L'algorithme LLL flottant

Damien STEHLÉ

Lyon, 08 Mars 2005

Travail en commun avec Phong NGUYEN

http://www.loria.fr/~stehle/

stehle@loria.fr

Le plan de l'exposé.

- Des rappels sur les réseaux Euclidiens et sur LLL.
- Qu'est-ce que c'est que LLL flottant?
- Présentation de LLL^2, un LLL flottant:

  $\Rightarrow$ prouvé,

  $\Rightarrow$ pratique,

  $\Rightarrow$ quadratique vis-à-vis de la taille des vecteurs.

  Est-ce la bonne variante de LLL?

Pourquoi les réseaux? Parce que c'est utile.

- Principal outil de cryptanalyse en clé publique
  (sacs-à-dos, RSA & DSA pour certains paramètres)
- Cryptosystèmes: NTRU, GGH, Ajtai-Dwork
- Calcul formel: factorisation des polynômes rationnels
- Théorie algorithmique des nombres: idéaux dans les corps de nombres, petites racines de polynômes (Coppersmith), ...
- Découverte de relations linéaires (meilleur que PSLQ?)

Pourquoi les réseaux? Parce que c'est curieux.

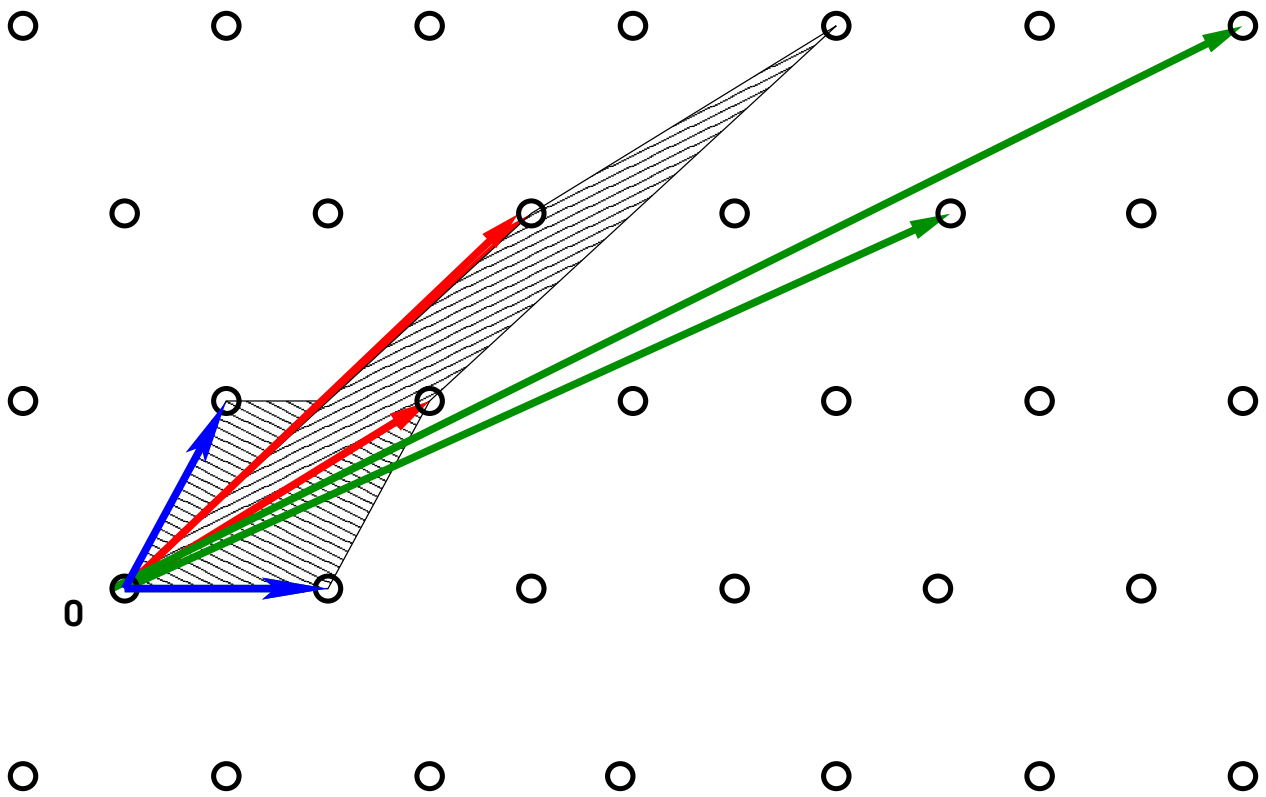
Très intéressant du point de vue de la complexité: γ-SVP est

