

H-LLL : Un LLL vectoriel

Ivan Morel, Damien Stehlé, Gilles Villard

LIP-University of Sydney
CNRS-INRIA-ENSL-UCBL-Université de Lyon

Séminaire ALGORITHMS
25 mai 2009

Contexte

- Réseaux euclidiens.
- LLL-réduction.
- Utilisation de flottants pour accélérer les calculs de Gram-schmidt.
- Garantir la terminaison et la correction.

Pourquoi ?

La réduction de réseau est un outil puissant :

- Cryptographie.
- Arithmétique des ordinateurs.

Pourquoi ?

La réduction de réseau est un outil puissant :

- Cryptographie.
- Arithmétique des ordinateurs.

LLL reste coûteux :

- L'arithmétique entière est coûteuse.
- On se tourne naturellement vers les flottants.

Plan

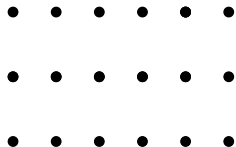
- 1 Réseaux euclidiens et décomposition QR
- 2 Les enjeux du flottant dans LLL
- 3 H-LLL : utiliser Householder dans LLL
- 4 Conclusion

Plan

- 1 Réseaux euclidiens et décomposition QR
- 2 Les enjeux du flottant dans LLL
- 3 H-LLL : utiliser Householder dans LLL
- 4 Conclusion

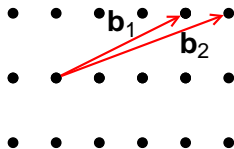
Réseaux euclidiens

- Un sous-groupe discret de \mathbb{Z}^n .



Réseaux euclidiens

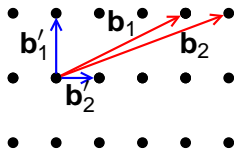
- Un sous-groupe discret de \mathbb{Z}^n .



- Décrit par une base $(\mathbf{b}_1, \dots, \mathbf{b}_d)$, représentée par une matrice B (colonnes).

Réseaux euclidiens

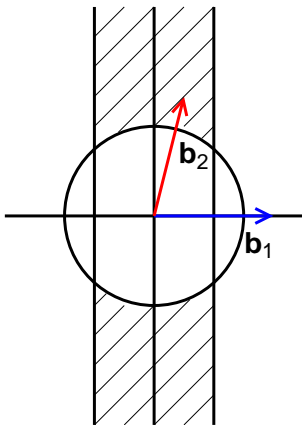
- Un sous-groupe discret de \mathbb{Z}^n .



- Décrit par une base $(\mathbf{b}_1, \dots, \mathbf{b}_d)$, représentée par une matrice B (colonnes).
- On passe d'une base à une autre par multiplication par une matrice unimodulaire.

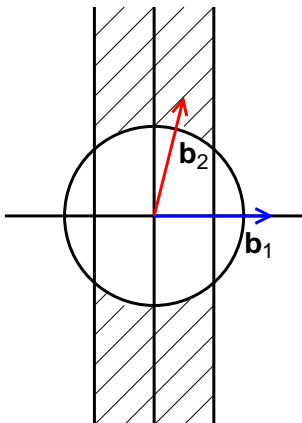
$$\begin{pmatrix} 3 & 4 \\ 1 & 1 \end{pmatrix} \cdot \underbrace{\begin{pmatrix} 4 & -1 \\ -3 & 1 \end{pmatrix}}_U = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Qualité d'une base en dimension 2



$$B = \begin{pmatrix} 1 & a \\ 0 & b \end{pmatrix}$$

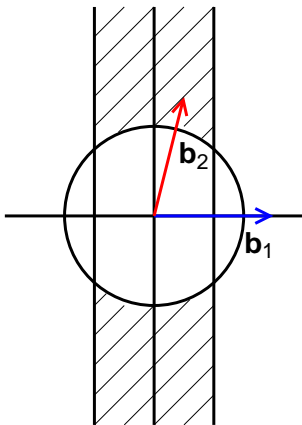
Qualité d'une base en dimension 2



$$B = \begin{pmatrix} 1 & a \\ 0 & b \end{pmatrix}$$

- Propreté : $a < \eta$.

