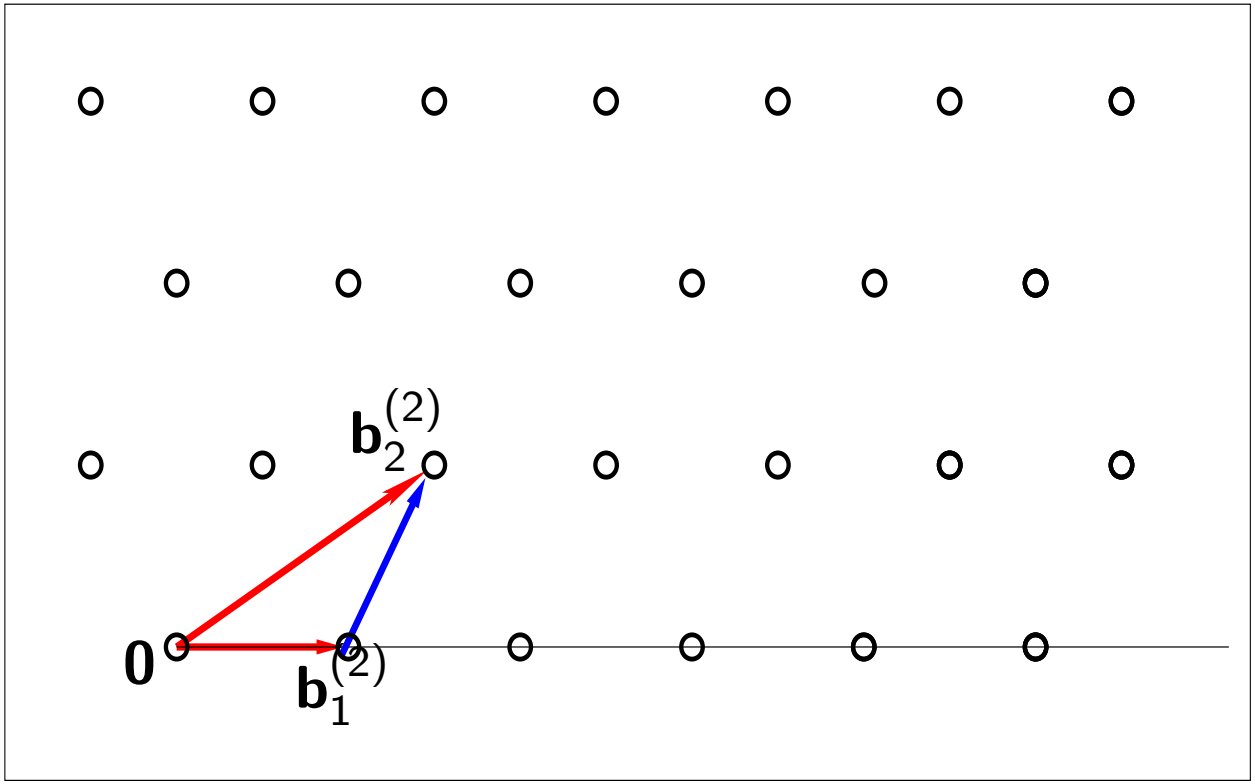
- That's almost Euclid's algorithm!
- Returns a vector reaching $\lambda(L)$.
- Runs in polynomial-time.

SVP is easy in small dimensions!



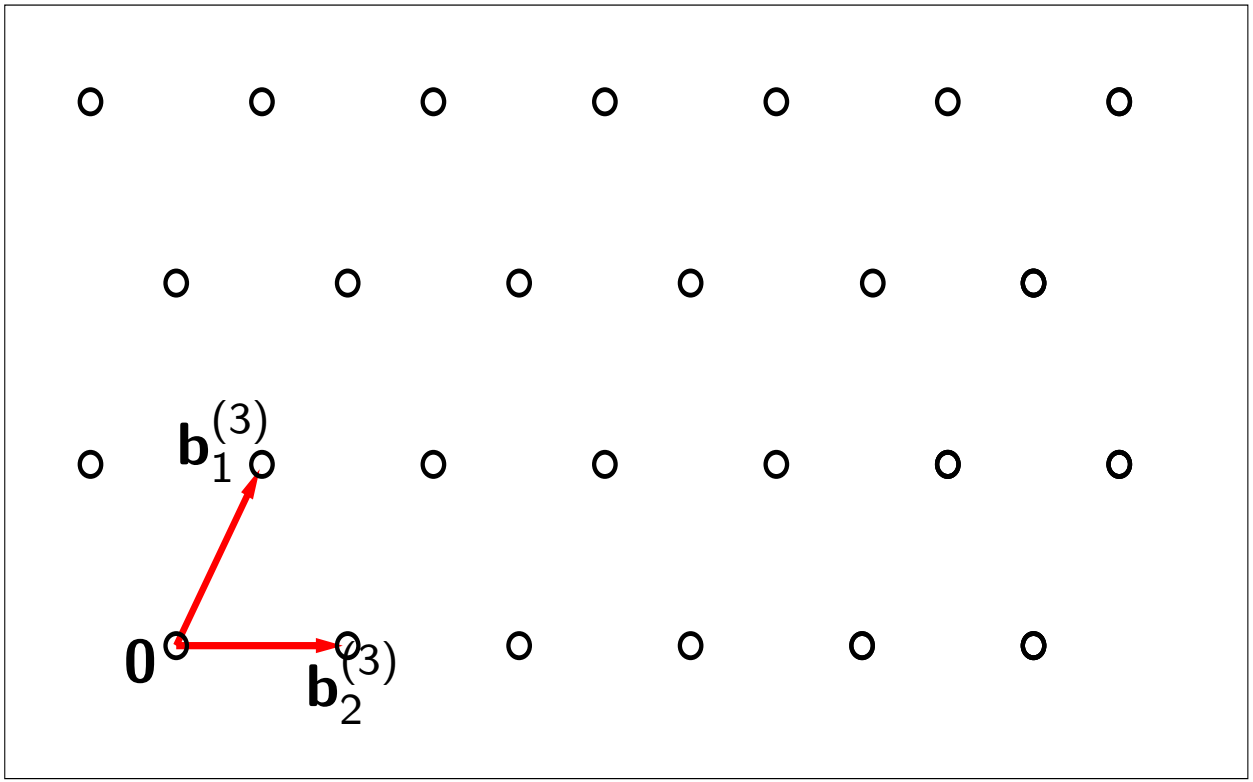
- That's almost Euclid's algorithm!
- Returns a vector reaching $\lambda(L)$.
- Runs in polynomial-time.

SVP is easy in small dimensions!



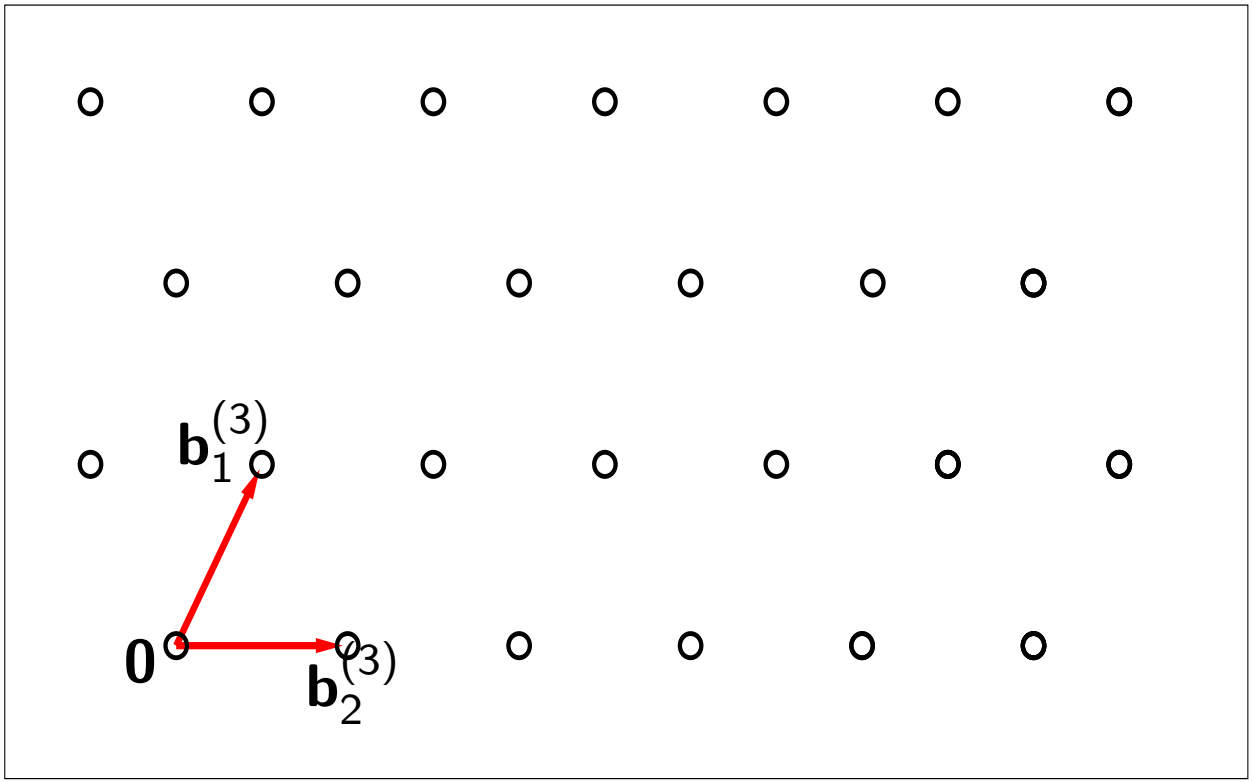
- That's almost Euclid's algorithm!
- Returns a vector reaching $\lambda(L)$.
- Runs in polynomial-time.