- $\gamma = 1$ et norme infinie: dét-NP-dur,
- $\gamma = 1$ et norme L_i: prob-NP-dur,
- $\gamma = O(1)$ et norme L_2: prob-NP-dur,
- $\gamma = O(2^{\sqrt{\log n} - \varepsilon})$ et norme L_2: probablement prob-NP-dur,
- $\gamma = O(\frac{\sqrt{n}}{\log n})$: probablement pas NP-dur,
- $\gamma = O(n^c)$: P-équivalent "cas le pire - cas moyen",
- $\gamma = 2^{O\left(n \frac{\log \log n}{\log n}\right)}$: dans BPP,
- $\gamma = 2^{O\left(n \frac{(\log \log n)^2}{\log n}\right)} = o\left(2^n\right)$: dans P.

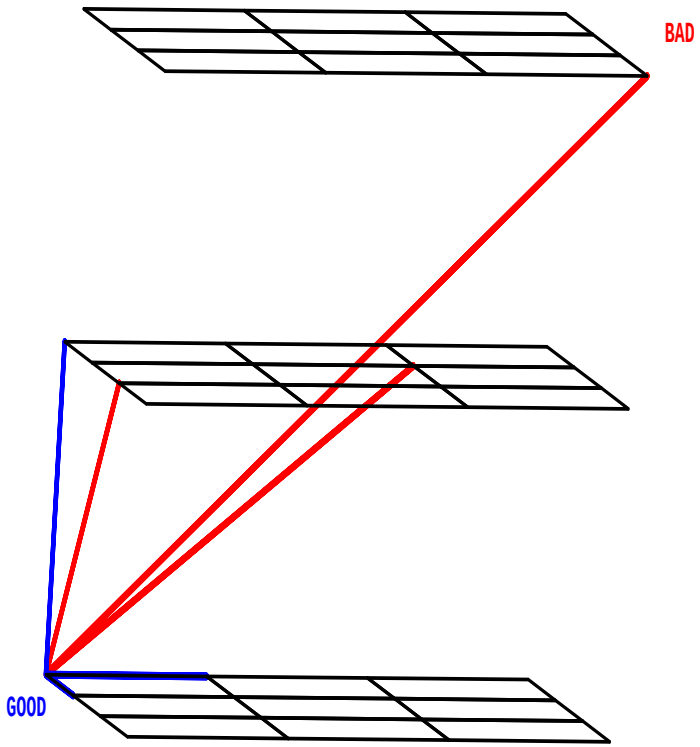
Quelques définitions

- Réseau == grille en dimension d == sous-groupe discret de $\mathbb{R}^d$.
- Soient $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{Z}^d$ linéairement indépendants.
  $L[\mathbf{b}_1, \ldots, \mathbf{b}_d] = \left\{ \sum_{i=1}^{d} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\}$
  $=$ combinaisons entières des vecteurs de la base.
- Matrice de Gram: $G(\mathbf{b}_1, \ldots, \mathbf{b}_d) = (\langle \mathbf{b}_i, \mathbf{b}_j \rangle)_{1 \leq i,j \leq d}$.
- Infinité de bases, liées par des transformations unimodulaires (matrices à coefficients entiers et déterminant ± 1).
- vol$(L) = |\det(L)| = \det(G)^{1/2}$ est indépendant de la base.

