INFORMATIQUE THÉORIQUE ET APPLICATIONS

BRIGITTE VALLÉE

La réduction des réseaux. Autour de l'algorithme de Lenstra, Lenstra, Lovász

Informatique théorique et applications, tome 23, n° 3 (1989), p. 345-376.

<http://www.numdam.org/item?id=ITA_1989__23_3_345_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 23, n° 3, 1989, p. 345 à 376)

LA RÉDUCTION DES RÉSEAUX. AUTOUR DE L'ALGORITHME DE LENSTRA, LENSTRA, LOVÁSZ (*)

par Brigitte VALLÉE (1)

Communiqué par J.-E. PIN

Résumé. – *Nous rappelons d'abord le cadre général de la Géométrie des Nombres, tant classique qu'algorithmique; nous définissons les principaux problèmes des réseaux et insistons sur leurs nombreuses applications.*

Nous définissons ensuite les différentes notions de réduction, et décrivons en particulier l'algorithme de Gauss qui résoud le problème parfaitement en dimension 2. Puis nous précisons la notion de réduction au sens de Lovász.

Nous décrivons alors l'algorithme dû à Lenstra, Lenstra et Lovász qui construit en temps polynomial une base réduite au sens de Lovász et nous analysons précisément sa complexité. Puis nous mentionnons d'autres algorithmes, qui en sont issus et qui permettent de calculer aisément dans les réseaux. Nous insistons enfin sur la simplicité de l'implémentation de tels algorithmes.

Nous continuons en présentant le champ d'application très étendu d'un tel algorithme, d'abord interne à la théorie des réseaux. Nous terminons en insistant sur des applications externes à la théorie et en décrivant des thèmes particuliers : théorie des nombres, algèbre ou cryptographie.

Cet exposé veut juste décrire l'ampleur des problèmes posés, aussi bien que la qualité des réponses qui y sont apportées. Pour plus de précisions, on pourra se reporter aux références bibliographiques finales.

Citons ici une référence générale, qui contient une bibliographie exhaustive sur le sujet et où la technique des problèmes est abordée de manière plus détaillée qu'ici : R. Kannan, Algorithmic Geometry of Numbers, *Annual Reviews in Computer Science,* 1988.

Abstract. – *We first recall the general framework of Geometry of Numbers, both classical and algorithmical; we define the main problems of lattices and we insist on their numerous applications.*

We then define different notions of reduction, and describe Gauss' algorithm that solves perfectly the problem in the two-dimensional case. After this, we precise the notion of Lovasz reduction. We then describe the Lenstra-Lenstra-Lovasz algorithm which builds in polynomial time a Lovasz reduced basis, and we analyse its complexity. Then, we mention other algorithms, stemming from it, which allow simple computations in lattices.

We continue with a description of the area of the applications of such an algorithm, starting with internal applications in lattice theory. We end with external applications in different areas: Theory of Numbers, Algebra or Cryptography.

Here, we just hope to describe the sheersize of the problems as well as the quality of the answers. A general reference, with an exhaustive bibliography, and a detailed technical treatment is: R. Kannan, Algorithmic Geometry of Numbers, *Annual Reviews in Computer Science,* 1988.

(*) Reçu novembre 1987, version finale en avril 1988.
(1) Département de Mathématiques, Université de Caen, 14032 Caen, France.

Informatique théorique et Applications/Theoretical Informatics and Applications
0296-1598 89/03 345 32/$5.20/

1. LA PROBLÉMATIQUE GÉNÉRALE DES RÉSEAUX

$\mathbf{R}^p$ est muni de sa structure euclidienne canonique, et on désigne par $|v|$ la norme de v et par $(u|v)$ le produit scalaire de u et v.

Par ailleurs $[r]$ désignera l'entier le plus proche du réel r.

Un réseau de $\mathbf{R}^p$ est l'ensemble des combinaisons linéaires à coefficients entiers de vecteurs linéairement indépendants de $\mathbf{R}^p$: c'est un sous-groupe discret de $\mathbf{R}^p$. Un tel réseau est dit *entier* s'il est inclus dans $\mathbf{Z}^p$.

Si $b=(b_1, b_2, \ldots, b_n)$ est un système de n vecteurs linéairement indépendants de $\mathbf{R}^p$, $(n \leqq p)$, le réseau engendré par b, noté $L(b)$, est l'ensemble $\left\{ \sum_{i=1}^{n} \lambda_i b_i \Big/ \lambda_i \in \mathbf{Z} \right\}$ et n est appelé rang ou dimension du réseau.

1.1. Le déterminant d'un réseau

Soient $(b_1, b_2, \ldots, b_n)$ et $(c_1, c_2, \ldots, c_n)$ deux bases du même réseau, et B et C leurs matrices $(n \times p)$ dans la base canonique de $\mathbf{R}^p$. Il existe alors une matrice unimodulaire U (matrice entière dont le déterminant vaut ± 1) vérifiant $C=UB$.

On remarque donc que le volume n-dimensionnel du parallélépipède construit sur une base quelconque du réseau est indépendant de la base : c'est un invariant du réseau, noté $d(L)$, et appelé le déterminant du réseau. Cette quantité est aisément calculable : si $G(b)$ désigne la matrice de Gram de la base b, c'est-à-dire la matrice $B^t B$ de coefficient général $(b_i|b_j)$, on a

$$\det G(b) = \det B^t B = d(L)^2.$$

On dira qu'un réseau est *pseudo-entier* s'il admet une base b dont la matrice de Gram $G(b)$ est à coefficients entiers — toutes ses bases vérifient alors la même propriété. Dans la pratique algorithmique, on se limite à des réseaux pseudo-entiers ou plus souvent encore à des réseaux entiers.

1.2. Les minima successifs des réseaux

Il existe, dans un réseau, d'autres objets qui ne dépendent que du réseau et non d'une base servant à le définir; en particulier, les minima successifs...

Pour un réseau L de $\mathbf{R}^p$ de dimension n, on définit le i-ème minimum $\Lambda_i(L)$ comme étant le plus petit réel positif t pour lequel il existe i vecteurs v linéairement indépendants de L vérifiant $|v|^2 \leqq t$.

Il est clair que les n nombres $\Lambda_i(L)$ sont bien définis et vérifient

$$\Lambda_1(L) \leqq \Lambda_2(L) \leqq \ldots \leqq \Lambda_n(L)$$

et qu'il existe un ensemble — non nécessairement unique — de n vecteurs linéairement indépendants de L, appelés aussi minima successifs, notés $\lambda_i(L)$ vérifiant

$$\text{pour tout } k, \qquad 1 \leqq k \leqq n, \qquad |\lambda_k(L)|^2 = \Lambda_k(L).$$

En particulier, $\lambda_1(L)$ désigne un plus court vecteur du réseau L.

1.3. Les problèmes théoriques et pratiques des réseaux

Les problèmes *théoriques* sont de deux types :

1. Chercher à relier les quantités intrinsèques d'un réseau, en particulier les $\Lambda_i(L)$ et $d(L)$.

2. Construire, exhiber, au moyen d'algorithmes, les objets intrinsèques d'un réseau, en particulier ses vecteurs minimaux successifs $\lambda_i(L)$.

Il arrive parfois que, en voulant résoudre le premier type de problèmes, on trouve une méthode explicite qui résolve du même coup le deuxième type de problèmes. Ce n'est jamais le cas pour cette théorie, et c'est ce qui explique la nécessité et l'importance de l'algorithmique en Géométrie des Nombres.

Dans la *pratique*, on se pose d'autres sortes de problèmes, qui sont essentiellement des problèmes de calcul dans les réseaux entiers; citons-en quelques-uns :

— Soit un réseau L donné par une base b et un vecteur v. Décider si v appartient au réseau $L(b)$.

