Introduction aux réseaux euclidiens

Plan

Teasing : sécurité de RSA avec Petits exposants secrets

Introduction aux réseaux euclidiens

Propriétés de base, plus court vecteur, volume

Problèmes algorithmiques dans les réseaux

Algorithme LLL

Plan

Signature RSA

Génération de clef classique

1. Choisir $p, q \approx 2^{n/2}$ aléatoires, premiers.
2. Calculer $N = pq$ et $\phi(N) = (p-1)(q-1)$
3. Choisir $e = 3$
4. Si e n'est pas inversible modulo $\phi(N)$, retourner en 1.
5. Calculer $d \leftarrow e^{-1} \bmod \phi(N)$

À la fin, $ed \equiv 1 \bmod \phi(N)$

Bilan : $N \approx 2^n$, $e = 3$, $d \approx 2^n$.

- Signature = $2n$ multiplications $\bmod N \rightsquigarrow \mathcal{O}\left(n^3\right)$.
- Vérification = 2 multiplications $\bmod N \rightsquigarrow \mathcal{O}\left(n^2\right)$.

Signer est plus lent que vérifier

La CB signe, le terminal vérifie... dommage !
Peut-on faire l'inverse ?

Signature RSA

Génération de clef avec petit exposant secret

1. Choisir $p, q \approx 2^{n/2}$ aléatoires, premiers
2. Calculer $N = pq$ et $\phi(N) = (p-1)(q-1)$
3. Choisir $\cancel{d = 3}$ $d \approx 2^{128}$ aléatoire
4. Si d n'est pas inversible modulo $\phi(N)$, retourner en 1.
5. Calculer $e \leftarrow d^{-1} \bmod \phi(N)$ À la fin, $ed \equiv 1 \bmod \phi(N)$

Bilan : $N \approx 2^n$, $e \approx 2^n$, $d \approx 2^{128}$.

- Signature = 256 multiplications $\bmod N \rightsquigarrow \mathcal{O}\left(n^2\right)$.
- Vérification = $2n$ multiplications $\bmod N \rightsquigarrow \mathcal{O}\left(n^3\right)$.

C'est 16× plus rapide, mais...

Est-ce sûr ?

Modélisation du problème

- On connaît e et N
- BUT : retrouver d

On sait que :

$$ed = 1 + k\phi(N)$$

$$0 \leq d \leq N^{\alpha} \qquad \alpha = \frac{1}{16}$$

$$0 \leq k \leq N^{\alpha} \qquad k = (ed - 1)/\phi(N) < d$$

Modélisation du problème

- On connaît e et N
- BUT : retrouver d

On sait que :

$$ed = 1 + k(N - \sigma + 1) \qquad \sigma = p + q$$

$$0 \leq d \leq N^{\alpha} \qquad \alpha = \frac{1}{16}$$

$$0 \leq k \leq N^{\alpha} \qquad k = (ed - 1)/\phi(N) < d$$

$$0 \leq \sigma \leq 2N^{\frac{1}{2}}$$

Modélisation du problème

- On connaît e et N
- BUT : retrouver d

On sait que :

$$ed = 1 + k(N+1) - k\sigma \qquad \sigma = p + q$$

$$0 \leq d \leq N^{\alpha} \qquad \alpha = \frac{1}{16}$$

$$0 \leq k \leq N^{\alpha} \qquad k = (ed-1)/\phi(N) < d$$

$$0 \leq \sigma \leq 2N^{\frac{1}{2}}$$

Modélisation du problème

- On connaît e et N
- BUT : retrouver d

On sait que :

$$ed = k(N+1) - y \qquad \sigma = p + q, y = k\sigma - 1$$
$$0 \leq d \leq N^{\alpha} \qquad \alpha = \frac{1}{16}$$
$$0 \leq k \leq N^{\alpha} \qquad k = (ed-1)/\phi(N) < d$$
$$0 \leq y \leq 2N^{\frac{1}{2}+\alpha}$$

Technique générale : linéarisation

Introduire $y = k\sigma - 1$ pour éliminer le terme non-linéaire.

Programmation linéaire

« Programme linéaire »

$$\begin{cases} ed - k(N+1) + y = 0 \\ 0 \leq d \leq N^{\alpha} \\ 0 \leq k \leq N^{\alpha} \\ 0 \leq y \leq 2N^{\frac{1}{2}+\alpha} \end{cases}$$

- La programmation linéaire sur les flottants est **polynomiale**
  - Khachiyan, 1979 – URSS
- La programmation linéaire sur les entiers est **NP-dure**
- Algorithmes polynomiaux si #variables est constant
  - exponentiel en #variables, polynomial en |coefficients|
  - H. Lenstra, 1981
  - Basé sur les réseaux euclidiens

Plan

Rappels

- Vecteur $\mathbf{x} = (x_1, \ldots, x_n)$.
- Norme (euclidienne) (« norme 2 », « longueur ») :

$$\|\mathbf{x}\| = \sqrt{{x_1}^2 + {x_2}^2 + \cdots + {x_n}^2}$$

$\mathbb{R}$-Espace vectoriel V engendré par $\mathbf{b}_1, \ldots, \mathbf{b}_m$

- Ensemble des combinaisons linéaires de $\mathbf{b}_1, \ldots, \mathbf{b}_m$

$$V = \left\{ \sum_{i=1}^{m} \lambda_i \mathbf{b}_i : (\lambda_1, \ldots, \lambda_m) \in \mathbb{R}^m \right\}$$

- $\mathbf{x}, \mathbf{y} \in V \Longrightarrow \lambda\mathbf{x} + \mu\mathbf{y}$ pour tout $\lambda, \mu \in \mathbb{R}$.

Réseau euclidien

Un réseau euclidien (« *Euclidean lattice* »), c'est la même chose, mais avec des entiers. Il suffit de remplacer $\mathbb{R}$ par $\mathbb{Z}$ dans la diapo précédente.

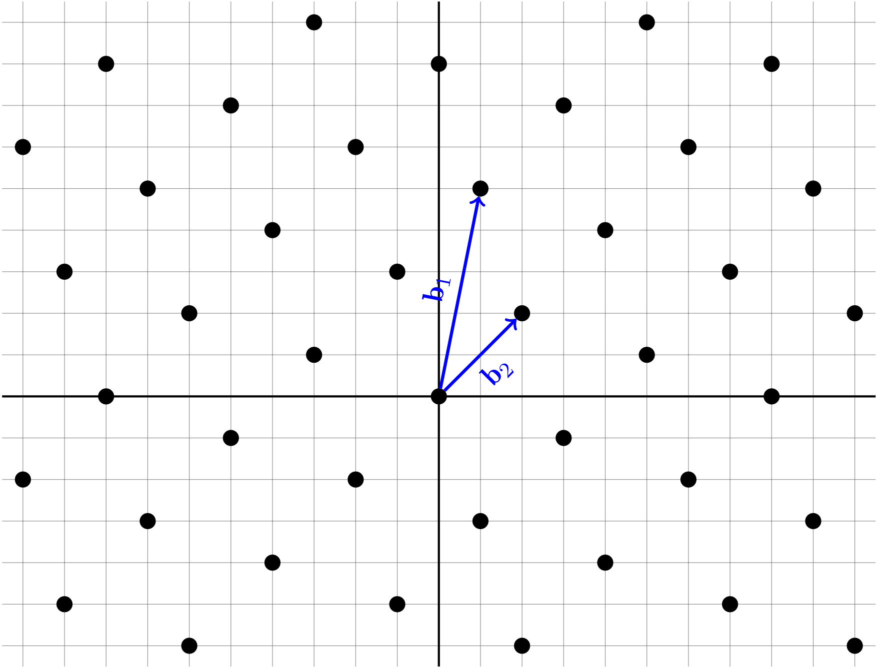
Definition (Réseau euclidien)