Trois bases d’un réseau de dimension 2

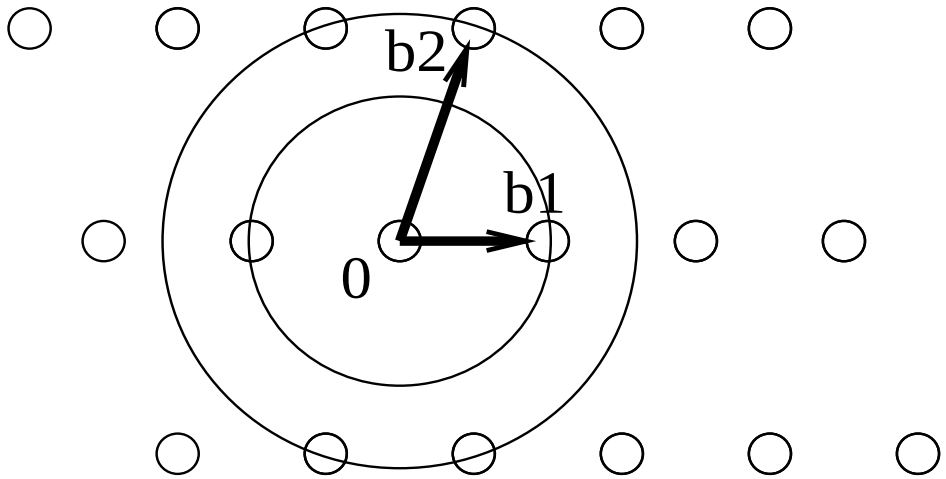


Deux bases d’un réseau de dimension 3



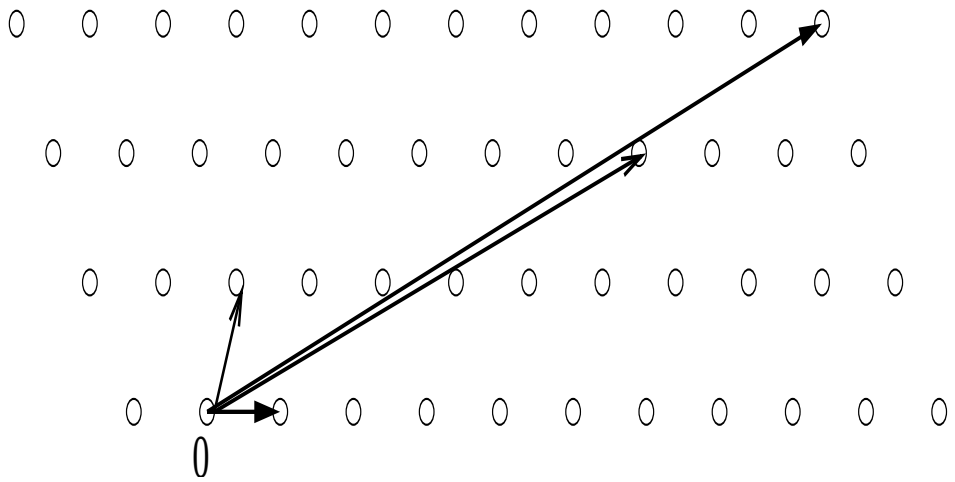
Le problème principal: trouver un vecteur court

- Minkowski: il existe un vecteur $\neq 0$ de norme $\leq \sqrt{d} \cdot \det(L)^{1/d}$.
- SVP: Partant d'une base, trouver un vecteur $\neq 0$ le plus court.
- γ-SVP: pas plus de γ fois plus long qu'une solution de SVP.



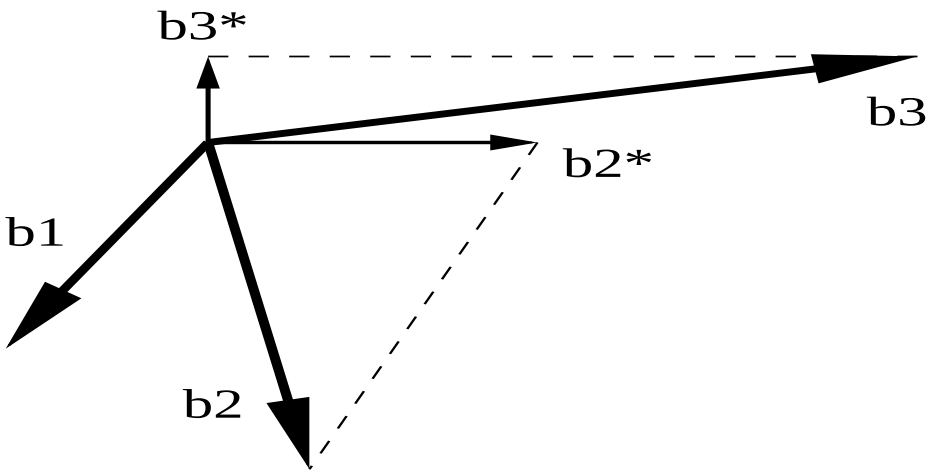
La solution générale: réduire le réseau

- Principe: trouver une base avec de petits vecteurs.
- Mais: pas vraiment de définition optimale/canonique.
- Moralement, une bonne base est plutôt orthogonale.



L'orthogonalisation de Gram-Schmidt

- Processus itératif d'orthogonalisation de $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$.
- $\mathbf{b}_i^*$ est $\mathbf{b}_i$ projeté sur l'orthogonal de $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$.
- $\mathbf{b}_1^* = \mathbf{b}_1$, $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*$, $\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{||\mathbf{b}_j^*||^2}$.