— Soit un réseau L engendré par un système c de vecteurs non nécessairement indépendants. Trouver une base b du réseau.

— Soit $b=(b_1, b_2, \ldots, b_n)$ un système de n vecteurs de $\mathbf{Z}^p$. Trouver le réseau L des relations, c'est-à-dire l'ensemble

$$\left\{ v=(v_1, v_2, \ldots, v_n) \in \mathbf{Z}^n \Big/ \sum_{i=1}^{n} v_i b_i = 0 \right\}.$$

1.4. Les résultats théoriques non constructifs : Hermite, Minkowski [7]

Hermite a montré l'existence d'une constante γ_n, dite constante d'Hermite vérifiant

$$\gamma_n = \text{Max}\left\{ \frac{\Lambda_1(L)}{d(L)^{2/n}} \middle/ L \text{ réseau de rang } n \right\}.$$

Par la suite, Minkowski a montré qu'on a aussi :

$$d^2(L) \leqq \prod_{i=1}^{n} \Lambda_i(L) \leqq \gamma_n^n d^2(L).$$

Les preuves de ces résultats sont non constructives, et la première majoration connue de γ_n, due à Hermite, est exponentielle en n :

$$\gamma_n \leqq \left(\frac{4}{3}\right)^{(n-1)/2}.$$

Cette borne est la meilleure possible seulement dans le cas $n=2$. Ensuite Minkowski a affiné cette borne en obtenant les inégalités suivantes :

$$\gamma_n \leqq \frac{2}{3}n \quad \text{si} \quad n \text{ est pair} \qquad \text{et} \qquad \gamma_n \leqq \frac{1}{2}n \quad \text{si } n \text{ est impair.}$$

Seules les huit premières valeurs de cette constante sont exactement connues.

1.5. Les problèmes algorithmiques des réseaux

Un réseau entier L étant donné par son rang n et une base b de longueur $M = \text{Max}_i |b_i|$, on se pose les problèmes suivants :

1. *Déterminer $\lambda_1(L)$, un plus court vecteur du réseau L.*

2. *Déterminer les $\lambda_i(L)$, une suite de vecteurs minimaux successifs du réseau L.*

L'intérêt porté à ces problèmes est dû à la conjonction de trois facteurs :

— ce sont des problèmes *probablement difficiles,*

— qui admettent pourtant des solutions *approchées,*

— permettant la résolution d'autres problèmes, *variés et essentiels,* en théorie des nombres, en algèbre ou en cryptographie.

Nous allons décrire, tout au long de cet exposé, le second facteur d'intérêt, dans les sections 2 et 3 puis le troisième facteur, dans la section 4.

Attardons-nous un instant sur le premier facteur d'intérêt :

1.6. La difficulté probable de ces problèmes.

Le second problème est NP-dur en la donnée $(n, \log M)$ [22].

Rien n'est connu sur la « facilité » du premier : on ne connaît, à l'heure actuelle, aucun algorithme polynomial en $(n, \log M)$ qui résolve ce problème, à première vue plus facile que le second. L'opinion courante semble croire aussi en sa « dureté », en vertu de trois arguments principaux :

– ce problème est NP-dur pour la norme sup [27];

– le problème non homogène associé, qui consiste à chercher le point d'un réseau L le plus proche d'un point donné de $\mathbf{Q}^n$, est lui aussi NP-dur, même pour la norme euclidienne [27];

– les inégalités de Minkowski ne sont pas plus fines pour le premier minimum que pour la moyenne géométrique des autres minima successifs.

Deux points de vue différents mais complémentaires peuvent alors exister pour tourner cette difficulté presque sûre :

1. Chercher des algorithmes *approchés* polynomiaux en la donnée $(n, \log M)$.

2. Chercher, *à dimension n fixée*, des algorithmes *exacts* polynomiaux en la donnée $\log M$.

Ce ne sont pas d'ailleurs des points de vue divergents.

Le second point de vue est fructueux en petite dimension : l'algorithme de Gauss est un algorithme polynomial qui trouve les deux minima d'un réseau de dimension 2. Sa complexité est parfaitement connue, et il admet une généralisation totale en dimension 3 [25].

Le premier point de vue utilise alors en procédures des algorithmes de ce type – exacts en petite dimension – pour construire en dimension supérieure des algorithmes approchés. On obtient alors ce qu'on appelle une *base réduite* : c'est une base formée de vecteurs « *assez courts* » et « *assez orthogonaux* » qui permet de bien décrire le réseau et donc

1. de donner une bonne approximation des objets intrinsèques du réseau,

2. de pouvoir calculer facilement dans le réseau.

Nous verrons, dans la section 4, comment une telle base peut résoudre, de manière étonnamment satisfaisante, la totalité des problèmes algorithmiques, tant théoriques que pratiques.

Nous décrivons maintenant plus précisément les notions de réduction des réseaux.

2. LA RÉDUCTION DES RÉSEAUX

On cherche donc une base formée de vecteurs assez orthogonaux; remarquons qu'un réseau ne possède pas, en général, de base orthogonale. Le procédé d'orthogonalisation de Gram-Schmidt associe bien à une base b d'un réseau de $\mathbf{R}^p$ une base orthogonale b^* du $\mathbf{Q}$-espace vectoriel engendré par b mais cette dernière n'appartient pas, en général au réseau $L(b)$.

2.1. Le procédé d'orthogonalisation de Gram-Schmidt

Il associe à un système ordonné $b=(b_1, b_2, \ldots, b_n)$ le système $b^*=(b_1^*, b_2^*, \ldots, b_n^*)$ et la matrice $m=(m_{ij})$ qui exprime le système b dans le système b^* définis comme suit :

(i) $b_1^*=b_1$,

(ii) pour $i \geqq 2$, b_i^* est le projeté de b_i orthogonalement au sous-espace H_{i-1} engendré par les $i-1$ premiers vecteurs de b. On peut donc écrire

$$b_i = b_i^* + \sum_{j<i} m_{ij} b_j^*$$

où m_{ij} est défini par la relation $m_{ij}=(b_i \mid b_j^*)/|b_j^*|^2$.

Ce qui permet de calculer facilement les b_i^* et les m_{ij} par récurrence sur i.

La matrice m est donc une matrice triangulaire inférieure, possédant une diagonale de 1.

Remarquons que $d(L)$ est égal au produit des longueurs des vecteurs b_i^*.

Si b est un système de n vecteurs de $\mathbf{Z}^p$, de longueur M, le calcul du couple (b^*, m) est polynomial en la taille de la donnée $(n, \log M)$. Soient L_i le réseau engendré par les i premiers vecteurs de b, et $d_i=d(L_i)^2$ le déterminant de Gram associé.

L'inégalité d'Hadamard permet d'affirmer que

$$d_i \leqq \prod_{j=1}^{i} |b_j|^2 \leqq M^{2i}.$$

D'autre part, les rationnels intervenant dans b^* ou dans m ont comme dénominateurs les d_j; plus précisément :

$$|b_i^*|^2 = \frac{d_i}{d_{i-1}} \quad \text{pour tout } i, \qquad 2 \leqq i \leqq n$$

$$d_{i-1}\, b_i^* \in \mathbf{Z}^p \quad \text{pour tout } i, \qquad 2 \leqq i \leqq n$$

$$d_j m_{ij} \in \mathbf{Z} \quad \text{pour tout couple } (i, j), \qquad 1 \leqq j < i \leqq n.$$

Remarquons donc que la quatité

$$D = \prod_{j=1}^{n-1} d_j$$

représente un dénominateur commun à tous les rationnels intervenant dans le couple (b^*, m).

