RÉDUCTION DES RÉSEAUX ET APPLICATIONS

GUILLAUME HANROT
PROJET SPACES, INRIA LORRAINE ET LORIA

Résumé. Le but de ce texte est de présenter l'algorithme LLL de réduction des réseaux, et de montrer divers domaines d'application de cet algorithme.

Table des matières

Date: 18 mars 2003.

1. Introduction

Après quelques brefs rappels sur les propriétés élémentaires des réseaux, principalement la définition et les notions de minima, défaut de longueur et d'orthogonalité, j'introduirai la problématique de la réduction des réseaux, qui consiste à construire des bases des réseaux qui soient de bonne qualité pour des calculs de type euclidien, c'est-à-dire le plus orthogonales possibles.

Lenstra, Lenstra et Lovàsz ont montré comment une de ces notions de réduction (due à Lovàsz) pouvait être réalisée en temps polynomial. On donnera une description de cet algorithme ainsi que ses généralisations et de ses versions flottantes dans la partie 6.

Enfin, une dernière partie essaiera de faire un tour d'horizon - – forcément sommaire – des très nombreuses applications de cet algorithme, dans tous les domaines dans lesquels la compétence de l'auteur s'exercent de façon non anecdotique. L'auteur serait reconnaissant aux lecteurs éventuels de lui signaler d'éventuels domaines d'application qui lui seraient inconnus.

Dans tout le texte, $\mathbb{R}^n$ est muni de sa structure euclidienne canonique et de la mesure de Lebesgue. On notera $(.|.)$ le produit scalaire canonique, i.e. $(u|v) = \sum_{i=1}^n u_i v_i$ et $vol(.)$ la mesure de Lebesgue.

Première partie 1. **Réseaux et algorithme LLL**

2. Réseaux

Le réseau L engendré par une famille libre $(e_1, \ldots, e_k)$ de $\mathbb{R}^n$ est l'ensemble des vecteurs $\{\sum_{i=1}^k \lambda_i e_i, \lambda_i \in \mathbb{Z}\}$. On dit que $(e_1, \ldots, e_k)$ est une *base* du réseau L, et que le réseau est de dimension k. La différence avec un espace vectoriel est donc le fait que les coefficients λ_i vivent dans $\mathbb{Z}$, et non dans $\mathbb{R}$.

De même que pour les espaces vectoriels, on peut se demander quel est l'ensemble des bases d'un réseau, et si la dimension est une grandeur intrinsèque, c'est-à-dire si elle ne dépend pas de la base choisie.

Proposition 2.1. *Si $(e_1, \ldots, e_k)$ et $(f_1, \ldots, f_j)$ sont deux familles libres qui engendrent le même réseau, alors $k = j$ et il existe une matrice M de dimension $k \times k$, à coefficients entiers, de déterminant ± 1 telle que $(e_i) = (f_i)M$.*

En particulier, un réseau a une infinité de bases. Toutes celles-ci sont loin d'apporter la même information sur la géométrie du réseau.

L'exemple suivant nous permettra d'introduire un premier exemple de réseau.

Posons $x = \sqrt{2} + \sqrt{3} \approx 3.1462643699419723423291350 66....$ La théorie nous prédit que x est racine d'un polynôme de degré 4. Considérons le réseau de $\mathbb{R}^5$ dont une base est

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ [Cx^4] & [Cx^3] & [Cx^2] & [Cx] & C \end{pmatrix}$$

avec C un grand entier. Qu'est-ce qu'un vecteur du réseau ? C'est un élément de $\mathbb{R}^5$ de la forme ${}^t(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_1[Cx^4] + \lambda_2[Cx^3] + \lambda_3[Cx^2] + \lambda_4[Cx] + C)$.

En particulier, la dernière composante est voisine de $C(\sum \lambda_i x^i)$. Si les λ_i sont les coefficients d'un polynôme annulateur de x, cette dernière composante est donc petite ; dans le cas contraire, comme C est très grand, la dernière composante est très grande. Par conséquent, si l'on sait trouver une base qui contient le vecteur non nul le plus court du réseau, on sait sans doute trouver un polynôme annulateur de x, alors même que la base donnée plus haut ne contient aucune information intéressante.

Noter que la “bonne” façon de calculer le polynôme minimal de x est par $\mathrm{Res}_Y(Y^2 - 2, (X-Y)^2 - 3)$, cf. le cours d'Éric Schost.

Remarque 2.1. *Un réseau peut être défini plus intrinsèquement comme un sous-groupe discret (i.e. l'intersection avec tout compact est finie) de $\mathbb{R}^n$. On montre que les deux définitions sont équivalentes.*

3. Invariants d'un réseau

Ce paragraphe, qui introduit des invariants importants des réseaux, peut être ignoré par le lecteur principalement intéressé par l'algorithme LLL et ses applications.

Définition 3.1. *Soit L un réseau de $\mathbb{R}^n$, $(e_1, \ldots, e_k)$ une base de L. On appelle discriminant de L, et on note $d(L)$ ou $\operatorname{disc}(L)$ la quantité $\sqrt{\det(e_i|e_j)}$ (rappelons que $(e_i|e_j)$ désigne le produit scalaire des deux vecteurs e_i et e_j).*

Ce discriminant est en fait le volume de $\{\sum \alpha_i e_i, \alpha_i \in [0,1]\}$, le *parallélogramme fondamental* du réseau. Si $n = k$, ce volume est exactement $|\det(e_i)|$.

3.1. **Vecteurs courts.** Soit L un réseau de dimension n. Pour k compris entre 1 et n, on définit le k-ème minimum de L comme étant $\min\{\rho \in \mathbb{R}; \dim \operatorname{Vect}(L \cap B(0,\rho)) \geq k\}$. En d'autres termes, le i-ème minimum est le plus petit réel ρ tel qu'il existe i vecteurs indépendants du réseau de norme au plus ρ.

En particulier, le premier minimum du réseau est la longueur d'un des vecteurs non nuls les plus courts. Le k-ème minimum du réseau est noté $\lambda_k(L)$.

Théorème 3.1 (Hermite)**.** *Soit $n \geq 1$. Il existe un réel γ_n tel que pour tout réseau de dimension n,*

$$\lambda_1(L) \leq \gamma_n (\det L)^{1/n}.$$

On peut prendre en particulier

$$\gamma_n \leq \frac{2}{\sqrt{\pi}} \Gamma(n/2+2)^{1/n} \sim \sqrt{\frac{2n}{\pi e}}.$$

Le nombre γ_n minimal est appelé *constante d'Hermite* en dimension n, et est étroitement relié au problème des empilements réguliers de sphères (il suffit de placer une sphère identique en chaque sommet d'un réseau atteignant la borne γ_n pour obtenir un empilement régulier optimal en dimension n). On connaît sa valeur exacte en dimension ≤ 8 :

n	1	2	3	4	5	6	7	8
γ_n^n	1	4/3	2	4	8	64/3	64	256

On peut en un certain sens étendre ce résultat aux minima d'ordre supérieur :

Théorème 3.2 (Minkowski)**.** *Pour tout réseau L de dimension n on a*

$$1 \leq \frac{\prod_{i=1}^{n} \lambda_i(L)}{d(L)} \leq \gamma_n^n.$$

Les vecteurs minimaux d'un réseau (ceux qui atteignent comme longueur les minima de ce même réseau) contiennent bien évidemment énormément d'information sur la structure de ce réseau. Il serait donc idéal de pouvoir les calculer, ne serait-ce même que le premier. Le résultat suivant montre que cet espoir est illusoire :

Théorème 3.3 (Ajtai (1997), Miccianco (1998))**.** *Le problème de trouver un vecteur v tel que $\|v\| = \lambda_1(L)$ est NP-difficile pour des réductions polynomiales probabilistes, et reste NP-difficile si on tolère un facteur d'approximation $< \sqrt{2}$.*

4. Bases réduites

L'exemple placé à la fin de la première partie illustre le fait que parmi toutes les bases d'un réseau, certaines nous sont plus utiles que d'autres.

On peut mettre en parallèle la situation euclidienne classique, où beaucoup de choses s'expriment de façon bien plus simple dans une base orthogonale (voire orthonormée) que dans une base quelconque (par exemple, la norme d'un vecteur s'évalue alors de façon très simple).

Les bases dotées de propriétés intéressantes seront appelées "bases réduites".

L'exemple présenté dans la première partie nous montre que des informations sur les vecteurs courts du réseau sont intéressantes ; en pratique, on verra des situations ou l'on souhaite que la base obtenue soit la plus orthogonale possible. Par chance, on verra que ces deux critères coïncident à peu près.

L'objet de cette partie est de présenter plusieurs notions de *base réduite*. Les deux parties suivantes montreront comment calculer certaines bases réduites en temps polynomial.