SVP is easy in small dimensions!



- That's almost Euclid's algorithm!
- Returns a vector reaching $\lambda(L)$.
- Runs in polynomial-time.

SVP is easy in small dimensions!



- That's almost Euclid's algorithm!
- Returns a vector reaching $\lambda(L)$.
- Runs in polynomial-time.

Plan of the talk

Plan of the talk:

1. Euclidean lattices
2. **Applications of euclidean lattices**
3. The LLL algorithm and its accelerations

1. Integer relation detection
2. Polynomial factorisation
3. Cryptanalysis

Plan of the talk

Plan of the talk:

1. Euclidean lattices
2. **Applications of euclidean lattices**
3. The LLL algorithm and its accelerations

1. Integer relation detection
2. Polynomial factorisation
3. Cryptanalysis

Finding small integer relations between real numbers

Lattices may be used to find the Bailey-Borwein-Plouffe formula:

$$\pi = \sum_{i \geq 0} \frac{1}{16^i} \left(\frac{4}{8i+1} - \frac{2}{8i+4} - \frac{1}{8i+5} - \frac{1}{8i+6} \right).$$

Interesting to compute a given hexadecimal digit of π

Assume we search a small integer relation between $y_1, \ldots, y_d \in \mathbb{R}$.

Consider $L := L[(\mathbf{b}_i)_i]$, with $B = \begin{bmatrix} y_1 & y_2 & \cdots & y_d \\ 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix}$

A relation $\mathbf{x} \in \mathbb{Z}^{d+1}$ leads to a small vector $(0, x_1, \ldots, x_d)^T$.

Using a large C makes it a shortest of L.

Finding small integer relations between real numbers

Lattices may be used to find the Bailey-Borwein-Plouffe formula:

$$\pi = \sum_{i \geq 0} \frac{1}{16^i} \left(\frac{4}{8i+1} - \frac{2}{8i+4} - \frac{1}{8i+5} - \frac{1}{8i+6} \right).$$

Interesting to compute a given hexadecimal digit of π

Assume we search a small integer relation between $y_1, \ldots, y_d \in \mathbb{R}$.

Consider $L := L[(\mathbf{b}_i)_i]$, with $B = \begin{bmatrix} y_1 & y_2 & \cdots & y_d \\ 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix}$

A relation $\mathbf{x} \in \mathbb{Z}^{d+1}$ leads to a small vector $(0, x_1, \ldots, x_d)^T$.
Using a large C makes it a shortest of $L \setminus 0$.

Finding small integer relations between real numbers

Lattices may be used to find the Bailey-Borwein-Plouffe formula:

$$\pi = \sum_{i \geq 0} \frac{1}{16^i} \left(\frac{4}{8i+1} - \frac{2}{8i+4} - \frac{1}{8i+5} - \frac{1}{8i+6} \right).$$

Interesting to compute a given hexadecimal digit of π
Assume we search a small integer relation between $y_1, \ldots, y_d \in \mathbb{R}$.

$$\text{Consider } L := L[(\mathbf{b}_i)_i], \quad \text{with } B = \begin{bmatrix} y_1 & y_2 & \cdots & y_d \\ 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix}$$

A relation $\mathbf{x} \in \mathbb{Z}^{d+1}$ leads to a small vector $(0, x_1, \ldots, x_d)^T$.
Using a large C makes it a shortest of $L \setminus \mathbf{0}$.

Finding small integer relations between real numbers

Lattices may be used to find the Bailey-Borwein-Plouffe formula:

$$\pi = \sum_{i \geq 0} \frac{1}{16^i} \left(\frac{4}{8i+1} - \frac{2}{8i+4} - \frac{1}{8i+5} - \frac{1}{8i+6} \right).$$

Interesting to compute a given hexadecimal digit of π
Assume we search a small integer relation between $y_1, \ldots, y_d \in \mathbb{R}$.

$$\text{Consider } L := L[(\mathbf{b}_i)_i], \quad \text{with } B = \begin{bmatrix} Cy_1 & Cy_2 & \ldots & Cy_d \\ 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix}$$

A relation $\mathbf{x} \in \mathbb{Z}^{d+1}$ leads to a small vector $(0, x_1, \ldots, x_d)^T$.
Using a large C makes it a shortest of $L \setminus \mathbf{0}$.

Factoring integer polynomials

This idea may be used to factor polynomials over $\mathbb{Z}[x]$.

Factoring polynomials with rational coefficients,
A. K. Lenstra, H. W. Lenstra Jr. and L. Lovász. Math. Ann., 1982

$\Rightarrow$ Cited 2544 times!

- Step 0: If $\deg P \leq 1$, then stop
- Step 1: Compute a root $\alpha \in \mathbb{C}$ of $P(x) \in \mathbb{Z}[x]$
- Step 2: Find the minimal polynomial $P_\alpha(x)$ of α, by searching for integer combinations between $1, \alpha, \ldots, \alpha^i$ for increasing i
- Step 3: Divide P by P_α and restart

Factoring integer polynomials

This idea may be used to factor polynomials over $\mathbb{Z}[x]$.

Factoring polynomials with rational coefficients,
A. K. Lenstra, H. W. Lenstra Jr. and L. Lovász. Math. Ann., 1982

$\Rightarrow$ Cited 2544 times!

- Step 0: If $\deg P \leq 1$, then stop
- Step 1: Compute a root $\alpha \in \mathbb{C}$ of $P(x) \in \mathbb{Z}[x]$
- Step 2: Find the minimal polynomial $P_\alpha(x)$ of α, by searching for integer combinations between $1, \alpha, \ldots, \alpha^i$ for increasing i
- Step 3: Divide P by P_α and restart

Cryptanalysis of RSA with small decryption exponent

RSA:

- $N = p \cdot q$ with $p, q \approx \sqrt{N}$ prime
- e, d with $e \cdot d = 1 \bmod (p-1) \cdot (q-1)$
- $PK = (N, e)$, $sk = d$

RSA with small decryption exponent: $d \ll N^{1/4}$, $e \approx N$.

Some orders of magnitude:

$$\begin{array}{rcl} e \approx N & \Rightarrow & k \approx d \ll N^{1/4} \\ ed = 1 + k\phi(N) & \Rightarrow & |ed - kN| \ll N^{3/4} \end{array}$$

Cryptanalysis: Consider $L(\mathbf{b}_1, \mathbf{b}_2)$ with $[\mathbf{b}_1 | \mathbf{b}_2] = \begin{bmatrix} e & N \\ \sqrt{N} & 0 \end{bmatrix}$.

$\lambda(L)$ expected to be $\approx \sqrt{\det(L)} = N^{3/4}$.
But $\|d\mathbf{b}_1 - k\mathbf{b}_2\| \ll N^{3/4}$ ...