2.2. Les défauts de longueur et d'orthogonalité

Les deux conditions cherchées — vecteurs assez courts, vecteurs assez orthogonaux — sont heureusement compatibles en vertu des résultats de Hermite et Minkowski.

Soit $b = (b_1, b_2, \ldots, b_n)$ une base d'un réseau L; les deux paramètres suivants mesurent la qualité de la base :

Le rapport $\rho(b) = \left(\prod_{i=1}^{n} |b_i|^2 \right) \Big/ d(L)^2$ s'appelle le *défaut d'orthogonalité* de la base b.

Le rapport $\mu_i(b) = |b_i|^2 / \Lambda_i(L)$ s'appelle le i-ième *défaut de longueur* de la base b.

Les résultats de 1.4 permettent de montrer la liaison entre ces deux paramètres, qui vérifient la double inégalité :

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

2.3. Les différentes notions de réduction

Il n'existe pas *une* réduction; historiquement, il s'est dégagé principalement quatre notions de réduction.

Minkowski cherche à minimiser directement les défauts de longueur.

Dans les trois autres réductions, c'est $|b_i^*|$ qu'on va chercher à minimiser en même temps que $|b_i|$; ces réductions se décrivent donc aisément sur la matrice m, qui exprime le système b en fonction du système b^* : ce sont les réductions au sens de Korkine-Zolotarev, Siegel et enfin Lovász.

On peut montrer que la réduction au sens de Siegel est la plus générale [7] : toute base réduite en un des trois autres sens est réduite au sens de Siegel. Cette réduction, qui est aussi la moins fine, est suffisante dans beaucoup d'applications, car la base réduite obtenue est d'assez bonne qualité.

Les deux premières réductions,

— celle de Minkowski, qui est en un sens la plus naturelle,

— celle de Korkine-Zolotarev qui paraît être la meilleure tant pour le défaut d'orthogonalité que les défauts de longueur [13],

ne peuvent pas être obtenues — semble-t-il — en temps polynomial. Nous privilégierons alors les deux autres réductions, et nous montrerons qu'elles sont obtenues « facilement ».

Si ces réductions diffèrent quelque peu en dimension quelconque, elles coïncident toutes en dimension 2 avec la célèbre réduction de Gauss que nous décrivons maintenant, avant de préciser ces différentes notions de réduction.

2.4. La réduction de Gauss en dimension 2 [4]

Les minima successifs forment toujours dans un réseau un système de vecteurs indépendants, mais n'engendrent pas en général le réseau; c'est cependant vrai en petite dimension :

Les minima successifs d'un réseau de dimension $n \leqq 4$ forment une base du réseau, appelée base minimale du réseau : c'est alors la « meilleure base » du réseau.

En dimension 2, l'algorithme de Gauss construit en temps polynomial une base minimale du réseau; il généralise, en dimension 2, l'algorithme d'Euclide centré :

$$a = bq + r \qquad \text{avec} \quad -\frac{b}{2} < r \leqq +\frac{b}{2}.$$

Algorithme de Gauss

Donnée : une base (u, v) d'un réseau L.

Résultat : une base minimale (u, v) du réseau L.

Répéter

1. *Échanger* éventuellement u et v pour que $|u| \leqq |v|$.
2. *Translater* v parallèlement à u de manière à le raccourcir au maximum : plus précisément, choisir dans l'ensemble $\{ w = \varepsilon(v - mu) \mid \varepsilon = \pm 1, m \in \mathbf{Z} \}$, le vecteur w qui vérifie $0 \leqq (w \mid u)/(u \mid u) \leqq 1/2$.
   Ce dernier est aisément calculable en fonction de $r = (v \mid u)/(u \mid u)$; on choisit $m = [r]$ et $\varepsilon = sign\, e\,(r - m)$

jusqu'à ce que $|v| \geqq |u|$.

On peut modifier le test d'arrêt, en le changeant en un test moins fin :

Si t est un nombre réel vérifiant la double inégalité $1 < t \leqq \sqrt{3}$, on obtient ainsi un algorithme dit de t-Gauss qui « tourne » un peu moins longtemps, mais qui possède une configuration de sortie comparable : le triangle construit sur la base contient les deux minima du réseau.

Algorithme de t-*Gauss*

Donnée : une base (u, v) d'un réseau L.

Résultat : une base « quasi-minimale » (u, v) du réseau L.

Répéter

1. *Échanger* éventuellement u et v pour que $|u| \leqq |v|$
2. *Translater* v parallèlement à u

jusqu'à ce que $|v| \geqq 1/t|u|$.

2.5. Étude de la complexité de l'algorithme de Gauss

Soient $k(t)$ et k le nombre d'itérations des algorithmes de t-Gauss et de Gauss sur la même base (u, v) de départ de longueur M. Il est clair que l'on a $k(t) \leqq \log_t M + 1$.

On peut montrer aussi que, pour $1 \leqq t \leqq \sqrt{2}$, on a : $k(t) \leqq k \leqq k(t) + 1$, ce qui démontre la complexité polynomiale — non tout à fait triviale — de l'algorithme de Gauss.

Une étude plus fine du plus mauvais cas de l'algorithme de Gauss [25] permet d'exhiber la meilleure borne possible : on obtient

$$k \leqq \log_{1+\sqrt{2}} M + 3$$

qui est une borne similaire à celle obtenue dans l'algorithme d'Euclide centré [3].

2.6. L'effet de l'algorithme de Gauss sur l'orthogonalisée (u^*, v^*)

A la sortie de l'algorithme de Gauss, le vecteur v vérifie les deux conditions

(i) $|v| \geqq 1/t\,|u|$,

(ii) $0 \leqq (v|u) \leqq 1/2\,(u|u)$.

La projection de v orthogonalement à u, égale par définition à v^*, vérifie donc

$$|v^*|^2 \geqq \left(\frac{1}{t^2} - \frac{1}{4}\right)|u^*|^2.$$

Posant $s = \sqrt{4t^2/(4-t^2)}$, nous obtenons donc $|u^*| \leqq s\,|v^*|$.

Remarquons que si l'on a $1 \leqq t \leqq \sqrt{2}$, on a $4/3 \leqq s^2 \leqq 4$. La valeur $s^2 = 2$, correspondant à la valeur $t = 2/\sqrt{3}$ est usuellement choisie pour simplifier les calculs.

2.7. La propreté d'une base

L'idée la plus simple, pour rapprocher b de b^* est de diminuer les coefficients de la matrice m sans modifier ni b^*, ni le réseau $L(b)$; cela justifie la définition suivante qui reprend les notations de 2.1 :

Une base $b = (b_1, b_2, \ldots, b_n)$ est propre si la matrice m associée à tous ses coefficients m_{ij} pour $j < i$ inférieurs, en valeur absolue, à 1/2.

Explicitons géométriquement cette condition : chaque vecteur b_i se projette orthogonalement sur l'hyperplan H_{i-1} à l'intérieur du parallélépipède rectangle construit sur les b_j pour $j < i$ et défini comme étant l'ensemble des vecteurs

$$\sum_{j=1}^{i-1} \alpha_j b_j^* \qquad \text{pour} \quad -\frac{1}{2} < \alpha_j \leqq \frac{1}{2}.$$

Il existe un algorithme Totalement-Propre qui, étant donné un système $b = (b_1, b_2, \ldots, b_n)$, le rend propre; c'est une succession d'appels à Propre (i)

qui généralise la seconde étape de l'algorithme de Gauss; cette procédure translate b_i parallèlement à chaque vecteur b_j pour $j<i$ et ne modifie donc ni b_i^* ni $L(b)$.

Algorithme Propre (i)
  Pour j *allant de* $i-1$ *à* 1 *faire*
    $r_j := [m_{ij}]$;
    $b_i := b_i - r_j b_j$.

Algorithme Totalement-Propre
  Pour i *allant de* 2 *à* n *faire* Propre (i).