4.1. Défaut de longueur.

Pour mesurer la qualité d'une base réduite en terme de longueur de ses vecteurs, on compare la longueur de ceux-ci aux minima du réseau. Cela conduit à définir les défauts de longueur :

Définition 4.1. *Soit L un réseau de base (b_k) ; on suppose que les b_k sont ordonnés par norme croissante. Le i-ème défaut de longueur de L est défini par*

$$\mu_i(L) = \frac{\|b_i\|}{\lambda_i(L)} \geq 1.$$

Un des objets de la réduction des réseaux est donc d'obtenir des bases pour lesquelles les défauts de longueur ne sont pas trop grands.

4.2. Orthogonalisation de Gram-Schmidt et défaut d'orthogonalité.

On mesure l'orthogonalité d'une base d'un réseau en la comparant à la base orthogonalisée correspondante.

On commence par rappeler le procédé d'orthogonalisation de Gram-Schmidt dans $\mathbb{R}^n$. Ce procédé permet, à partir d'une base (b_i) de $\mathbb{R}^n$, de construire une base (b_i^*) vérifiant

– b_i^* est orthogonale ;
– $\mathrm{Vect}(b_1^*, \dots, b_k^*) = \mathrm{Vect}(b_1, \dots, b_k)$ pour tout k.

Cela revient à dire en pratique que $b_1^* = b_1$ et que b_k^* est le projeté orthogonal de b_k orthogonalement au sous-espace $\mathrm{Vect}(b_1, \dots, b_{k-1})$ de $\mathrm{Vect}(b_1, \dots, b_k)$.

On obtient les formules en cherchant b_k^* sous la forme $b_k + \sum_{i=0}^{k-1} m_{ik} b_i^*$, ce qui impose nécessairement $(b_k|b_i^*) + m_{ik}\|b_i^*\|^2 = 0$, d'où $m_{ik} = -(b_k|b_i^*)/\|b_i^*\|^2$.

Notons que voir b_i^* comme un projeté de b_i signifie en particulier que $\|b_i^*\| \leq \|b_i\|$ pour tout i. On définit alors :

Définition 4.2. *Le défaut d'orthogonalité d'une base $(b_1, \dots, b_n)$ est*

$$\rho(b) = \prod_{i=1}^{n} \frac{\|b_i\|}{\|b_i^*\|}.$$

Une base est orthogonale si et seulement si son défaut d'orthogonalité vaut 1 ; dans le cas contraire, son défaut d'orthogonalité est > 1. Noter que, bien entendu, un réseau n'admet que rarement une base orthogonale.

Les défauts de longueur et d'orthogonalité sont reliés par la formule suivante, conséquence immédiate du théorème de Minkowski.

Proposition 4.1. *On a*

$$1 \leq \frac{\rho(b)}{\prod_{i=1}^n \mu_i(b)} \leq \gamma_n^n.$$

En particulier, « base formée de vecteurs courts » et « base presque orthogonale » sont deux notions très voisines. Les diverses notions de réduction s'attachent donc principalement à réaliser une de ces deux conditions, l'autre se trouvant alors *ipso facto* vérifiée.

4.3. **Bases propres.** On ne peut pas appliquer le procédé de Gram-Schmidt à une base d'un réseau, car il implique des combinaisons linéaires à coefficients rationnels. On peut toutefois tenter de se rapprocher de l'orthogonalité en appliquant le procédé de Gram-Schmidt en arrondissant les coefficients à l'entier le plus proche. Cela conduit à la définition suivante :

Définition 4.3. *Une base (b_i) d'un réseau L est dite* propre *si les coefficients m_{ij} du procédé de Gram-Schmidt vérifient*

$$|m_{ij}| \leq \frac{1}{2}.$$

Une base quelconque peut aisément être rendue propre, en faisant :

Pour k de 2 à n faire
    Pour j de $k-1$ à 1 faire
        $b_k \leftarrow b_k - \lfloor (b_k|b_j^*)/\|b_j^*\|^2 \rceil b_j$

4.4. **Diverses notions de réduction.** Selon que l'on s'intéresse plutôt au défaut de longueur ou au défaut d'orthogonalité, on aboutit aux notions de base réduite au sens de Minkowski, et de base réduite au sens de Korkine-Zolotarev.

4.5. **Base réduite au sens de Minkowski.** On souhaiterait définir une telle base réduite comme une base (b_i) telle que $\|b_i\| = \lambda_i(L)$. Malheureusement, une telle base n'existe pas toujours (sauf en dimension 2 et 3). On se rabat donc sur la définition suivante :

Définition 4.4. *Une base $(b_i)_{1\leq i\leq n}$ d'un réseau L est dite* réduite au sens de Minkowski *si pour tout $1 \leq i \leq n$, b_i est un plus court vecteur tel que $(b_1, \ldots, b_{i-1}, b_i)$ soit une famille libre pouvant être complétée en une base du réseau L.*

4.6. **Base réduite au sens de Korkine-Zolotarev.**

Définition 4.5. *Une base $(b_i)_{1\leq i\leq n}$ d'un réseau L est dite* réduite au sens de Korkine-Zolotarev *(on trouve parfois* au sens de Hermite*) si*

– (b_i) *est propre ;*

– $\forall i$, b_i^* *est un élément de longueur minimale de* $proj_{<b_1,\ldots,b_{i-1}>^\perp} < b_1, \ldots, b_i >$

Noter en particulier que b_1 est dans ce cas un vecteur non nul le plus court du réseau.

Il n'existe hélas pas à l'heure actuelle d'algorithmes polynomiaux pour ces deux notions de réduction. Le résultat d'Ajtai montre qu'il est extrêmement improbable qu'un tel algorithme existe. On renvoie à [5] pour des algorithmes (non polynomiaux en la dimension) existants.

4.7. **Base réduite au sens de s-Siegel.**

Définition 4.6. *Soit s un paramètre réel positif inférieur à $\sqrt{3}/2$. Une base $(b_i)_{1\leq i\leq n}$ d'un réseau L est dite* réduite au sens de s-Siegel *si*

– (b_i) *est propre ;*

– $\forall i$, $\|b_{i+1}^*\|/\|b_i^*\| \geq s$.

4.8. **Base réduite au sens de t-Lovász.**

Définition 4.7. *Soit t un paramètre réel > 1. Une base $(b_i)_{1\leq i\leq n}$ d'un réseau L est dite* réduite au sens de t-Lovász *si*

– (b_i) *est propre ;*

– $\forall i$, $\|b_{i+1}^*\|^2 + m_{i+1,i}^2\|b_i^*\|^2 \geq \|b_i^*\|^2/t^2$.

La proposition suivante relie les deux dernières définitions :

Proposition 4.2. *Une base t-Lovász réduite est s-Siegel réduite, avec $s^2 = 1/t^2 - 1/4$.*

Démonstration. La condition de réduction se récrit

$$\frac{\|b_{i+1}^*\|}{\|b_i^*\|} \geq \sqrt{\frac{1}{t^2} - m_{i+1,i}^2},$$

et l'hypothèse de propreté, qui garantit $|m_{ij}| \leq 1/2$, permet de conclure. □

Les bases s-Siegel et t-Lovász réduites ont, par rapport aux bases de Minkowski ou Korkine-Zolotarev, l'avantage de pouvoir se calculer en temps polynomial, c'est ce que nous verrons dans la partie suivante. En revanche, leurs propriétés vis-à-vis des défauts de longueur ou d'orthogonalité sont moins bonnes.

On peut toutefois – c'est un des intérêts de ces bases – contrôler lesdits défauts encore assez précisément.

Théorème 4.1. *Soit (b_i) une base réduite au sens de s-Siegel d'un réseau de dimension n. Alors on a*

$$\mu_i(b) \leq s^{1-n}.$$

et

$$\rho(b) \leq s^{p(1-p)/2}.$$

On utilise le plus souvent ce résultat sous la forme particulière $\|b_1\| \leq s^{1-n}\lambda_1(L)$; dans la plupart des textes, la définition d'une base Lovász réduite correspond à $t = 2/\sqrt{3}$, soit $s = 1/\sqrt{2}$.

5. Algorithmes de réduction

Cette partie a pour objet de présenter l'algorithme LLL, dans une version naïve. On donnera quelques indications et pointeurs vers divers raffinements et idées d'implantation plus sophistiqués.

5.1. **Dimension 2 : l'algorithme de Gauss.** On va traiter dans cette partie le cas de la dimension 2, qui illustre un certain nombre d'idées du cas général, tout en restant relativement simple, puisqu'il s'agit d'une variation sur l'algorithme d'Euclide.

On part d'un réseau L disposant d'une base de 2 vecteurs, u_0 et v_0. Le principe est, à chaque étape, de réduire v_0 modulo u_0 comme dans l'algorithme d'Euclide, c'est-à-dire de translater v_0 parallèlement à u_0 jusqu'à ce que l'angle de droites (u_0, v_0) soit dans $[\pi/3, 2\pi/3]$. Cela revient en fait à rendre la base (u_0, v_0) propre, dans la terminologie de la partie précédente.