Cryptanalysis of RSA with small decryption exponent

RSA:

- $N = p \cdot q$ with $p, q \approx \sqrt{N}$ prime
- e, d with $e \cdot d = 1 \bmod (p-1) \cdot (q-1)$
- $PK = (N, e)$, $sk = d$

RSA with small decryption exponent: $d \ll N^{1/4}$, $e \approx N$.

Some orders of magnitude:

$$\begin{array}{rcl} e \approx N & \Rightarrow & k \approx d \ll N^{1/4} \\ ed = 1 + k\phi(N) & \Rightarrow & |ed - kN| \ll N^{3/4} \end{array}$$

Cryptanalysis: Consider $L(\mathbf{b}_1, \mathbf{b}_2)$ with $[\mathbf{b}_1|\mathbf{b}_2] = \begin{bmatrix} e & N \\ \sqrt{N} & 0 \end{bmatrix}$.

$\lambda(L)$ expected to be $\approx \sqrt{\det(L)} = N^{3/4}$.
But $\|d\mathbf{b}_1 - k\mathbf{b}_2\| \ll N^{3/4} \ldots$

Cryptanalysis of RSA with small decryption exponent

RSA:

- $N = p \cdot q$ with $p, q \approx \sqrt{N}$ prime
- e, d with $e \cdot d = 1 \bmod (p-1) \cdot (q-1)$
- $PK = (N, e)$, $sk = d$

RSA with small decryption exponent: $d \ll N^{1/4}$, $e \approx N$.

Some orders of magnitude:

$$\begin{array}{rcl} e \approx N & \Rightarrow & k \approx d \ll N^{1/4} \\ ed = 1 + k\phi(N) & \Rightarrow & |ed - kN| \ll N^{3/4} \end{array}$$

Cryptanalysis: Consider $L(\mathbf{b}_1, \mathbf{b}_2)$ with $[\mathbf{b}_1|\mathbf{b}_2] = \begin{bmatrix} e & N \\ \sqrt{N} & 0 \end{bmatrix}$.

$\lambda(L)$ expected to be $\approx \sqrt{\det(L)} = N^{3/4}$.
But $\|d\mathbf{b}_1 - k\mathbf{b}_2\| \ll N^{3/4} \ldots$

Cryptanalysis of RSA with small decryption exponent

RSA:

- $N = p \cdot q$ with $p, q \approx \sqrt{N}$ prime
- e, d with $e \cdot d = 1 \bmod (p-1) \cdot (q-1)$
- $PK = (N, e)$, $sk = d$

RSA with small decryption exponent: $d \ll N^{1/4}$, $e \approx N$.

Some orders of magnitude:

$$\begin{array}{rcl} e \approx N & \Rightarrow & k \approx d \ll N^{1/4} \\ ed = 1 + k\phi(N) & \Rightarrow & |ed - kN| \ll N^{3/4} \end{array}$$

Cryptanalysis: Consider $L(\mathbf{b}_1, \mathbf{b}_2)$ with $[\mathbf{b}_1 | \mathbf{b}_2] = \begin{bmatrix} e & N \\ \sqrt{N} & 0 \end{bmatrix}$.

$\lambda(L)$ expected to be $\approx \sqrt{\det(L)} = N^{3/4}$.
But $\|d\mathbf{b}_1 - k\mathbf{b}_2\| \ll N^{3/4} \ldots$

Cryptographic design

Lattice-based cryptography:

- Secret key: very short basis of a lattice
- Public key: long basis of the same lattice
- Relies on the assumed hardness of SVP

Very popular research topic, because LBC is:

- More secure: post-quantum
- More efficient: no modular exponentiation
- More flexible: fully homomorphic encryption

Lattice reduction can be used to attack these protocols.

Cryptographic design

Lattice-based cryptography:

- Secret key: very short basis of a lattice
- Public key: long basis of the same lattice
- Relies on the assumed hardness of SVP

Very popular research topic, because LBC is:

- More secure: post-quantum
- More efficient: no modular exponentiation
- More flexible: fully homomorphic encryption

Lattice reduction can be used to attack these protocols.

Cryptographic design

Lattice-based cryptography:

- Secret key: very short basis of a lattice
- Public key: long basis of the same lattice
- Relies on the assumed hardness of SVP

Very popular research topic, because LBC is:

- More secure: post-quantum
- More efficient: no modular exponentiation
- More flexible: fully homomorphic encryption

Lattice reduction can be used to attack these protocols.

Plan of the talk

Plan of the talk:

1. Euclidean lattices
2. Applications of euclidean lattices
3. **The LLL algorithm and its accelerations**

Gram-Schmidt orthogonalization (GSO)

$(\mathbf{b}_i)_i$ linearly independent.

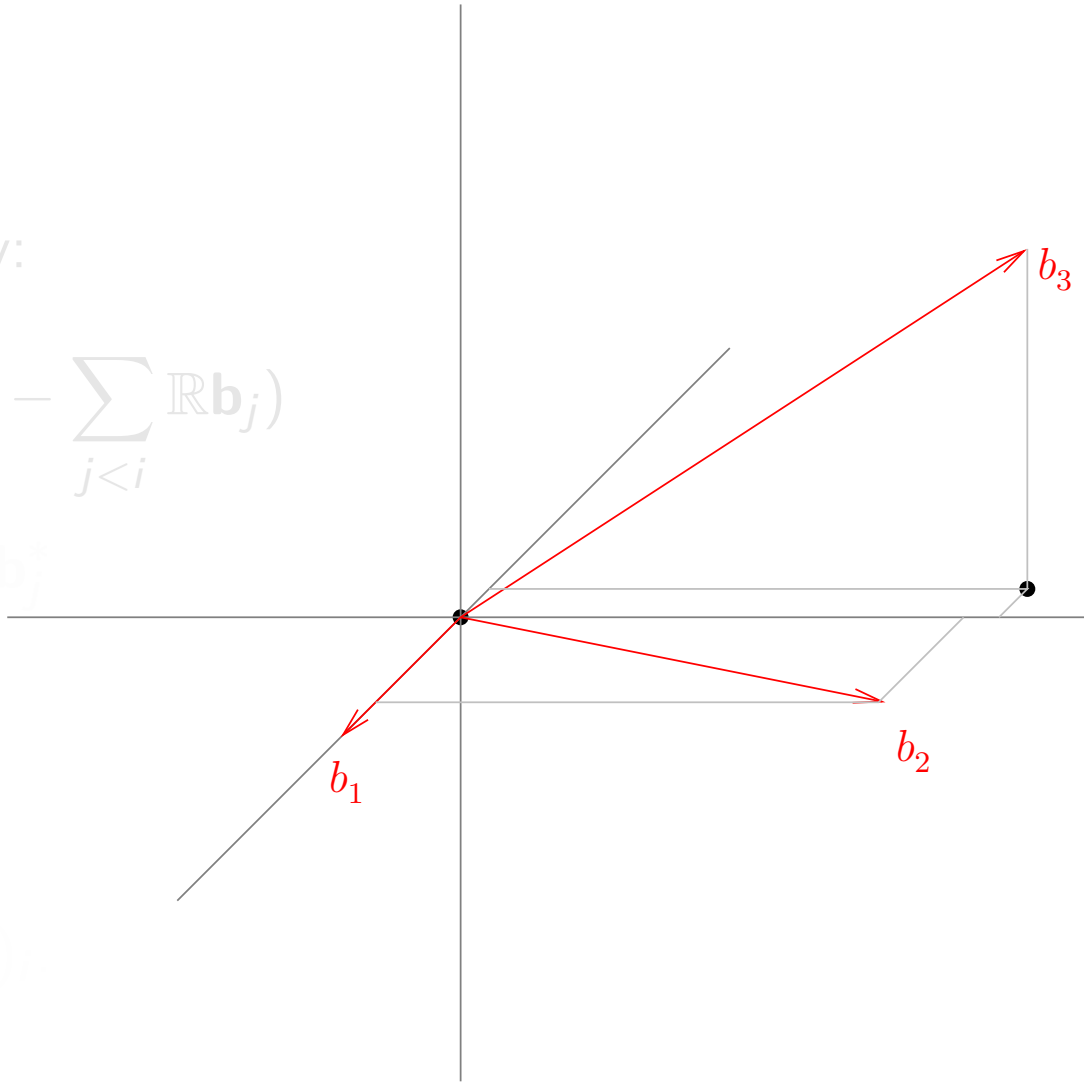
The GSO $(\mathbf{b}_i^*)_i$ is defined by:

$$\forall i, \quad \mathbf{b}_i^* = \text{argmin}_{\|\cdot\|}(\mathbf{b}_i - \sum_{j<i} \mathbb{R}\mathbf{b}_j)$$

$$= \mathbf{b}_i - \sum_{j<i} \mu_{i,j}\mathbf{b}_j^*$$

$$\forall i > j, \mu_{i,j} = \frac{(\mathbf{b}_i, \mathbf{b}_j^*)}{\|\mathbf{b}_j^*\|^2}.$$

Triangularization of B : $(\mathbf{b}_i)_i$.