La proprification

- Algorithme de l'hyperplan le plus proche de Babai.
- Version "entière" de Gram-Schmidt.
- Étant donnés $(\mathbf{b}_1, \ldots, \mathbf{b}_k)$, ça enlève à $\mathbf{b}_k$ une combinaison entière des précédents pour que: $\forall i < k,\ |\mu_{k,i}| \leq 1/2$.

1. Calculer les $\mu_{k,i}$ pour $i < k$.
2. Pour $i = (k-1)..1$,
3. $\quad x_i := \lceil \mu_{\kappa,i} \rfloor$, $\mathbf{b}_k := \mathbf{b}_\kappa - x_i \mathbf{b}_i$,
4. $\quad$ Pour $j = 1..i$,
5. $\quad\quad \mu_{\kappa,j} := \mu_{\kappa,j} - x_i \mu_{i,j}$.

La LLL-réduction (1982)

- $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ est LLL-réduite ssi:
  1) $\forall i > j, |\mu_{i,j}| \leq 1/2$ [Hermite].
  2) $\forall i, 1^- ||\mathbf{b}_{i-1}^*|| \leq ||\mathbf{b}_i^* + \mu_{i,i-1}\mathbf{b}_{i-1}^*||$ [Lovász].

- 2) signifie: dans $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-2})^{orth}$, $\mathbf{b}_{i-1}$ est plus court que $\mathbf{b}_i$.

- 1) et 2) impliquent $||\mathbf{b}_i^*|| \geq \sqrt{\frac{3}{4}^-}||\mathbf{b}_{i-1}^*||$.
  "Les orthogonalisés de Gram-Schmidt ne chutent pas trop vite."

Pourquoi la LLL-réduction est-elle intéressante?

- On peut l'obtenir en temps polynomial.
- Et le premier vecteur n'est pas "trop" long: $||\mathbf{b}_1|| \leq (4/3)^{d/2}\lambda$, avec λ la longueur d'un vecteur le plus court $\neq 0$.
- On a aussi: $||\mathbf{b}_1|| \leq (4/3)^{d/4}\mathrm{det}(L)^{1/d}$.
- "En moyenne", $(4/3)^{1/4}$ devient ≈ 1.04.