Une fois ceci effectué, on permute u_0 et v_0 (comme dans l'algorithme d'Euclide). On arrête quand on ne peut plus améliorer la base choisie.

Cela donne le pseudo-code suivant :

Données : $(u_0, v_0) \in \mathbb{Z}^2$, linéairement indépendants, avec $\|u_0\| \leq \|v_0\|$.
- $(u, v) \leftarrow (u_0, v_0)$.
- Tant que $\|u\| \leq \|v\|$
  - $q = \lfloor (u|v)/\|u\| \rceil$
  - $(u, v) \leftarrow (v - qu, u)$.
- Renvoyer $(u_1, v_1) = (v, u)$.

Théorème 5.1. *L'algorithme de Gauss termine, renvoie une base du réseau $L = (u_0, v_0)$ de la forme (u_1, v_1) avec $\|u_1\| \leq \|v_1\| \leq \|x\|$ pour tout $x \in L- < u_1 >$.*

On est donc en présence d'un algorithme calculant une base réduite au sens de Minkowski, en temps polynomial.

6. L'algorithme LLL

Nous abordons maintenant la description de l'algorithme LLL. Rappelons qu'une base t-Lovász réduite est propre, et vérifie la condition

$$\frac{\|b_{i+1}^*\|}{\|b_i^*\|} \geq \sqrt{\frac{1}{t^2} - m_{i+1,i}^2}.$$

6.1. **Description sommaire.** Le principe de l'algorithme LLL consiste à partir d'une base (e_i) et de la transformer en une base réduite incrémentalement. Si l'on suppose la réduction effectuée jusqu'à un certain i_0, alors on commence par modifier e_{i_0+1} pour rendre $e_1, \ldots, e_{i_0}, e_{i_0+1}$ propre. Une fois cela fait, on teste sur e_{i_0}, e_{i_0+1} la condition de Lovász. Si elle est réalisée, on incrémente i_0 ; sinon on permute e_{i_0} et e_{i_0+1} (penser à l'algorithme de Gauss) et on décrémente i_0.

6.2. **Terminaison.** Il est bien clair que ce processus, s'il termine, fournira une base propre et t-Lovász réduite. La terminaison n'en est toutefois pas évidente. Nous prouvons dès maintenant cette terminaison ; quelques indications sur les très nombreux détails d'implantation permettant de transformer la stratégie ci-dessus en véritable algorithme efficace sont données après la preuve.

Théorème 6.1. *L'algorithme LLL termine, en temps polynomial.*

Démonstration. Soit $M = \max_i \|b_i\|$ le plus grand des coefficients au début de l'algorithme. Notons $(b_1, \ldots, b_n)$ la base du réseau telle qu'elle évolue au cours de l'algorithme. On considère les sous-réseaux commençants $L_j =< b_1, \ldots, b_j >$, et on pose $d_j = \mathrm{disc} L_j$. On considère la quantité entière

$$D = \prod_{j=1}^{n-1} d_j^2 = \prod_{j=1}^{n-1} \prod_{k=1}^{j} \|b_k^*\|^2.$$

Il est bien clair que $D \leq M^{n(n-1)}$. Observons maintenant que si le test de t-Lovász est vérifié, la valeur de la quantité D ne change pas. Si le test n'est pas vérifié, regardons le résultat d'un échange de b_i et b_{i+1} : l'échange de b_i et de b_{i+1} se traduit sur la base de Gram-Schmidt correspondante par le remplacement de b_i^* par $b_{i+1}^* + m_{i+1,i} b_i^*$, dont la norme est $\leq \|b_i^*\|/t$, puisque précisément la condition de Lovász n'est pas vérifiée. Il s'ensuit que d_i^2 est remplacé par $d_i'^2 \leq d_i^2/t^2$ et que les autres d_j ne sont pas modifiés. La quantité D décroît donc d'un facteur t^2 au moins, et on peut donc faire au plus $n(n-1)(\log_t M)/2$ retours en arrière d'un indice. Le nombre de pas en avant est au plus égal à $n-1$ plus le nombre de pas en arrière, donc le nombre total d'étapes de l'algorithme LLL est au plus

$$n(n-1)(\log_t M) + (n-1).$$

Chacune de ces étapes est de complexité polynomiale (il faut dans la pratique être un peu soigneux pour contrôler la croissance des coefficients lors de l'algorithme), d'où le résultat. □

6.3. **Implémentation.** Tel que l'algorithme est présenté, l'étape limitante est le recalcul à chaque étape d'une base de Gram-Schmidt, dont le coût est essentiellement quadratique en la dimension, afin de tester la condition de Lovász. On peut éviter ce lourd calcul en remarquant (on l'a déjà fait dans la preuve) que les transformations effectuées dans LLL sur la base b_i se traduisent simplement sur la base de Gram-Schmidt ; cela implique qu'un seul calcul initial quadratique, suivi de mises à jour linéaires, est nécessaire. On renvoie le lecteur intéressé à l'ouvrage de Cohen [3] pour des formules.

Le paramètre t est classiquement choisi égal à $2/\sqrt{3}$. Plus t est petit, meilleure est – en principe – la qualité de la base réduite, mais plus le processus est lent. L'idéal, en théorie, serait de prendre $t = 1$; on ne sait alors plus garantir que l'algorithme termine en temps polynomial. Dans la pratique, la valeur de t semble avoir un peu, mais moins que prévu, d'incidence, tant sur la qualité de la base que sur le temps d'exécution. On peut même envisager de faire varier la valeur de t en cours d'exécution, en l'approchant de 1.

7. Variantes

On présente ici quelques-unes des principales variantes de l'algorithme LLL. Les deux premières sont basées sur la remarque que ce qui est réellement coûteux est le calcul de la matrice de Gram. Cette dernière a en effet des coefficients rationnels qui croissent, obligeant à un choix : faire des pgcd continuellement pour réduire la taille des numérateurs et dénominateurs, ou manipuler ces très grands entiers sans chercher à les réduire.

7.1. **LLL entier.** Cette version est due à de Weger, et consiste essentiellement à remarquer que l'on sait contrôler multiplicativement les dénominateurs intervenant lors de l'exécution de l'algorithme. Elle repose sur le lemme suivant :

Lemme 7.1. *Soit b_i une base d'un réseau L de dimension n. Posons*

$$d_i = \det(b_j|b_k)_{1\leq j,k\leq i} = \prod_{1\leq j\leq i} \|b_j^*\|^2.$$

Alors $d_{i-1}\|b_i^*\|^2 \in \mathbb{Z}$ pour tout i, $d_j\mu_{i,j} \in \mathbb{Z}$ et pour $j < m \leq i$,

$$d_j \sum_{1\leq k\leq j} \mu_{i,k}\mu_{m,k}\|b_k^*\|^2 \in \mathbb{Z}.$$

On peut alors facilement calculer les $\lambda_{ij} = d_j m_{ij}$ en construisant la suite $u_0 = (b_i|b_j)$, $u_k = (d_k u_{k-1} - \lambda_{ik}\lambda_{jk})/d_{k-1}$ pour $k < j$. Tous les u_k sont alors entiers, et on a $u_{j-1} = \lambda_{ij}$.

Noter qu'il faut aussi modifier les formules de mise à jour de la base de Gram-Schmidt pour tenir compte du fait qu'on stocke non plus les m_{ij} mais les $d_j m_{ij}$.

7.2. **Versions flottantes.** Dans cette partie, nous aborderons à peine un sujet qui a connu un grand essor ces dernières années. Une solution différente au problème du calcul de la matrice de Gram-Schmidt (m_{ij}) consiste à calculer cette matrice sous une forme approchée, par exemple en simple précision machine, dans la mesure où on n'a besoin en pratique que d'information sommaire sur ces nombres (partie entière, comparaisons).

Hélas, cette idée naïve est dans la pratique vouée à l'échec, pour une raison bien simple. Dans la base d'entrée – a priori tout sauf orthogonale – les coefficients de Gram-Schmidt sont très gros. À l'issue de la réduction, ils doivent être nettement plus petits ($< 1/2$). Les opérations de « proprification » consistent essentiellement au cours du calcul à remplacer ces nombres par leur distance à $\mathbb{Z}$, c'est-à-dire à effectuer une soustraction entre deux grands nombres réels très proches, opération notoirement – et évidemment – condamnable sur le plan de la stabilité numérique : ainsi en double précision machine (53 bits), si l'un des coefficients est $> 10^{16}$ (ce qui est courant dans les applications), l'étape de « proprification fg conduira à un résultat entièrement non significatif.