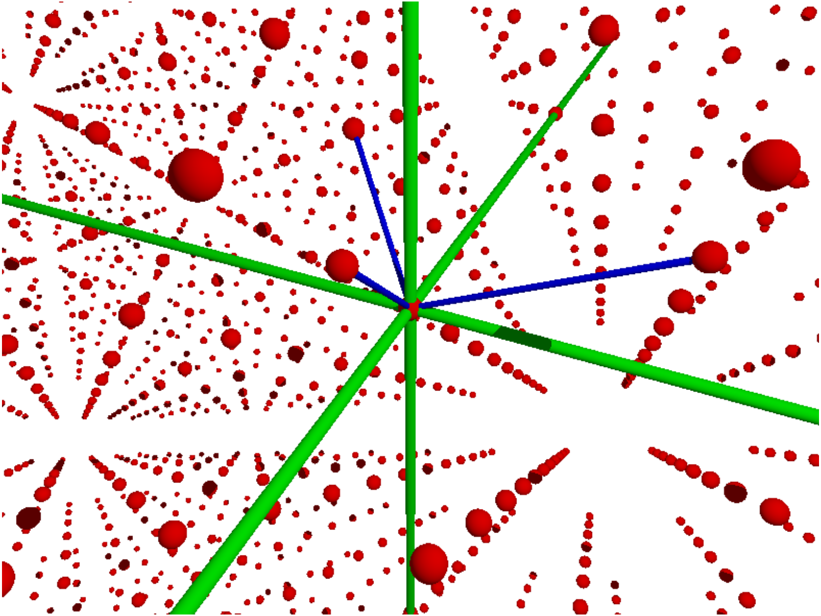
$\mathbf{b}_1, \ldots, \mathbf{b}_d$: vecteurs de $\mathbb{Z}^n$. Ils *engendrent* le réseau euclidien :

$$\mathcal{L} = \left\{ \sum_{i=1}^{d} \mu_i \mathbf{b}_i : (\mu_1, \ldots, \mu_d) \in \mathbb{Z}^d \right\}.$$

$\mathbf{b}_i$ linéairement indépendants $\rightsquigarrow$ *base* de $\mathcal{L}$ (d = dimension).
Si $d = n$, on dit que le réseau est *de rang plein*.

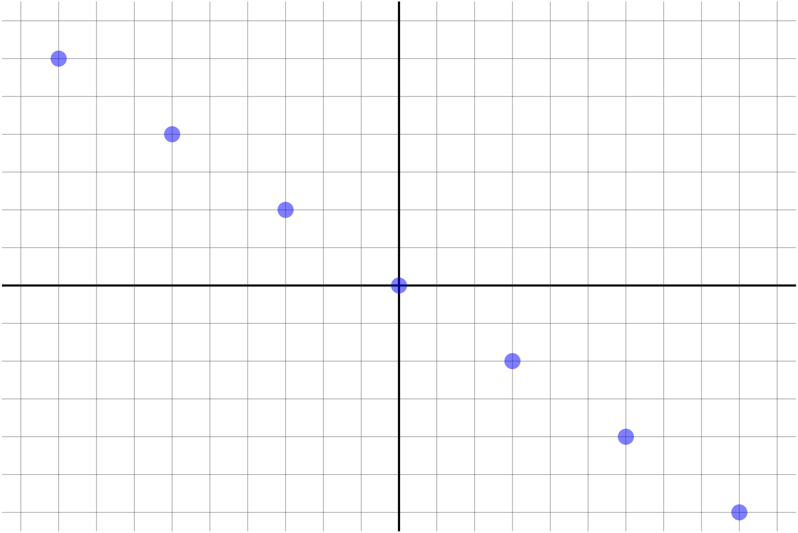
Ce sont les sous-groupes de $(\mathbb{Z}^n, +)$.


$\mathbf{b}_1$
$\mathbf{b}_2$



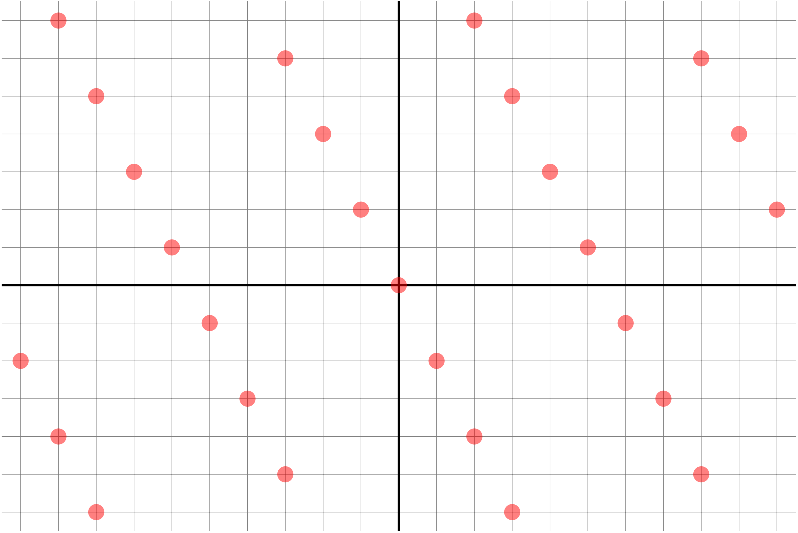
Équations diophantiennes linéaires

$$2x + 3y = 0$$



Équations diophantiennes linéaires

$$3x + 5y \equiv 0 \mod 11$$



Équations diophantiennes linéaires

- A = matrice $n \times m$ à coefficients dans $\mathbb{Z}$
- $\mathcal{S} = \{x \in \mathbb{Z}^m : Ax = 0 \mod q\}$
  - C'est un réseau euclidien !

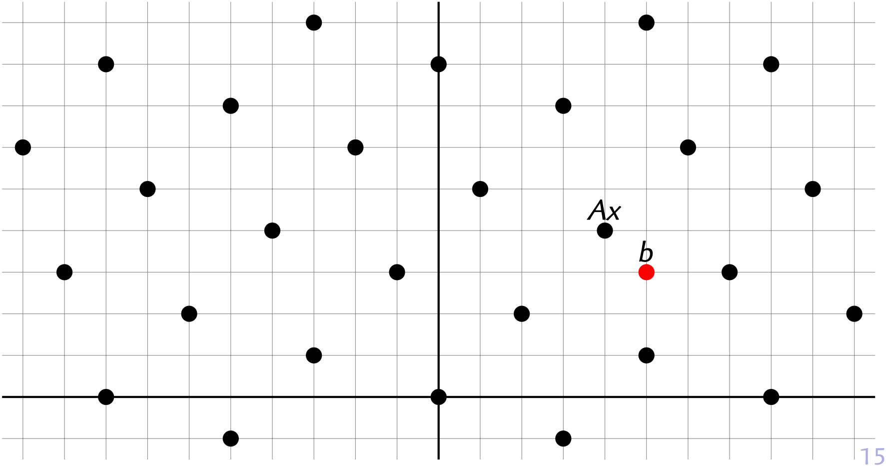
Conséquence

- Ajtai, LWE, subset-sum, etc. reviennent à des problèmes simples dans des réseaux
- Bien d'autres questions sur RSA aussi

Learning With Errors

Rappel

- A = matrice $n \times m$ à coefficients dans $\mathbb{Z}_q$
- $b = Ax + e$



Plan

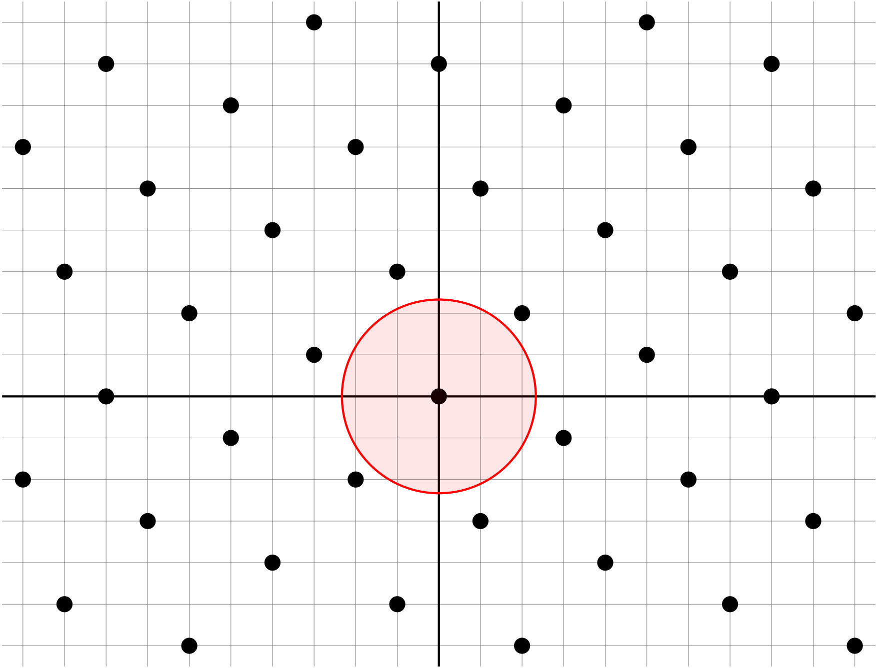
Teasing : sécurité de RSA avec Petits exposants secrets