Qualité d'une base en dimension 2



$$B = \begin{pmatrix} 1 & a \\ 0 & b \end{pmatrix}$$

- Propreté : $a < \eta$.
- Lovász : $\delta \leq a^2 + b^2$.

LLL-réduction

On étend la définition en utilisant la décomposition QR de B :

$$B = \begin{pmatrix} Q & \text{orthog.} \end{pmatrix} \cdot \begin{pmatrix} * & * & R \\ & * & * \\ (0) & & * \end{pmatrix}$$

- Propreté : $R_{i,j} \leq \eta R_{i,i}$ ($\eta \geq \frac{1}{2}$).
- Lovász : $\delta R_{i,i}^2 \leq R_{i,i+1}^2 + R_{i+1,i+1}^2$ ($1 - \eta^2 < \delta < 1$).

L'algorithme LLL

La structure classique de l'algorithme LLL est la suivante:

- $\kappa := 2$.
- Proprifier le κ -ième vecteur par rapport aux précédents.
- Si la condition de Lovász est vérifiée entre $\kappa - 1$ et κ :
 - Incrémenter κ .
 - Sinon échanger $\mathbf{b}_{\kappa-1}$ et \mathbf{b}_{κ} et décrémenter κ .

Plan

- 1 Réseaux euclidiens et décomposition QR
- 2 Les enjeux du flottant dans LLL**
- 3 H-LLL : utiliser Householder dans LLL
- 4 Conclusion

Introduction de flottants dans LLL

Le calcul des coefficients de Gram-Schmidt (R) et leur mise à jour dominant le coût de LLL.

Introduction de flottants dans LLL

Le calcul des coefficients de Gram-Schmidt (R) et leur mise à jour dominant le coût de LLL.

→ Introduction de flottants pour calculer R .

Introduction de flottants dans LLL

Le calcul des coefficients de Gram-Schmidt (R) et leur mise à jour dominant le coût de LLL.

→ Introduction de flottants pour calculer R .

Difficulté : prouver la cohérence des calculs effectués en se basant sur des approximations calculées de R .

Introduction de flottants dans LLL

Le calcul des coefficients de Gram-Schmidt (R) et leur mise à jour dominant le coût de LLL.

→ Introduction de flottants pour calculer R .

Difficulté : prouver la cohérence des calculs effectués en se basant sur des approximations calculées de R .

Historiquement :

- Les premières approches furent heuristiques.
- Schnorr 88 : première variante prouvée (grande précision).
- Nguyen, Stehlé 05 : L^2 utilisant Cholesky pour calculer R .

L^2

L^2 utilise l'algorithme de Cholesky pour le calcul de R :

- Calcul de R à partir de la matrice de Gram : $B^T B = R^T R$.
- Donne une bonne approximation de R dans LLL.
- Précision linéaire en d .

Intérêt de Householder

- Pas besoin de la matrice de Gram.
- Nombre d'opérations plus faible.
- Meilleur conditionnement : précision potentiellement plus faible.

Intérêt de Householder

- Pas besoin de la matrice de Gram.
- Nombre d'opérations plus faible.
- Meilleur conditionnement : précision potentiellement plus faible.

Le problème réside dans le type d'erreur commis par Householder.