Les versions flottantes sont donc basées sur des idées de détection des instabilités numériques potentielles et de recalcul [9], ou de préconditionnement/reconditionnement de la base en cours d'algorithme pour garantir que les phénomènes ci-dessus ne se produiront pas [8].

7.3. **LLL par blocs.** Un travail récent de Koy et Schnorr essaie de réaliser la réduction LLL « par blocs », ce qui permet en principe, en appliquant récursivement le découpage par blocs, de gagner un ordre de grandeur en complexité. Les premières expériences réalisées semblent indiquer que cette variante récursive n'est pour le moment pas compétitive en pratique.

7.4. **Recherche d'un vecteur minimal.** Il arrive dans certaines circonstances que la connaissance d'un vecteur court ou d'un minorant pour le vecteur le plus court ne soit pas suffisante, et qu'on veuille effectivement déterminer le vecteur le plus court. Le théorème suivant est décourageant de ce point de vue.

Théorème 7.1. *Soit L un réseau de dimension n. Le problème de décider s'il existe dans L un vecteur dont la norme euclidienne (resp. L^1, resp. L^∞) est au plus égale à l est NP-difficile.*

Il existe toutefois un algorithme qui, bien qu'exponentiel, est assez efficace en petite dimension, dû à Fincke et Pohst. Partant d'une base (b_i) « bien conditionnée »(quasi-orthogonale, et en pratique LLL-réduite), il obtient des bornes sur les coefficients λ_i du vecteur le plus court dans la base b_i et énumère les diefférentes possibilités. On renvoie à [3] pour une description complète.

7.5. **Quelques heuristiques.** Avant d'aborder les applications, il est important de donner quelques idées heuristiques sur les propriétés d'une base LLL-réduite. Dans le cas où le réseau d'entrée est un réseau aléatoire, les vecteurs du réseau de sortie sont sensiblement tous de la même longueur ; comme de surcroît ils sont presque orthogonaux, leur norme est voisine de $\|b_i\|^*$; comme le produit des $\|b_i^*\|$ est le discriminant du réseau, on en déduit que la norme des vecteurs en sortie est de l'ordre de $\mathrm{disc}(L)^{1/n}$.

Une autre heuristique consiste à dire que dans la norme du plus court vecteur, chacun des termes est à peu près du même ordre de grandeur. Ainsi si $u = (u_1, \dots u_n)$ est un vecteur d'une base LLL-réduite, $\|u\| \approx \sqrt{n}|u_1|$.

Deuxième partie 2. **Applications**

8. Recherche de relations linéaires homogènes

On discute dans cette partie de l'application la plus directe de LLL, qui est la recherche de relation linéaires courtes (et généralement approchées) entre nombres réels ou entiers. C'est probablement aussi la plus importante, et la plus versatile. On la verra apparaître en filigrane dans de multiples autres applications.

8.1. **Principe général.** Considérons d'abord la recherche de relations linéaires homogènes. On suppose donnés n nombres réels $\alpha_1, \ldots, \alpha_n$. On cherche des entiers $\ell_1, \ldots, \ell_k$ tels que $\sum_{j=1}^n \ell_j \alpha_j$ soit « petit ». Il est alors naturel d'appliquer, comme dans la partie 1, l'algorithme LLL au réseau

$$M(C) = \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \ldots & 1 & 0 \\ \lfloor C\alpha_1 \rceil & \lfloor C\alpha_2 \rceil & \ldots & \lfloor C\alpha_{n-1} \rceil & \lfloor C\alpha_n \rceil \end{pmatrix},$$

où C est un réel positif.

S'il existe une relation linéaire avec de petits coefficients – le réseau n'est pas "aléatoire" – on espère trouver la relation linéaire comme un vecteur court du réseau.

Si aucune relation linéaire à petits coefficients n'existe, on s'attend à trouver, en vertu de l'heuristique, des vecteurs de l'ordre de grandeur de $C^{1/n}$ (α_n étant fixé, on néglige la dépendance en α_n). La limite entre les deux cas consiste donc à choisir C un peu plus grand que la taille de la relation linéaire cherchée à la puissance n.

Si enfin on cherche une relation linéaire $\sum \lambda_i x_i$ de l'ordre de grandeur de t, l'heuristique nous dit que le vecteur le plus court, qui vaut environ $(\lambda_1, \ldots, \lambda_{n-1}, C \sum \lambda_i x_i)$ a tous ses coefficients de taille analogue. Ce vecteur le plus court est de l'ordre de $C^{1/n}$, donc la combinaison linéaire trouvée a des vecteurs de l'ordre de grandeur de $C^{1/n}$ et est de taille $C^{1/n-1}$.

On peut montrer un résultat précis, et non heuristique : s'il n'existe pas de vecteur très court dans la base LLL-réduite, alors il n'existe pas de relation linéaire avec de petits coefficients : si θ_1 est la norme du premier vecteur d'une base s-Siegel réduite du réseau $M(C)$, correspondant à des entiers $(l_1, \ldots, l_n)$, on obtient :

Théorème 8.1. *Soit $(b_1, \ldots, b_n)$ des entiers vérifiant $|b_i| \leq B$, avec $B \leq \theta_1 \sqrt{2^{1-n}/(n-1)}$, on a*

$$\left|\sum b_i \alpha_i\right| \geq \frac{1}{C}\left(\sqrt{2^{1-n}\theta_1^2 - (n-1)B^2} - nB/2\right).$$

Démonstration. Le théorème 4.1 montre que

$$b_1^2 + \cdots + b_{n-1}^2 + \left(\sum b_i \lfloor C\alpha_i \rceil\right)^2 \geq s^{2(1-n)}\theta_1^2,$$

Soit

$$\left|\sum b_i \lfloor C\alpha_i \rceil\right| \geq \sqrt{s^{2(1-n)}\theta_1^2 - (n-1)B^2},$$

ou enfin

$$\left|\sum b_i \alpha_i\right| \geq \frac{1}{C}\left(\sqrt{s^{2(1-n)}\theta_1^2 - (n-1)B^2} - nB/2\right).$$

□

Noter que si l'on est amené à augmenter C, on peut réutiliser la réduction déjà effectuée : on aura obtenu une matrice de passage $U \in GL_n(\mathbb{Z})$ telle que $M(C_1)U$ est LLL-réduite. Mais cela signifie que la matrice $M(C_2)U$ est bien mieux conditionnée que $M(C_2)$ elle-même, et rend *a priori* le travail à effectuer sur $M(C_2)U$ beaucoup moins important.

Autre remarque technique qui peut être importante : si l'on sait a priori qu'un des b_i est de taille plus petite que les autres (e.g. $b_i/b_1 \approx D$), il est bon de remplacer le 1 de la ligne correspondante par un facteur D ; en vertu de l'heuristique donnée plus haut on peut s'attendre à ce que $D\ell_1$ et ℓ_i soient du même ordre de grandeur dans la base LLL-réduite.

8.2. **Reconstruction de nombres algébriques.** Une des applications classiques de cette technique est la reconstruction de nombres algébriques : si l'on se donne un nombre algébrique réel α calculé numériquement, dont le degré est au plus égal à n, on peut appliquer la technique ci-dessus sur les nombres α^n, α^{n-1}, …, α, 1 pour tenter de deviner le polynôme minimal (ou un de ses multiples) de α. Noter que α doit être connu avec une précision assez grande pour éviter que des vecteurs courts aberrants ne surgissent, et qu'il est également bon de connaître le degré n le plus précisément possible.

8.3. **Relations linéaires simultanées.** La technique ci-dessus se généralise *mutatis mutandis* à la recherche de relations linéaires simultanées entre k n-uplets de nombres réels $(\alpha_{i1}, \ldots, \alpha_{in})_{1\le i\le k}$. On considère dans ce cas la matrice

$$M(C) = \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & & \ddots & \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & 1 & 0 & \ldots & 0 \\ \lfloor C\alpha_{11}\rceil & \lfloor C\alpha_{12}\rceil & \ldots & \lfloor C\alpha_{1(n-k)}\rceil & \lfloor C\alpha_{1(n-k+1)}\rceil & \ldots & \lfloor C\alpha_{1n}\rceil \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \lfloor C\alpha_{k1}\rceil & \lfloor C\alpha_{k2}\rceil & \ldots & \lfloor C\alpha_{k(n-k)}\rceil & \lfloor C\alpha_{k(n-k+1)}\rceil & \ldots & \lfloor C\alpha_{kn}\rceil \end{pmatrix}.$$

On laisse au lecteur le soin d'établir un théorème analogue à celui donné ci-dessus (noter que le choix optimal de C doit aussi être repensé).