Introduction aux réseaux euclidiens

Propriétés de base, plus court vecteur, volume

Problèmes algorithmiques dans les réseaux

Algorithme LLL

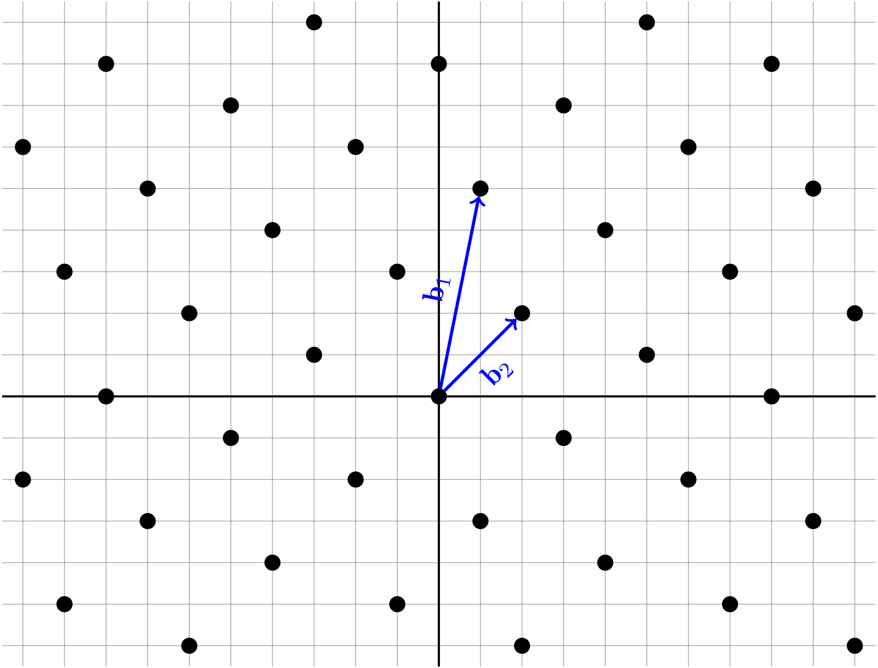


Plus court vecteur

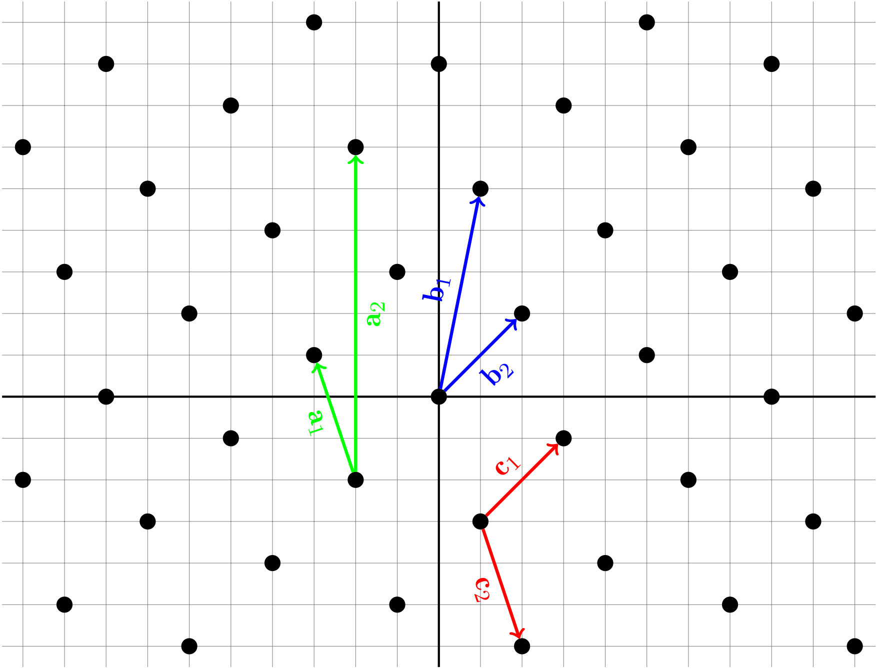
- Un réseau possède un plus court vecteur (non-nul).
- Sa longueur : $\lambda_1(\mathcal{L})$ — caractéristique importante de $\mathcal{L}$

Stay Tuned

On peut apprendre des choses sur sa longueur sans trop d'effort.



$\mathbf{b}_1$
$\mathbf{b}_2$

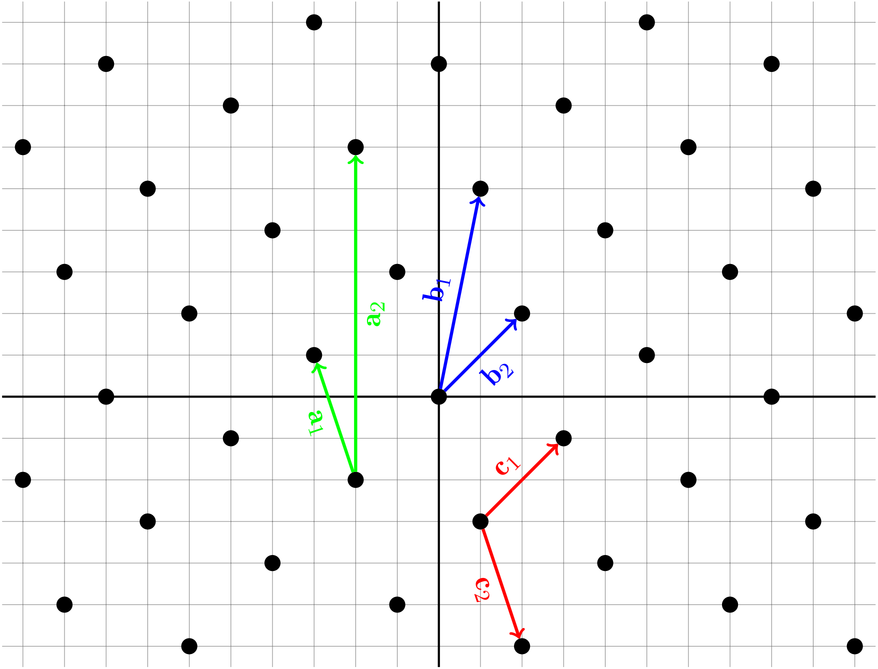

$\mathbf{a}_1$
$\mathbf{a}_2$
$\mathbf{b}_1$
$\mathbf{b}_2$
$\mathbf{c}_1$
$\mathbf{c}_2$