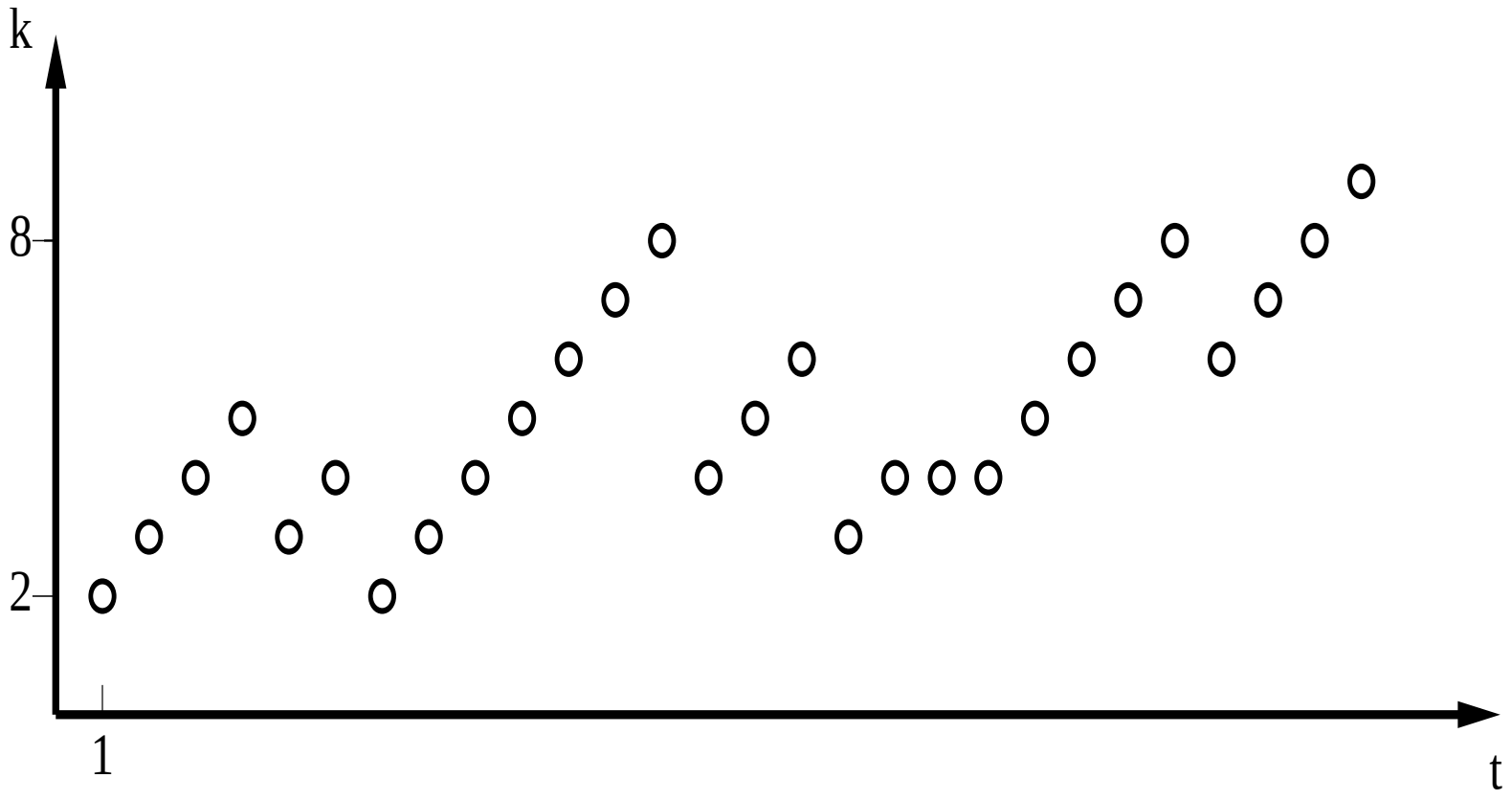
L'algorithme LLL

Entrée: $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ linéairement indépendants.

Sortie: Une base LLL-réduite de $L[\mathbf{b}_1, \ldots, \mathbf{b}_d]$.

1. [GS] Calculer les $\mu_{i,j}$ et $\|\mathbf{b}_i^*\|^2$.
2. $\kappa := 2$. Tant que $\kappa \leq d$,
3. [Hermite?] Proprifi er $\mathbf{b}_\kappa$ avec Babai.
4. [Lovász?] Si $(1^- - \mu_{\kappa,\kappa-1}^2)\|\mathbf{b}_{\kappa-1}^*\|^2 \leq \|\mathbf{b}_\kappa^*\|^2$, alors $\kappa := \kappa + 1$.
5. Sinon échanger $\mathbf{b}_{\kappa-1}$ et $\mathbf{b}_\kappa$, $\kappa := \max(\kappa - 1, 2)$.

Un exemple d'exécution de LLL en dimension 8



Les ronds sont les $k(t)$.

Analyse rapide de LLL

- Soit $B = \max_i ||\mathbf{b}_i||$.
- $O(d^2 \log B)$ itérations de boucle.
- Dans chaque itération, Babai domine: $O(d^2)$ *op.ar.*
- Les entiers (ou rationnels) sont de taille $O(d \log B)$.

- D'où: $O((d^2 \log B) \cdot d^2 \cdot (d \log B)^2) = O(d^6 \log^3 B)$.

C'est beaucoup trop cher!

- Cas de la factorisation d'un module RSA $N = p \cdot q$ quand la moitié des bits de p sont connus:

$$\log B \approx 1000,\ d \approx 64,\ \text{“}d^6 \log^3 B = 2^{66}\text{”}.$$

- L'arithmétique est beaucoup trop coûteuse.
- De Weger: rationnels $\longrightarrow$ entiers, même problème.
- LLL n'est pas utilisé pour de grands réseaux.

Une hiérarchie d'algorithmes

- Dim 1: Euclide, Euclide centré. Quadratique.
- Dim 2: Gauss (Legendre?). Quadratique.
- Dims 3 et 4: généralisation gloutonne de Gauss. Quadratique.
  [Nguyen-Stehlé, ANTS VI, 2004]
- Dim d: LLL. Cubique!

LLL flottant

- Dans LLL, les calculs liés à Gram-Schmidt dominent.
- Idée: faire les calculs de Gram-Schmidt (et Babai) en arithmétique flottante. Ça concerne les $\mu_{i,j}$ et les $\|\mathbf{b}_i^*\|^2$ uniquement.
- Calculs sur la base: toujours avec des entiers.
- L'idée date du début des années '80, avec la cryptanalyse des cryptosystèmes "sacs-à-dos".

L'arithmétique flottante?

- $x = \pm m_x \cdot 2^{e_x}$, avec $m_x \in [1/2, 1[$ sur ℓ bits.
- On veut que $\diamond(a \text{ op } b)$ soit un flottant le plus proche de $(a \text{ op } b)$, pour op $\in \{+, -, *, /\}$.
- Double précision: $1 + (52 + 1) + 11$ bits, LLL_{FP, XD} de NTL.
- Quadruple précision: $1 + (112 + 1) + 15$ bits, LLL_QP de NTL.
- Précision arbitraire: MPFR.

