Trouver un vecteur le plus court dans un réseau euclidien

Damien STEHLÉ

http://perso.ens-lyon.fr/damien.stehle

Travail en commun avec Guillaume HANROT (INRIA Lorraine)

CNRS/LIP/INRIA/ÉNS Lyon/Université de Lyon

Plan de l’exposé

- Les réseaux et le problème du plus court vecteur

Plan de l'exposé

- Les réseaux et le problème du plus court vecteur
- La cryptographie reposant sur les réseaux

Plan de l'exposé

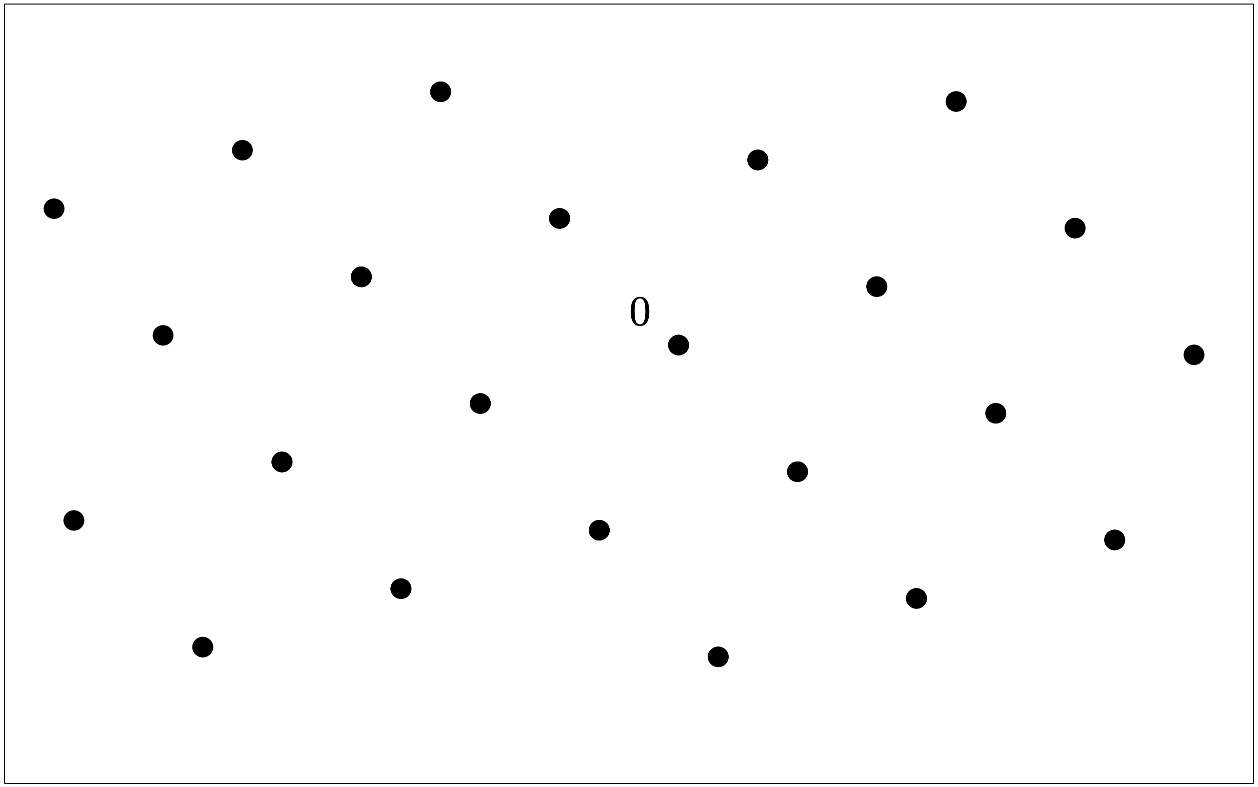
- Les réseaux et le problème du plus court vecteur
- La cryptographie reposant sur les réseaux
- L'algorithme d'énumération de vecteurs courts

Plan de l'exposé

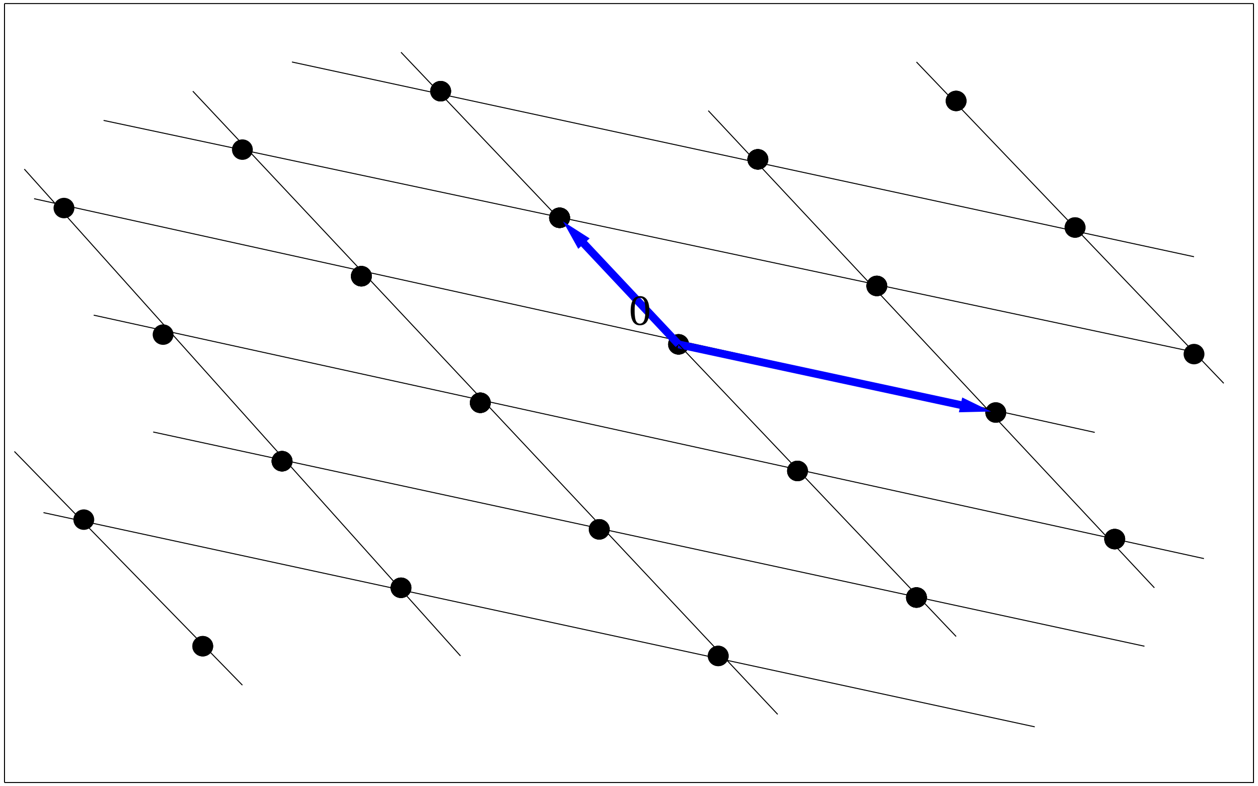
- Les réseaux et le problème du plus court vecteur
- La cryptographie reposant sur les réseaux
- L'algorithme d'énumération de vecteurs courts
- Complexité de la résolution du problème du plus court vecteur (bornes inférieures et supérieures)