Exercise: For any basis $(\mathbf{b}_i)_i$ of L, we have $\lambda(L) \geq \min_i \|\mathbf{b}_i^*\|$.

Gram-Schmidt orthogonalization (GSO)

$(\mathbf{b}_i)_i$ linearly independent.

The GSO $(\mathbf{b}_i^*)_i$ is defined by:

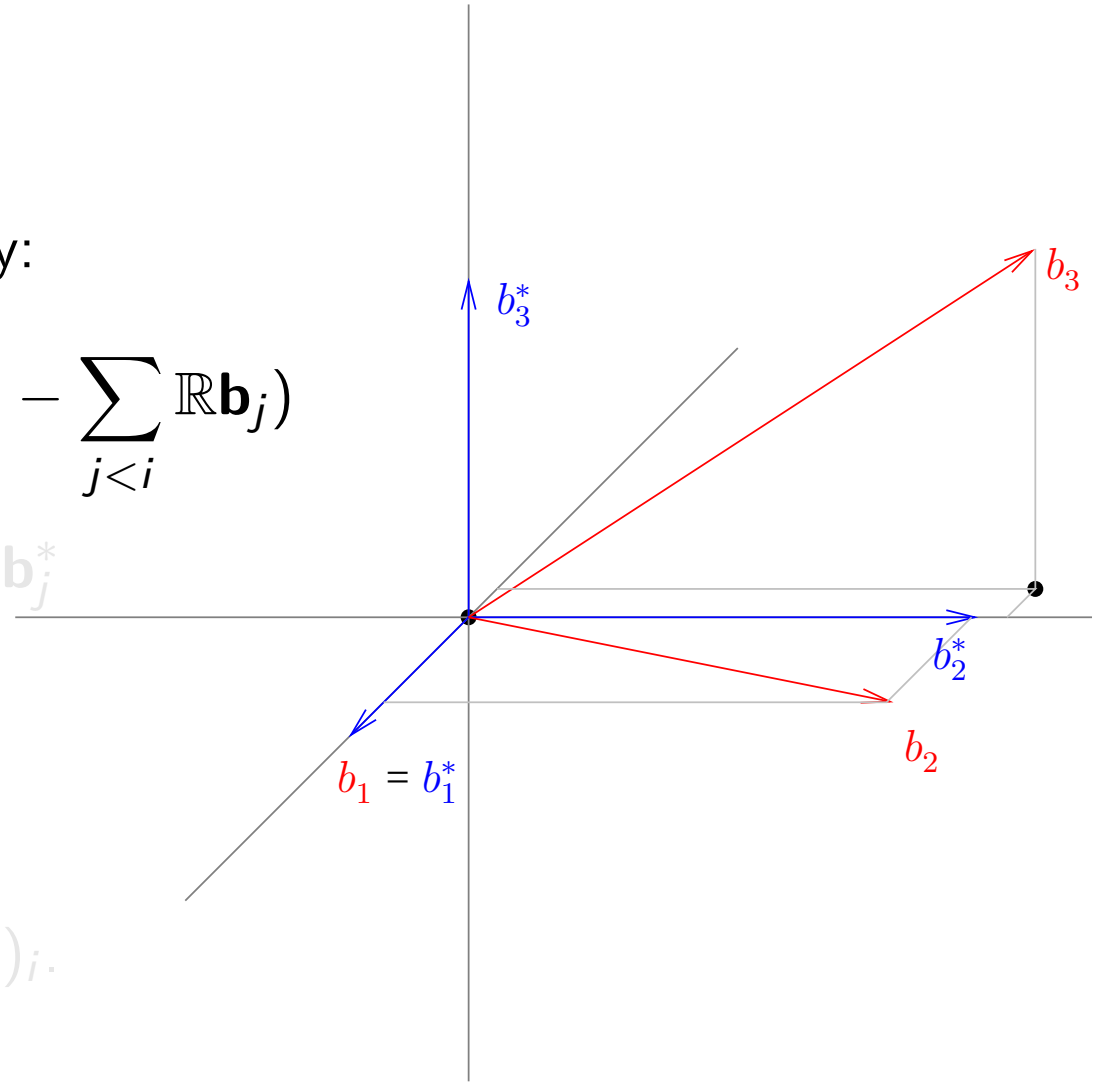
$$\forall i, \quad \mathbf{b}_i^* = \text{argmin}_{\|\cdot\|}(\mathbf{b}_i - \sum_{j<i} \mathbb{R}\mathbf{b}_j)$$

$$= \mathbf{b}_i - \sum_{j<i} \mu_{i,j}\mathbf{b}_j^*$$

$$\forall i > j, \mu_{i,j} = \frac{(\mathbf{b}_i, \mathbf{b}_j^*)}{\|\mathbf{b}_j^*\|^2}.$$

Triangularisation of $B = (\mathbf{b}_i)_i$.

Exercise: For any basis $(\mathbf{b}_i)_i$ of L, we have $\lambda(L) \geq \min_i \|\mathbf{b}_i^*\|$.