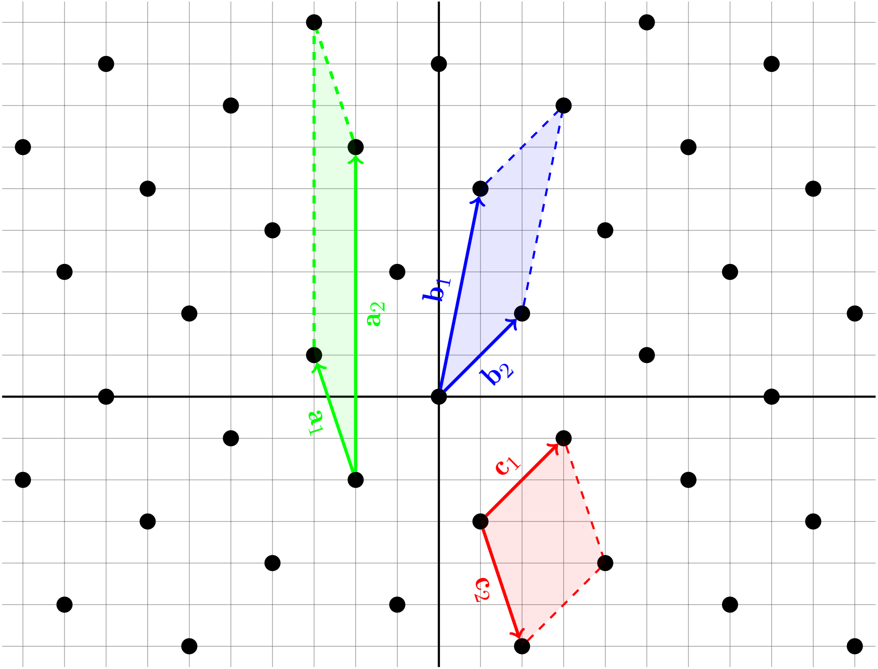
- Réseau précédent engendré par $\mathbf{b_1} = (1, 5), \mathbf{b_2} = (2, 2)$.
- Représenté de manière commode par :

$$B = \begin{pmatrix} 1 & 5 \\ 2 & 2 \end{pmatrix}$$

  Alors $\mathcal{L} = \{\mathbf{x} \cdot B \text{ avec } \mathbf{x} \in \mathbb{Z}^d\}$.
- Plusieurs bases possibles. Par ex. :

$$A = \begin{pmatrix} -1 & 3 \\ 0 & 8 \end{pmatrix} \quad \text{et} \quad C = \begin{pmatrix} 2 & 2 \\ 1 & -3 \end{pmatrix}$$


$\mathbf{a}_2$
$\mathbf{b}_1$
$\mathbf{b}_2$
$\mathbf{a}_1$
$\mathbf{c}_1$
$\mathbf{c}_2$



$\mathbf{a}_1$
$\mathbf{a}_2$
$\mathbf{b}_1$
$\mathbf{b}_2$
$\mathbf{c}_1$
$\mathbf{c}_2$

On fixe une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$.

- parallélepipède fondamental :

$$\left\{ \sum_{i=1}^{d} x_i \mathbf{b}_i : x_1, \ldots, x_d \in [0; 1[\right\}$$

- Son **volume** ne dépend pas du choix de la base

$\Rightarrow$ caractéristique (importante) du réseau

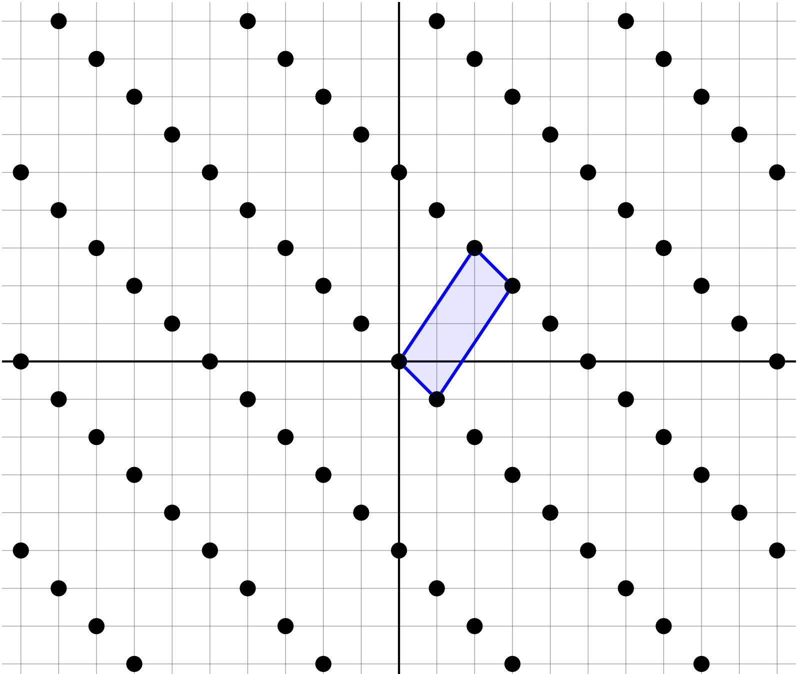
Theorem

Soit $\mathcal{L}$ un réseau de base B. On a $\mathrm{Vol}(\mathcal{L}) = \sqrt{\det BB^t}$.

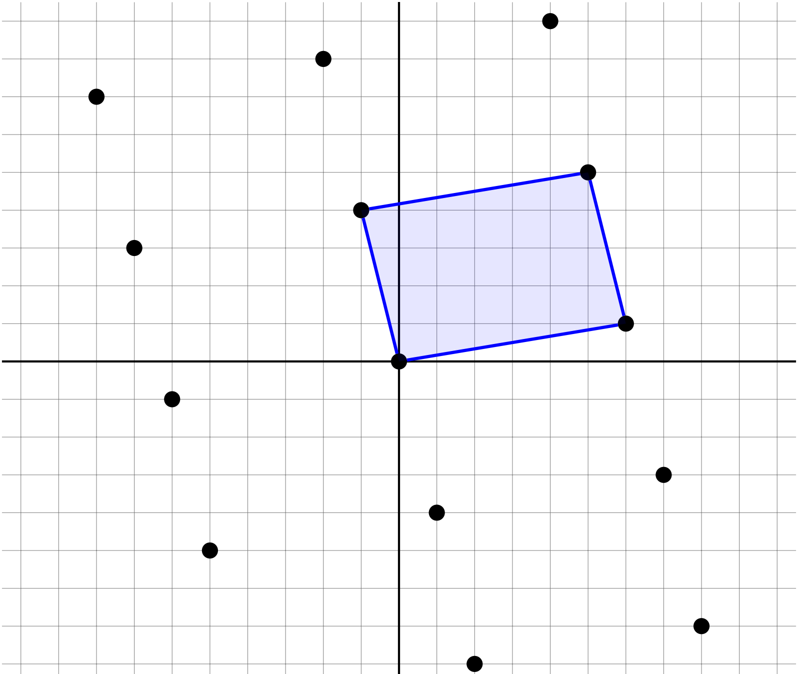
Si $\mathcal{L}$ est de rang plein, ça se simplifie en $\mathrm{Vol}(\mathcal{L}) = |\det B|$.

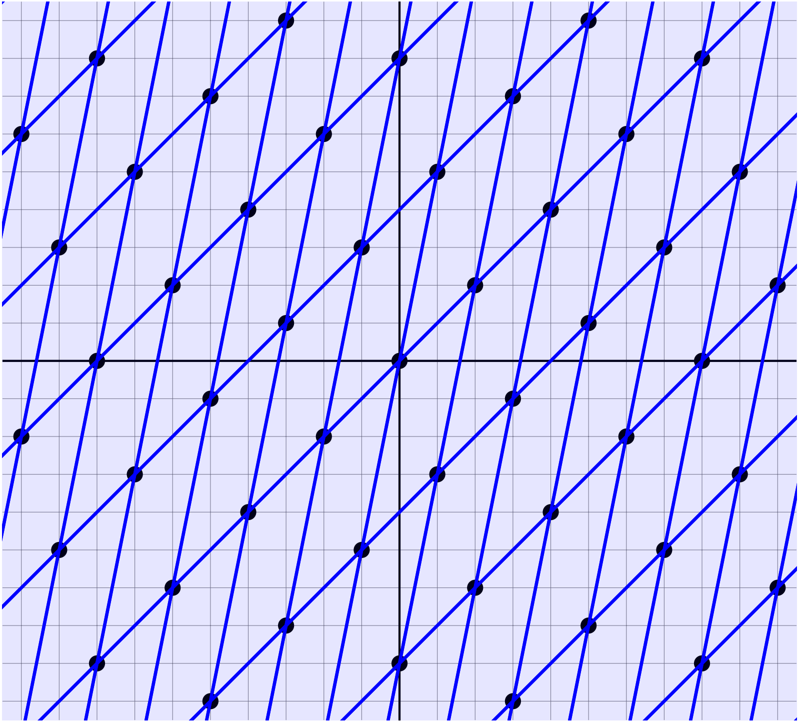
$\Rightarrow$ Le volume est (souvent) facile à déterminer