2.8. La réduction au sens de Siegel

Le fait qu'une base soit propre ne garantit pas le fait qu'elle soit presque orthogonale; on sait seulement pour le moment que la projection de chaque b_{i+1} sur H_i est assez petite. Pour pouvoir minorer l'angle θ_{i+1} que forme b_{i+1} avec le plan H_i, on a besoin de minorer en plus la projection de b_{i+1} orthogonalement à H_i, c'est-à-dire la longueur de b_{i+1}^*. C'est l'objet de la condition de réduction de Siegel :

$$|b_{i+1}^*| \geqq \frac{1}{s}|b_i^*| \quad \text{pour tout } i, \qquad 1 \leqq i \leqq n-1.$$

Une base propre qui vérifie de plus la condition de Siegel pour le paramètre s est dite s-réduite au sens de Siegel.

Cette condition est suffisante pour assurer la qualité de la base et pour majorer les défauts de longueur et d'orthogonalité. On obtient les résultats suivants :

$$|\sin(\theta_i)| \geqq \frac{1}{s^{i-1}}$$

et donc

$$|b_i| \leqq |b_i^*| s^{i-1} \quad \text{pour tout } i, \qquad 1 \leqq i \leqq n-1.$$

On en déduit

$$\rho(b) \leqq s^{n(n-1)/2}$$

et aussi

$$s^{-2(i-1)} \leqq \mu_i(b) \leqq s^{2(n-1)} \quad \text{pour tout } i, \qquad 1 \leqq i \leqq n.$$

2.9. La réduction au sens de Lovász

Il reste deux questions essentielles

(i) Tout réseau admet-il une base réduite au sens de Siegel? Si oui, pour quelles valeurs du paramètre s?

(ii) Si oui, existe-t-il un algorithme qui, partant d'une base quelconque b de longueur M d'un réseau L de rang n, construise une base de Siegel du réseau L en un temps polynomial en la donnée $(n, \log M)$?

Toutes ces questions vont recevoir des réponses positives. Puisque l'existence et la constructibilité d'une base propre ne posent pas de problèmes, la question se résume ainsi :

Comment assurer la condition de Siegel?

On va chercher à assurer une condition un peu plus forte : la condition de Lovász. Nous avons remarqué en 2.6 que l'algorithme de Gauss — en dimension 2 — permet d'obtenir une inégalité sur les orthogonalisés proche de la condition de Siegel; précisons :

Soit P_i l'orthogonal de H_{i-1} dans H_{i+1} et B_i le système (la « boîte ») formé par les projections u_i et v_i de b_i et de b_{i+1} sur P_i. Par définition de b^*_{i+1}, on a $b^*_{i+1} = v^*_i$. Si donc, nous appliquons l'algorithme de t-Gauss aux systèmes B_i, nous obtenons, à la sortie, les trois conditions suivantes valables pour tout i, $1 \leqq i \leqq n-1$ [les paramètres t et s sont liés, comme en 2.6, par la relation $s = \sqrt{4t^2/(4-t^2)}$] :

(i) $0 \leqq (v_i \mid u_i) \leqq 1/2\,(u_i \mid u_i)$;

(ii) $|v_i| \geqq 1/t\,|u_i|$;

(iii) $|u^*_i| \leqq s\,|v^*_i|$.

La première condition est une condition de propreté; la troisième est la condition de Siegel; la deuxième est appelée condition de Lovász.

D'après le paragraphe 2.6, on a : (i) + (ii) ⇒ (i) + (iii). Tout ceci justifie la définition suivante :

Une base propre qui vérifie la condition de Lovász pour le paramètre t est dite t-réduite au sens de Lovász.

et permet d'affirmer

Soient s et t deux paramètres liés par la relation $s = \sqrt{4t^2/(4-t^2)}$. *Une base t-réduite au sens de Lovász est s-réduite au sens de Siegel.*

3. L'ALGORITHME DE LENSTRA LENSTRA LOVASZ [15]

Cet algorithme construit, à partir d'une base b de longueur M d'un réseau L de rang n, une base t-réduite au sens de Lovász, en un temps polynomial en la taille de la donnée $(n, \log_t M)$. Pour $t=1$, cet algorithme se termine sans qu'on sache, à l'heure actuelle, préciser davantage sa complexité.

3.1. Les principales phases de l'algorithme

Cet algorithme se compose de trois phases principales :

— une phase d'*initialisation*; elle consiste essentiellement à calculer le système b^*, la matrice m et la liste l formée des éléments $l_i = |b_i^*|^2$, par la procédure d'orthogonalisation de Gram-Schmidt décrit en 2.1. Ces deux derniers objets — et eux seulement — vont d'ailleurs être essentiels tout au long de l'algorithme;

— des phases de *translation* des vecteurs b_i parallèlement à H_{i-1} qui s'effectuent par les procédures *Propre* décrites en 2.7. Rappelons aussi que ces phases ne modifient pas le système b^*;

— des phases d'*échange* des vecteurs b_i et b_{i+1} afin de réaliser :

(i) sur le système B_i la condition de t-Gauss,

(ii) et donc sur le système b^* la condition de s-Siegel.

Le triplet (b^*, m, l) est modifié lors de cet échange et nous devons en recalculer une partie, au moyen d'une procédure *Nouvortho* que nous décrirons plus loin.

Le choix — *translater ou échanger* — se fait en effectuant le test de l'algorithme de t-Gauss sur le système B_i défini en 2.9 et ce choix est répercuté ensuite sur les vecteurs $(b_1, b_2, \ldots, b_n)$ de la base b.

Remarquons d'abord que les vecteurs u_i et v_i du système B_i se lisent sur la matrice m : ce sont les vecteurs-ligne de la boîte B_i visualisée par un encadré ci-dessous.

$$
m = \begin{array}{c} \\ b_1 \\ b_2 \\ \vdots \\ b_i \\ b_{i+1} \\ \vdots \\ b_n \end{array}
\begin{array}{c}
\begin{array}{ccccccccc} b_1^* & b_2^* & \ldots & b_i^* & b_{i+1}^* & \ldots & b_n^* \end{array} \\
\begin{pmatrix}
1 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
m_{21} & 1 & \ldots & 0 & 0 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
m_{i1} & m_{i2} & \ldots & 1 & 0 & \ldots & 0 \\
m_{i+1,1} & m_{i+1,2} & \ldots & m_{i+1,i} & 1 & \ldots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
m_{n1} & m_{n2} & \ldots & m_{ni} & m_{n,i+1} & \ldots & 1
\end{pmatrix}
\end{array}
$$

En particulier, nous avons les relations suivantes :

$$u_i = b_i^*, \qquad v_i = b_{i+1}^* + m_{i+1,i} b_i^*$$

et donc les trois quantités intervenant dans l'algorithme de Gauss effectué sur B_i se calculent facilement en fonction de la liste l et de la matrice m :

$$|u_i|^2 = l_i, \qquad |v_i|^2 = l_{i+1} + m_{i+1,i}^2 l_i \quad \text{et aussi} \quad \frac{(v_i | u_i)}{(u_i | u_i)} = m_{i+1,i}.$$

3.2. La description générale de l'algorithme

Algorithme LLL (t)
Donnée : une base b d'un réseau L de rang n de $\mathbf{R}^p$.
Résultat : une base b de L t-réduite au sens de Lovász.
Gram;
$i := 1$;
Tant que $i < n$ *répéter*
1. *Translater* v_i parallèlement à u_i et donc b_{i+1} parallèlement à b_i: calculer $r_i = [m_{i+1,i}]$ et *faire* $v_i := v_i - r_i u_i$; $b_{i+1} := b_{i+1} - r_i b_i$;
2. *Tester si* $|v_i|^2 \geq 1/t^2 |u_i|^2$.
   Si *oui* la boîte B_i est réduite au sens de t-Gauss; *faire :*
   Translater alors b_{i+1} parallèlement aux b_j pour $j < i$ au moyen de *Propre* $(i+1)$.
   Modifier l'indice $i := i+1$.
   Sinon faire :
   Échanger b_i et b_{i+1}.
   Recalculer par la procédure *Nouvortho* le triplet (b^*, m, l).
   La boîte B_{i-1} n'est plus nécessairement réduite.
   Modifier éventuellement l'indice *si* $i \neq 1$ *alors* $i := i-1$.

