1 Introduction

On a souvent besoin de trouver rapidement un vecteur de "petite" norme dans un réseau. Par exemple, pour trouver un facteur d'un élément de $\mathbb{Z}[x]$ lorsqu'on en connait un dans $\mathbb{Z}/p^n\mathbb{Z}[x]$, mais aussi pour briser le cryptosystème de type sac à dos. La méthode décrite ci dessous ne donne pas forcémément un vecteur de norme minimale, mais elle donne en revanche rapidement un vecteur de petite norme. Elle fut publiée en 1982 par Lenstra-Lenstra-Lovasz pour factoriser des polynômes à coefficients entiers, mais elle fut appliquée ensuite dans bien d'autres situations.

2 Bases réduites d'un réseau

On dit que L est un réseau de $\mathbb{R}^n$ s'il existe une base $b_1, \dots, b_n$ de $\mathbb{R}^n$ telle que L soit un $\mathbb{Z}$-module libre de base $(b_1, \dots, b_n)$. On appellera déterminant de L le nombre[1]

$$d(L) = |\det(b_1, \dots, b_n)|$$

Pour toute base $(b_1, \dots, b_n)$ de $\mathbb{R}^n$, on notera $(b_1^*, \dots, b_n^*)$ la base obtenue à partir de (b) par orthogonalisation de Gram-Schmidt (pour le produit scalaire usuel sur $\mathbb{R}^n$). Autrement dit, les vecteurs b_i^* sont définis par les formules de récurrence :

$$b_1^* = b_1,\ \forall i > 1,\ b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*;\ \mu_{i,j} = \frac{(b_i, b_j^*)}{(b_j^*, b_j^*)}$$

où $(\ ,\ )$ désigne le produit scalaire choisi sur $\mathbb{R}^n$. On remarquera que b_i^* n'est rien d'autre que le projeté orthogonal de b_i sur l'orthogonal de l'espace vectoriel engendré par $b_1, \dots, b_{i-1}$.

Définition 1 *On dira d'une base $(b_1, \dots, b_n)$ d'un réseau L qu'elle est réduite, si les deux conditions suivantes sont réalisées :*

$$\forall i, j,\ 1 \leq j < i \leq n,\ |\mu_{i,j}| \leq \frac{1}{2} \tag{1}$$

$$\forall i,\ 1 < i \leq n,\ ||b_i^* + \mu_{i,i-1} b_{i-1}^*||^2 \geq \frac{3}{4} ||b_{i-1}^*||^2 \tag{2}$$

On remarquera que $b_i^* + \mu_{i,i-1} b_{i-1}^*$ et b_{i-1}^* sont les projections de b_i et b_{i-1} sur l'orthogonal de l'espace vectoriel engendré par $(b_1, \dots, b_{i-2})$. De plus, on a les deux propositons suivantes, qui "justifient" le fait qu'une base réduite donne un vecteur de "petite" norme du réseau :

Proposition 1 *Si $(b_1, \dots, b_n)$ est une base réduite d'un réseau L, alors on a :*

$$\forall (i, j) \in \mathbb{N}^2,\ 1 \leq j \leq i \leq n \Rightarrow ||b_j||^2 \leq 2^{i-1}.||b_i^*||^2 \tag{3}$$

$$d(L) \leq \prod_{i=1}^{n} ||b_i|| \leq 2^{\frac{n.(n-1)}{4}}.d(L) \tag{4}$$

$$||b_1|| \leq 2^{\frac{n-1}{4}} d(L)^{\frac{1}{n}} \tag{5}$$

En effet, si b est réduite, alors pour i, j tels que $1 \leq j \leq i \leq n$, on a $||b_j^*||^2 \leq 2^{i-j}||b_i^*||^2$, qui remplacé dans les formules d'orthogonalisation donne (**??**). On remarque ensuite que $d(L)$ est aussi égal à $|\det(b_1^*, \dots, b_n^*)|$ qui vaut $\prod_{i=1}^{n} ||b_i^*||$ ce qui permet d'obtenir (**??**) en utilisant[2] (**??**). On déduit (**??**) de (**??**) et (**??**).