Stabilité de Householder et perturbation de QR

Théorème [D. Stehlé, G. Villard & X.-W. Chang]

Soit une base $B = (\mathbf{b}_1, \dots, \mathbf{b}_d)$ LLL-réduite. Si, pour ε petit :

$$\forall i : \|\mathbf{b}_i - \tilde{\mathbf{b}}_i\| < \varepsilon \|\mathbf{b}_i\|$$

alors si R et \tilde{R} sont respectivement les facteurs R de B et \tilde{B} :

$$|R_{i,j} - \tilde{R}_{i,j}| = 2^{O(d)} \cdot \varepsilon \cdot R_{j,j}.$$

Perturbation de QR

Propreté

$$\begin{pmatrix} \dots & & & & & & \\ & R_{i,i} & \leftarrow \eta & & R_{i,j} & & \\ & & \dots & & & & \\ & (0) & & & R_{j,j} & & \\ & & & & & & \dots \end{pmatrix}$$

Perturbation de QR

Nouvelle propriété

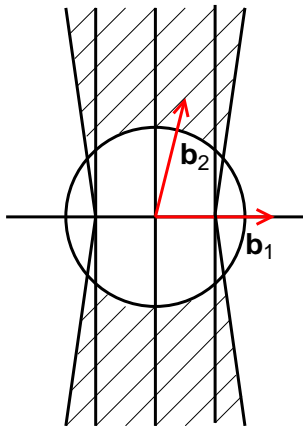
$$\begin{pmatrix} \ddots & & & & & & \\ & R_{i,i} & \leftarrow & R_{i,j} & & & \\ & & \ddots & \downarrow & \theta & & \\ & (0) & & R_{j,j} & & & \\ & & & & & \ddots & \end{pmatrix}$$

La LLL-réduction classique n'est pas stable par perturbation.

Nouvelle définition de propriété :

$$R_{i,j} \leq \eta R_{i,i} \rightarrow R_{i,j} \leq \eta R_{i,i} + \theta R_{j,j}$$

Qualité d'une base en dimension 2



Résultat principal

L'algorithme H-LLL renvoie une base (δ, η, θ) -LLL-réduite du réseau engendré par la base initiale $(\mathbf{b}_1, \dots, \mathbf{b}_d) \in \mathbb{Z}^{n \times d}$. De plus, sa complexité binaire est :

$$O(nd^4 \log \|B\| (d + \log \|B\|))$$

où $\|B\| = \max \|\mathbf{b}_i\|$.

Tableau comparatif

	Complexité	Précision
LLL rationnel	$O(nd^4 \log^2 \ B\ (d + \log \ B\))$	$d \log \ B\ $
Schnorr 88	$O(nd^3 \log \ B\ (d + \log \ B\)^2)$	$O(d + \log \ B\)$
L^2	$O(nd^4 \log \ B\ (d + \log \ B\))$	$d \log 3$
H-LLL	$O(nd^4 \log \ B\ (d + \log \ B\))$	d

Plus précisément, le coût de H-LLL est :

$$O \left[\left(d + \log \prod \frac{d_i^{initial}}{d_i^{final}} + \frac{1}{d} \log \prod \frac{\|\mathbf{b}_i^{initial}\|}{\|\mathbf{b}_i^{final}\|} \right) n\mathcal{M}(d)(d + \log \|B\|) \right],$$

où d_i est le déterminant de la matrice $i \times i$ ($\langle \mathbf{b}_j, \mathbf{b}_k \rangle \rangle_{j,k \leq i}$).

Difficultés

- Qualité de la sortie (introduction de θ).

Difficultés

- Qualité de la sortie (introduction de θ).
- Complique l'analyse numérique de l'algorithme.

Difficultés

- Qualité de la sortie (introduction de θ).
- Complique l'analyse numérique de l'algorithme.
- La preuve de correction est lourde.

Difficultés

- Qualité de la sortie (introduction de θ).
- Complique l'analyse numérique de l'algorithme.
- La preuve de correction est lourde.
- Mais l'analyse de complexité est simplifiée par la structure de l'algorithme.

Plan

- 1 Réseaux euclidiens et décomposition QR
- 2 Les enjeux du flottant dans LLL
- 3 H-LLL : utiliser Householder dans LLL**
- 4 Conclusion

H-LLL

Principe :

- Effectuer la proprification en utilisant \tilde{R} calculé avec Householder.
- Tester Lovász en utilisant les approximations calculées par Householder.