État de l'art

- [Schnorr, J. of A., 1988]: calcul de $(\mu_{i,j})_{i,j}^{-1}$ et itération de Schultz.
  $\Rightarrow$ Précision des calculs $\approx 12d + 7\log B$.
  $\Rightarrow O(d^4 \log B(d \log B)^2) \ \rightarrow O(d^4 \log B(d + \log B)^2)$.
- 3 problèmes: algo. dur à décrire, modèle "arithmétique flottante" non précisé, précision requise élevée. Pas implanté.
- [Koy-Schnorr, CALC, 2001]: Givens-Householder plutôt que Gram-Schmidt. Preuve fausse.
- Schnorr'04 (non publié): même complexité que Schnorr'88, avec une précision $\approx 5d + 2\log B$.

Et en pratique...

- Utilisation de variantes heuristiques non prouvées basées sur [Schnorr-Euchner, FCT, 1991].
- NTL et Magma utilisent Schnorr-Euchner.
- Pari est beaucoup plus lent.
- Koy-Schnorr'01 prétend que Schnorr-Euchner'91 est correct avec 53 bits de précision jusqu'en dimension 250.

Est-ce vrai?

On a un réseau de dimension 55 avec des entrées de 100 bits pour lequel NTL renvoie:

```
LLL_FP:warning--infinite loop?
LLL_FP:warning--infinite loop?
LLL_FP:warning--infinite loop?
```

Se méfier dès la dimension 35.

Nos résultats

Un LLL flottant prouvé dans un modèle précis.

	LLL'82	Schnorr'88	NS'05
Précision	$O(d \log B)$	$> 12d + 7 \log_2 B$	$d \log_2 3 \approx 1.58d$
Complexité	$O(d^6 \log^3 B)$	$O(d^4 (d + \log B)^2 \log B)$	$O(d^5 (d + \log B) \log B)$

LLL^2 est simple à décrire. Bonne version de LLL?

Ça fait tout!

- On n’a que la forme quadratique.
- Les vecteurs sont linéairement dépendants.
- Moins d’itérations de boucle que la borne la pire (sacs-à-dos).
- Conditions d’Hermite et de Lovász affaiblies.

Et ça semble efficace

- Codé en GNU-MP (arithmétique entière) et MPFR/DPE (arithmétique flottante en précision arbitraire/ “double”).
- Temps comparables à NTL et Magma, avec garanties sur le temps dans le cas le pire et sur la qualité de la base.
- Version heuristique plus rapide en cours d’écriture.

Les trois idées principales

- Utilisation exclusive de la matrice de Gram
  (pour éviter les pertes de précision liées aux produits scalaires).
- Version paresseuse de l'algorithme de Babai
  (pour éviter d'utiliser plus de bits de précision qu'il n'en faut).
- Généralisation en dimension d de la "cascade" de l'analyse de l'algorithme d'Euclide.

L'algorithme LLL^2

Même structure que LLL, $0.5 \rightarrow 0.5^+$ (dans Babai).

Entrée: Une base $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$, une précision ℓ.

Sortie: Une base LLL-réduite de $L[\mathbf{b}_1, \ldots, \mathbf{b}_d]$.

Var.: G, deux matrices fbttantes $\bar{r}$ et $\bar{\mu}$ et un vecteur fbttant $\bar{s}$.

1. Calculer la matrice de Gram $G = G(\mathbf{b}_1, \ldots, \mathbf{b}_d)$.
2. $\kappa := 2$. Tant que $\kappa \leq d$,
3. Proprifi er $\mathbf{b}_\kappa$ avec Babai itératif (mise jour de $\bar{\mu}$, $\bar{r}$ et $\bar{s}$).
4. Si $(1^- \bar{r}_{\kappa-1,\kappa-1} \leq \bar{s}_{\kappa-1})$, $\kappa := \kappa + 1$.
5. Sinon échanger $\mathbf{b}_{\kappa-1}$ et $\mathbf{b}_\kappa$, $\kappa := \max(\kappa - 1, 2)$.

Babai itératif: faire Babai jusqu'à ce que plus rien ne se passe.