1) Les réseaux et le problème du plus court vecteur

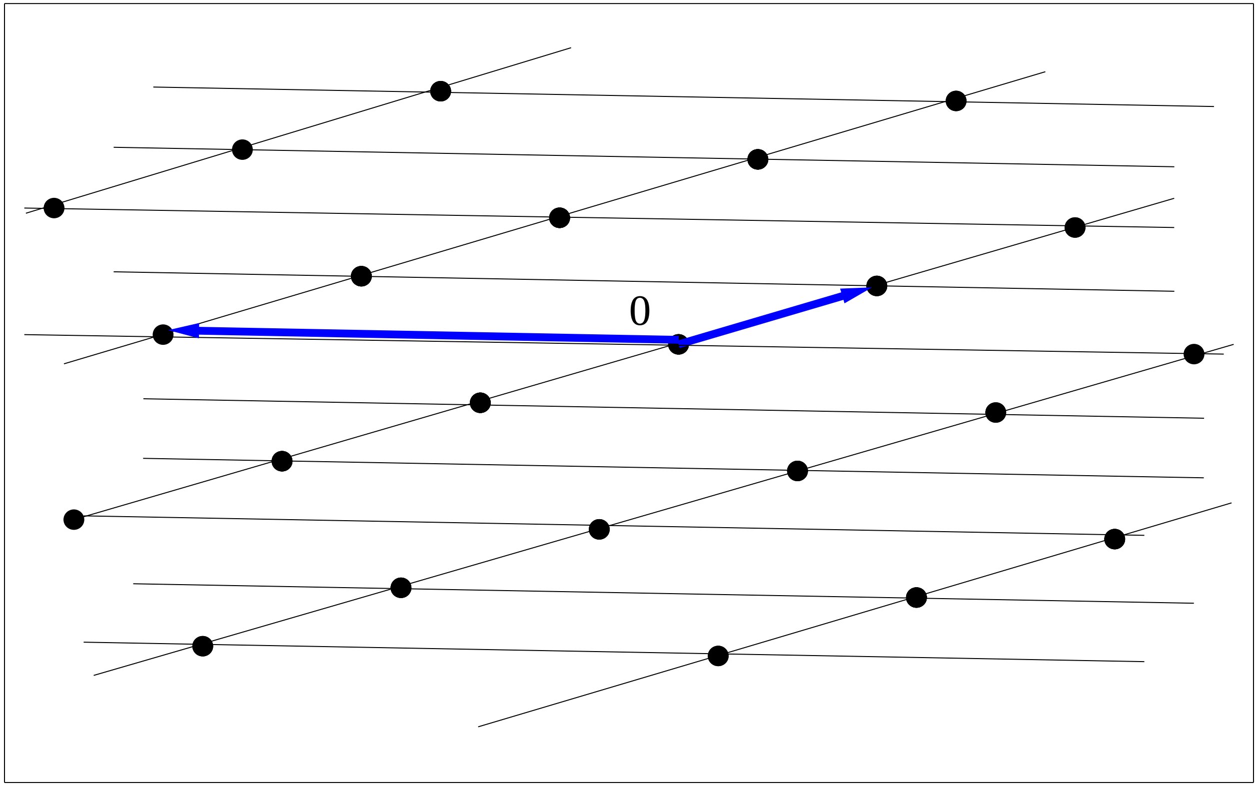
Un réseau est une grille infinie



On le représente par une base

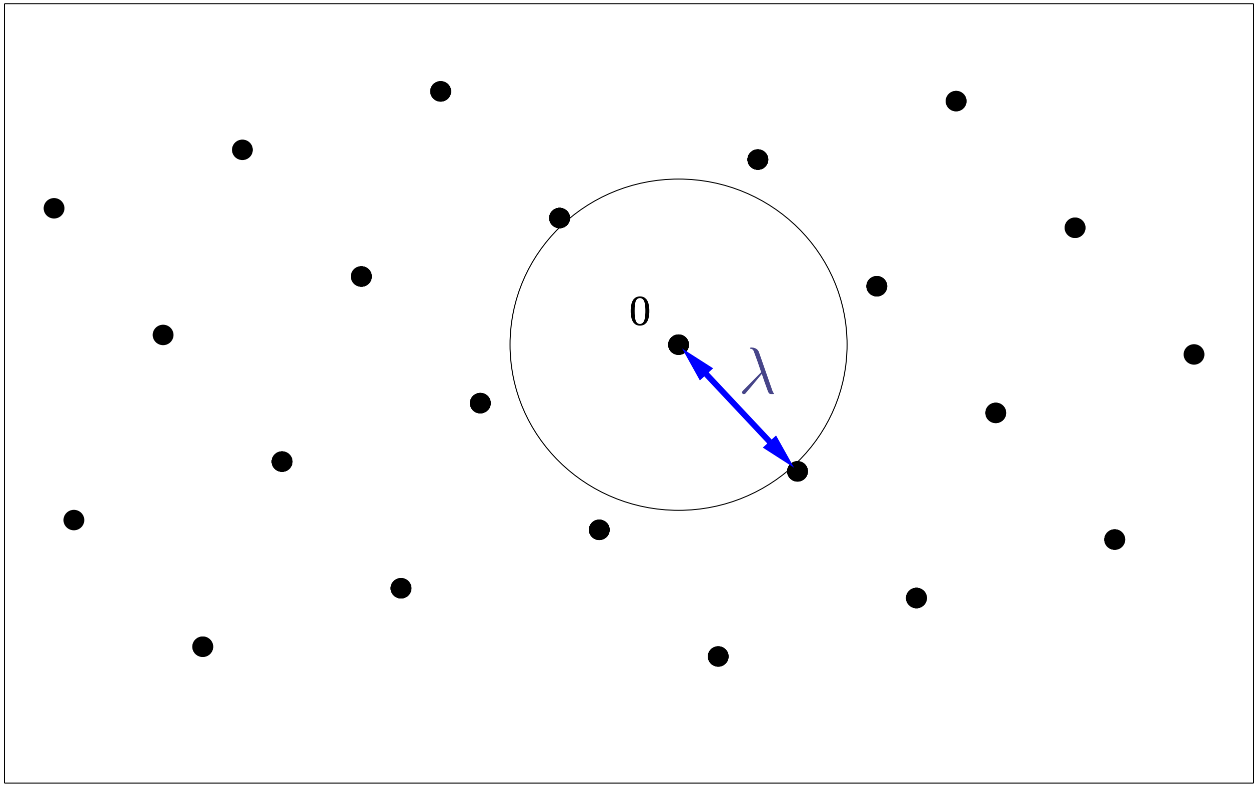


Il n'y a pas unicité . . .



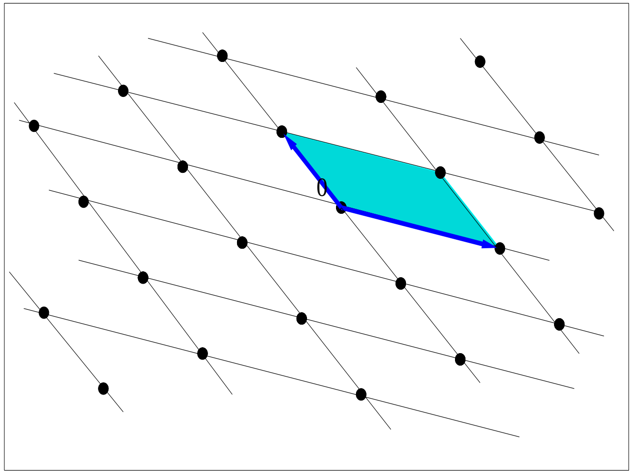
Le premier minimum

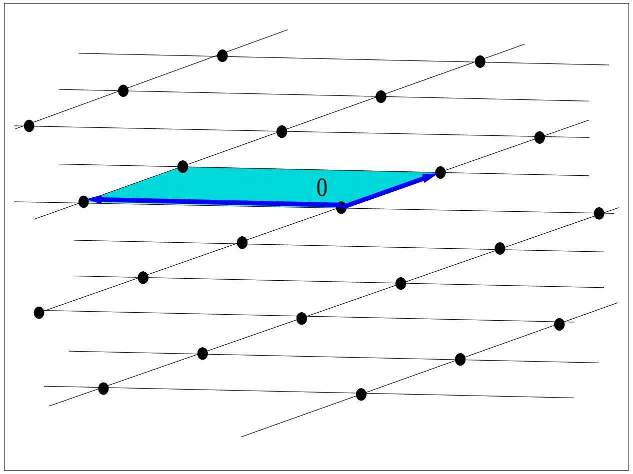
Longueur d'un plus petit vecteur non nul.