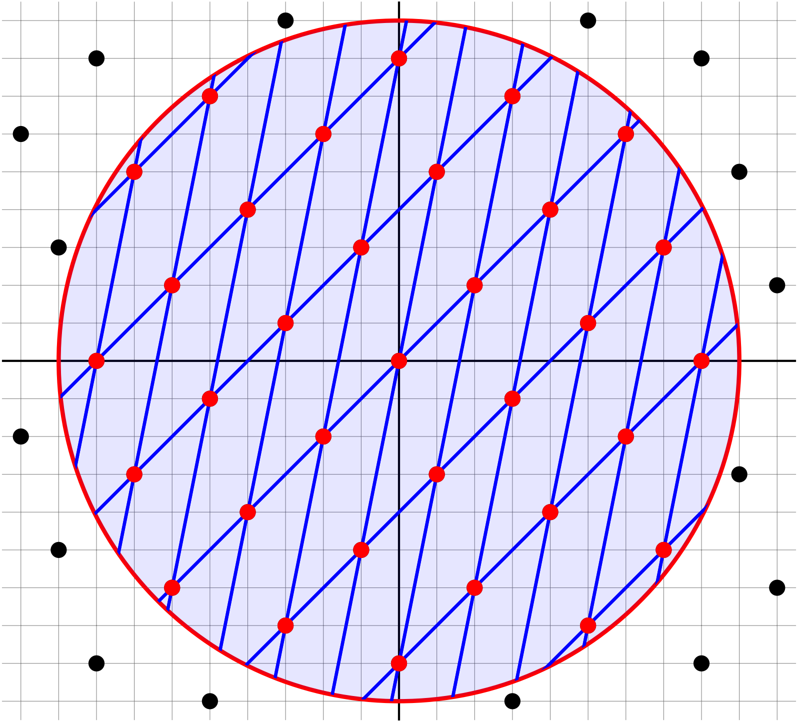
Volume = 5



Volume = 31







Un des principaux intérêts du concept de volume

$$\#\text{ points dans la boule} \approx \frac{\text{Volume de la boule}}{\text{Volume du réseau}}.$$

« Boule » = ensembles des points à distance r de l'origine.

Plus court vecteur

- Un réseau possède un plus court vecteur (non-nul).
- Sa longueur : $\lambda_1(\mathcal{L})$ — caractéristique importante de $\mathcal{L}$

Heuristique gaussienne

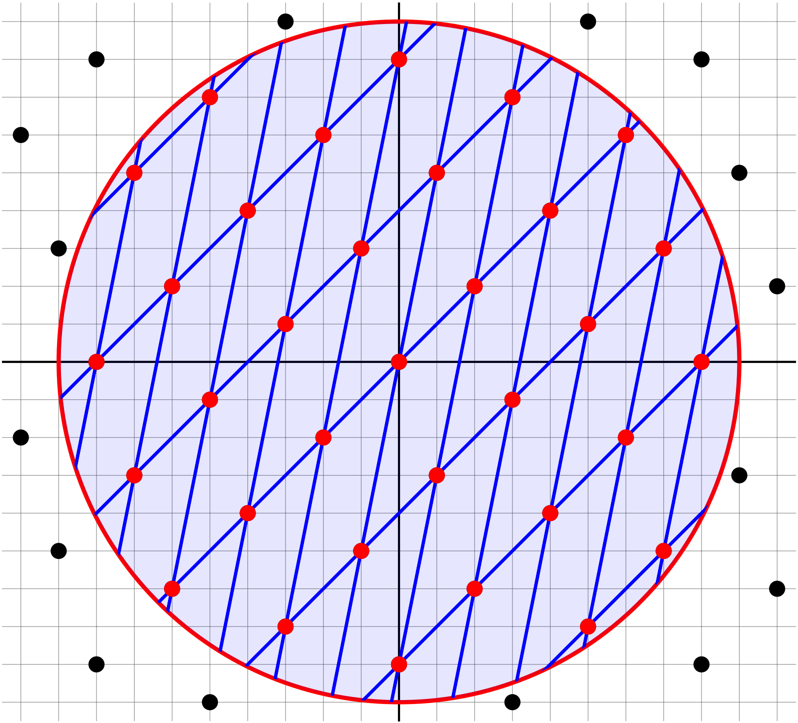
$$\#\text{ points dans la boule} \approx \frac{\text{Volume de la boule}}{\text{Volume du réseau}}$$

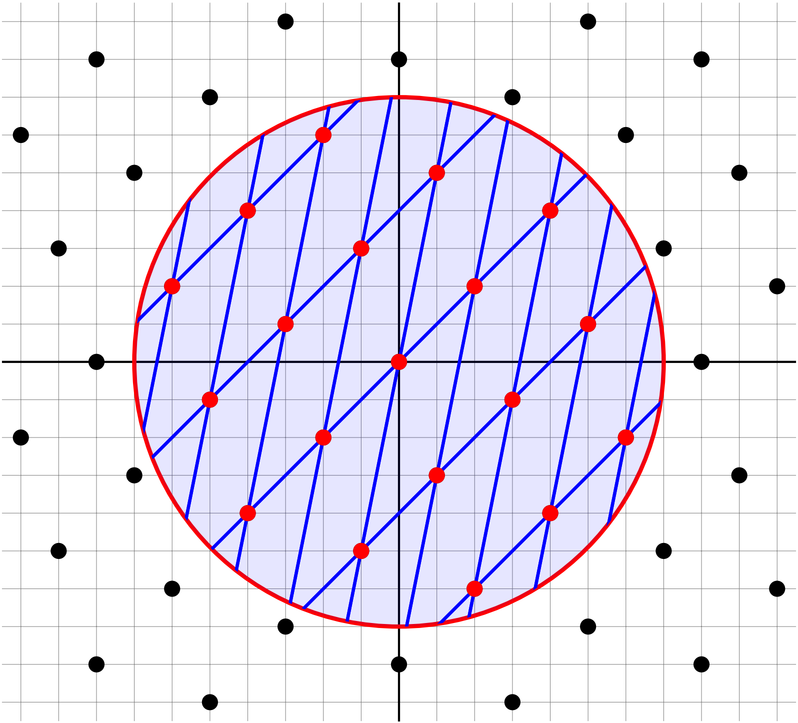
Le + grand rayon pour lequel la boule de rayon *r* ne contient plus qu'un point est (à peu près) la taille du plus court vecteur.

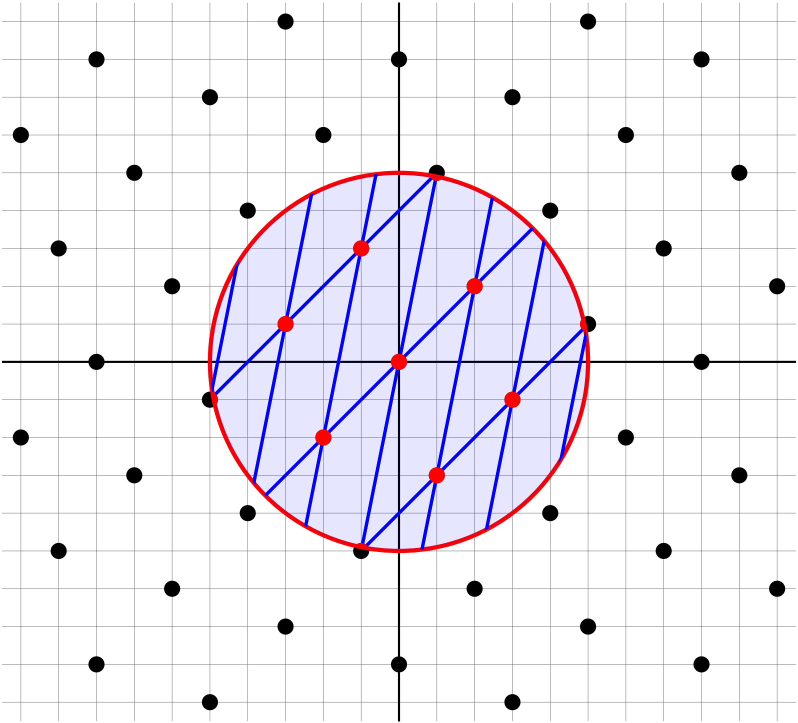
$$\lambda_1(\mathcal{L}) \approx \sqrt{n}\,(\mathrm{Vol}(\mathcal{L}))^{\frac{1}{n}}\ .$$