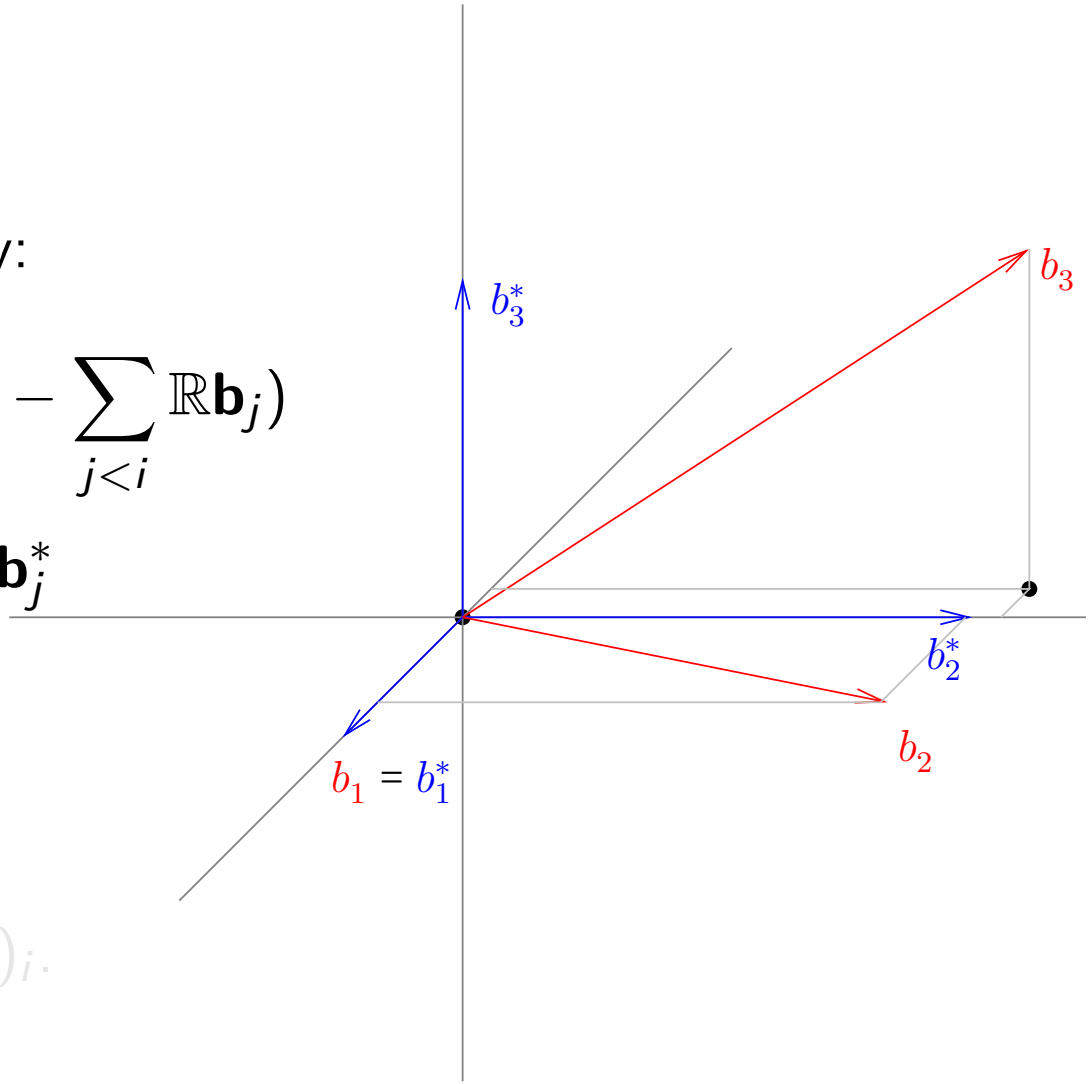


Gram-Schmidt orthogonalization (GSO)

$(\mathbf{b}_i)_i$ linearly independent.

The GSO $(\mathbf{b}_i^*)_i$ is defined by:

$$\begin{aligned} \forall i, \quad \mathbf{b}_i^* &= \text{argmin}_{\|\cdot\|}(\mathbf{b}_i - \sum_{j<i} \mathbb{R}\mathbf{b}_j) \\ &= \mathbf{b}_i - \sum_{j<i} \mu_{i,j}\mathbf{b}_j^* \\ \forall i > j, \mu_{i,j} &= \frac{(\mathbf{b}_i, \mathbf{b}_j^*)}{\|\mathbf{b}_j^*\|^2}. \end{aligned}$$



Triangularisation of $B = (\mathbf{b}_i)_i$.

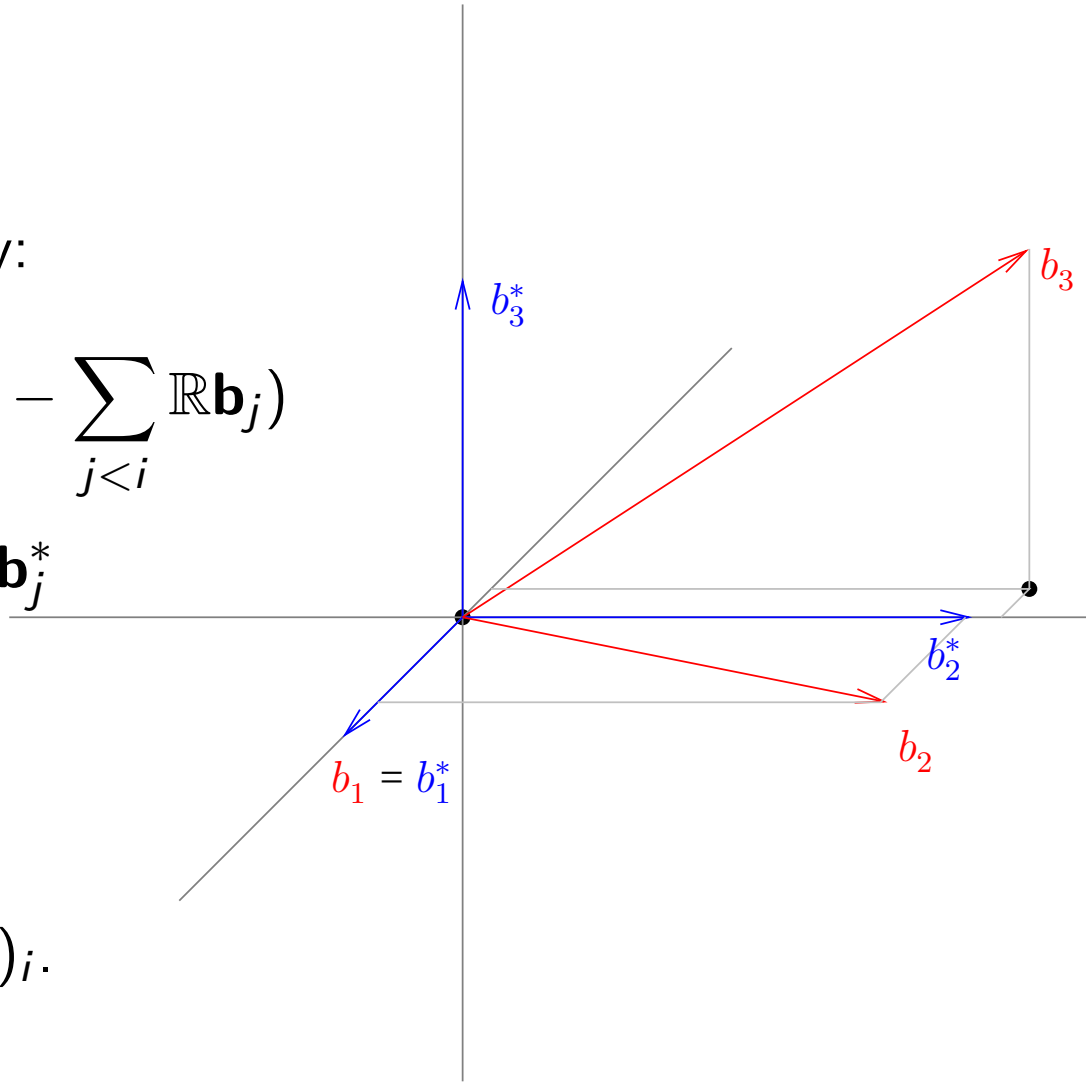
Exercise: For any basis $(\mathbf{b}_i)_i$ of L, we have $\lambda(L) \geq \min_i \|\mathbf{b}_i^*\|$.

Gram-Schmidt orthogonalization (GSO)

$(\mathbf{b}_i)_i$ linearly independent.

The GSO $(\mathbf{b}_i^*)_i$ is defined by:

$$\begin{aligned} \forall i, \quad \mathbf{b}_i^* &= \text{argmin}_{\|\cdot\|}(\mathbf{b}_i - \sum_{j<i} \mathbb{R}\mathbf{b}_j) \\ &= \mathbf{b}_i - \sum_{j<i} \mu_{i,j}\mathbf{b}_j^* \\ \forall i > j, \mu_{i,j} &= \frac{(\mathbf{b}_i, \mathbf{b}_j^*)}{\|\mathbf{b}_j^*\|^2}. \end{aligned}$$



Triangularisation of $B = (\mathbf{b}_i)_i$.