Le volume

$\det(L) = \mathrm{vol}(L)$: volume d-dimensionnel de tout parallélépipède fondamental.





Les réseaux euclidiens

- Réseau = sous-groupe discret de $\mathbb{R}^n$:

$$L[\mathbf{b}_1, \ldots, \mathbf{b}_d] = \left\{ \sum_{i=1}^{d} x_i \mathbf{b}_i, x_i \in \mathbb{Z} \right\}$$

- Si les b_i sont indépendants sur $\mathbb{R}$, ils forment une base.
- Dimension = d.
- Minimum = Longueur minimale d'un vecteur non-nul.
- Déterminant = $\det(L)$ = Volume du parallélépipède engendré par une base quelconque.
- Les bases sont liées entre elles par des transformations unimodulaires (entières de déterminant ± 1).

SVP

SVP : Étant donnée une base d'un réseau, trouver un vecteur non-nul le plus court.

- Conjecturé NP-difficile par van Emde Boas en 1982.
- Prouvé NP-difficile sous des réductions randomisées par Ajtai en 1997.
- Théorème de Minkowski :

$$\exists \mathbf{b} \in L \setminus \{\mathbf{0}\}, \quad \|\mathbf{b}\| \leq \sqrt{d} \cdot \det(L)^{1/d}.$$

Réduction : un problème de représentation

Trouver une “bonne” base à partir d’une base quelconque.

Réduction : un problème de représentation

Trouver une “bonne” base à partir d’une base quelconque.

- LLL : Lenstra, Lenstra et Lovász 1982.
  Fournit un vecteur relativement court.
  En temps polynomial.

Réduction : un problème de représentation

Trouver une “bonne” base à partir d’une base quelconque.

- LLL : Lenstra, Lenstra et Lovász 1982.
  Fournit un vecteur relativement court.
  En temps polynomial.
- HKZ : Hermite, Korkine et Zolotarev.
  Le premier vecteur atteint le minimum,
  et orthogonalement à ce dernier, la base
  est HKZ-réduite.
  Coûte un temps exponentiel.

Algorithmes résolvant SVP

- Fincke-Pohst ('83) : énumeration de points à coordonnées entières dans des hyper-ellipsoïdes après une LLL-réduction .
- Kannan ('83), Helfrich ('85) : énumeration dans des hyper-parallélépipèdes après une quasi-HKZ-réduction.
- Ajtai-Kumar-Sivakumar ('01) : repose essentiellement sur le principe des tiroirs.

Algorithmes résolvant SVP

	FP	KH	AKS
	déterministe	déterministe	probabiliste
Temps	$\left(2^{O(d^2)}\right)$	$d^{d/2}$	$2^{O(d)}$
Espace	polynomial	polynomial	$2^{O(d)}$

Comportement pratique de AKS étudié par Nguyen et Vidick (J. of Math. Crypto, 2008).
La constante du $O(\cdot)$ est relativement petite.
Ici : On va étudier précisément la complexité de KH.

2) Motivations cryptographiques

Les deux facettes des réseaux en crypto

[voir Nguyen-Stern, Calc'01]

- Cryptanalyse:
  - Depuis le début des années 1980.
  - Sacs-à-dos, générateurs pseudo-aléatoires, variantes de RSA, ...
  - Ces attaques reposent le plus souvent sur LLL.
- Cryptosystèmes:
  - Depuis le milieu des années 1990, après les résultats d'Ajtai sur la complexité de SVP.
  - Ajtai-Dwork, GGH, NTRU, ...
  - LLL ne suffit pas pour les casser.

NTRU `http://www.ntru.com`

- Clé publique $= B = \left[\begin{array}{cccc|cccc} 1 & 0 & \ldots & 0 & h_1 & h_2 & \ldots & h_N \\ 0 & 1 & \ldots & 0 & h_N & h_1 & \ldots & h_{N-1} \\ \vdots & \vdots & \ldots & \vdots & \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & 1 & h_2 & h_3 & \ldots & h_1 \\ \hline 0 & 0 & \ldots & 0 & q & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & 0 & q & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots & \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & q \end{array}\right].$

NTRU http://www.ntru.com

- Clé publique $= B =$ $$\left[\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & h_1 & h_2 & \dots & h_N \\ 0 & 1 & \dots & 0 & h_N & h_1 & \dots & h_{N-1} \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 1 & h_2 & h_3 & \dots & h_1 \\ \hline 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{array}\right].$$
- Clé privée : excellente base du même réseau.

- Clé publique $= B = \left[\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & h_1 & h_2 & \dots & h_N \\ 0 & 1 & \dots & 0 & h_N & h_1 & \dots & h_{N-1} \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 1 & h_2 & h_3 & \dots & h_1 \\ \hline 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{array}\right]$.
- Clé privée : excellente base du même réseau.
- Chiffrement : $\mathbf{m} \to \mathbf{m} \cdot B + \mathbf{e}$.

NTRU http://www.ntru.com

- Clé publique $= B = \left[\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & h_1 & h_2 & \dots & h_N \\ 0 & 1 & \dots & 0 & h_N & h_1 & \dots & h_{N-1} \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 1 & h_2 & h_3 & \dots & h_1 \\ \hline 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{array}\right].$
- Clé privée : excellente base du même réseau.
- Chiffrement : $\mathbf{m} \to \mathbf{m} \cdot B + \mathbf{e}$.
- Déchiffrement : à l'aide de la bonne base, trouver $\mathbf{b} \in L$ proche de $\mathbf{c}$. Si tout va bien, $\mathbf{m} = \mathbf{b} \cdot B^{-1}$.

SWIFFT

[Lyubashevsky, Micciancio, Peikert et Rosen, FSE'08]
Fonction de hachage. Pour hacher $\mathbf{x} \in \mathbb{Z}_d^{mn}$:

$$\mathbf{x} \cdot \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{pmatrix} \in \mathbb{Z}_p^n, \text{ avec } A_i = \begin{pmatrix} a_0^{(i)} & a_1^{(i)} & \dots & a_{n-1}^{(i)} \\ -a_{n-1}^{(i)} & a_0^{(i)} & \dots & a_{n-2}^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ -a_1^{(i)} & -a_2^{(i)} & \dots & a_0^{(i)} \end{pmatrix}.$$

Les A_i et p sont publics. Trouver une collision est au moins aussi difficile qu'un certain problème sur les réseaux, dans le cas le pire.

Comment casser SWIFFT

Il suffit de trouver un vecteur à coordonnées dans $[-d+1, d-1]$ qui est dans le réseau engendré par les lignes d'une matrice $mn \times mn$ du type :

$$\begin{pmatrix} G_1 & \mathsf{Id}_n & 0 & \dots & 0 \\ G_2 & 0 & \mathsf{Id}_n & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ G_{m-1} & 0 & 0 & \dots & \mathsf{Id}_n \\ p\mathsf{Id}_n & 0 & 0 & \dots & 0 \end{pmatrix},$$

avec les G_i similaires aux A_i.

Paramètres pratiques

- NTRU-251 : $N = 251, q = 128$.
  Dimension : 502.
- NTRU-503 : $N = 503, q = 256$.
  Dimension : 1006.
- SWIFFT-Mini : $n = 128, m = 8, d = 3, p = 257$.
  Dimension : 1024.

Autres fonctions reposant sur les réseaux

- LASH : Hachage à l'aide des réseaux. Cassé par Contini, Matusiewicz, Pieprzyk et Steinfeld (FSE'08).
- NTRUSign.
- Gentry, Peikert, Vaikuntanathan (eprint 2007/432): "Hash-and-sign", chiffrement reposant sur l'identité, etc.
- Aguilar-Melchor et Gaborit : PIR.