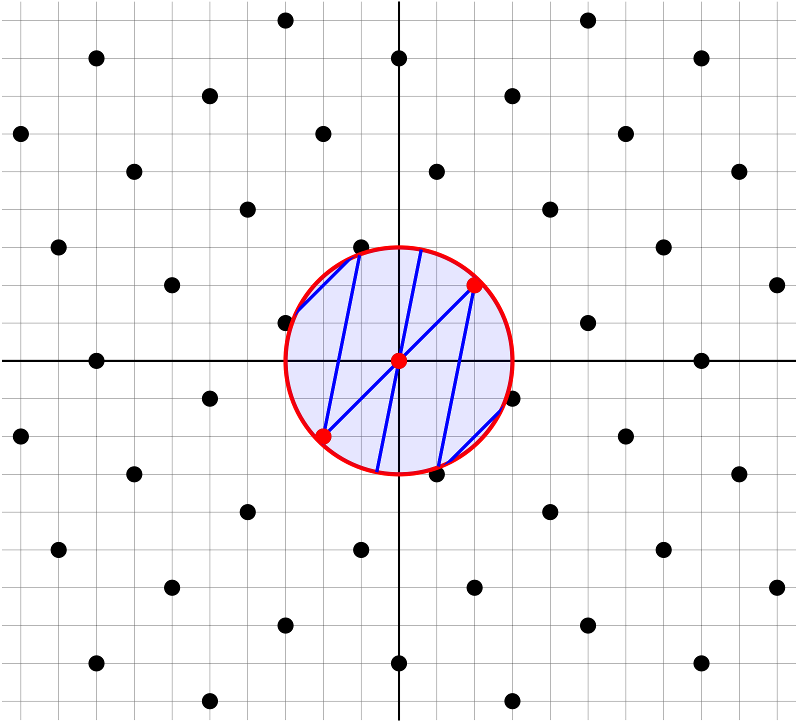
Theorem (Borne de Minkowski simplifiée)

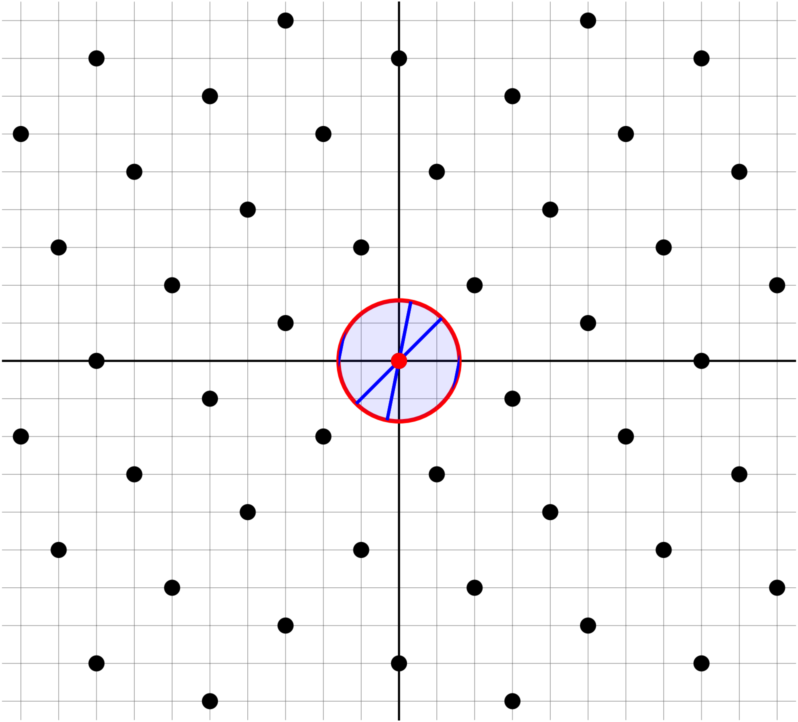
Si $\mathcal{L}$ est un réseau de dimension d, alors $\lambda_1(\mathcal{L}) \leq \sqrt{d}\,(\mathrm{Vol}(\mathcal{L}))^{\frac{1}{d}}$.

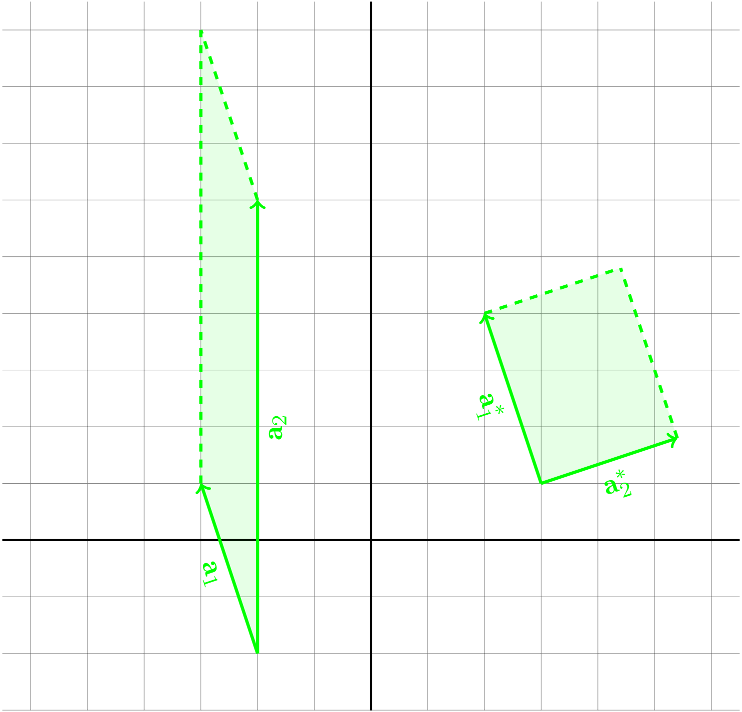












$\mathbf{a}_2$
$\mathbf{a}_1$
$\mathbf{a}_1^*$
$\mathbf{a}_2^*$

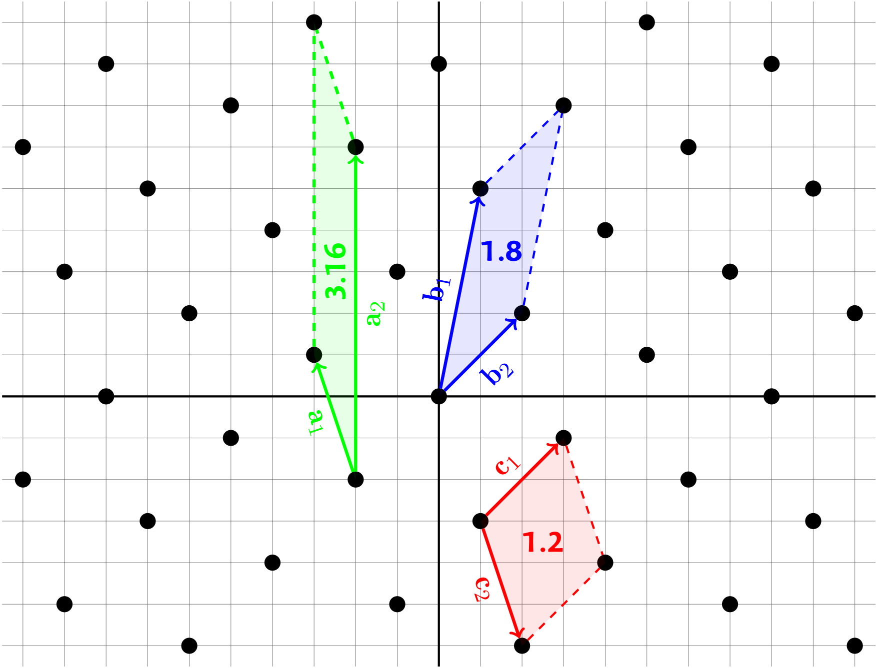
Orthogonalité

- Orthogonal = « perpendiculaire »
- Deux vecteurs $\mathbf{x}$ et $\mathbf{y}$ sont orthogonaux ssi $\mathbf{x} \cdot \mathbf{y} = 0$
- Un réseau n'admet pas forcément (et même rarement) une base orthogonale

Défaut d'orthogonalité

$$\frac{\|\mathbf{b}_1\| \times \cdots \times \|\mathbf{b}_d\|}{\mathrm{Vol}(\mathcal{L})}$$

- Plus il est grand, moins la base est orthogonale / courte
- Il vaut 1 sur un parallélépipède


3.16
a_2
a_1
1.8
b_1
b_2
c_1
1.2
c_2

Plan

Teasing : sécurité de RSA avec Petits exposants secrets

Introduction aux réseaux euclidiens

Propriétés de base, plus court vecteur, volume

Problèmes algorithmiques dans les réseaux

Algorithme LLL

Shortest Vector Problem (SVP) — NP-dur

Étant donné une base d'un réseau $\mathcal{L}$, calculer un vecteur non-nul $\mathbf{x}$ tel que $\|\mathbf{x}\| = \lambda_1(\mathcal{L})$.