La variable i désigne un indice, l'indice courant de l'algorithme qui va varier de 1 à $n-1$: c'est le plus grand indice k pour lequel le système $(b_1, b_2, \ldots, b_k)$ est t-réduit au sens de Lovász. La matrice m_i formée par les i premières lignes et les i premières colonnes de la matrice m et la liste formée par les i premiers termes de la liste l ont déjà les formes souhaitées.

On considère alors la $i+1$-ième ligne de m, représentant le vecteur b_{i+1} et on effectue les opérations de translation ou d'échange suivant le résultat du test de t-Gauss.

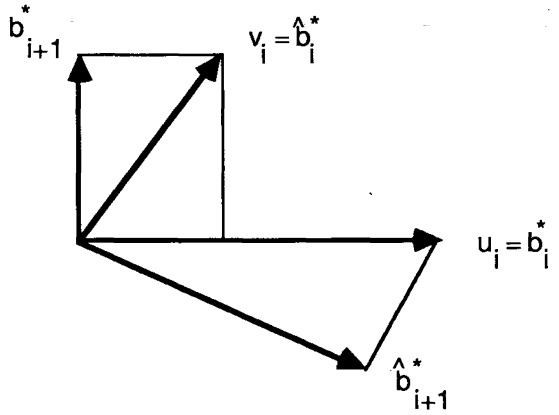
Il reste à préciser maintenant la procédure *Nouvortho.*

3.3. Les modifications de b^* de l et de m effectuées dans la seconde étape

Si le test est positif, ni b^* ni l ne sont modifiés; seule la $i+1$-ième ligne de la matrice m est remplacée par une combinaison linéaire de la $i+1$-ième et des lignes précédentes, conformément à la description de la procédure *Propre* $(i+1)$.

Si, par contre, le test est négatif, l'échange des vecteurs b_i et b_{i+1} dans le système b modifie les deux vecteurs b_i^* et b_{i+1}^*. Soit $(\hat{b}, \hat{b}^*, \hat{m})$ le nouveau triplet recalculé par la procédure *Nouvortho* : v_i est la projection de $\hat{b}_i = b_{i+1}$ sur P_i : on a donc $\hat{b}_i^* = v_i$; $\hat{b}_{i+1}^*$ est la projection de $\hat{b}_{i+1} = b_i$ orthogonalement à $\hat{H}_i$ et donc la projection de u_i sur l'orthogonal de v_i.

On a donc le schéma suivant :



On obtient aussi les égalités suivantes :

$$\hat{m}_{i+1,\,i} = m_{i+1,\,i}\,\frac{|u_i|^2}{|v_i|^2}$$

et aussi

$$\begin{pmatrix} \hat{b}_{i+1}^* \\ \hat{b}_i^* \end{pmatrix} = \begin{pmatrix} -\hat{m}_{i+1,\,i} & 1-\hat{m}_{i+1,\,i}\,m_{i+1,\,i} \\ 1 & m_{i+1,\,i} \end{pmatrix} \begin{pmatrix} b_{i+1}^* \\ b_i^* \end{pmatrix}$$

qui permettent de recalculer les colonnes i et $i+1$ de la matrice m et les éléments i et $i+1$ de la liste l selon les formules ci-dessous :

$$\hat{m}_{j,\,i} = m_{j,\,i}\hat{m}_{i+1,\,i} + m_{j,\,i+1}\frac{|b_{i+1}^*|^2}{|\hat{b}_i^*|^2}$$

$$\hat{m}_{j,\,i+1} = m_{j,\,i} - m_{j,\,i+1}m_{i+1,\,i}.$$

3.4. Le nombre de tours effectués par l'algorithme

L'indice i est un indice qui croît si le test en 2 est positif et qui décroît si ce test est négatif.

On va d'abord majorer en fonction de t, M, n le nombre de fois k_- où l'on passe en 2 avec un test négatif. Cette majoration suffira à conclure car le nombre de fois k_+ où l'on passe en 2 avec un test positif vérifie :

$$k_+ \leqq k_- + n - 1$$

Le nombre total k d'itérations de l'algorithme vérifiera donc $k \leqq n - 1 + 2k_-$.

Comme dans l'étude de la complexité de la procédure *Gram* étudiée en 2.1, c'est la quantité

$$D = \prod_{j=1}^{n-1} d_j$$

définie dans ce paragraphe 2.1 qui va jouer un rôle essentiel. Remarquons qu'à chaque passage en 2, avec un test négatif, avec une valeur de i donnée :

— les réseaux L_j pour $j<i$ et pour $j \geqq i+1$ ne sont pas modifiés; donc les d_j correspondants non plus;

— par contre le réseau L_i est modifié et son déterminant de Gram également.

On avait précédemment :

$$d_i = \prod_{j=1}^{i} |b_i^*|^2 = |u_i|^2 \prod_{j=1}^{i-1} |b_i^*|^2.$$

On a maintenant :

$$\hat{d}_i = \prod_{j=1}^{i} |\hat{b}_i^*|^2 = |v_i|^2 \prod_{j=1}^{i-1} |b_i^*|^2.$$

Le test de t-Gauss étant négatif, on en déduit que

$$\hat{d}_i < \frac{1}{t^2} d_i \quad \text{et donc} \quad \hat{D} < \frac{1}{t^2} D$$

D'autre part, lors de chaque passage en 2 avec un test positif, aucun des réseaux L_i n'est modifié. Donc D reste inchangé lors de cette étape. Finalement, D décroît tout au long de l'algorithme, et :

$$\text{au départ, on a : } D \leqq M^{n(n-1)},$$

$$\text{à l'arrivée, on a : } D \geqq 1.$$

On obtient donc la majoration : $k_- \leqq n(n-1)/2 \log_t M$.

3.5. La complexité de l'algorithme

Il reste à borner la taille des nombres apparaissant dans l'algorithme en fonction de la donnée $(n, \log M)$.

Il est clair que les nombres l_i sont des rationnels dont numérateur et dénominateur décroissent tout au long de l'algorithme.

Par contre, la situation est moins claire pour les entiers $|b_i|^2$ et la matrice rationnelle m : en effet, tout se passe bien en projection sur les plans P_i, mais, lors du relèvement, même choisi de manière minimale, on ne peut affirmer que la taille de ces quantités décroissent.

Un peu de technique, que nous ne développerons pas ici, permet de montrer que les majorations suivantes ont cours tout au long de l'algorithme — y compris dans les phases de relèvement — [voir [15] formules (1.30)-(1.34)].

On a toujours :

$$|m_{ij}| \leqq \sqrt{n}(2M)^{n-1} \quad \text{et aussi} \quad |b_i| \leqq n(2M)^n$$

et donc $|m_{ij}|$ est un rationnel dont la taille des numérateur et dénominateur est majorée par une quantité polynomiale en $(n, \log M)$. Notons d'ailleurs que ces bornes ont été améliorées par Schnorr [20].

Les opérations effectuées sur ces nombres sont très simples : calculs d'entier le plus proche, élévations au carré.