Cryptanalyse de ces fonctions

- Les vecteurs intéressants sont si petits qu'ils sont difficiles à obtenir :

LLL ne suffit pas.

Cryptanalyse de ces fonctions

- Les vecteurs intéressants sont si petits qu'ils sont difficiles à obtenir :

LLL ne suffit pas.

- Ils sont significativement plus courts que les autres :

HKZ est trop puissant.

Cryptanalyse de ces fonctions

- Les vecteurs intéressants sont si petits qu'ils sont difficiles à obtenir :

  LLL ne suffit pas.

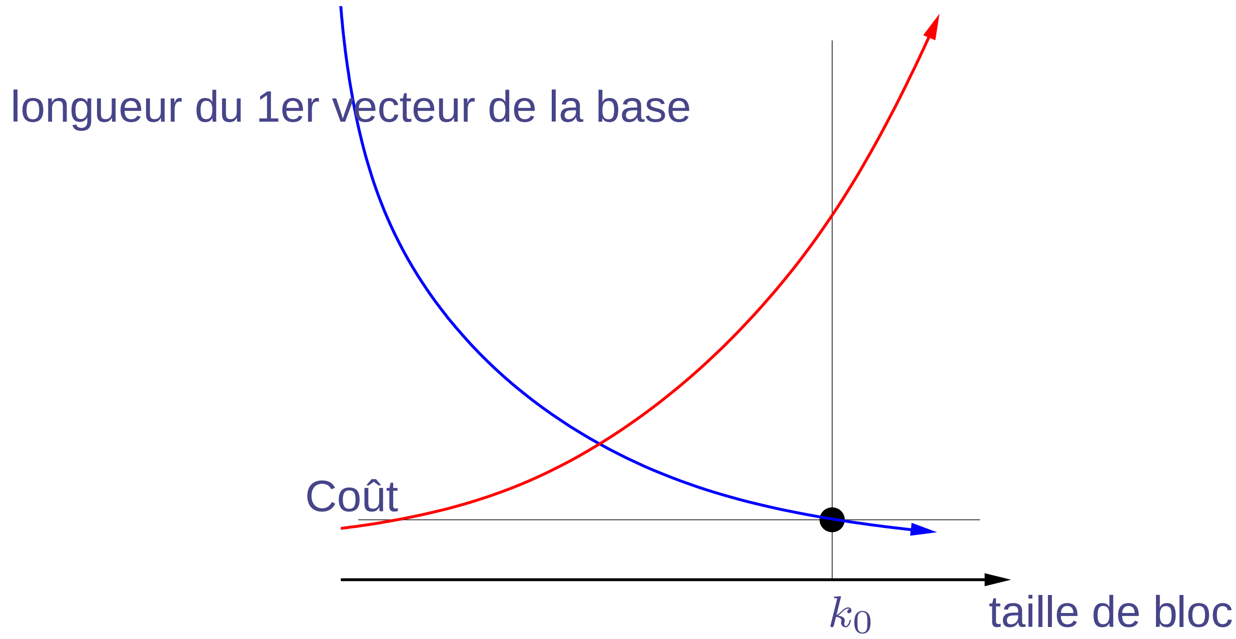
- Ils sont significativement plus courts que les autres :

  HKZ est trop puissant.

- On utilise la hiérarchie d'algorithmes de Schnorr .
  - Ils travaillent sur des blocs plutôt que sur des vecteurs.
  - À l'intérieur des blocs, on résout des instances de SVP/HKZ.

Hiérarchie de Schnorr

- Étant donné un réseau provenant de la cryptographie, quelle est la plus petite taille de bloc k_0 qui fournit des vecteurs intéressants?



- La sécurité est donnée par k_0. Nombre polynomial de résolutions de HKZ/SVP en dimension k_0.
- Quelle est la plus grosse taille de bloc que l'on peut considérer en pratique ?

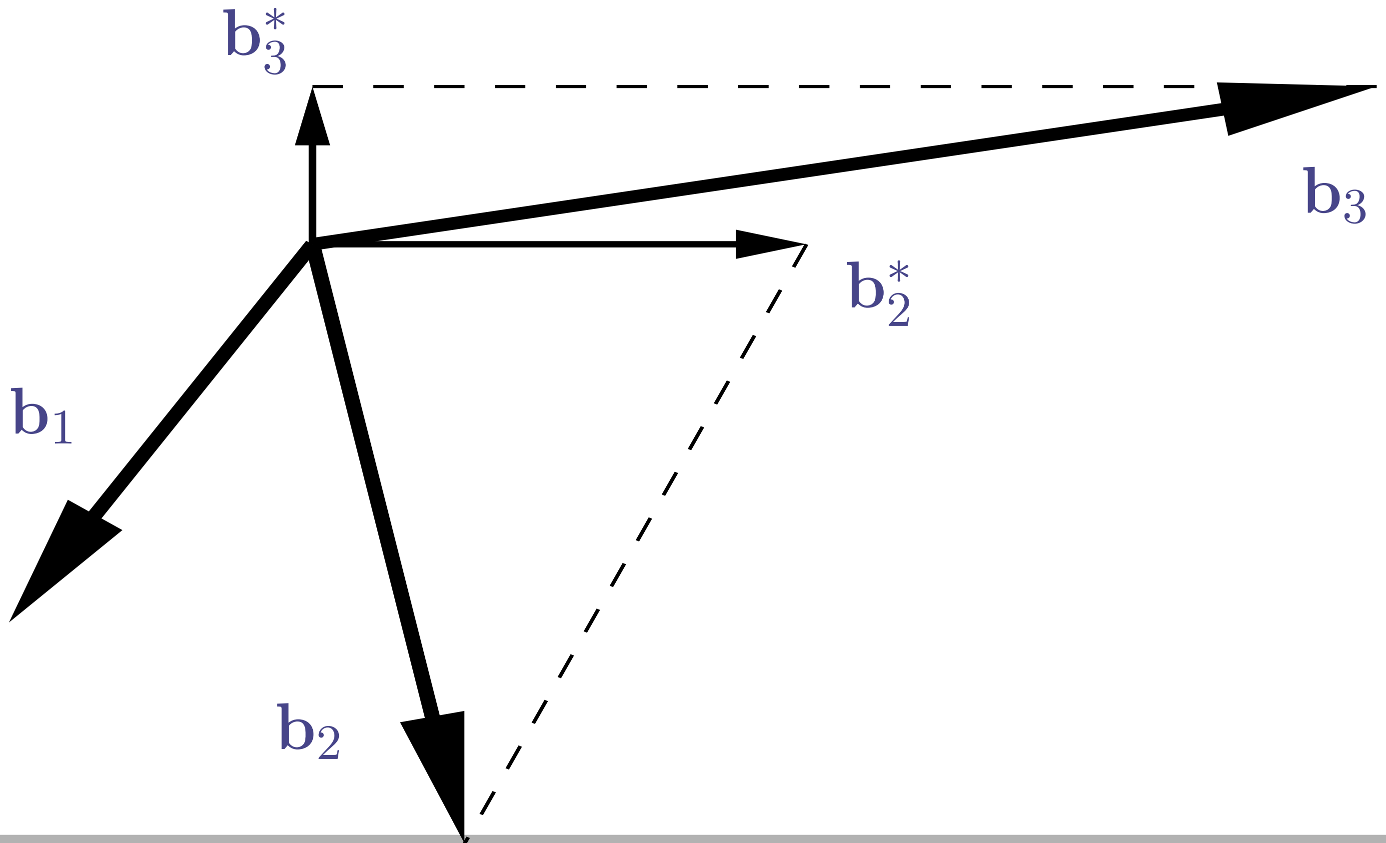
Motivations non cryptographiques

- Théorie algorithmique des nombres : calculer le groupe des unités dans un corps de nombres.
- Géométrie des nombres : minimas, kissing number, séries théta de réseaux, etc.
- Théorie des communications : décodage MIMO.