8.4. **Approximation diophantienne simultanée.** Parmi les applications de cette dernière technique, citons la recherche de meilleure approximation simultanée, i.e., étant donnés r nombres réels ρ_1, …, ρ_r, trouver un entier q et r entiers $p_1, \ldots, p_r$ tels que $|q\rho_i - p_i|$ soient simultanément petits. Il suffit en effet de prendre $n = r+1$, $k = r$, $\alpha_{j1} = \rho_j$, $\alpha_{j(j+1)} = -1$, $\alpha_{ij} = 0$ sinon. Un vecteur $(q, p_1, \ldots, p_n)$ du réseau est alors de norme $q^2 + \sum_{1\le i\le r} |q\lfloor C\rho_i\rceil - Cp_i|^2$, de l'ordre de $C^{1-1/(r+1)}$, avec q — et donc les p_i — et $C|q\rho_i - p_i|$ de ce même ordre, donc $|q\rho_i - p_i| \approx q^{-1/(r+1)}$, de l'ordre de grandeur prévu par le principe des tiroirs, ce qui montre que LLL généralise en un certain sens les algorithmes de fractions continues. En dimension 2, l'algorithme de Gauss appliqué au réseau correspondant coïncide d'ailleurs exactement avec l'algorithme décomposant un rationnel en fraction continue.

On peut aussi mentionner le cas des nombres algébriques complexes, pour lesquels on construit deux combinaisons linéaires simultanément petites en séparant partie réelle et partie imaginaire.

9. Relations linéaires inhomogènes

Le traitement des relations linéaires inhomogènes est un peu différent du paragraphe précédent. Il s'agit maintenant de trouver, non le vecteur le plus court ou même un vecteur court du réseau, mais le vecteur le plus proche, ou un vecteur proche.

L'idée pour y parvenir est simple : on cherche à trouver un vecteur x à coordonnées entières tel que $\|Mx - y\|$ soit le plus petit possible. Pour ce faire, on va calculer $M^{-1}y$, et arrondir à l'entier le plus proche les coordonnées de ce vecteur.

Si la base M du réseau L est quelconque, on obtiendra de cette façon un très mauvais résultat. En revanche, avec une base LLL-réduite, on peut obtenir un résultat intéressant :

Théorème 9.1. *Soit $\boldsymbol{x} = (x_i)$ un vecteur de $\mathbb{R}^n$, et $A = (b_1, \dots, b_n)$ une base LLL-réduite d'un réseau L. Posons $s = (s_i) = A^{-1}(x_i)$. Alors*

$$d(x, \Lambda) \geq 2^{(1-n)/2} d(s_{i^*}, \mathbb{Z}) \|b_1\|_2,$$

où i^* est le plus grand entier i tel que $d(s_i, \mathbb{Z}) \neq 0$.

Démonstration. On considère la base $(b_1^*, \dots, b_n^*)$, orthogonalisée de Gram-Schmidt de la base $(b_1, \dots, b_n)$. Les b_i sont donnés dans la base (b_i^*) par

$$b_i = b_i^* + \sum_{j=1}^{i-1} \mu_{ij} b_j^*,$$

où $\mu_{ij} = (b_i, b_j^*)/\|b_j^*\|_2^2$. On définira dans la suite $\mu_{ii} = 1$.

Soit $\boldsymbol{k} \in \mathbb{Z}^n$, et formons $\|Ak - x\|_2 = \|A(k - s)\|_2$:

$$\begin{aligned} A(k-s) &= \sum_{i=1}^{n} (k_i - s_i) b_i, \\ &= \sum_{i=1}^{n} \sum_{j=1}^{i} (k_i - s_i) \mu_{ij} b_j^*, \\ &= \sum_{j=1}^{n} \left(\sum_{i=j}^{n} \mu_{ij} (k_i - s_i) \right) b_j^*. \end{aligned}$$

Par suite,

$$\|A(k-s)\|_2^2 = \sum_{j=1}^{n} \left(\sum_{i=j}^{n} \mu_{ij} (k_i - s_i) \right)^2 \|b_j^*\|_2^2.$$

Posons $i_1 := \max\{i : k_i \neq s_i\}$. Il est clair que $i_1 \geq i^*$. Si $i_1 > i^*$, on a $|k_{i_1} - s_{i_1}| \geq 1 \geq d(s_{i^*}, \mathbb{Z})$, et sinon $|k_{i_1} - s_{i_1}| \geq d(s_{i^*}, \mathbb{Z})$. Dans tous les cas, on a donc

$$\|A(k-s)\|_2^2 \geq d(s_{i^*}, \mathbb{Z})^2 \|b_{i_1}^*\|_2^2,$$

puis on utilise le théorème 4.1 pour conclure. □

Ce résultat sert de substitut au théorème 4.1 dans le cas inhomogène ; on retrouve alors les mêmes applications que dans la section précédente, dans le cas de la recherche de relations linéaires inhomogènes. Noter que ce cas pourrait se traiter en "homogénéisant" la relation et en ne gardant que les relations où le terme correspondant est 1, mais le traitement présenté ici est plus direct et fournit presque toujours des résultats de meilleure qualité.

10. Factorisation des polynômes

On présente dans cette partie l'application de l'algorithme LLL à la factorisation des polynômes de $\mathbb{Z}[x]$ (les idées s'étendent aisément à diverses situations dans $\mathbb{A}[x]$, où $\mathbb{A}$ est e.g. l'anneau des entiers d'un corps de nombres ou $\mathbb{Z}[y]$).

Historiquement, l'algorithme LLL a été introduit par Lenstra, Lenstra et Lovász aux fins de prouver que la factorisation des polynômes à coefficients entiers pouvait se calculer en temps polynomial. Toutefois, l'algorithme proposé n'est pas utilisable en pratique ; il est dans la plupart des cas battu par la méthode dite de Cantor-Zassenhaus. En 2000, van Hoeij a proposé une variante de la méthode de Cantor-Zassenhaus, qui, bien qu'il paraisse difficile d'en prouver le caractère polynomial, améliore en pratique nettement la méthode de Cantor-Zassenhaus.

10.1. **Méthode de Cantor-Zassenhaus.** Rappelons tout d'abord comment procède la méthode dite de Cantor-Zassenhaus. On commence par choisir un premier p ne divisant pas le discriminant du polynôme $f \in \mathbb{Z}[x]$ à factoriser. On calcule (par exemple en utilisant l'algorithme de Berlekamp) la factorisation de f dans $\mathbb{F}_p$. On relève ensuite cette factorisation par une méthode de type Hensel dans $\mathbb{Z}/p^e\mathbb{Z}[x]$, où e est choisi de sorte que tout facteur de f dans $\mathbb{Z}[x]$ ait ses coefficients au plus égaux à p^e ; diverses bornes existent qui permettent de garantir cette propriété.

On a donc une écriture $f = \prod_{i=1}^s f_i$, où $f_i \in \mathbb{Z}/p^e\mathbb{Z}[x]$. Si l'on compare cette écriture à la factorisation de f sur $\mathbb{Z}$ de la forme $f = \prod_{j=1}^\ell g_j$, on voit que chaque g_j est produit de certains des f_i. On teste alors les recombinaisons possibles de façon exhaustive, c'est-à-dire que l'on calcule les $\prod_{i=1}^s f_i^{\varepsilon_i}$ dans $\mathbb{Z}/p^e\mathbb{Z}[x]$ et que l'on teste si le résultat divise bien f, auquel cas on a trouvé un facteur de f.

Il va de soi que ce qui précède n'est qu'une présentation sommaire de l'idée, et que diverses améliorations de détail permettent à l'algorithme de bien se comporter. Une analyse rapide de la complexité montre que la phase de factorisation et de relèvement est polynomiale (probabiliste), mais que la seconde phase est de complexité 2^s, où s est le nombre de facteurs p-adiques de f. On essaiera donc usuellement divers p, pour en trouver un modulo lequel f a peu de facteurs. Toutefois, on peut construire une famille d'exemples (polynômes de Swinnerton-Dyer) irréductibles, n'ayant que des facteurs quadratiques modulo tous les bons premiers p. Cela montre que la méthode de Cantor-Zassenhaus est exponentielle dans le cas le pire.

10.2. **Méthode de van Hoeij.** Tout récemment, un algorithme dû à van Hoeij améliore la phase de recombinaison en utilisant une méthode basée sur LLL. Nous allons tenter de présenter les grandes idées de sa méthode. Pour tous les détails techniques, le lecteur consultera son travail [12].

Notons $f = \prod_{i=1}^{s} f_i$ la factorisation de f dans $\mathbb{Q}_p$, et soit $v_1, v_2, \dots, v_k$ les vecteurs de $\{0,1\}^s$ tels que $\prod_{i=1}^{s} f_i^{v_{ji}}$ soient les facteurs de f dans $\mathbb{Z}[x]$.