Processus d'orthogonalisation (1)

- Gram-Schmidt $\longrightarrow$ Cholesky (on travaille sur la matrice de Gram).
- Minimiser le nombre d'opérations arithmétiques.
- Schnorr-Euchner: $\mu_{k,j} = \frac{\langle \mathbf{b}_k, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2} = \frac{\langle \mathbf{b}_k, \mathbf{b}_j \rangle - \sum_{i<j} \mu_{j,i}\mu_{k,i}\|\mathbf{b}_i^*\|^2}{\|\mathbf{b}_j^*\|^2}$.
- On stocke en plus les $r_{i,j} = \mu_{i,j}\|\mathbf{b}_j^*\|^2$, d'où les formules:

$$r_{i,j} = \langle \mathbf{b}_k, \mathbf{b}_j \rangle - \sum_{i<j} \mu_{j,i} r_{k,i} \text{ et } \mu_{i,j} = r_{i,j}/r_{j,j}.$$

Processus d'orthogonalisation (2)

- Et Householder? Et Givens?

  En général, seulement "backward stability".

- Ici, les premiers vecteurs sont LLL-réduits, et on veut utiliser la matrice de Gram.

- Avec Cholesky, on perd $\log_2 3 \cdot d$ bits, et la borne semble fine.

$$\begin{aligned} L_{i,i} &= (\sqrt{4/3})^{d-i} \\ L_{i,j} &= (-1)^{i-j+1} L_{i,i} \cdot \mathrm{random}[0.49, 0.5] \quad \text{si} \quad j > i \\ L_{i,j} &= 0 \quad \text{si} \quad j < i. \end{aligned}$$

L'algorithme de Babai, rappel

1. Calculer les $\mu_{k,i}$ pour $i < k$.
2. Pour $i = (k-1)..1$,
3. $\quad x_i := \lceil \mu_{\kappa,i} \rfloor$, $\mathbf{b}_k := \mathbf{b}_\kappa - x_i \mathbf{b}_i$,
4. $\quad$ Pour $j = 1..i$,
5. $\quad\quad \mu_{\kappa,j} := \mu_{\kappa,j} - x_i \mu_{i,j}$.

Résultat de stabilité pour Babai

- Précision de t bits

  $\Rightarrow$ les $\approx t - \log_2 3 \cdot d$ premiers bits des x_i corrects.

- Les x_i peuvent être de taille $O(d + \log B)$ dans le cas le pire, d'où une précision requise de $O(d + \log B)$ bits. C'est trop.

L'algorithme de Babai itératif

- Prendre une précision de $(\log_2 3 + c) \cdot d$ bits avec $c > 0$.
- Éxécuter Babai tant que les x_i de l'itération ne sont pas nuls.
- Calcul progressif des x_i en temps:
  $O\left(\left(1 + \frac{\log M}{cd}\right) d^3(d + \log B)\right)$
  $= O(d^2(d + \log B)(d + \log M)),$
  avec $M = \max|\mu_{k,i}| \approx \max|x_i|$.

Première analyse de complexité

- $O(d^2 \log B)$ itérations de boucle de LLL^2.
- Dans chaque itération, Babai itératif domine.
- Chaque Babai itératif coûte $O(d^2(d + \log B)(d + \log M))$.
- $\log M = O(d + \log B) \implies d^4 \log B(d + \log B)^2$.
- C'est comme Schnorr'88 !!!

Une analyse d'Euclide plutôt intéressante

- Euclide: pour $r_0 > r_1 > 0$ donnés, calculer pgcd(r_0, r_1).
- Deux suites q_i et r_i définies par:

$$q_i = \lfloor r_{i-1}/r_i \rfloor \text{ et } r_{i+1} = r_{i-1} - q_i r_i, \text{ jusqu'à } r_{\tau+1} = 0.$$

- $\tau = O(\log r_0)$ *op.ar.* sur des entiers de taille $O(\log r_0)$, d'où:

$$O(\log^3 r_0).$$

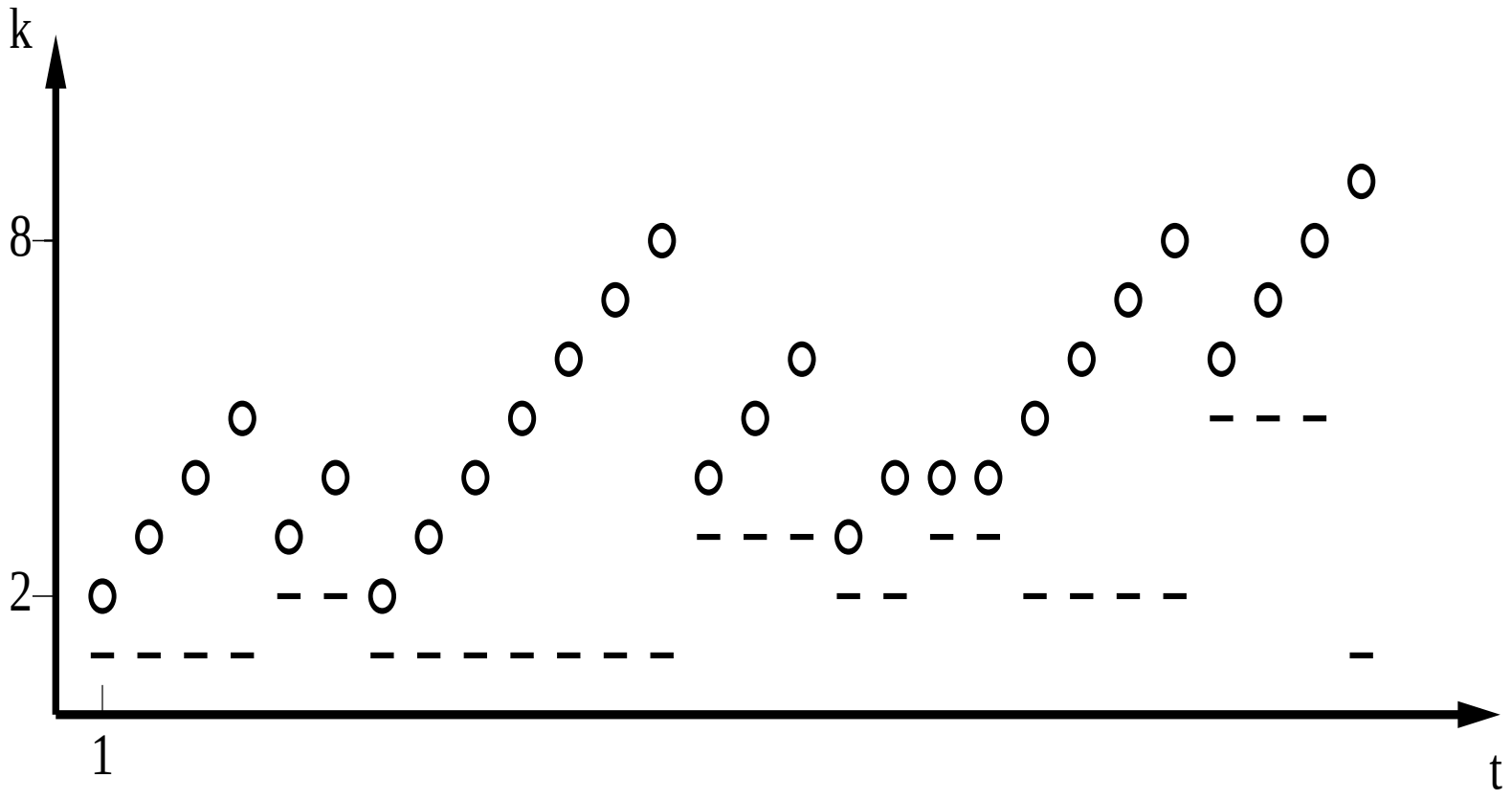
- En fait, moins que:

$$\log r_0 \cdot [(\log r_0 - \log r_1 + 1) + \ldots + (\log r_{\tau-1} - \log r_\tau + 1)]$$
$$= O(\log^2 r_0).$$

Comment faire apparaître une annulation ici?

- Babai itératif coûte $O(d^2(d+\log B)(d+\log M(t)))$.
- Coût total: $O(d^2(d+\log B)\sum_{\text{étape } t}(d+\log M(t))$.
- $M(t) = \max_{j<\kappa(t)}|\mu_{\kappa(t),j}| \leq d + \log\|\mathbf{b}_{\kappa(t)}\| - \log\|\mathbf{b}_{\alpha(t)}\|$.
- Avec $\alpha(t)$ le plus petit indice i tel que $\mathbf{b}_i$ a changé depuis la dernière fois que κ a valu $\kappa(t)$ au moins.
- Moralement: entre ces deux moments, on a travaillé dans l'orthogonal des $\mathbf{b}_i$ pour $i < \alpha$.

Un exemple d'exécution de LLL en dimension 8



Les ronds sont les $k(t)$ et les traits les $\alpha(t)$.

Problèmes ouverts et travaux en cours

- Implantation prouvée, implantation efficace.
- Complexité quasi-linéaire, comme pour le pgcd.
- Diminution du coût "algèbre linéaire".
- Généralisation à BKZ.
- Meilleure compréhension du comportement pratique de LLL.