3) L'algorithme d'énumération de vecteurs courts

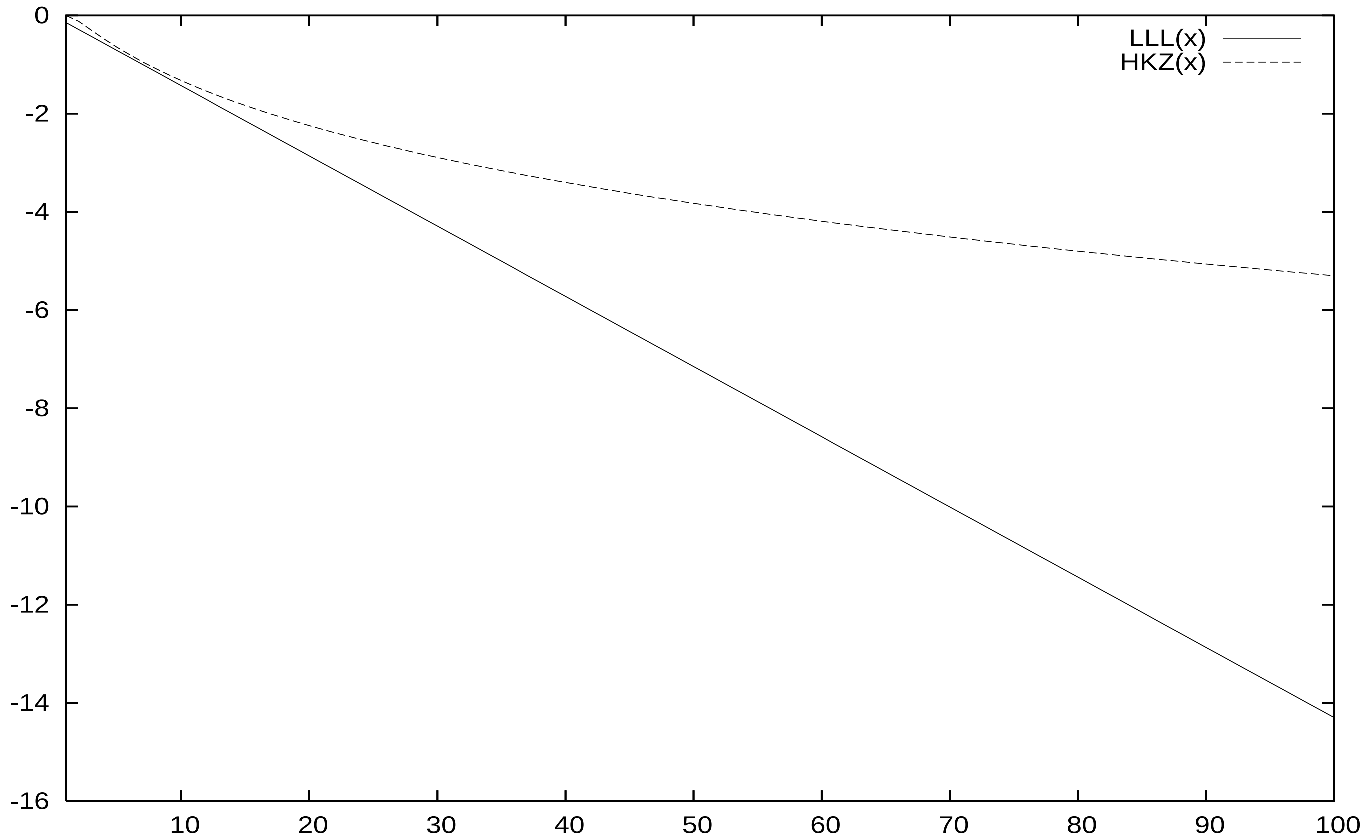
L'orthogonalisation de Gram-Schmidt

- Procédé itératif pour orthogonaliser $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$.
- $\mathbf{b}_1^* = \mathbf{b}_1, \;\; \mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*, \;\; \mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2}$.
- $\det(L[\mathbf{b}_i]) = \prod \|\mathbf{b}_i^*\|$.



Quantifier la qualité d'une base

Moins les $\|\mathbf{b}_i^*\|$ décroissent, plus la base est orthogonale.



Courbes bornant les cas le pire de $\ln \|\mathbf{b}_i^*\|$.

Le principe de l'énumération

Étant donnés $\mathbf{b}_1, \ldots, \mathbf{b}_d$, on cherche les $x_i \in \mathbb{Z}$ tels que :

$$\|x_1\mathbf{b}_1 + \ldots + x_d\mathbf{b}_d\|^2 = \sum_i (x_i + \sum_{j>i} \mu_{j,i} x_j)^2 \|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_1\|^2$$

Le principe de l'énumération

Étant donnés $\mathbf{b}_1, \ldots, \mathbf{b}_d$, on cherche les $x_i \in \mathbb{Z}$ tels que :

$$\|x_1\mathbf{b}_1 + \ldots + x_d\mathbf{b}_d\|^2 = \sum_i (x_i + \sum_{j>i} \mu_{j,i} x_j)^2 \|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_1\|^2$$

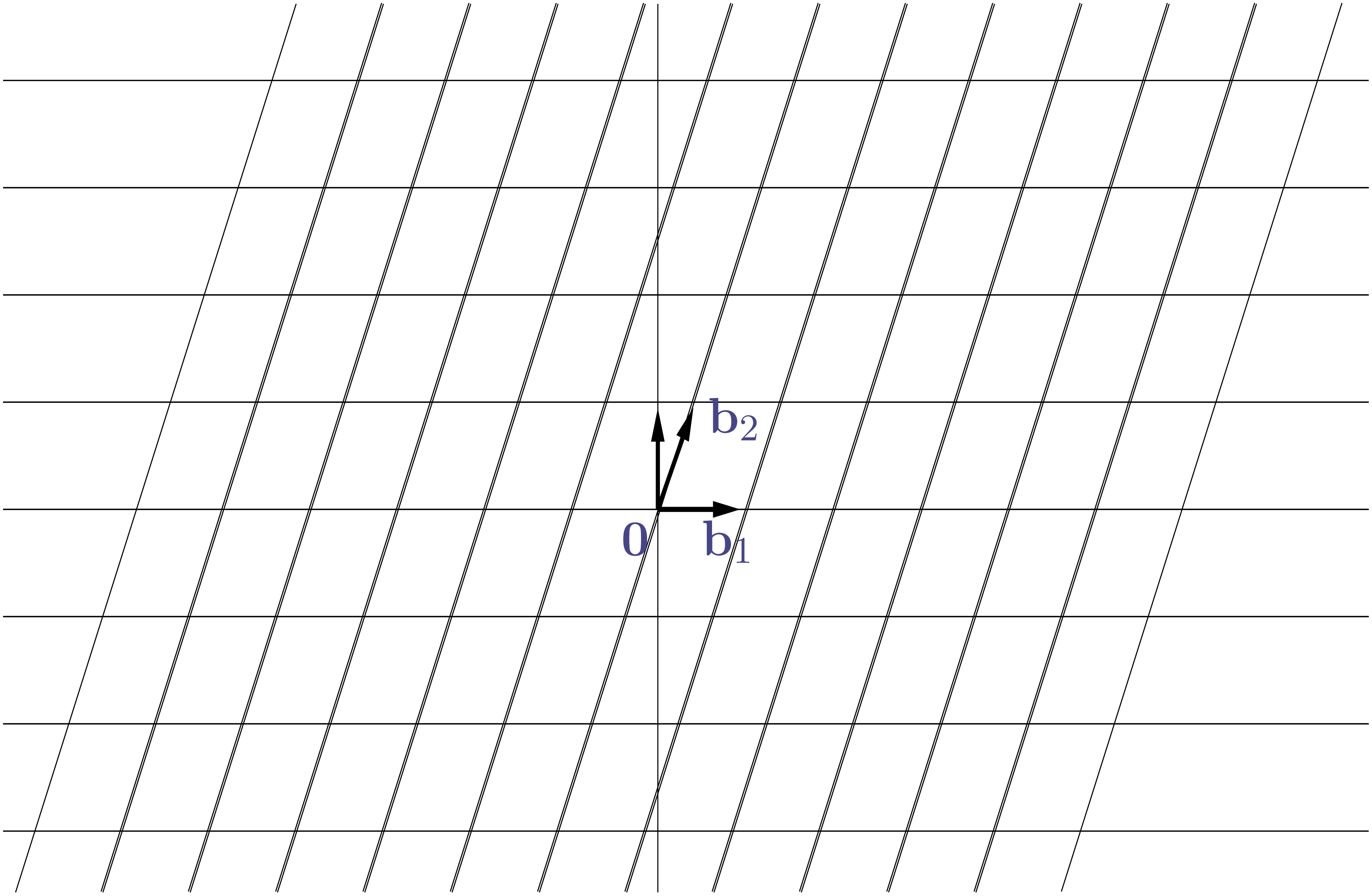
En regardant les composantes sur les $\mathbf{b}_i^*$:

$$\begin{aligned}
x_d^2\|\mathbf{b}_d^*\|^2 &\leq \|\mathbf{b}_1\|^2 \\
(x_{d-1} + \mu_{d,d-1}x_d)^2\|\mathbf{b}_{d-1}^*\|^2 + x_d^2\|\mathbf{b}_d^*\|^2 &\leq \|\mathbf{b}_1\|^2 \\
&\ldots \\
\sum_{j\geq i}(x_j + \sum_{k>j}\mu_{k,j}x_k)^2\|\mathbf{b}_j^*\|^2 &\leq \|\mathbf{b}_1\|^2
\end{aligned}$$

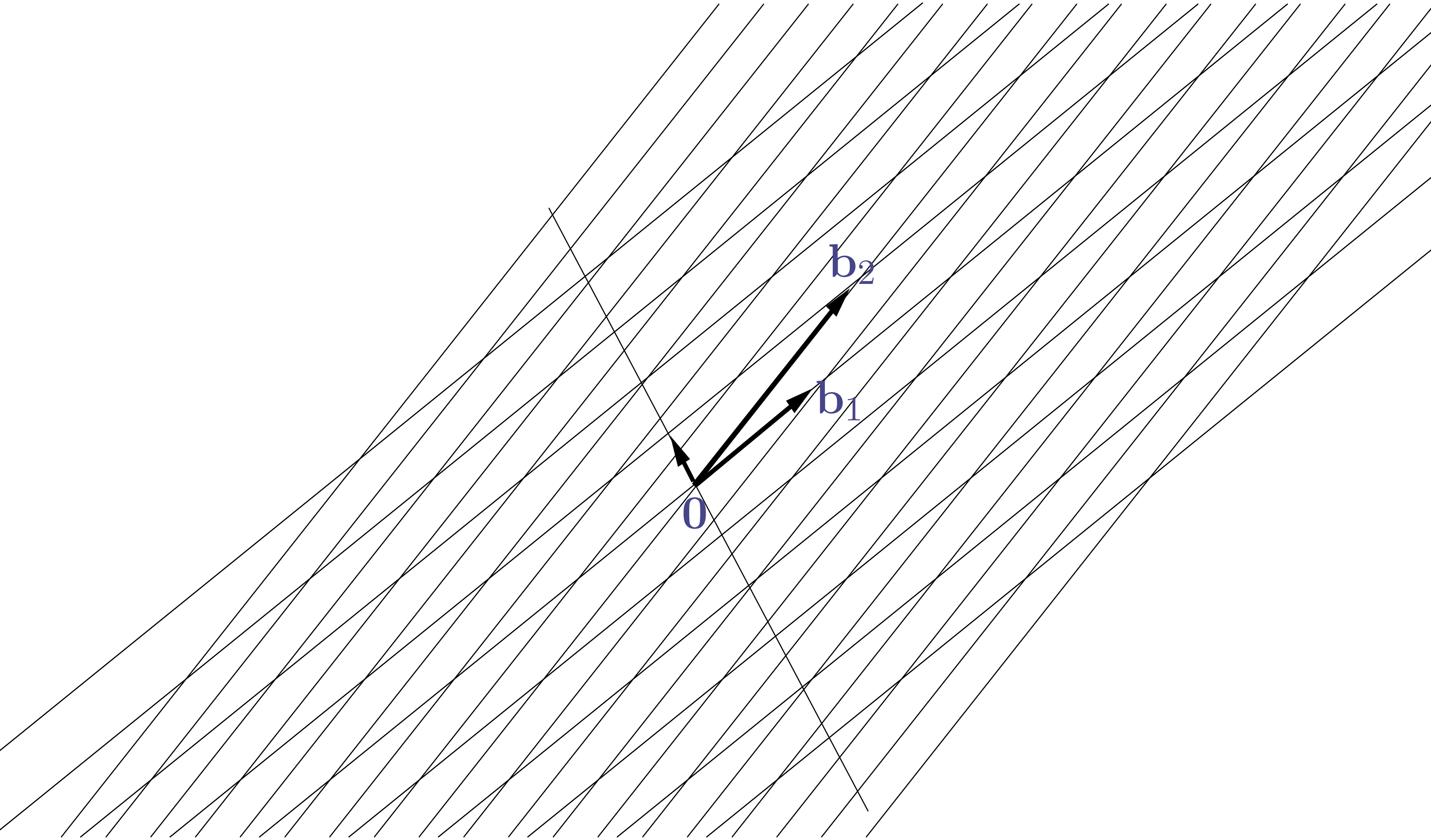
Trois interprétations

- Des points du réseau dans des hyper-boules.
- Des points entiers dans des hyper-ellipsoïdes.
- Un parcours d'arbre.