Une des deux remarques importantes de van Hoeij est le fait que l'ensemble des vecteurs w de $\mathbb{Z}^s$ tels que $\prod_{i=1}^{s} f_i^{w_i}$ soit dans $\mathbb{Q}(x)$ est un sous-réseau L_f de $\mathbb{Z}^s$, engendré par les vecteurs $v_1, \dots, v_k$. L'algorithme de van Hoeij consiste alors à construire L_f itérativement, en le faisant décroître à chaque étape. On commence par $L_0 = \mathbb{Z}^s$, et on construit une suite de réseaux $L_i \subset L_{i-1}$. Une fois calculée une base de L_i, on tente de mettre cette base sous forme colonne échelon (pivot de Gauss) ; si dans cette forme on trouve exactement un 1 par ligne et des 0 en regard, on a un candidat pour la recombinaison, que l'on teste comme dans Cantor-Zassenhaus. Dans le cas contraire (ou si le candidat n'est pas valide), on continue à faire décroître L_i.

Reste à expliquer comment on procède pour calculer L_i. C'est à ce stade qu'intervient LLL. On va construire des formes linéaires qui sont petites sur L_f ; on a pour cela des candidats naturels, qui sont les sommes de Newton associées à f. Soit $S_i(\varphi) = \sum_{k=0}^{\deg \varphi} \alpha_k^i$ la i-ème somme de Newton d'un polynôme φ, où les α_k sont les racines de φ, qui s'évalue aisément en terme des coefficients du polynôme φ lui-même par les formules de Newton.

On construit alors $\tau_i(w) = \sum_{j=1}^{s} w_j S_i(f_j) \in \mathbb{Q}_p$ une forme linéaire sur L_0. On peut s'apercevoir que w est dans L_f tout bonnement en calculant $\tau_i(w)$ à des précisions p-adiques croissantes et en constatant que le résultat stationne, disons reste inférieur à p^f. Comme on ne peut pas faire opérer directement LLL sur les p-adiques, on va regarder ces sommes « modulo p^g ». Cette troncature n'est pas linéaire, et on aboutit donc à une inégalité vérifiée par toutes les recombinaisons de L_f :

$$|\sum_{j=1}^{s} w_j (S_i(f_j) \bmod p^g) + \delta p^g| \le p^f,$$

où $\delta < s/2$ et les « modulo » sont centrés. Cela amène à construire la matrice

$$M = \begin{pmatrix} C\mathrm{Id}_n & 0 \\ S & p^g \mathrm{Id}_s \end{pmatrix} L_t, \text{ où } S := \begin{pmatrix} S_{i_1} f_1 \bmod p^g \dots & S_{i_1} f_s \bmod p^g \\ \vdots \quad\quad \vdots & \vdots \\ S_{i_k} f_1 \bmod p^g \dots & S_{i_k} f_s \bmod p^g \end{pmatrix}$$

où $i_1, \dots, i_k$ sont des entiers que l'on peut faire varier au cours des différentes itérations de l'algorithme. Le résultat de LLL va permettre de faire décroître la dimension du réseau courant L_i, grâce au lemme suivant :

Lemme 10.1. *Soit (b_i) une base d'un réseau. Si $\|b_j^*\| > B$ pour $j \ge j_0$, alors si v est tel que $\|v\| \le B$, $v \in < b_1, \dots, b_{j_0-1} >$.*

Ce lemme n'est efficace, bien entendu, que si la base est bien choisie, i.e., si les premiers vecteurs – qui vont concentrer les solutions – sont courts, et les derniers longs. C'est en pratique l'allure d'une base LLL-réduite.

On peut donc, en appliquant ce lemme, supprimer une partie des vecteurs de la base LLL-réduite calculée, et ce faisant, obtenir un réseau L_{t+1} de dimension plus petite que L_t. En pratique, on remplacera lors de l'itération suivante les sommes $S_j(f_i)$ par $\sum c_{ki} S_j(f_i)$ où les (c_k) sont les vecteurs ayant survécu à l'application du lemme.

11. Réduction polynomiale

Soit P un polynôme **unitaire** irréductible sur $\mathbb{Q}$. On sait alors qu'une racine α de P définit une extension algébrique de $\mathbb{Q}$, de degré $\deg P$. Il arrive parfois que P soit très gros, ce qui rend les opérations de base dans $\mathbb{Q}(\alpha)$ coûteuses ; on peut alors se demander s'il existe P_2 ayant de petits coefficients engendrant la même extension. Pour trouver P_2, on peut utiliser le réseau :

$$\begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \dots & \alpha_1^{d-1} \\ 1 & \alpha_2 & \alpha_2^2 & \dots & \alpha_2^{d-1} \\ 1 & \alpha_3 & \alpha_3^2 & \dots & \alpha_3^{d-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha_d & \alpha_d^2 & \dots & \alpha_d^{d-1} \end{pmatrix}$$

Dans cette matrice, les α_i sont les différentes racines du polynôme P, et d est le degré de P.

Ce réseau est celui qui correspond au cas où le polynôme n'a que des racines réelles. Dans le cas où l'on a des racines imaginaires, il convient de ne conserver qu'une seule des deux racines conjuguées, et de séparer partie réelle et partie imaginaire. On obtient ainsi à nouveau un réseau de la bonne dimension.

Un vecteur court du réseau correspond alors à un élément de la forme $\sum b_i \alpha^i$ – qui est donc un entier algébrique – dont tous les conjugués sont simultanément petits. On peut en déduire que les fonctions symétriques élémentaires de ces conjugués sont également petites, et donc que le polynôme minimal de $\sum b_i \alpha^i$, s'il n'engendre pas une sous-extension de $\mathbb{Q}(\alpha)$, est à petits coefficients. Noter qu'on ne produit pas ainsi un polynôme dont tous les coefficients sont sensiblement de la même taille, mais un polynôme $\sum q_i X^i$ avec $q_i \approx M^{d-i}$, pour une certaine constante i.

En pratique, la solution ci-dessus marche mal, parce qu'elle ne permet pas, par exemple, de considérer des polynômes comme $P(X/2)$ qui peuvent dans certains cas être à coefficients entiers et parfaitement valables. Une meilleure façon de procéder consiste alors à utiliser non pas les α^i, mais les ω_j, où $\omega_1, \dots, \omega_d$ constitue une *base d'entiers* de $\mathbb{Q}(\alpha)$. Dans ce cas, et si on utilise ensuite l'algorithme de Fincke-Pohst, on trouve en un sens la meilleure matrice possible. C'est ce que fait en pratique la fonction `polredabs` de `pari`.

12. Recherche de petites racines modulo N

Les techniques exposées dans cette partie sont issues de travaux de Vallée, Girault, Toffin et de Coppersmith.

Supposons donnés un polynôme P et un module N, et que l'on recherche les x_0 entiers positifs petits, disons $|x_0| < M$ tels que $P(x_0) = 0 \bmod N$. Cette question est, mathématiquement, assez peu naturelle : la contrainte selon laquelle x_0 doit être petit est de nature très différente de la contrainte modulo N.

L'idée est donc de supprimer le "modulo N" de cette dernière contrainte : une racine entière est *a fortiori* racine modulo N. Pour être certain de ne pas rater de racine, il faut toutefois pouvoir assurer la réciproque. C'est vrai dès que P a des coefficients assez petits pour que $P(x_0) < N$ dès que $x_0 \leq M$, ce qui est à son tour vrai dès que par exemple $||P(Mx)||_2 < N/\sqrt{d}$: en effet,

$$P(x_0) = \sum a_i x_0^i \leq \sum |a_i M^i| \leq \sqrt{d}||P(xM)||_2 < N.$$

Malheureusement P lui-même ne vérifie que rarement cette inégalité, et si c'est le cas, le problème est trivial. On est donc amené à chercher un polynôme auxiliaire dont toutes les racines de P modulo N sont également racines, et qui a de petits coefficients. Cela amène à considérer un réseau de polynômes qui sont congrus modulo N à un multiple de P. On peut par exemple considérer le réseau engendré par les polynômes $N, \dots, NX^{d-1}, P(X)$; toutefois, réduire le réseau revient alors simplement à réduire les coefficients de P modulo N.

Notons que la contrainte sur la taille du polynôme produit est fortement liée à N ; on peut alors imaginer de remplacer N par des puissances supérieures de N et de construire des polynômes Q tels que les racines de P modulo N sont racines de Q modulo N^2, ou même N^m. Ainsi, on peut utiliser les polynômes de base $N^2, \dots, N^2X^{d-1}, NP(X), \dots, NX^{d-1}P(X), P^2(X)$ pour obtenir un réseau dans lequel tous les polynômes Q vérifient $Q(x_0) = 0 \bmod N^2$.

Plus généralement, on considère usuellement le réseau engendré par un sous-ensemble de la famille $g_{i,k} = N^{m-k}X^iP^k(X)$, où m est un entier fixé ; m et le sous-ensemble sont choisis selon les applications. Toute racine de P modulo N est alors racine des $g_{i,k}$ modulo N^m.