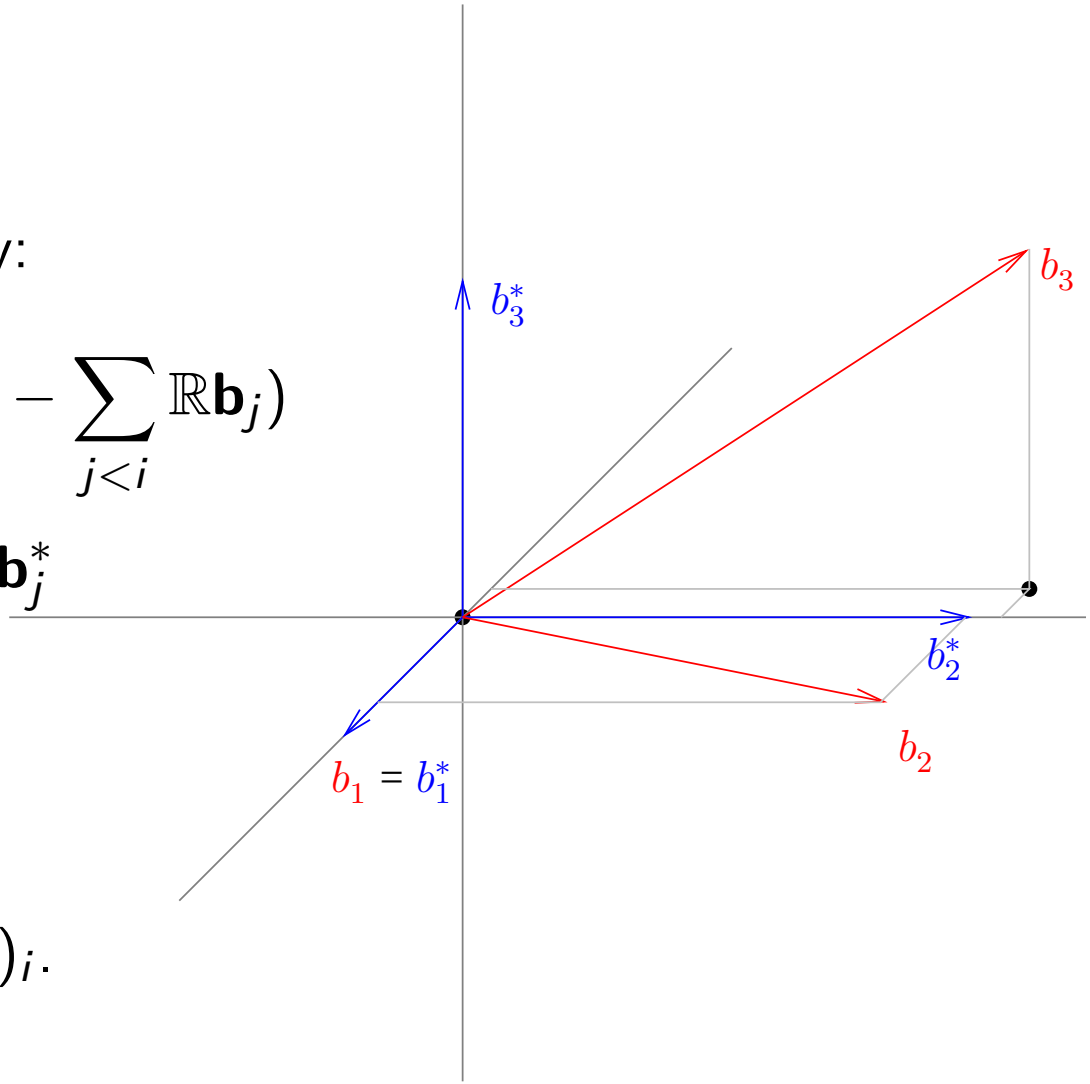
Exercise: For any basis $(\mathbf{b}_i)_i$ of L, we have $\lambda(L) \geq \min_i \|\mathbf{b}_i^*\|$.

Gram-Schmidt orthogonalization (GSO)

$(\mathbf{b}_i)_i$ linearly independent.

The GSO $(\mathbf{b}_i^*)_i$ is defined by:

$$\begin{aligned} \forall i, \quad \mathbf{b}_i^* &= \text{argmin}_{\|\cdot\|}(\mathbf{b}_i - \sum_{j<i} \mathbb{R}\mathbf{b}_j) \\ &= \mathbf{b}_i - \sum_{j<i} \mu_{i,j}\mathbf{b}_j^* \\ \forall i > j, \mu_{i,j} &= \frac{(\mathbf{b}_i, \mathbf{b}_j^*)}{\|\mathbf{b}_j^*\|^2}. \end{aligned}$$



Triangularisation of $B = (\mathbf{b}_i)_i$.

Exercise: For any basis $(\mathbf{b}_i)_i$ of L, we have $\lambda(L) \geq \min_i \|\mathbf{b}_i^*\|$.

The Lenstra-Lenstra-Lovász reduction

Let $\delta \in (1/4, 1)$. A basis $B = (\mathbf{b}_i)_{i \leq n} \in \mathbb{R}^{n \times n}$ is said LLL-reduced if:

- $\forall i, j : \ |\mu_{i,j}| \leq 1/2$ [size-reduction]
- $\forall i : \ \delta \cdot \|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|\mathbf{b}_i^*\|^2$ [Lovász' condition]

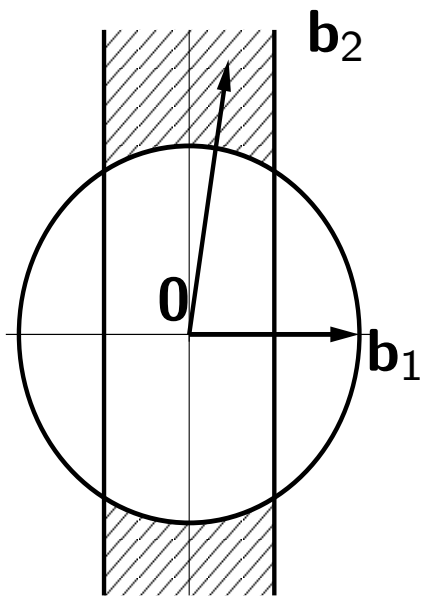
The Lenstra-Lenstra-Lovász reduction

Let $\delta \in (1/4, 1)$. A basis $B = (\mathbf{b}_i)_{i \leq n} \in \mathbb{R}^{n \times n}$ is said LLL-reduced if:

- $\forall i, j : \ |\mu_{i,j}| \leq 1/2$ [size-reduction]
- $\forall i : \ \delta \cdot \|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|\mathbf{b}_i^*\|^2$ [Lovász' condition]

The $\|\mathbf{b}_i^*\|$'s can't drop too fast:
For all i: $\|\mathbf{b}_{i+1}^*\|^2 \geq (\delta - \frac{1}{4})\|\mathbf{b}_i^*\|^2$.
$\Rightarrow \lambda(L) \leq \|\mathbf{b}_1\| \leq 2^{\mathcal{O}(n)} \cdot \lambda(L)$

$\delta < 1$ is crucial to get a polynomial complexity.



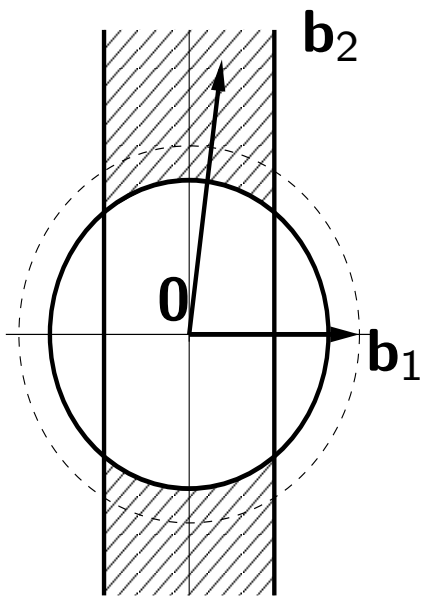
The Lenstra-Lenstra-Lovász reduction

Let $\delta \in (1/4, 1)$. A basis $B = (\mathbf{b}_i)_{i \leq n} \in \mathbb{R}^{n \times n}$ is said LLL-reduced if:

- $\forall i, j : \ |\mu_{i,j}| \leq 1/2$ [size-reduction]
- $\forall i : \ \delta \cdot \|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|\mathbf{b}_i^*\|^2$ [Lovász' condition]

The $\|\mathbf{b}_i^*\|$'s can't drop too fast:
For all i: $\|\mathbf{b}_{i+1}^*\|^2 \geq (\delta - \frac{1}{4})\|\mathbf{b}_i^*\|^2$.
$\Rightarrow \lambda(L) \leq \|\mathbf{b}_1\| \leq 2^{\mathcal{O}(n)} \cdot \lambda(L)$

$\delta < 1$ is crucial to get a polynomial complexity.