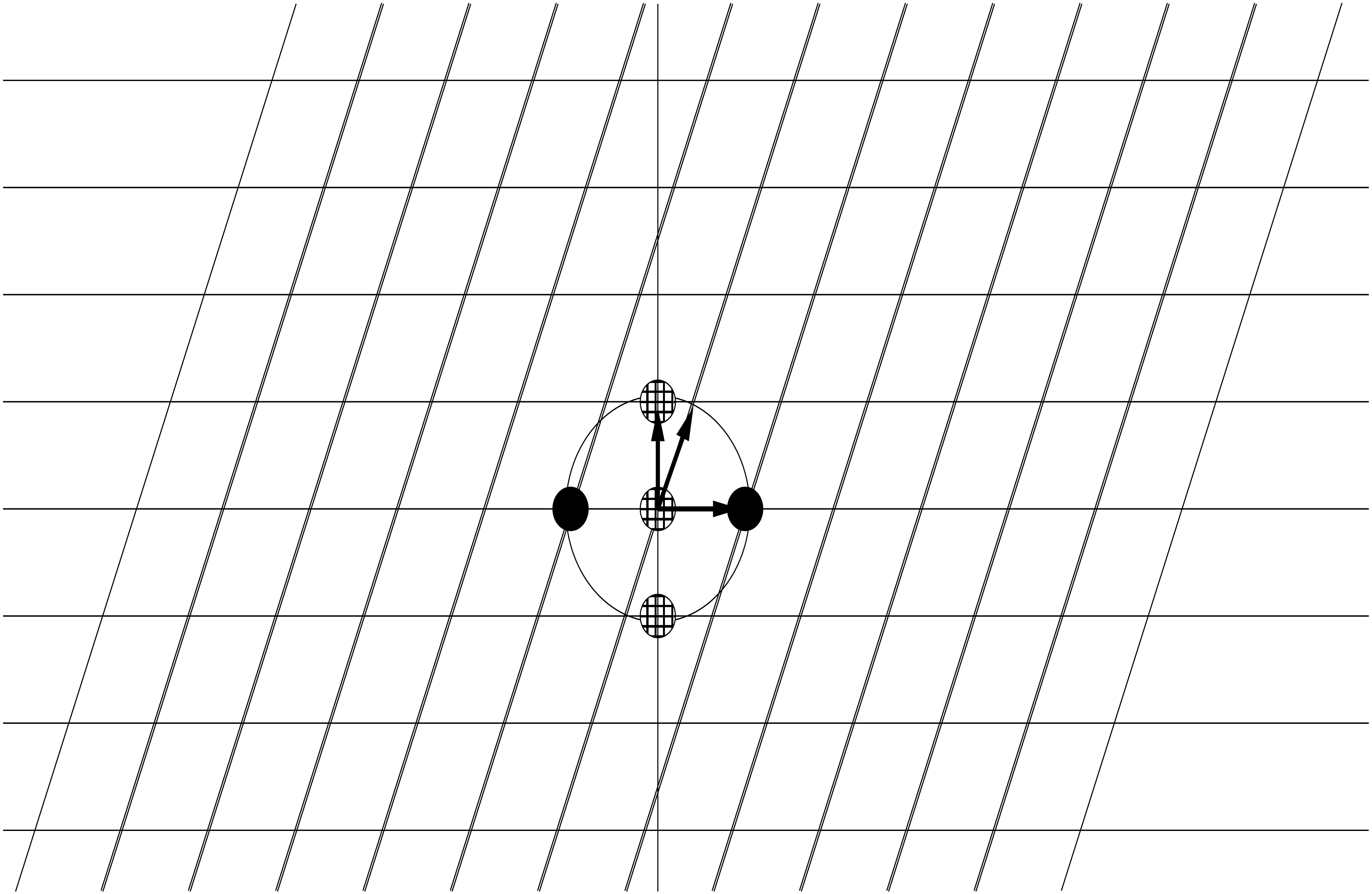
Le principe de Kannan : pré-processer



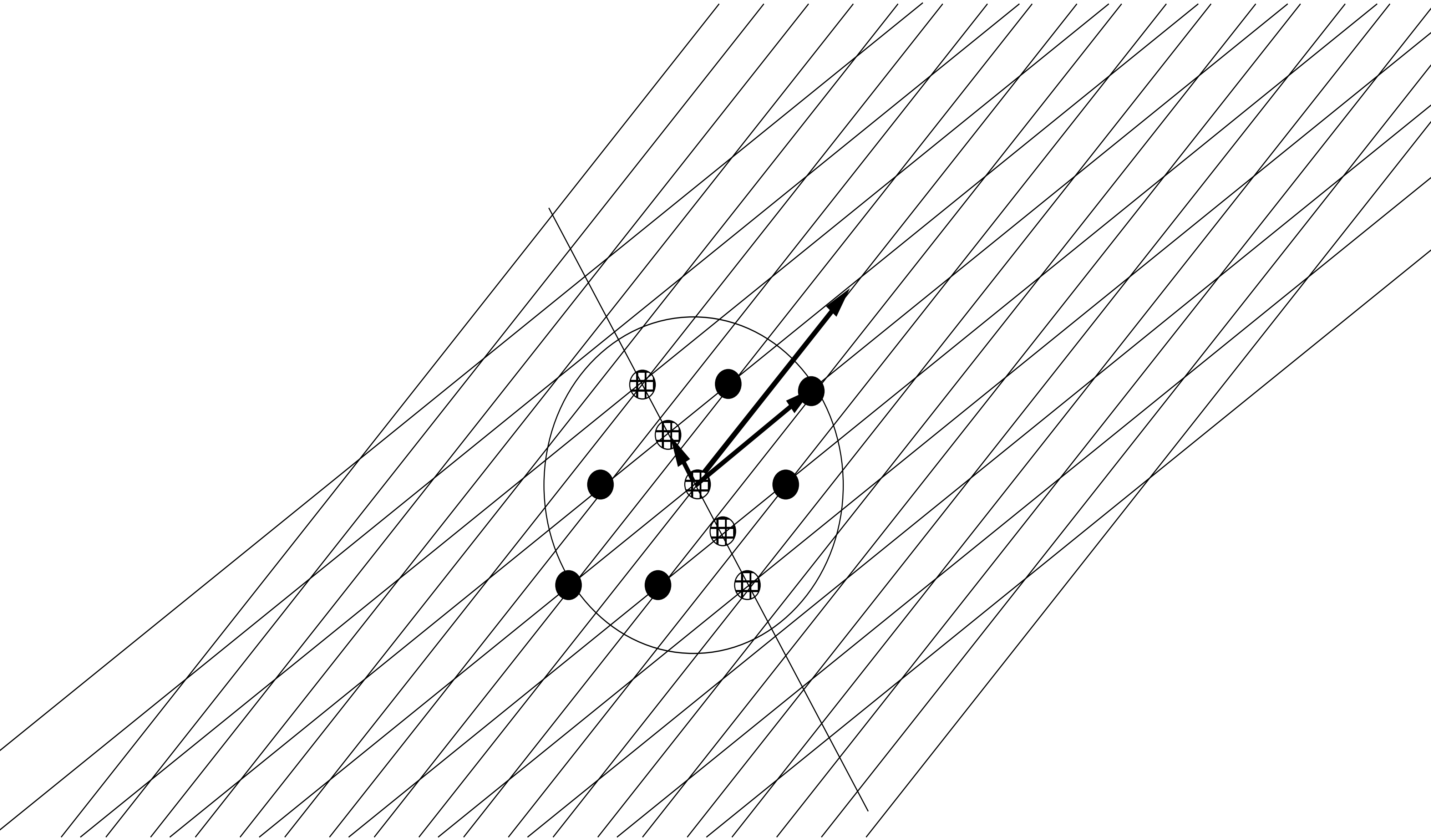
Le principe de Kannan : pré-processer



Le principe de Kannan : pré-processer



Le principe de Kannan : pré-processer



4) Nouveaux résultats

- Improved analysis of Kannan’s shortest lattice vector algorithm, Crypto 2007, avec G. Hanrot.
- Worst-case Hermite-Korkine-Zolotarev Reduced Lattice Bases, 2008, avec G. Hanrot.

Résumé des résultats

1. Meilleure borne supérieure du coût de l'algorithme de Kannan pour SVP:

$$d^{\frac{d}{2}} \longrightarrow d^{\frac{d}{2e}} \approx d^{0.182 \cdot d}.$$

2. Construction probabiliste de bases pour lesquelles l'algorithme va avoir ce temps d'exécution.

Résumé des résultats

1. Meilleure borne supérieure du coût de l'algorithme de Kannan pour SVP:

$$d^{\frac{d}{2}} \longrightarrow d^{\frac{d}{2e}} \approx d^{0.182 \cdot d}.$$

2. Construction probabiliste de bases pour lesquelles l'algorithme va avoir ce temps d'exécution.
3. Estimation efficace et a priori du coût d'une instance.