En pratique, souvent, on utilise non pas $g_{ik}(X)$ mais $g_{ik}(Mx)$. Cela vient du fait que l'on souhaite non pas trouver un polynôme dont les coefficients sont minimaux, mais dont la valeur sur $[-M, M]$ est minimale. En vertu de l'heuristique usuelle, le résultat de LLL appliqué sur des $g_{ik}(Mx)$ donnera un polynôme $\sum p_i(Mx)^i$ pour lequel chacun des termes sera sensiblement du même ordre de grandeur ; une application directe aurait donné $\sum p_i x^i$, avec les p_i du même ordre de grandeur, et donc la somme aurait été écrasée par le terme de degré dominant.

Noter que la méthode est limitée par le fait que l'on souhaite trouver un vecteur de petite norme, en vertu de la condition de taille sur $||P||_2$. Comme heuristiquement LLL fournit un vecteur de taille $(\mathrm{disc}L)^{1/\dim(L)}$, on obtient ainsi une limite – heuristique, mais souvent vérifiée en pratique – sur M et N pour lesquelles la méthode peut s'appliquer.

12.1. **Factorisation de nombres de la forme $p^r q$.** Il s'agit d'un travail de Boneh, Durfee et Howgrave-Graham. On suppose donné un entier N de la forme $p^r q$, et un entier π qui correspond aux bits les plus significatifs de p, ce qui signifie que $|\pi - p| < M$, avec M petit devant p. Il s'ensuit que si $P(X) = (\pi + X)^r$, on a $P(\pi - p) = 0 \bmod p^r$ avec $|\pi - p| < M$. On applique alors le procédé ci-dessus avec la famille $g_{ik}, k \in [0, m-1], i \in [0, r-1]$ et $g_{jm}, j = 0, \ldots, d - mr - 1$, où m et d sont à choisir. On trouve alors un polynôme h tel que $h(x) = 0 \bmod N^m$, et si ce polynôme est assez petit, on trouve la vraie racine p comme racine de $h(X)$ sur $\mathbb{Z}$. On vérifie le résultat en faisant le pgcd avec N.

Un calcul simple montre que le discriminant du réseau est dans ce cas de l'ordre de $N^{rm(m+1)/2} M^{d^2/2}$, et la dimension est d ; la norme du plus court vecteur est donc de l'ordre de $N^{rm(m+1)/(2d)} M^{d/2}$, et on veut qu'elle soit au plus de l'ordre de $p^{rm}/\sqrt{d}$. Posons $c = \log q / \log p$, de sorte que $N = p^{r+c}$. En comparant les deux termes et en cherchant m de façon à maximiser M, on trouve $m \approx d/(r+c) - 1/2$, et $X < p^{1 - c/(r+c) - 2r/d}$.

12.2. **Attaque de RSA avec petit exposant de déchiffrement.** Le célèbre cryptosystème RSA procède en choisissant un module N produit de deux nombres premiers p et q, et un exposant de chiffrement e premier avec $(p-1)(q-1)$, et chiffre alors en calculant $m^e \bmod N$; le déchiffrement se fait par le calcul de $x^d \bmod N$, où $ed = 1 \bmod (p-1)(q-1)$. Dans ce schéma, N et e sont publics, d, p, q sont secrets.

Pour des raisons de commodité, on choisit fréquemment e petit et de forme telle que l'élévation à la puissance e puisse s'effectuer de façon peu coûteuse. 3, 17, 257, 65537 sont des valeurs fréquemment rencontrées. Cela permet à l'émetteur du message d'avoir un coût de chiffrement assez réduit.

Remarquons que les techniques ci-dessus montrent qu'il est facile d'attaquer dans ce cas le système RSA dans le cas où le message chiffré est petit devant N ; si le message émis est m, il suffit d'appliquer les techniques ci-dessus au polynôme $P(X) = X^e - m \bmod N$. On évite ce problème en rajoutant des bits aléatoires au message à chiffrer pour le rendre de l'ordre de grandeur de N.

On peut imaginer, à l'inverse, de faciliter la tâche du destinataire du message en choisissant d petit. Toutefois, la méthode s'applique à nouveau, et ce indépendamment du message émis. C'est un résultat de Boneh et Durfee que nous décrivons maintenant.

En effet, $ed = 1 \bmod \varphi(N)/2$, soit, comme $\varphi(N) = N + 1 - p - q$, il existe k tel que $ed + k\left((N+1)/2 - (p+q)/2\right) = 1$. Si l'on pose $s = -(p+q)/2$ et $A = (N+1)/2$, il vient $k(A+s) = 1 \bmod e$. Dans cette équation, A et e sont connus, k et s sont inconnus. Par hypothèse, e est de l'ordre de grandeur de N.

On applique dans ce cas une version bivariée de la méthode générale décrite plus haut, qui cherche une petite solution $f(x_0, y_0) = 0 \bmod e$, avec $|x_0| \leq M, |y_0| < M'$. Les arguments donnés plus haut s'étendent sans difficulté ; on produit alors un réseau L formé sur un sous-ensemble des familles $g_{ik} = x^i f^k(x,y) e^{m-k}$ et $h_{jk} = y^j f^k(x,y) e^{m-k}$, m convenablement choisi.

On utilise alors non plus un, mais deux vecteurs courts produits par l'algorithme LLL, qui sont tels que $P_1(x_0, y_0) = 0$ et $P_2(x_0, y_0) = 0$, cette fois dans $\mathbb{Z}$. En prenant leur résultant –

en pratique non nul, même si on ne sait pas le garantir – on trouve un polynôme univarié dont x_0 (ou y_0, selon la variable éliminée) est racine.

Un choix optimal de la famille de polynômes choisit permet alors de casser RSA dès lors que $d \leq N^{0.292}$.

12.3. **Dilemme du fabricant de tables.** Le dilemme du fabricant de tables est un problème que l'on rencontre naturellement quand on se pose la question de l'évaluation des fonctions transcendantes. Des théorèmes généraux montrent pour de nombreuses fonctions usuelles que si x est un argument algébrique (et donc a fortiori rationnelles), $f(x)$ n'est pas algébrique (et donc encore moins rationnel). Il s'ensuit donc qu'il faut arrondir la valeur de $f(x)$.

Plus formellement, soit $x = m/2^p$ un nombre flottant en précision p, avec $m \in [2^{p-1}, 2^p[$. Supposons que l'on cherche à arrondir $f(x)$ en précision p'. Une idée naturelle consiste à calculer $f(x)$ en précision $p' + \delta$, avec δ petit.

On obtient alors $2^{p'} f(x) = A + m/2^\delta$, où A entier. Arrondir f est alors décider, selon les cas, si $m > 0$ ou si $m < 2^{\delta-1}$. En d'autres termes, si $f(x)$ est très voisin d'un nombre de la forme $A/2^{p'}$, on risque de devoir utiliser de très grandes valeurs de δ. L'entier δ mesure simplement le nombre de zéros (ou de 1) consécutifs situés juste après la virgule dans le développement binaire de $2^{p'} f(x)$ (dans le cas de l'arrondi au plus proche, c'est le nombre de 0/1 situés après le 1 initial).

Chercher les arguments x qui vont poser problème pour une valeur de δ fixé, c'est chercher les entiers m tels que $2^{p'} f(m/2^p) \bmod 1 < 2^{-\delta}$, ou $2^{p'+1} f(m/2^p) \bmod 1 < 2^{-\delta}$ dans le cas de l'arrondi au plus proche.

Si l'on suppose que f est un polynôme (on peut toujours se ramener à ce cas en coupant l'ensemble des valeurs de m à explorer en sous-intervalles et en remplaçant dans chaque intervalle f par une approximation polynomiale locale, par exemple son développement limité), et que $\delta > p$, on est ramené à une forme $\varphi(x) \bmod 1$ petit, soit $\varphi(x) - y = 0 \bmod 1$, ce qui modulo une renormalisation pour rendre y entier, rentre dans la situation précédente. Il s'agit d'un résultat de Stehlé, Lefèvre et Zimmermann[10].

13. Applications en cryptographie

13.1. **Le problème du sac-à-dos.** Le cryptosystème du sac-à-dos a été introduit par Merklé et Hellman dans les années 1980. Sa sécurité reposait heuristiquement sur la NP-complétude du problème du sac-à-dos,[1] défini de la façon suivante :

Données : $a_1, \ldots, a_n \in \mathbb{N}$, et S.
Résultat : existe-t-il $\varepsilon_i \in \{0,1\}^n$ tel que $\sum \varepsilon_i a_i = S$?