On déduit de tout cela le théorème suivant :

L'algorithme de Lenstra, Lenstra, Lovász, associé au paramètre t construit, à partir d'une base b de longueur M d'un réseau L de rang n, une base t-réduite au sens de Lovász en un temps polynomial en $(n, \log_t M)$ *: plus précisément, le nombre d'opérations arithmétiques effectuées dans cet algorithme est* $O(n^4 \log_t M)$ *et les entiers sur lesquels on opère sont de taille* $O(n \log M)$.

3.6. Des améliorations de l'algorithme

Schonhage [19] observe que, lors de l'algorithme original, on perd parfois du temps en aller et retour inutiles le long de la diagonale. Son idée principale est alors de procéder le plus souvent à des échanges locaux à l'intérieur de blocs (où l'indice i ne peut varier que dans un petit intervalle de longueur k), et de temps en temps seulement à des échanges globaux (pendant lesquels l'indice i peut varier de 1 à n). En choisissant $k = \sqrt{n}$, il obtient une amélioration de la complexité de l'algorithme, puisqu'il n'effecte plus que $O(n^3 \log M)$ opérations arithmétiques. L'idée de Schnorr [20] est d'arrondir les rationnels utilisés dans l'algorithme, tout en préservant l'exactitude des résultats. On utilise alors une plus forte réduction correspondant à la valeur $t = 1.05$ du paramètre et surtout une nouvelle méthode d'auto-correction dans le calcul approché de l'inverse d'une matrice. Il montre qu'on peut effectivement diminuer la taille des entiers utilisés jusqu'à $O(n + \log M)$.

On peut aussi combiner les deux méthodes...

3.7. Le cas particulier de la valeur $t = 1$ du paramètre t

On ne sait pas prouver la complexité polynomiale de l'algorithme **LLL**(1). On conjecture actuellement que le nombre d'itérations de cet algorithme est encore polynomial en la taille de la donnée $(n, \log M)$. Plusieurs arguments plaident en la faveur d'une telle conjecture :

— Lagarias et Odlyzko [12] ont effectué une étude expérimentale de cette conjecture : en pratique, le nombre d'itérations de **LLL**(1) ne dépasse pas trois fois le nombre d'itérations de **LLL**(t) pour $t^2 = 4/3$, valeur usuelle du paramètre.

— Nous avons montré en 2.5 combien le nombre d'itérations de l'algorithme de t-Gauss dépendait faussement du paramètre t.

3.8. La pratique de l'algorithme

Le succès de l'algorithme provient aussi de la simplicité de sa mise en œuvre : cet algorithme est plus simple à programmer qu'à comprendre, ce qui n'est pas si usuel pour un algorithme !

Toutes les opérations de l'algorithme s'effectuent sur le système b ou sur le triplet (b^*, m, l); il est d'ailleurs facile de se convaincre que b^* doit être seulement calculé lors de l'initialisation, et qu'il n'est plus nécessaire, ni de le conserver, ni de le mettre à jour ensuite. On peut donc seulement travailler sur les trois données b, m, l.

Nous montrons maintenant comment des adaptations simples de l'algorithme **LLL** permettent de résoudre de manière satisfaisante des problèmes pratiques de base :

- trouver une base d'un réseau;
- trouver des relations linéaires entières.

3.9. La recherche d'une base d'un réseau donné par un système de générateurs [5]

Soit $b=(b_1, b_2, \ldots, b_n)$ un système de générateurs d'un réseau L de rang s; on veut trouver une base du réseau, pour pouvoir calculer, par exemple, le déterminant du réseau. L'idée est de faire « comme si » les b_i étaient indépendants.

On peut en effet généraliser le procédé d'orthogonalisation de Gram-Schmidt à un système de vecteurs non indépendants : on obtient un couple (b^*, l) et une liste I formée des indices i pour lesquels b_i^* et donc l_i sont nuls. Par définition, le réseau L' est le réseau engendré par le système $\{b_i / i \notin I\}$ qui, par définition est un système de vecteurs indépendants : les deux réseaux L et L' engendrent bien sûr le même espace vectoriel de dimension s et L' est inclus dans L; on a donc $d(L) \leqq d(L')$.

L'idée est alors de faire décroître les indices $i \in I$, jusqu'à ce qu'ils soient tous au début; alors, les deux réseaux seront les mêmes et le système des vecteurs correspondant aux indices finaux sera le système cherché.

Pour cela, on utilise un algorithme **LLL** modifié, dont la structure générale est proche de celle de l'algorithme initial :

Algorithme Construction de base
Donnée : un système générateur $b=(b_1, b_2, \ldots, b_n)$ d'un réseau L de $\mathbf{R}^p$.
Résultat : une base b de L.
Gram;
$q:=0$.
Pour $i \in I$ *répéter*
$q:=q+1$.
Pour j allant de $i-1$ à q *faire*
1. *Tant que* $l_j \neq 0$ faire
Translater v_j parallèlement à u_j et donc b_{j+1} parallèlement à b_j.
Échanger b_j et b_{j+1}.
Recalculer par la procédure *Nouvortho* le triplet (b^*, m, l).
2. *Translater* alors b_{j+1} parallèlement aux b_k pour $k<j$ au moyen de *Propre* $(j+1)$.

Dans la procédure d'initialisation, on calcule les colonnes d'indice $i \notin I$ de la matrice m comme précédemment. Les colonnes d'indice $i \in I$ seront, par convention, égales à celles de la matrice identité correspondante.

Puis on procède là aussi par une succession d'échanges et de translations, mais uniquement sur les boîtes B_i associées à un indice i vérifiant $i+1 \in I$; pour ces indices-là, ces boîtes contiennent deux vecteurs u_i et v_i colinéaires : puisqu'elles sont « aplaties », l'algorithme de Gauss y coïncide avec l'algorithme d'Euclide centré, et la procédure Nouvortho est juste un échange entre u_i^* et v_i^*. L'étude de la complexité de cet algorithme est assez proche de l'algorithme classique. La quantité qui décroît ici tout au long de l'algorithme est la suivante :

$$D = \prod_{i \in I} 2^i \prod_{i \notin I} |b_i^*| = \prod_{i \in I} 2^i d(L').$$

Lors de chaque boucle interne, la partie I n'est pas modifiée, mais L', lui, est modifié par l'échange des vecteurs de la boîte et $d(L')$ est divisé par 2. Lors d'une boucle externe, L' n'est pas modifié, mais par contre, un indice $i \in I$ diminue d'une unité : c'est au tour de la première quantité d'être divisée par 2.

3.10. La recherche d'une relation linéaire courte entre n vecteurs de $\mathbf{Z}^p$

Soit $y=(y_1, y_2, \ldots, y_n)$ le système formé par ces vecteurs, Y la matrice dont les colonnes sont les y_i. Soit $x=(x_1, x_2, \ldots, x_p)$ le système formé par les lignes de la matrice Y et L le réseau de $\mathbf{Z}^n$ engendré par x qu'on suppose de rang q ($q \leqq p$). On veut construire un vecteur court de ce qu'on appelle *le réseau des relations* c'est-à-dire le réseau R des vecteurs $v=(v_1, v_2, \ldots, v_n)$ de $\mathbf{Z}^n$ vérifiant

$$\sum_{i=1}^{n} v_i y_i = 0 \quad \text{et donc} \quad (v \mid x_i) = 0 \quad \text{pour tout } i, \qquad 1 \leqq i \leqq p.$$

On procède de la manière suivante :

1. On construit une base $b=(b_1, b_2, \ldots, b_n)$ du réseau $\mathbf{Z}^n$ tel que les q premiers vecteurs de b engendrent le même $\mathbf{Q}$-sous-espace vectoriel H que x.

2. Les derniers $n-q$ vecteurs de la base $c=(c_1, c_2, \ldots, c_n)$ duale de la base b sont alors une base du réseau des relations.