Résumé des résultats

1. Meilleure borne supérieure du coût de l'algorithme de Kannan pour SVP:

$$d^{\frac{d}{2}} \longrightarrow d^{\frac{d}{2e}} \approx d^{0.182 \cdot d}.$$

2. Construction probabiliste de bases pour lesquelles l'algorithme va avoir ce temps d'exécution.
3. Estimation efficace et a priori du coût d'une instance.
4. Meilleure borne supérieure du coût de l'algorithme de Kannan pour CVP:

$$d^{d} \longrightarrow d^{\frac{d}{2}}.$$

5. Des mauvaises bases pour la hiérarchie de Schnorr.

Le coût de l'algorithme de Kannan

- L'énumération domine.
- Le coût de l'étage $i+1$ est essentiellement le nombre de solutions entières de :

$$\sum_{j \geq i} (x_j + \sum_{k>j} \mu_{k,j} x_k)^2 \|\mathbf{b}_j^*\|^2 \leq \|\mathbf{b}_1\|^2$$

- Nombre de points entiers dans un ellipsoïde
$\Rightarrow$ volume de l'ellipsoïde :

$$\approx \frac{\|\mathbf{b}_1\|^{d-i}}{\sqrt{d-i}^{d-i} \prod_{j>i} \|\mathbf{b}_j^*\|}$$

Le coût de l'algorithme de Kannan

- On part d'une base quasiment HKZ-réduite, donc :

$$\|\mathbf{b}_1\| \lesssim \sqrt{d}\left(\prod_{j=1}^{d} \|\mathbf{b}_j^*\|\right)^{1/d}$$
$$\|\mathbf{b}_2^*\| \lesssim \sqrt{d}\left(\prod_{j=2}^{d} \|\mathbf{b}_j^*\|\right)^{1/(d-1)}$$
$$\|\mathbf{b}_i^*\| \lesssim \sqrt{d}\left(\prod_{j=i}^{d} \|\mathbf{b}_j^*\|\right)^{1/(d-i+1)}$$

- Cela implique que $\|\mathbf{b}_1\|^{d-i} \leq \sqrt{d}^{(d-i)(1+\log\frac{d}{d-i})} \prod_{j>i} \|\mathbf{b}_j^*\|$.
- La complexité est :

$$\lesssim \max_i \frac{\|\mathbf{b}_1\|^{d-i}}{\sqrt{d-i}^{d-i} \prod_{j>i} \|\mathbf{b}_j^*\|} \lesssim \max_i \sqrt{d}^{(d-i)\log\frac{d}{d-i}} \lesssim \sqrt{d}^{\frac{d}{e}}.$$

Et de manière rigoureuse...

- Le nombre de points entiers n'est pas toujours le volume, en particulier quand il y a de "gros" $\|\mathbf{b}_i^*\|$.
- Besoin de propriétés plus fines sur les bases HKZ-réduites.
- Analyse amortie.

Timings (en secondes, Pentium (R) 3.0 GHz)

pre-processing	$d = 40$	$d = 46$	$d = 52$	$d = 58$
LLL	1.8	110	$5.0 \cdot 10^3$	$-$
BKZ_{10}	0.36	6.7	160	$-$
BKZ_{20}	0.40	4.7	96	$2.5 \cdot 10^3$
BKZ_{30}	0.57	5.2	68	$1.6 \cdot 10^3$

Timings (en secondes, Pentium (R) 3.0 GHz)

pre-processing	$d = 40$	$d = 46$	$d = 52$	$d = 58$
LLL	1.8	110	$5.0 \cdot 10^3$	$-$
BKZ_{10}	0.36	6.7	160	$-$
BKZ_{20}	0.40	4.7	96	$2.5 \cdot 10^3$
BKZ_{30}	0.57	5.2	68	$1.6 \cdot 10^3$

- Record actuel : dimension 75 en $\approx$ 15 heures.

Timings (en secondes, Pentium (R) 3.0 GHz)

pre-processing	$d = 40$	$d = 46$	$d = 52$	$d = 58$
LLL	1.8	110	$5.0 \cdot 10^3$	$-$
BKZ_{10}	0.36	6.7	160	$-$
BKZ_{20}	0.40	4.7	96	$2.5 \cdot 10^3$
BKZ_{30}	0.57	5.2	68	$1.6 \cdot 10^3$

- Record actuel : dimension 75 en $\approx$ 15 heures.
- On peut s'attendre à ce que les instances apparaissant dans la hiérarchie de Schnorr soient plus faciles.

Timings (en secondes, Pentium (R) 3.0 GHz)

pre-processing	$d = 40$	$d = 46$	$d = 52$	$d = 58$
LLL	1.8	110	$5.0 \cdot 10^3$	$-$
BKZ_{10}	0.36	6.7	160	$-$
BKZ_{20}	0.40	4.7	96	$2.5 \cdot 10^3$
BKZ_{30}	0.57	5.2	68	$1.6 \cdot 10^3$

- Record actuel : dimension 75 en $\approx$ 15 heures.
- On peut s'attendre à ce que les instances apparaissant dans la hiérarchie de Schnorr soient plus faciles.
- C'est nettement au-delà des tailles de blocs utilisées dans toutes les attaques effectuées contre NTRU.

Borne inférieure : la génération d'Ajtai

- Une base est HKZ-réduite si pour tout $i < d$, $\mathbf{b}_i^*$ est un plus court vecteur de $L[\mathbf{b}_i^*, \ldots, \mathbf{b}_d^{\perp 1,\ldots,i-1}]$. Ou encore :

$$\forall i < j, x_j \neq 0 \implies \| \sum_{k \in [i,j]} x_k \mathbf{b}_k^{\perp 1,\ldots,i-1} \| \geq \|\mathbf{b}_i^*\|.$$

- On fixe $\|\mathbf{b}_i^*\| = f_d(i)$ for $i \leq d$ et on génère les $\mu_{i,j}$ uniformément et indépendamment dans $[-1/2, 1/2]$.
- Quelle est la probabilité que ces $\frac{d(d-1)}{2}$ conditions soient satisfaites simultanément?

Les conditions primaires

- Pour une base HKZ-réduite :

$$\|\mathbf{b}_i^*\| \leq \sqrt{d-i+1}\left(\prod_{j\geq i}\|\mathbf{b}_j^*\|\right)^{\frac{1}{d-i+1}}.$$

Les conditions primaires

- Pour une base HKZ-réduite :

$$\|\mathbf{b}_i^*\| \leq \sqrt{d-i+1}\left(\prod_{j\geq i}\|\mathbf{b}_j^*\|\right)^{\frac{1}{d-i+1}}.$$

- Ces conditions, avec l'égalité, définissent la fonction f. Pour obtenir une probabilité > 0, il suffit de les renforcer d'un facteur constant.

Les conditions primaires

- Pour une base HKZ-réduite :

$$\|\mathbf{b}_i^*\| \leq \sqrt{d-i+1} \left(\prod_{j \geq i} \|\mathbf{b}_j^*\| \right)^{\frac{1}{d-i+1}}.$$

- Ces conditions, avec l'égalité, définissent la fonction f. Pour obtenir une probabilité > 0, il suffit de les renforcer d'un facteur constant.
- On peut montrer que l'algorithme de Kannan fera au moins $d^{\frac{d}{2e}}$ opérations sur les bases générées, avec probabilité > 0.

Travail en cours, problèmes ouverts

Travail en cours, problèmes ouverts

- Utilisation rigoureuse de l'arithmétique flottante dans l'orthogonalisation de Gram-Schmidt.

Travail en cours, problèmes ouverts

- Utilisation rigoureuse de l'arithmétique flottante dans l'orthogonalisation de Gram-Schmidt.
- Une implantation efficace de l'énumération.

Travail en cours, problèmes ouverts

- Utilisation rigoureuse de l'arithmétique flottante dans l'orthogonalisation de Gram-Schmidt.
- Une implantation efficace de l'énumération.
- Simplifier et améliorer la hiérarchie de Schnorr.

Travail en cours, problèmes ouverts

- Utilisation rigoureuse de l'arithmétique flottante dans l'orthogonalisation de Gram-Schmidt.
- Une implantation efficace de l'énumération.
- Simplifier et améliorer la hiérarchie de Schnorr.
- De réelles attaques contre NTRU et SWIFFT, pour en mesurer la sécurité pratique.

Travail en cours, problèmes ouverts

- Utilisation rigoureuse de l'arithmétique flottante dans l'orthogonalisation de Gram-Schmidt.
- Une implantation efficace de l'énumération.
- Simplifier et améliorer la hiérarchie de Schnorr.
- De réelles attaques contre NTRU et SWIFFT, pour en mesurer la sécurité pratique.
- Comment exploiter le résultat de borne inférieure?