Closest Vector Problem (CVP) — NP-dur

Étant donné une base d'un réseau $\mathcal{L}$ et un vecteur $\mathbf{y}$, calculer un vecteur $\mathbf{x}$ tel que $\|\mathbf{y} - \mathbf{x}\| \leq \|\mathbf{y} - \mathbf{z}\|$ pour tout $\mathbf{z} \in \mathcal{L}$.

Les classiques

Approximate Shortest Vector Problem (SVP_γ)

Étant donné une base d'un réseau $\mathcal{L}$, calculer un vecteur non-nul $\mathbf{x}$ tel que $\|\mathbf{x}\| \leq \gamma \times \lambda_1(\mathcal{L})$.

Approximate Closest Vector Problem (CVP_γ)

Étant donné une base d'un réseau $\mathcal{L}$ et un vecteur $\mathbf{y}$, calculer un vecteur $\mathbf{x}$ tel que $\|\mathbf{y} - \mathbf{x}\| \leq \gamma \times \|\mathbf{y} - \mathbf{z}\|$ pour tout $\mathbf{z} \in \mathcal{L}$.

Les classiques

Approximate Shortest Vector Problem (SVP_γ)

Étant donné une base d'un réseau $\mathcal{L}$, calculer un vecteur non-nul $\mathbf{x}$ tel que $\|\mathbf{x}\| \leq \gamma \times \lambda_1(\mathcal{L})$.

Hermite Shortest Vector Problem (HSVP_γ)

Étant donné une base d'un réseau $\mathcal{L}$, calculer un vecteur non-nul $\mathbf{x}$ tel que $\|\mathbf{x}\| \leq \gamma \times \mathrm{Vol}(\mathcal{L})^{1/n}$.

Approximate Closest Vector Problem (CVP_γ)

Étant donné une base d'un réseau $\mathcal{L}$ et un vecteur $\mathbf{y}$, calculer un vecteur $\mathbf{x}$ tel que $\|\mathbf{y} - \mathbf{x}\| \leq \gamma \times \|\mathbf{y} - \mathbf{z}\|$ pour tout $\mathbf{z} \in \mathcal{L}$.

CVP est NP-dur

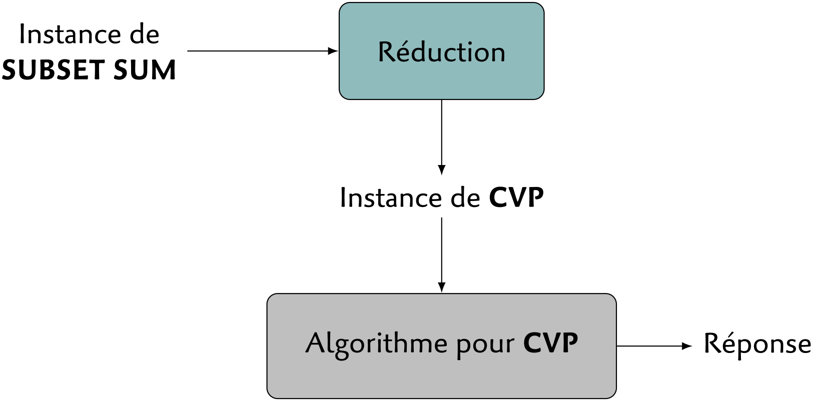
Closest Vector Problem (CVP) — version décisionnelle

Étant donné une base d'un réseau $\mathcal{L}$, un vecteur $\mathbf{y}$ et une distance d, déterminer s'il existe $\mathbf{x} \in \mathcal{L}$ tel que $\|\mathbf{y} - \mathbf{x}\| \leq d$.

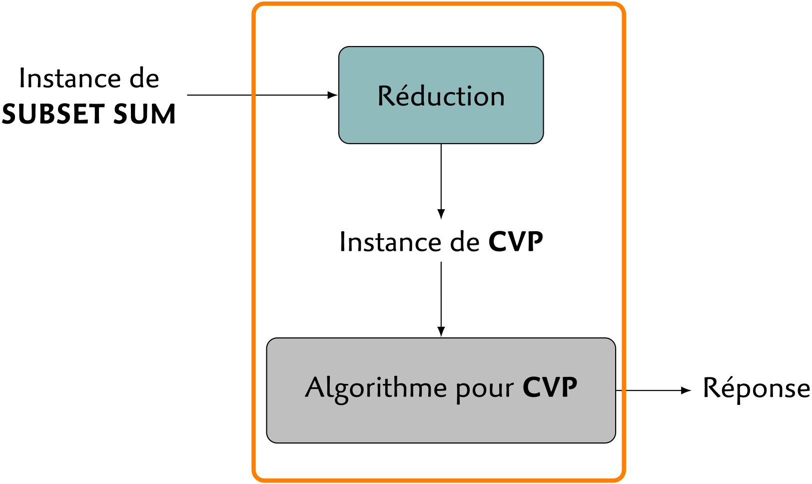
Theorem

CVP est ***NP-dur***

NP-difficulté de CVP : Stratégie



NP-difficulté de CVP : Stratégie



NP-difficulté de CVP : preuve

Démonstration.

- Réduction : CVP $\rightarrow$ SUBSET SUM $\rightarrow$ VERTEX COVER
- On part d'une instance $(A_1, \ldots, A_n), t$ de SUBSET SUM
  - Existe-t-il $\mathbf{x} \in \{0,1\}^n$ tel que $\sum x_i A_i = t$?
- On considère un réseau spécial :

$$B = \begin{pmatrix} A_1 & 2 & & & \\ A_2 & & 2 & & \\ A_3 & & & 2 & \\ \vdots & & & & \ddots & \\ A_n & & & & & 2 \end{pmatrix}$$

- Si $\sum x_i A_i = t$, alors $xB = (t, 2x_1, 2x_2, 2x_3, \ldots, 2x_n)$
- xB est proche de $\mathbf{y} = (t, 1, 1, \ldots, 1)$. Distance $= \sqrt{n}$

NP-difficulté de CVP : preuve (suite)

Démonstration.

$$B = \begin{pmatrix} A_1 & 2 & & & & \\ A_2 & & 2 & & & \\ A_3 & & & 2 & & \\ \vdots & & & & \ddots & \\ A_n & & & & & 2 \end{pmatrix}$$

$$xB = \left(\sum_{i=1}^{n} x_i A_i, 2x_1, 2x_2, 2x_3, \ldots, 2x_n \right)$$

- Décider CVP pour B, $\mathbf{y} = (t, 1, 1, \ldots, 1)$, distance $\sqrt{n}$
  - Ça va révéler la solution du SUBSET SUM de départ
- Si $\sum x_i A_i = t$, $\|\mathbf{x}B - \mathbf{y}\| = \sqrt{n}$
- Si $z \in L$, n dernières coordonnées paires $\Rightarrow \|\mathbf{y} - \mathbf{z}\| \geq \sqrt{n}$
- $\|\mathbf{y} - \mathbf{z}\| = \sqrt{n} \implies z_1 = t$ et $z_i \in \{0, 2\}$ pour $i \geq 2$
  - $\mathbf{z}$ décrit une solution valide au SUBSET SUM de départ

$\square$

Les classiques

CVP / SVP **exact**

- Complexité : $\mathcal{O}\left(2^{d \log d}\right)$ (méthodes d'énumération)
- Facile en petite dimension (≤ 50 disons)
  - Algorithmes disponibles dans `fpylll`
- Impossible en grande dimension (≥ 300 disons)

CVP / SVP **approché** en grande dimension

- « Facile » d'obtenir γ exponentiel en la dimension
  - Algorithmes disponibles dans `fpylll`, `SageMath`, etc.
- Difficile d'obtenir γ faible

Plan

Algorithme LLL

Hendrik Lenstra, Arjen Lenstra et László Lovász (1982)

Spécification

- ENTRÉE : une base d'un réseau euclidien
- SORTIE : une meilleure base du même réseau
- Temps d'exécution : polynomial (cher : d^5...)