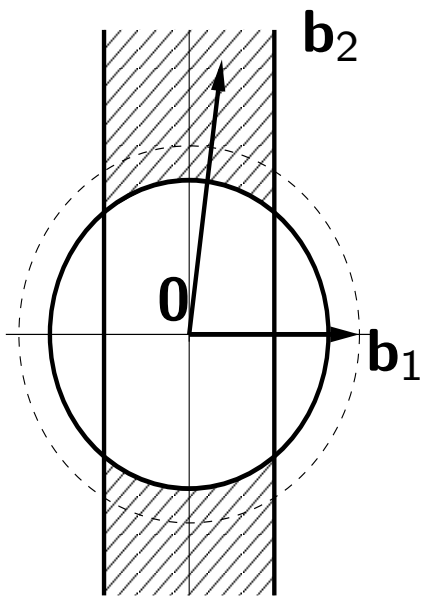
The Lenstra-Lenstra-Lovász reduction

Let $\delta \in (1/4, 1)$. A basis $B = (\mathbf{b}_i)_{i \leq n} \in \mathbb{R}^{n \times n}$ is said LLL-reduced if:

- $\forall i, j : \ |\mu_{i,j}| \leq 1/2$ [size-reduction]
- $\forall i : \ \delta \cdot \|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|\mathbf{b}_i^*\|^2$ [Lovász' condition]

The $\|\mathbf{b}_i^*\|$'s can't drop too fast:
For all i: $\|\mathbf{b}_{i+1}^*\|^2 \geq (\delta - \frac{1}{4})\|\mathbf{b}_i^*\|^2$.
$\Rightarrow \lambda(L) \leq \|\mathbf{b}_1\| \leq 2^{\mathcal{O}(n)} \cdot \lambda(L)$

$\delta < 1$ is crucial to get a polynomial complexity.



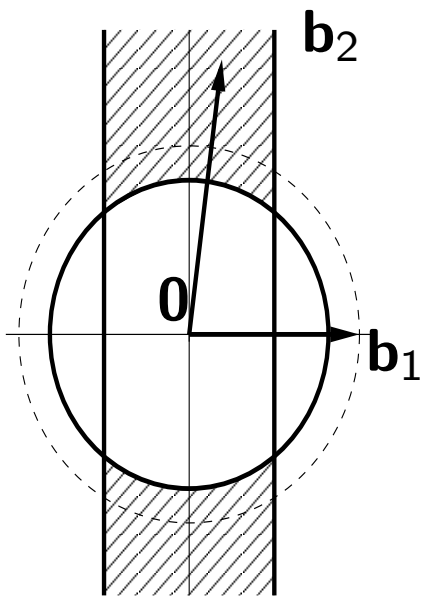
The Lenstra-Lenstra-Lovász reduction

Let $\delta \in (1/4, 1)$. A basis $B = (\mathbf{b}_i)_{i \leq n} \in \mathbb{R}^{n \times n}$ is said LLL-reduced if:

- $\forall i, j : \ |\mu_{i,j}| \leq 1/2$ [size-reduction]
- $\forall i : \ \delta \cdot \|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|\mathbf{b}_i^*\|^2$ [Lovász' condition]

The $\|\mathbf{b}_i^*\|$'s can't drop too fast:
For all i: $\|\mathbf{b}_{i+1}^*\|^2 \geq (\delta - \frac{1}{4})\|\mathbf{b}_i^*\|^2$.
$\Rightarrow \lambda(L) \leq \|\mathbf{b}_1\| \leq 2^{\mathcal{O}(n)} \cdot \lambda(L)$

$\delta < 1$ is crucial to get a polynomial complexity.



The Lenstra-Lenstra-Lovász algorithm

Let $\delta \in (1/4, 1)$. A basis $B = (\mathbf{b}_i)_{i \leq n} \in \mathbb{R}^{n \times n}$ is said LLL-reduced if:

- $\forall i, j : \ |\mu_{i,j}| \leq 1/2$ [size-reduction]
- $\forall i : \ \delta \cdot \|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|\mathbf{b}_i^*\|^2$ [Lovász' condition]

1. Enforce size-reduction, using a modified Gaussian elimination
2. If there is an i with $\delta \cdot \|\mathbf{b}_i^*\|^2 > \|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|\mathbf{b}_i^*\|^2$, swap $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$, then go to Step 1
3. Return the current basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$

$\Rightarrow$ Correctness is trivial

$\Rightarrow$ Termination is much less so: $\mathcal{O}(n^2 \log \max_i \|\mathbf{b}_i^{init}\|)$ iterations

The Lenstra-Lenstra-Lovász algorithm

Let $\delta \in (1/4, 1)$. A basis $B = (\mathbf{b}_i)_{i \leq n} \in \mathbb{R}^{n \times n}$ is said LLL-reduced if:

- $\forall i, j : \ |\mu_{i,j}| \leq 1/2$ [size-reduction]
- $\forall i : \ \delta \cdot \|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|\mathbf{b}_i^*\|^2$ [Lovász' condition]

1. Enforce size-reduction, using a modified Gaussian elimination
2. If there is an i with $\delta \cdot \|\mathbf{b}_i^*\|^2 > \|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|\mathbf{b}_i^*\|^2$, swap $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$, then go to Step 1
3. Return the current basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$

$\Rightarrow$ Correctness is trivial

$\Rightarrow$ Termination is much less so: $\mathcal{O}(n^2 \log \max_i \|\mathbf{b}_i^{init}\|)$ iterations

The Lenstra-Lenstra-Lovász algorithm

Let $\delta \in (1/4, 1)$. A basis $B = (\mathbf{b}_i)_{i \leq n} \in \mathbb{R}^{n \times n}$ is said LLL-reduced if:

- $\forall i, j : \ |\mu_{i,j}| \leq 1/2$ [size-reduction]
- $\forall i : \ \delta \cdot \|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|\mathbf{b}_i^*\|^2$ [Lovász' condition]

1. Enforce size-reduction, using a modified Gaussian elimination
2. If there is an i with $\delta \cdot \|\mathbf{b}_i^*\|^2 > \|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|\mathbf{b}_i^*\|^2$, swap $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$, then go to Step 1
3. Return the current basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$

$\Rightarrow$ Correctness is trivial

$\Rightarrow$ Termination is much less so: $\mathcal{O}(n^2 \log \max_i \|\mathbf{b}_i^{init}\|)$ iterations

Complexity of LLL and practical run-time

[LLL82,Kaltofen83]

The LLL algorithm terminates within $\mathcal{O}(n^4\beta^2(n+\beta))$ bit operations, with $\beta = \log \max_i \|\mathbf{b}_i^{init}\|$.

With MAGMA V2.16:

```
> n := 25; B := RMatrixSpace(Integers(),n,n)!0;
> for i:=1 to 25 do
>    B[i][i]:=1; B[i][1]:=RandomBits(2000);
> end for;
> time C := LLL(B:Method:=``Integral'');
```

Time: 11.700

```
> time C := LLL(B);
```

Time: 0.240

LLL in practice: the numeric-symbolic approach

The Gram-Schmidt computations dominate the cost

Odlyzko's hybrid approach

Replace the rational computations on the GSO by floating-point approximations, but keep the basis operations exact

Floating-point numbers: $x_1.x_2x_3\ldots x_p \cdot B^e$, where:

- p is the precision
- B is the base, and $x_i \in \{0, \ldots, B-1\}$ for all i
- $e \in \mathbb{Z}$ is the exponent
- $x_1.x_2\ldots x_p$ is called the mantissa

Arithmetic: $fl(a\ op\ b)$ is a nearest floating-point number to $a\ op\ b$, for any $op \in \{+, -, /, \times\}$.

LLL in practice: the numeric-symbolic approach

The Gram-Schmidt computations dominate the cost

Odlyzko's hybrid approach

Replace the rational computations on the GSO by floating-point approximations, but keep the basis operations exact

Floating-point numbers: $x_1.x_2x_3\ldots x_p \cdot B^e$, where:

- p is the precision
- B is the base, and $x_i \in \{0, \ldots, B-1\}$ for all i
- $e \in \mathbb{Z}$ is the exponent
- $x_1.x_2\ldots x_p$ is called the mantissa

Arithmetic: $fl(a\ op\ b)$ is a nearest floating-point number to $a\ op\ b$, for any $op \in \{+, -, /, \times\}$.

LLL in practice: the numeric-symbolic approach

The Gram-Schmidt computations dominate the cost

Odlyzko's hybrid approach

Replace the rational computations on the GSO by floating-point approximations, but keep the basis operations exact

Floating-point numbers: $x_1.x_2x_3 \ldots x_p \cdot B^e$, where:

- p is the precision
- B is the base, and $x_i \in \{0, \ldots, B-1\}$ for all i
- $e \in \mathbb{Z}$ is the exponent
- $x_1.x_2 \ldots x_p$ is called the mantissa

Arithmetic: $fl(a\ op\ b)$ is a nearest floating-point number to $a\ op\ b$, for any $op \in \{+, -, /, \times\}$.