Pour construire un cryptosystème sur ce principe, il faut d'abord disposer d'une instance facile. On définit une suite supercroissante par $a_i > \sum_{j<i} a_j$. Dans ce cas, le problème du sac-à-dos admet une solution évidente en temps linéaire. On va déguiser cette instance en choisissant un $M > \sum a_i$ premier, un u inversible modulo M, et publier la clé $(M, b_1 =$

[1]La variante du sac-à-dos utilisée ici est souvent appelée *subset-sum problem*, et est elle-même NP-complète

$u^{-1}a_1, \ldots, b_n = u^{-1}a_n)$. Le chiffrement s'effectue alors via $(\varepsilon_i) \mapsto \sum \varepsilon_i b_i \bmod M = S$, et le déchiffrement en résolvant le problème du sac-à-dos pour la suite a_i et la valeur $Su \bmod M$.

Afin de présenter les attaques basées sur LLL, on va définir la densité d'un sac-à-dos :

Définition 13.1. *La densité du sac* $(a_1, \ldots, a_n)$ *est*

$$d = \frac{n}{\log_2 \max_i a_i}.$$

Le cas intéressant est le cas $d \leq 1$; en effet, pour $d > 1$, le problème du sac-à-dos a, a priori, des solutions multiples, et ne peut donc pas être utilisé pour construire un cryptosystème.

Selon le principe affiché plus haut, on est naturellement amené à considérer le réseau engendré par les colonnes de la matrice

$$M(C) = \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \ldots & 1 & 0 \\ Ca_1 & Ca_2 & \ldots & Ca_n & CS \end{pmatrix}.$$

On peut montrer que la donnée du vecteur le plus court de ce réseau fournit une solution au problème du sac-à-dos avec grande probabilité dès lors que la densité du sac est au plus 0.6463. Comme LLL fournit souvent de bons vecteurs courts – et que de surcroît on peut s'attendre à ce que le réseau ait un vecteur, correspondant à la solution, beaucoup plus court que les autres et donc trouvé automatiquement par LLL – on voit qu'on a de bonnes chances de trouver la solution.

Coster et al. [1] ont montré qu'on pouvait faire mieux en modifiant légèrement le réseau.

$$M'(C) = \begin{pmatrix} 1 & 0 & \ldots & 0 & 1/2 \\ 0 & 1 & \ldots & 0 & 1/2 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \ldots & 1 & 1/2 \\ Ca_1 & Ca_2 & \ldots & Ca_n & CS \end{pmatrix}.$$

On peut montrer dans ce cas qu'un vecteur le plus court du réseau fournit une solution avec haute probabilité pour les sacs de densité au plus 0.9408.

La modification du réseau vient de l'idée suivante : les cas d'échec correspondent à la situation où le vecteur le plus court du réseau ne provient pas de la solution du problème initial. Il faut donc construire un réseau dans lequel la solution au problème donne un vecteur le plus court possible par rapport au discriminant, avec l'espoir que cela fournira bien le vecteur le plus court.

Dans le premier réseau, si $\epsilon_i \in \{0, 1\}$ est une solution au problème initial, alors

$${}^t(\epsilon_1, \ldots, \epsilon_n, 0) = M(C)\ {}^t(\epsilon_1, \ldots, \epsilon_n, -1)$$

est de norme $\lambda = \sqrt{\#\{i; \epsilon_i = 1\}}$.

Dans le second cas,

$$^t(\epsilon_1 - 1/2, \ldots, \epsilon_n - 1/2, 0) = M'(C)\ ^t(\epsilon_1, \ldots, \epsilon_n, -1)$$

est dans le réseau, et de longueur $\sqrt{n/2}$. Le vecteur qui correspond à la solution est donc significativement plus court que dans le cas précédent, avec un discriminant qui est pourtant analogue ; on a donc plus de chances que le vecteur le plus court du réseau soit bien le vecteur provenant de la solution.

13.2. **Attaque du générateur linéaire congruentiel.** La cryptologie, et plus spécifiquement les protocoles cryptographiques, sont gros consommateurs de bits aléatoires. Pour assurer la sécurité, il faut dans ce cas que les bits aléatoires soient *imprévisibles*, c'est-à-dire qu'étant donnés un nombre arbitraire de bits produits par le générateur, il soit impossible, en temps raisonnable, de deviner le bit suivant avec une probabilité significativement supérieure à 1/2.

Un des premiers exemples de générateur aléatoire proposé (par Knuth) dans un cadre plus général est le générateur linéaire congruentiel, qui se cache encore derrière la fonction `rand` dans certains systèmes Unix. Ce générateur est défini par la donnée de (a, b, m, x_0) et la relation de récurrence $x_{n+1} = ax_n + b \bmod m$ (avec l'abus de notation usuel, qui désigne par $\alpha \bmod \beta$ le reste de la division euclidienne de α par β). Il a longtemps été utilisé en cryptographie sous une forme tronquée, c'est-à-dire en utilisant non pas x_n mais $y_n = \lfloor x_n/2^\lambda \rfloor$, pour un certain λ de l'ordre d'une proportion de la longueur de x.

Supposons que l'on connaît k valeurs consécutives de y_n. L'idée pour retrouver les paramètres, due à Stern [11], consiste tout d'abord à chercher une relation linéaire courte entre les $y_{i+1} - y_i$, au moyen des techniques pour les relations homogènes. Supposons que $Y = \sum \alpha_i(y_{i+1} - y_i)$ est petit. On forme alors $\sum \alpha_i(x_{i+1} - x_i)$, qui est proche de $2^\lambda Y$. On remarque maintenant que $(x_{i+j+1} - x_{i+j})_{0 \leq j < k}$ est dans le réseau engendré par les colonnes de

$$\begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ a & m & 0 & \ldots & 0 \\ a^2 & 0 & m & \ldots & 0 \\ \vdots & \vdots & & \ddots & 0 \\ a^{k-1} & 0 & 0 & \ldots & m \end{pmatrix},$$

dont le discriminant est m^{k-1}. Par suite, il est probable que le vecteur non nul le plus court soit de l'ordre de $m^{(k-1)/k}$, et donc si $2^\lambda Y$ est significativement plus petit que cette quantité, la combinaison linéaire $\sum \alpha_i(x_{i+1} - x_i)$ doit être nulle ; on montre que cela se produit effectivement avec une probabilité élevée. Cette identité se récrit $\sum \alpha_i a^i = 0$; en appliquant la méthode pour différentes valeurs initiales de i, on obtient donc une famille de polynômes annulateurs de $a \bmod m$, de degré $k - 1$.

L'ensemble des polynômes annulateurs de $a \bmod m$ de degré $\leq k - 1$ est en fait un réseau, engendré par les polynômes $X^i - a^i$, pour $i \in [1, k - 1]$ et le polynôme constant m. On vérifie facilement que son discriminant est m. Si lors de la première phase on calcule donc suffisamment de polynômes annulateurs, on peut espérer qu'ils engendrent le réseau tout entier, et donc en calculant le déterminant correspondant, trouver la valeur de m. On peut, là encore, montrer que cela se produit avec probabilité élevée ; trouver a est ensuite aisé.

14. Remerciements

Merci à Paul Zimmermann pour ses commentaires et sa relecture attentive sur des versions préliminaires de ce document.

Références

[1] M.J. Coster, A. Joux, B.A. LaMacchia, J. Stern, Improved low-density subset sum algorithms, Computational Complexity **2** (1992), 111-128.

[2] A. K. Lenstra, H. W. Lenstra, Jr., L. Lovász, Factoring polynomials with rational integers coefficients.

[3] H. Cohen, *A course in computational algebraic number theory*, Springer-Verlag Graduate Texts in Mathematics **138**, 1993.

[4] B.M.M. de Weger, Solving exponential diophantine equations using lattice basis reduction algorithms, *J. Number Th.* **26** (1987), 325–367.

[5] B. Helfrich, Algorithms to construct Minkowski-reduced and Hermite-reduced lattice bases, *Theoret. Comput. Sci.* **41** (1985), 125–139.

[6] A. Joux, J. Stern, Lattice reduction : a toolbox for the cryptanalyst. *J. Cryptology* **11** (1998), 161–185.

[7] H. Koy, C. P. Schonrr, Segment LLL-reduction of lattice basis, *Proceedings of CaLC 2001*, Springer-Verlag LNCS 2146, 67–80.

[8] H. Koy, C. P. Schnorr, Segment LLL-reduction of lattice basis with floating point orthogonalization, *Proceedings of CaLC 2001*, Springer-Verlag LNCS 2146, 81–96.

[9] C. P. Schnorr, M. Euchner. Lattice basis reudction and solving subset sum problems. *Math Programming Studies* **66A**, 1994, 181–199.

[10] D. Stehlé, V. Lefevre, P. Zimmermann, Worst cases and lattice reduction, rapport de recherche INRIA RR-4586 (2002).

[11] J. Stern, Secret linear congruential generators are not cryptographically secure, *Proc. IEEE Symposium on Foundations of Computer Science* (1987), 421–426.

[12] M. van Hoeij Factoring polynomials and the knapsack problem, *Journal of Number Theory* **95** (2002), 167–189.