A t'on utilisé le fait que b soit réduite pour l'inégalité $d(L) \leq \prod_{i=1}^{n} ||b_i||$?

[1] indépendant du choix de la base (b) de L

[2] comparer $||b_i||$ et $||b_i^*||$

Proposition 2 *Soit L un réseau de $\mathbb{R}^n$, et $(b_1, \ldots, b_n)$ une base réduite de L, alors :*

$$\forall x \in L, x \neq 0 \Rightarrow ||b_1||^2 \leq 2^{n-1}.||x||^2 \tag{6}$$

3 L'algorithme de réduction

Soit (b) une base d'un réseau L de $\mathbb{R}^n$.

• On initialise l'algorithme en calculant (b^*). On stockera alors pour la suite $(\mu_{i,j})$ et les nombres $||b_i^*||^2$, les vecteurs (b^*) ne seront plus utilisés.

Nous allons effectuer une suite d'étape (qui modifie un élément k de $\{1, \ldots, n+1\}$) commençant et se terminant de telle sorte que les propriétés suivantes soient vraies :

$$|\mu_{i,j}| \leq \frac{1}{2} \text{ pour } 1 \leq j < i < k \tag{7}$$

$$||b_i^* + \mu_{i,i-1} b_{i-1}^*||^2 \geq \frac{3}{4}||b_{i-1}^*||^2 \text{ pour } 1 < i < k \tag{8}$$

On constate que pour $k = 2$, il n'y a rien à vérifier pour satisfaire ces conditions. On peut donc commencer l'algorithme avec $k = 2$. Si les conditions précédentes sont satisfaites pour $k = n + 1$, alors la base est réduite, et l'on a terminé.

• On répète alors les étapes suivantes tant que $k \leq n$:

◇ On aimerait avoir $|\mu_{k,k-1}| \leq \frac{1}{2}$. Si cette inégalité n'est pas vérifiée, alors on remplace b_k par $b_k - r b_{k-1}$ où r est l'entier le plus proche de $\mu_{k,k-1}$. Ce changement impose alors les transformations suivantes pour retrouver l'orthogonalisé : pour $j < k-1$ on remplace $\mu_{k,j}$ par $\mu_{k,j} - r\mu_{k-1,j}$ et $\mu_{k,k-1}$ par $\mu_{k,k-1} - r$ les autres $\mu_{i,j}$ étant inchangés.

On distingue ensuite deux cas :

◇ Cas 1 : Si[3] l'on est dans la (mauvaise) situation suivante : $||b_k^* + \mu_{k,k-1} b_{k-1}^*||^2 < \frac{3}{4}||b_{k-1}^*||^2$

Nous allons alors faire diminuer $||b_{k-1}^*||$. Etudions ce qu'il se passe si l'on échange b_k et b_{k-1} sans toucher aux autres b_i. Notons c la base déduite de b où l'on a échangé b_k et b_{k-1}, c^* son orthonorgonalisée, et $\nu_{i,j} = \frac{(c_i, c_j^*)}{c_j^*, c_j^*}$. Puisque c_{k-1}^* est la projection de b_k sur l'orthogonal de $\sum_{j=1}^{k-2} \mathbb{R} b_j$, on a :

$$c_{k-1}^* = b_k^* + \mu_{k,k-1} b_{k-1}^*$$

ce qui donne $||c_{k-1}^*||^2 < \frac{3}{4}||b_{k-1}^*||^2$, confirmant le fait que l'on a bien réussi à faire diminuer $||b_{k-1}^*||$. On obtient ensuite c_k^* en projetant b_{k-1}^* sur l'orthogonal de c_{k-1}^*, ce qui donne :

$$\nu_{k,k-1} = \frac{(b_{k-1}^*, c_{k-1}^*)}{(c_{k-1}^*, c_{k-1}^*)} = \mu_{k,k-1} \frac{||b_{k-1}^*||^2}{||c_{k-1}^*||^2} \quad \text{et} \quad c_k^* + \nu_{k,k-1}.c_{k-1}^* = b_{k-1}^*$$