Odlyzko's hybrid approach is only heuristic

Odlyzko's hybrid approach

Replace the rational computations on the GSO by floating-point approximations, but keep the basis operations exact

Rationale: If p is small, floating-point arithmetic can be used to efficiently simulate rational arithmetic.
$\Rightarrow$ In practice: we aim for 53-bit machine precision

But Odlyzko's approach is heuristic:

- Fp arithmetic is inexact: $fl(a\ op\ b)$ differs from $a\ op\ b$ by little, but errors can be amplified in case of a sequence of operations.
- Odlyzko's LLL suffers from infinite loops
- Odlyzko's LLL can return incorrect answers

Odlyzko's hybrid approach is only heuristic

Odlyzko's hybrid approach

Replace the rational computations on the GSO by floating-point approximations, but keep the basis operations exact

Rationale: If p is small, floating-point arithmetic can be used to efficiently simulate rational arithmetic.
$\Rightarrow$ In practice: we aim for 53-bit machine precision

But Odlyzko's approach is heuristic:

- Fp arithmetic is inexact: $fl(a\ op\ b)$ differs from $a\ op\ b$ by little, but errors can be amplified in case of a sequence of operations.
- Odlyzko's LLL suffers from infinite loops
- Odlyzko's LLL can return incorrect answers

Odlyzko's hybrid approach is only heuristic

Odlyzko's hybrid approach

Replace the rational computations on the GSO by floating-point approximations, but keep the basis operations exact

Rationale: If p is small, floating-point arithmetic can be used to efficiently simulate rational arithmetic.
$\Rightarrow$ In practice: we aim for 53-bit machine precision

But Odlyzko's approach is heuristic:

- Fp arithmetic is inexact: $fl(a\ op\ b)$ differs from $a\ op\ b$ by little, but errors can be amplified in case of a sequence of operations.
- Odlyzko's LLL suffers from infinite loops
- Odlyzko's LLL can return incorrect answers

Making the numeric-symbolic approach rigorous

Underlying mathematical phenomenon

Any LLL-reduced basis is well-conditioned with respect to GSO computations

- Well-conditioned? The GSO computed in small precision fp arithmetic is close to the true GSO

$\Rightarrow$ We'd like to rely on LLL-reduced bases as much as we can

Use a greedy LLL algorithm:

- Consider the first i s.t. $\mathbf{b}_1, \ldots, \mathbf{b}_i$ is not LLL-reduced
- $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$ is well-conditioned, and its related approximate GSO are meaningful

Making the numeric-symbolic approach rigorous

Underlying mathematical phenomenon

Any LLL-reduced basis is well-conditioned with respect to GSO computations

- Well-conditioned? The GSO computed in small precision fp arithmetic is close to the true GSO

$\Rightarrow$ We'd like to rely on LLL-reduced bases as much as we can

Use a greedy LLL algorithm:

- Consider the first i s.t. $\mathbf{b}_1, \ldots, \mathbf{b}_i$ is not LLL-reduced
- $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$ is well-conditioned, and its related approximate GSO are meaningful

Bit complexity of floating-point LLL

Small precision? $\mathcal{O}(n)$ bits of precision suffice for correctness.
Bit-complexity:

$$\mathcal{O}(n^2\beta) \cdot \mathcal{O}(n^2) \cdot \left[\mathcal{O}(n\beta) + \mathcal{O}(n^2)\right] = \mathcal{O}(n^5\beta(n+\beta)).$$

Can we do better?

The totally numeric approach

LLL can be accelerated further by using approximations for the bases too, in a recursive manner! $\Rightarrow \widetilde{\mathcal{O}}(n^5\beta)$ bit operations.

The totally numeric approach with blocking

Restrict to sub-matrices of the GSO, and use fast matrix multiplication $\Rightarrow \widetilde{\mathcal{O}}(n^4\beta)$ bit operations.

Bit complexity of floating-point LLL

Small precision? $\mathcal{O}(n)$ bits of precision suffice for correctness.
Bit-complexity:

$$\mathcal{O}(n^2\beta) \cdot \mathcal{O}(n^2) \cdot \left[\mathcal{O}(n\beta) + \mathcal{O}(n^2)\right] = \mathcal{O}(n^5\beta(n+\beta)).$$

Can we do better?

The totally numeric approach

LLL can be accelerated further by using approximations for the bases too, in a recursive manner! $\Rightarrow \widetilde{\mathcal{O}}(n^5\beta)$ bit operations.

The totally numeric approach with blocking

Restrict to sub-matrices of the GSO, and use fast matrix multiplication $\Rightarrow \widetilde{\mathcal{O}}(n^4\beta)$ bit operations.

Bit complexity of floating-point LLL

Small precision? $\mathcal{O}(n)$ bits of precision suffice for correctness.
Bit-complexity:

$$\mathcal{O}(n^2\beta) \cdot \mathcal{O}(n^2) \cdot \left[\mathcal{O}(n\beta) + \mathcal{O}(n^2)\right] = \mathcal{O}(n^5\beta(n+\beta)).$$

Can we do better?

The totally numeric approach

LLL can be accelerated further by using approximations for the bases too, in a recursive manner! $\Rightarrow \widetilde{\mathcal{O}}(n^5\beta)$ bit operations.

The totally numeric approach with blocking

Restrict to sub-matrices of the GSO, and use fast matrix multiplication $\Rightarrow \widetilde{\mathcal{O}}(n^4\beta)$ bit operations.

Plan of the talk

Plan of the talk:

1. Euclidean lattices
2. Applications of euclidean lattices
3. The LLL algorithm and its accelerations
4. **Conclusion**

Research topics

- Lowering the cost of LLL-reduction
- Algorithms computing shorter vectors than LLL

  If paying $\mathcal{P}oly(n, 2^k)$, we can replace LLL's $2^{O(n)}$ by $k^{O(n/k)}$.

  Can we achieve a better time/quality trade-off?
- Quantum algorithms for lattice problems
- Hardness proofs

  NP hardness of SVP under deterministic reductions?
- Lattice-based cryptography

Research topics

- Lowering the cost of LLL-reduction
- Algorithms computing shorter vectors than LLL

  If paying $\mathcal{P}oly(n, 2^k)$, we can replace LLL's $2^{O(n)}$ by $k^{O(n/k)}$.

  Can we achieve a better time/quality trade-off?
- Quantum algorithms for lattice problems
- Hardness proofs

  NP hardness of SVP under deterministic reductions?
- Lattice-based cryptography

Research topics

- Lowering the cost of LLL-reduction
- Algorithms computing shorter vectors than LLL

  If paying $\mathcal{P}oly(n, 2^k)$, we can replace LLL's $2^{O(n)}$ by $k^{O(n/k)}$.

  Can we achieve a better time/quality trade-off?
- Quantum algorithms for lattice problems
- Hardness proofs

  NP hardness of SVP under deterministic reductions?
- Lattice-based cryptography

Research topics

- Lowering the cost of LLL-reduction
- Algorithms computing shorter vectors than LLL

  If paying $\mathcal{P}oly(n, 2^k)$, we can replace LLL's $2^{O(n)}$ by $k^{O(n/k)}$.

  Can we achieve a better time/quality trade-off?
- Quantum algorithms for lattice problems
- Hardness proofs

  NP hardness of SVP under deterministic reductions?
- Lattice-based cryptography

Research topics

- Lowering the cost of LLL-reduction
- Algorithms computing shorter vectors than LLL

  If paying $\mathcal{P}oly(n, 2^k)$, we can replace LLL's $2^{O(n)}$ by $k^{O(n/k)}$.

  Can we achieve a better time/quality trade-off?
- Quantum algorithms for lattice problems
- Hardness proofs

  NP hardness of SVP under deterministic reductions?
- Lattice-based cryptography

Further readings

L. Lovász: *An Algorithmic Theory of Numbers, Graphs, and Convexity*

D. Micciancio & S. Goldwasser: *Complexity of Lattice Problems, A Cryptographic Perspective*

I. Morel, D. Stehlé & G. Villard: *Analyse numérique et réduction de réseaux*

D. Micciancio: *The Geometry of Lattice Cryptography*

F. Eisenbrand: *Integer Programming and Algorithmic Geometry of Numbers*