Facilement accessible (SageMath, fpylll, NTL, ...)

Exemple

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 & 2121390710 \\ 0 & 1 & 0 & 0 & 365500767 \\ 0 & 0 & 1 & 0 & 2647889201 \\ 0 & 0 & 0 & 1 & 1407355715 \\ 0 & 0 & 0 & 0 & 2585271343 \end{pmatrix}$$

Algorithme LLL

Hendrik Lenstra, Arjen Lenstra et László Lovász (1982)

Spécification

- ENTRÉE : une base d'un réseau euclidien
- SORTIE : une meilleure base du même réseau
- Temps d'exécution : polynomial (cher : d^5...)

Facilement accessible (SageMath, fpylll, NTL, ...)

Exemple

$$LLL(B) = \begin{pmatrix} -7 & 28 & -35 & 3 & -36 \\ -4 & -51 & 24 & -2 & -22 \\ 50 & -8 & -29 & -72 & 2 \\ -36 & 33 & 41 & -59 & -48 \\ -70 & -41 & -65 & 7 & 11 \end{pmatrix}$$

Algorithme LLL

Hendrik Lenstra, Arjen Lenstra et László Lovász (1982)

Theorem

Soit $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ *une base LLL-reduite de* $\mathcal{L}$*. Alors :*

1. $\|\mathbf{b}_1\| \leq 2^{(d-1)/4} \left(\mathrm{Vol}(\mathcal{L})\right)^{1/d}$.
2. $\|\mathbf{b}_1\| \leq 2^{(d-1)/2} \lambda_1(\mathcal{L})$.
3. $\|\mathbf{b}_1\| \times \cdots \times \|\mathbf{b}_d\| \leq 2^{d(d-1)/4}\, \mathrm{Vol}(\mathcal{L})$.

Bonus : si $d = 2$, alors $\|\mathbf{b}_1\| = \lambda_1(\mathcal{L})$.

Conséquence

1. HSVP_γ avec $\gamma = (2^{1/4})^n$ en temps polynomial
2. SVP_γ avec $\gamma = (2^{1/2})^n$ en temps polynomial
3. Le défaut d'orthogonalité des bases LLL-réduite est borné

Pierre angulaire de tout ce qui tourne autour des réseaux !

Algorithme LLL
Approx-SVP en temps polynomial

LLL : en théorie

- LLL approxime HSVP avec $\gamma = (2^{1/4})^n = 1.19^n$
- LLL approxime SVP avec $\gamma = (2^{1/2})^n = 1.41^n$

LLL : en pratique

- HSVP : on observe $\gamma = 1.0219^n$ en général (pas de preuve)
- SVP : on observe $\gamma = 1.04428^n$ en général (pas de preuve)

BKZ-24 : en pratique

- HSVP : on observe $\gamma = 1.0125^n$ en général (pas de preuve)
- SVP : on observe $\gamma = 1.0252^n$ en général (pas de preuve)

Et CVP ?

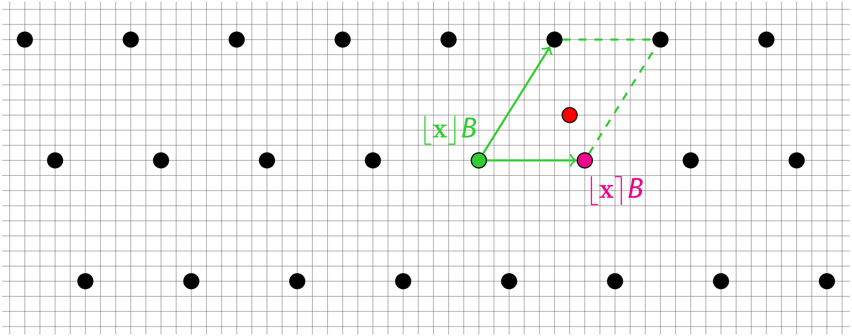
Méthode de l'arrondi

1. Résoudre $\mathbf{x}B = \mathbf{y}$ avec des *flottants* (ou des rationnels)
2. **Arrondir** les coefficients de $\mathbf{x}$
3. $\lfloor \mathbf{x} \rceil B$ est peut-être proche de $\mathbf{y}$

Demonstration

- $B = \begin{pmatrix} -4 & 1 & -2 & -1 \\ 6 & 0 & -8 & 0 \\ 3 & 12 & 2 & 1 \\ 2 & 2 & 1 & -14 \end{pmatrix}$
- $\mathbf{y} = (42, 0, 0, 0)$.
- $\mathbf{x} = (-6.967..., 1.931..., 0.491..., 0.532...)$
- $\lfloor \mathbf{x} \rceil B = (-7, 2, 0, 1)B = (42, -5, -1, -7)$

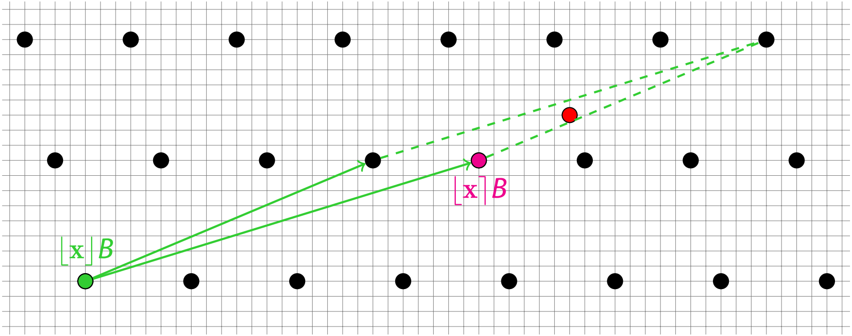
Méthode de l'arrondi



Interprétation géométrique

- $\lfloor \mathbf{x} \rfloor B$ = base d'un parallélépipède fondamental contenant $\mathbf{y}$.
- arrondir = renvoyer son sommet le plus proche de $\mathbf{y}$
- Avec une bonne base, ça marche (assez) bien

Méthode de l'arrondi



Interprétation géométrique

- $\lfloor \mathbf{x} \rfloor B$ = base d'un parallélépipède fondamental contenant $\mathbf{y}$.
- arrondir = renvoyer son sommet le plus proche de $\mathbf{y}$
- Avec une bonne base, ça marche (assez) bien
- Avec une mauvaise base, ça marche (très) mal

Et CVP ?

Méthode de l'arrondi

1. Résoudre $\mathbf{x}B = \mathbf{y}$ avec les *flottants*
2. Arrondir les coefficients de $\mathbf{x}$
3. $\lfloor \mathbf{x} \rceil B$ doit être proche de $\mathbf{y}$

Bonne base ?

Theorem (Babai, 1986)

La méthode de l'arrondi utilisée avec des bases LLL-réduites approxime CVP avec facteur $\gamma = 1 + 2d(9/2)^{d/2}$.

Autre méthodes (cf. poly)

- Hyperplan le plus proche : $\gamma = 2^{(d-1)/2}$
- Plongement : $\gamma = 2^{(d-1)/2}$ (heuristique)

CVP : méthode du plongement

Idée : ramener CVP à SVP, puis utiliser LLL

Trouver $\mathbf{x}$ proche de $\mathbf{y}$ dans le réseau engendré par B

1. Former le nouveau réseau engendré par les lignes de

$$B' = \begin{pmatrix} \boxed{\quad B \quad} & \boxed{0} \\ \boxed{\quad \mathbf{y} \quad} & 1 \end{pmatrix}$$

   - $(\mathbf{y} - \mathbf{x}, 1)$ appartient au nouveau réseau
   - $\mathbf{x}$ très proche de $\mathbf{y} \iff (\mathbf{y} - \mathbf{x}, 1)$ est un vecteur très court

2. Lancer LLL sur B'
3. **SI** on trouve un vecteur $(\mathbf{z}, 1)$, alors renvoyer $\mathbf{y} - \mathbf{z}$ (=$\mathbf{x}$).

Ça peut rater...