Pour $i \neq k-1$, on a $c_i^* = b_i^*$, et en exprimant pour $i > k$ les vecteurs b_i^* en fonction de (c^*) dans l'égalité $b_i = b_i^* + \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$, on trouve que les transformations à faire sont :

$$\nu_{k,k-1} = \mu_{k,k-1} \frac{||b_{k-1}^*||^2}{||c_{k-1}^*||^2}$$

$$\text{pour } k+1 \leq i \leq n \begin{cases} \nu_{i,k-1} &= \mu_{i,k-1}.\nu_{k,k-1} + \mu_{i,k}.\dfrac{||b_k^*||^2}{||c_{k-1}^*||^2} \\ \nu_{i,k} &= \mu_{i,k-1} - \mu_{i,k}.\mu_{k,k-1} \end{cases}$$

[3] On pourra trouver une condition équivalente ne dépendant que des normes des vecteurs de b^*. On remarquera que dans tout l'algorithme on peut se contenter de n'utiliser que les normes des vecteurs b^*, et que l'on n'a pas besoin des vecteurs b^*

$$\text{pour } 1 \leq j \leq k-2 \qquad \nu_{k-1,j} = \mu_{k,j} \text{ et } \nu_{k,j} = \mu_{k-1,j}$$
$$\nu_{i,j} = \mu_{i,j} \text{ pour } 1 \leq j < i \leq n \text{ et } \{i,j\} \cap \{k-1,k\} = \emptyset$$
$$c_{k-1} = b_k; c_k = b_{k-1} \text{ et } \forall i \notin \{k-1,k\},\ c_i = b_i$$
$$||c^*_{k-1}||^2 = ||b^*_k||^2 + (\mu_{k,k-1})^2||b^*_{k-1}||^2, \text{ et } ||c^*_k||^2 = \frac{||b^*_{k-1}||^2.||b^*_k||^2}{||c^*_{k-1}||^2}$$

Les vecteurs $(b_1, \ldots, b_{k-2})$ n'ayant pas été changés, les propriétés **??** et **??** restent vraies au rang précédent, on peut alors remplacer k par $\sup(2, k-1)$ et continuer l'algorithme.

$\diamond$ Cas 2 : Si $||b^*_k + \mu_{k,k-1}b^*_{k-1}||^2 \geq \frac{3}{4}||b^*_{k-1}||^2$

On aimerait maintenant avoir la propriété **??** : $|\mu_{k,j}| \leq \frac{1}{2}$ pour $1 \leq j \leq k-1$. Comme c'est vrai pour $j = k-1$, on obtient les autres inégalités en regardant pour l variant de $k-2$ à 1 si $|\mu_{k,l}| > \frac{1}{2}$. Et si cette condition est vérifiée, alors on considère l'entier r le plus proche de $\mu_{k,l}$, et l'on remplace b_k par $b_k - rb_l$. Pour $j < l$, les nombres $\mu_{k,j}$ sont alors remplacés par $\mu_{k,j} - r.\mu_{l,j}$ et $\mu_{k,l}$ par $\mu_{k,l} - r$. Les autres μ et b^* restent inchangés.

On a maintenant une bonne situation au niveau k et l'on peut remplacer k par $k+1$ et continuer l'algorithme.

4 Terminaison et coût

Il faut d'abord remarquer que l'algorithme précédent se termine. On considère pour celà les nombres[4] $d_i = \det((b_j, b_l)_{1 \leq j,l \leq i}) = \prod_{j=1}^{i} ||b^*_j||^2$. On pose $D = \prod_{i=1}^{n-1} d_i$. Le nombre D n'est modifié que si l'un des vecteurs de b^* est changé, ce qui n'arrive que dans le cas 1. Dans ce cas, d_{k-1} est réduit d'un facteur $\frac{3}{4}$, et les autres ne sont pas modifiés.