3. Il reste alors à chercher un vecteur court de ce réseau.

Il est clair que **LLL** résoud l'étape 3. Il est vrai aussi qu'un algorithme assez semblable à celui du paragraphe précédent permet de résoudre la première étape :

Partant de la base canonique b de $\mathbf{Z}^n$, nous définissons

1. le système b^* formé par les vecteurs b_i^*, projections des vecteurs b_i orthogonalement aux sous-espaces $K_{i-1}=H+H_{i-1}$;

2. le couple (m, l) et la partie I correspondant dont le cardinal est q, dimension de H.

Travaillant alors sur le triplet (b^*, m, l), nous cherchons par une succession d'échanges et de translations à faire décroître les indices $i \in I$ jusqu'à ce que $I=\{1, 2, \ldots, q\}$: nous avons ainsi obtenu la base b cherchée.

Remarquons que la première phase de cet algorithme permet aussi de calculer une base normale d'Hermite d'un réseau entier, c'est-à-dire une base b vérifiant la propriété suivante : pour tout i, b_i est un vecteur de l'hyperplan engendré par les i premiers vecteurs de la base canonique.

Cette même première phase permet aussi de compléter un vecteur primitif en une matrice unimodulaire.

4. LE CHAMP D'APPLICATIONS DE L'ALGORITHME

Il s'agit de montrer ici comment l'algorithme **LLL** permet de résoudre de manière satisfaisante les problèmes internes à la théorie des réseaux mais aussi beaucoup d'autres problèmes externes.

Les applications internes, les trois premières, permettent de résoudre polynomialement, à dimension fixée, les problèmes difficiles de la théorie.

Les applications externes, au moins les trois premières d'entre elles, sont si essentielles en algorithmique qu'elles ont été un moteur puissant pour l'élaboration même de l'algorithme **LLL**.

Les dernières ont agi après coup en utilisant l'algorithme déjà existant.

Cet exposé ne prétend pas à l'exhaustivité sur ce sujet : on veut juste donner un aperçu de l'importance de l'utilisation de cet algorithme.

4.1. Le vecteur le plus court du réseau

Le premier vecteur b_1 de la base réduite obtenue est assez court, grâce à la majoration du premier défaut de longueur d'une base s-réduite au sens de Siegel. Pour les valeurs usuelles des paramètres s et t, on obtient :

$$|b_1|^2 \leqq 2^{n-1} \Lambda_1(L).$$

Nous verrons plus tard comment ce vecteur b_1 peut jouer, dans les applications, le même rôle que le vecteur $\lambda_1(L)$, même s'il est en général plus long que lui.

Ici, nous nous posons la question :

Comment, à partir d'une base réduite au sens de Siegel, trouver $\lambda_1(L)$*?*

Rappelons qu'à ce jour, aucun algorithme polynomial n'est connu pour résoudre ce problème. Il existe essentiellement trois algorithmes de complexité décroissante mais de complication croissante : les deux premiers sont dus à Lenstra, et le troisième à Kannan.

Le premier opère une simple mais longue recherche systématique. Exprimant $\lambda_1(L)$ dans la base b, sous la forme $\lambda_1(L) = \sum_{i=1}^{n} \beta_i b_i$, les formules de Cramer donnent :

$$\beta_i = \frac{\det(b_1, b_2, \ldots, b_{i-1}, \lambda_1(L), b_{i+1}, \ldots, b_n)}{\det(b_1, b_2, \ldots, b_n)}.$$

En utilisant l'inégalité d'Hadamard, et la définition de $\lambda_1(L)$, on obtient : $|\beta_i| \leqq \rho(b)$.
Puisque b est de Siegel, le défaut d'orthogonalité $\rho(b)$ est majoré et on obtient :

$$\rho(b) \leqq 2^{n(n-1)/4}$$

on en déduit :

$$|\beta_i| \leqq 2^{n(n-1)/4} \quad \text{pour tout } i, \qquad 1 \leqq i \leqq n.$$

On en déduit donc un algorithme qui doit calculer la longueur de 2^{n^3} vecteurs du réseau.

Le second procède de manière récursive en utilisant un argument géométrique assez simple : la longueur $|b_n^*|$ mesure la distance entre deux hyperplans consécutifs du réseau parallèles à H_{n-1}. Or, puisque b est s-réduite au sens de Siegel, ces hyperplans sont assez « espacés » et on a, d'après le paragraphe 2.8, pour les valeurs usuelles de s et t :

$$|b_n^*|^2 \geqq \frac{1}{2^{n-1}} |b_n|^2 \quad \text{et donc} \quad |b_n^*|^2 \geqq \frac{1}{2^{n-1}} |\Lambda_1(L)|.$$

Par conséquent, $\lambda_1(L)$ ne peut se trouver que dans un petit nombre d'hyperplans de direction parallèle à H_{n-1} (ce « petit » nombre est de l'ordre de $2^{(n+1)/2}$) : on projette successivement dans ce nombre fini d'hyperplans affines, et dans chacun d'eux on peut utiliser le même genre d'arguments car l'inégalité précédente est vraie quand on remplace n par $n-1$ et $\lambda_1(L)$ par ses projetés dans ces hyperplans.

On obtient ainsi un algorithme qui considère $2^{n(n+1)/4}$ vecteurs du réseau. Du fait que cet algorithme est affine et non pas vectoriel comme les deux autres, nous y reviendrons dans le paragraphe suivant 4.2.

Le troisième [9] construit, à partir d'une base réduite au sens de Siegel, une base réduite au sens de Korkine-Zolotarev : nous y reviendrons dans le paragraphe 4.3.

4.2. La recherche du vecteur d'un réseau L le plus proche d'un point donné N

Rappelons que l'on sait que ce problème est NP-dur.

Il existe pour le résoudre un algorithme dû à Babai [1], qui reprend les mêmes principes que le second algorithme de la section précédente et qui a la même complexité.

Par contre, si on cherche seulement un point « assez » proche, on possède un algorithme polynomial, fondé aussi sur les mêmes principes, qui trouve un point A du réseau vérifiant

$$d(N, A) \leqq 2^{(n-1)/2} d(N, L)$$

où, par définition,

$$d(N, L) = \text{Min } \{ d(N, A) / A \in L \}.$$

4.3. Les autres réductions

Nous avions mentionné dans la section 1 les réductions au sens de Minkowski et au sens de Korkine-Zolotarev. Il est prouvé que la première réduction est NP-dure et il est probable que la seconde le soit également puisqu'elle a la même « dureté » que la recherche du plus court vecteur.

Bien que cette seconde réduction soit donc moralement NP-dure, nous avons déjà mentionné en 4.1 qu'un algorithme de Kannan [9] résoud le problème polynomialement à la dimension n fixée.

Cet algorithme a été amélioré par Schnorr [21] qui a introduit une hiérarchie de réductions intermédiaires entre la réduction de Lovasz et celle de Korkine-Zolotarev.

Helfrich-Just [6] a également construit, en utilisant une technique similaire à celle de Kannan, un algorithme polynomial à dimension n fixée, qui, partant d'une base réduite au sens de Siegel, détermine une base réduite au sens de Minkowski. Par ailleurs, en dimension 3, on peut construire un algorithme polynomial qui, généralisant exactement l'algorithme de Gauss, construit directement une base réduite au sens de Minkowski, sans réduction préalable au sens de Siegel [25].

4.4. La factorisation des polynômes à coefficients entiers

L'idée fondamentale est la suivante :

Étant donné un polynôme $f(X)$ à coefficients entiers de degré n et de hauteur $M(f) = \max |f_i|$ et une assez bonne approximation d'une racine α de f on peut déterminer h le polynôme minimal du nombre algébrique α qui est par définition un facteur irréductible de f.