H-LLL

Principe :

- Effectuer la propprification en utilisant \tilde{R} calculé avec Householder.
- Tester Lovász en utilisant les approximations calculées par Householder.

Difficultés :

- Le vecteur courant n'est pas (encore) réduit.
- L'erreur faite sur la colonne associée de R est liée à la norme du vecteur.

Proprification paresseuse

- Initialement on ne connaît que les bits de poids forts de la κ -ième colonne de R .

Proprification paresseuse

- Initialement on ne connaît que les bits de poids forts de la κ -ième colonne de R .
- La proprification est effectuée en plusieurs passes.
- Chaque passe diminue la longueur du vecteur.

Proprification paresseuse

- Initialement on ne connaît que les bits de poids forts de la κ -ième colonne de R .
- La proprification est effectuée en plusieurs passes.
- Chaque passe diminue la longueur du vecteur.
- On s'arrête lorsque la longueur du vecteur ne diminue plus.

Algorithme de proprification

- Calculer une approximation \tilde{R}_κ de la κ -ième colonne de R .
- Pour i allant de $\kappa - 1$ à 1 :
 - $X_i = \left\lfloor \diamond \left(\frac{\tilde{R}_{i,\kappa}}{\tilde{R}_{i,i}} \right) \right\rfloor$.
 - Pour j allant de 1 à $i - 1$: $\bar{R}_{j,\kappa} := \diamond \left(\tilde{R}_{j,\kappa} - \diamond \left(X_i \tilde{R}_{j,i} \right) \right)$.
- $\mathbf{b}_\kappa := \mathbf{b}_\kappa - \sum_{i < \kappa} X_i \mathbf{b}_i$.
- Recommencer si $\|\mathbf{b}_\kappa\|^2$ a suffisamment diminué.

Une proprification ad-hoc

L'algorithme de proprification n'effectue pas à proprement parler une proprification.

En sortie, avec une précision $\approx d$, on a :

- Un vecteur suffisamment petit pour pouvoir tester Lovász.
- Et si Lovász est vérifié, alors le vecteur est propre.
- Sinon, κ est décrémenté.

Complexité de la proprification

Pour réduire le vecteur \mathbf{b}_{κ} :

- Le nombre de passes est $O\left(1 + \frac{1}{d} \log \frac{\|\mathbf{b}_{\kappa}^{initial}\|}{\|\mathbf{b}_{\kappa}^{final}\|}\right)$.
- Chaque passe coûte $O(n\mathcal{M}(d)(d + \log \|B\|))$.

La nouvelle structure de l'algorithme permet ainsi de majorer facilement le coût total des proprifications.

Complexité de H-LLL

- Le nombre d'itérations de LLL est $O\left(d + \log \prod \frac{d_i^{initial}}{d_i^{final}}\right)$.
- Expression de la complexité en fonction du travail effectué :

$$O\left[\left(d + \log \prod \frac{d_i^{initial}}{d_i^{final}} + \frac{1}{d} \log \prod \frac{\|\mathbf{b}_i^{initial}\|}{\|\mathbf{b}_i^{final}\|}\right) n\mathcal{M}(d)(d + \log \|B\|)\right],$$

où d_i est le déterminant de la matrice $(\langle \mathbf{b}_j; \mathbf{b}_k \rangle)_{j,k \leq i}$.

Plan

- 1 Réseaux euclidiens et décomposition QR
- 2 Les enjeux du flottant dans LLL
- 3 H-LLL : utiliser Householder dans LLL
- 4 Conclusion**

Conclusion et travaux en cours

- Un algorithme flottant plus naturel.
- Une précision requise moins élevée.
- Un nouvel algorithme de proprification.
- Travail en cours pour améliorer la précision requise.