D'autre part le nombre D est minoré. On se limitera ici au cas où les b_i sont à coefficients entiers. Dans ce cas, si B est une constante telle que $\forall i ||b_i||^2 \leq B$ alors $d_i \leq B^i$, et

le nombre de fois où l'on passe par le cas 1 est alors borné (et de l'ordre de $O(n^2.\log B)$), donc celui où l'on passe par le cas 2 est aussi borné en $O(n^2.\log B)$, et l'algorithme se termine en $O(n^4.\log B)$ opérations arithmétiques sur le corps de base.

5 Le problème du sac à dos ; Cryptosystème de Merkle et Hellman

Soit $(a_1, \ldots, a_n, c)$ des entiers. Le problème du sac à dos est le suivant : Quelles sont les $(\epsilon_1, \ldots, \epsilon_n) \in \{0,1\}^n$ tels que $\sum_{i=1}^{n} \epsilon_i.a_i = c$. Autrement dit, quelles sont les différentes façons de remplir un sac à dos de hauteur c avec des batons de taille a_i ? Sous cette généralité ce problème est considéré comme très difficile, et l'on n'en connait pas de solution polynômiale en n. (On a en revanche une solution triviale exponentielle en n).

En revanche, dans le cas particulier d'une suite $(a_1, \ldots, a_n)$ surpercroissante (ie telle que $\forall i \in [2,n], a_i > \sum_{j=1}^{i-1} a_j$), alors le problème a une solution unique que l'on trouve rapidement.

L'idée de Merkle et Hellman était donc de prendre pour clef secrète une suite supercroissante (a_i), de choisir des entiers $M > \sum_{i=1}^{n} a_i$ et w tels que $M \wedge w = 1$, et de briser la supercroissance en posant : $b_i = w * a_i[M]$. (NB pour simplifier le raisonnement ici, on ne réordonne pas les (b_i) dans l'ordre croissant).

[4] On pourra justifier cette égalité

La clef publique est alors la donnée de (b_i), et pour crypter un message écrit en base 2 $\epsilon_1, \ldots, \epsilon_n$, on transmet grâce à la clef publique : $c = \sum_{i=1}^{n} \epsilon_i.b_i$. Résoudre ce problème semblait difficile sans connaitre $M, w, (a)$ puisque (b) n'est pas supercroissante, alors que la connaissance de $M, w, (a)$ permet de trouver facilement le message.

Shamir a cependant trouvé une faille au problème du sac à dos en (b) reposant sur LLL. Considérons la matrice publique $M = \begin{pmatrix} 0 & 1 & 0 & . \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & 1 \\ c & b_1 & \ldots & b_n \end{pmatrix}$. Alors le vecteur suivant caractérisant le message $\begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \\ 0 \end{pmatrix}$ est un vecteur du réseau engendré par les colonnes de M, de norme petite puisqu'elle est majorée par $\sqrt{n}$. On a donc de bonnes chances de le trouver par LLL.

6 Développements

On donne ici une liste indicative de développements et d'illustrations que vous pourrez utiliser ou non. Il s'agit d'un menu à la carte et non de passages obligatoires.

1) a) On pourra implémenter l'algorithme d'orthogonalisation, pour obtenir (b^*) et (μ).

b) On pourra éventuellement implémenter l'algorithme décrit ci dessus.

2) Expliquer pourquoi l'algorithme se termine, et le calcul du coût.

3) a) On pourra implémenter le cryptosystème de Merkle-Hellmann.

b) Expliquer et illustrer la faille du sac à dos. Si l'on n'a pas programmé l'algorithme, on pourra regarder si l'algorithme LLL n'est pas déjà implémenté dans votre logiciel favori[5]

4) Expliquer le lien entre les réseaux et le problème de trouver un diviseur d'un élément de $\mathbb{Z}[X]$ lorsque l'on en connait un diviseur dans $\mathbb{Z}/p^k\mathbb{Z}[X]$.

[5]par exemple maple7, xcas, ...(Attention maple7 n'a pas le package `IntegerRelations`). Pour xcas on utilisera la commande de pari pour l'instant plus fiable que la commande native lll (Un bug a été corrigé le 20/5/2007)