L'approximation $\bar{\alpha}$ de α sera :

— soit complexe et obtenue par l'algorithme de Newton [10];

— soit p-adique, et obtenue alors par l'algorithme de factorisation mod p dû à Berlekamp suivi d'un relèvement par le lemme de Hensel [15].

Si l'approximation est suffisamment fine, on peut alors appliquer un principe de séparation qui affirme : il existe δ dont la taille est polynomiale en la taille de la donnée $(n, \log M)$ tel que les deux propositions soient équivalentes :

(i) g est multiple de h,

(ii) $|g(\bar{\alpha})| \leqq \delta$ (la valeur absolue est archimédienne ou p-adique selon le cas envisagé).

Dans le premier cas, la proposition (ii) incite donc à chercher un vecteur court du réseau L engendré par les lignes v_i $(0 \leqq i \leqq n)$ de la matrice

$$A = \begin{pmatrix} C & 0 & 0 & \dots & 0 & 1 & 0 \\ 0 & C & 0 & \dots & 0 & \beta_1 & \gamma_1 \\ 0 & 0 & C & \dots & 0 & \beta_2 & \gamma_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & C & \beta_n & \gamma_n \end{pmatrix}$$

où $\beta_i = \mathscr{R}(\bar{\alpha}^i)$, $\gamma_i = \mathscr{F}(\bar{\alpha}^i)$, et C est une constante dépendant polynomialement de $M(f)$ et de δ.

On montre effectivement qu'en appliquant l'algorithme **LLL** à la matrice ci-dessus, on peut construire exactement h : le premier vecteur de la base réduite obtenue, mis sous la forme $v = \sum_{i=0}^{n} h_i v_i$ permet de construire le polynôme h sous la forme $h = \sum_{i=0}^{n} h_i X^i$.

Dans le second cas, on considère un nombre premier p, et on détermine, par l'algorithme de Berlekamp, un polynôme g de degré m vérifiant les deux propriétés :

(i) g mod p est irréductible dans $\mathbf{F}_p[X]$,

(ii) g mod p divise f mod p dans $\mathbf{F}_p[X]$.

On choisit alors une puissance de p, suffisamment grande, de la forme p^l et on « relève » g mod p en un polynôme g mod p^l. Nous cherchons alors

un polynôme $h \in \mathbf{Z}[X]$ de degré inférieur ou égal à q tel que :

(i) h soit un facteur irréductible de f dans $\mathbf{Z}[X]$,

(ii) $h \bmod p^l$ soit un multiple de $g \bmod p^l$.

Nous travaillons donc dans le réseau

$$L = \{ \varphi \in \mathbf{Z}[X] \text{ de degré } q / \varphi = p^l b + ag \text{ pour } a \text{ et } b \in \mathbf{Z}[X] \}$$

qui admet la base $V = \{ p^l, p^l X, p^l X^2, \ldots, p^l X^{m-1}, g, gX, gX^2, \ldots, gX^{q-m} \}$. Remarquons donc que si p^l est suffisamment grand, en fonction de la hauteur de g qu'on sait borner en fonction de celle de f, les vecteurs courts de L auront, dans la base V, leurs m premières composantes nulles et seront donc des petits multiples de g.

4.5. Les approximations diophantiennes simultanées

Le problème à résoudre est le suivant :

Soit $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ *un n-uplet de nombres réels. On cherche n nombres entiers* $(p_1, p_2, \ldots, p_n)$ *et un nombre entier q tels que les n nombres rationnels* $(p_1/q, p_2/q, \ldots, p_n/q)$ *soient de bonnes approximations des nombres donnés.*

On connaît une réponse à cette question, due à Dirichlet, fondée sur le théorème de Minkowski, et donc non constructive :

Pour tout n, pour tout n-uplet $(\alpha_1, \alpha_2, \ldots, \alpha_n)$, *pour tout couple* (ε, Q) *vérifiant* $\varepsilon > 0$ *et* $Q \geqq \varepsilon^{-n}$, *il existe des entiers* $(p_1, p_2, \ldots, p_n)$ *et un entier q vérifiant*

$$0 < q \leqq Q \qquad \text{et} \qquad |q\alpha_i - p_i| < \varepsilon \quad \text{pour tout } i, \qquad 1 \leqq i \leqq n.$$

Lagarias [11] a pu donner une version approchée mais constructive à ce théorème en appliquant l'algorithme **LLL** au réseau L engendré par les lignes v_i de la matrice

$$A = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0 \\ \alpha_1 & \alpha_2 & \alpha_3 & \ldots & \alpha_n & \varepsilon/Q \end{pmatrix}$$

Là encore, le premier vecteur de la base réduite obtenue, mis sous la forme

$$v = \sum_{i=1}^{n} p'_i v_i + q' v_{n+1}$$

permet de construire une bonne approximation

$$\left(\frac{p'_1}{q'}, \frac{p'_2}{q'}, \ldots, \frac{p'_n}{q'} \right).$$

On peut préciser le théorème constructif obtenu avec l'algorithme **LLL** associé à la valeur usuelle du paramètre :

Pour tout n, pour tout n-uplet $(\alpha_1, \alpha_2, \ldots, \alpha_n)$, *pour tout couple* (ε, Q) *vérifiant* $\varepsilon > 0$ *et* $Q \geqq 2^{n^2} \varepsilon^{-n}$, *on peut construire des entiers* $(p'_1, p'_2, \ldots, p'_n)$ *et un entier* q' *vérifiant*

$$0 < q' \leqq Q \qquad \text{et} \qquad |q\alpha_i - p_i| \leqq \varepsilon \quad \text{pour tout } i, \qquad 1 \leqq i \leqq n.$$

4.6. La programmation linéaire en nombres entiers

Le problème principal est le suivant :

Étant donné un polytope P de $\mathbf{R}^n$ *de volume non nul, déterminer les points de coordonnées entières situés à l'intérieur de ce polytope.*

On sait que ce problème est NP-dur en général. Mais, là encore, on peut chercher un algorithme qui soit polynomial quand la dimension n est fixée. C'est la démarche de Lenstra [16], bien décrite dans [17], qui utilise à la fois des arguments géométriques liés à la réduction des réseaux — l'espacement des hyperplans du réseau parallèles à H_{n-1} — et d'autres arguments liés à la méthode de l'ellipsoïde en programmation linéaire — la possibilité de coincer un polytope entre deux ellipsoïdes concentriques et homothétiques.

On commence par considérer que P est un ellipsoïde, puis on se ramènera à ce cas, en coinçant un polytope entre deux ellipsoïdes.

Soit f la transformation linéaire qui transforme P en une sphère unité S. Soit L le transformé du réseau $\mathbf{Z}^n$ par f. Le problème est alors transformé en le suivant :

Déterminer les points de L situés à l'intérieur de S.

On réduit le réseau L en lui appliquant l'algorithme **LLL** : on obtient ainsi une base $(b_1, b_2, \ldots, b_n)$. Puis, on procède de manière récursive, en utilisant

l'argument d'espacement qui permet de borner le nombre d'hyperplans affines parallèles à H_{n-1} qui rencontrent la sphère S.

4.7. L'attaque du système de Merkle-Hellmann

Le système de cryptographie de Merkle-Hellmann est fondé sur la difficulté du problème dit du « sac à dos ».

Soient n entiers positifs a_i — les paquets — et un entier M — le sac —, trouver un élément $X=(x_i)_{1\leq i\leq n}$ élément de $\{0, 1\}^n$ solution de l'équation

$$\sum_{i=1}^{n} a_i x_i = M.$$

Quels paquets doit-on mettre pour pouvoir remplir exactement le sac?

Ce problème est facile quand la suite a_i est super-croissante : $a_i \geqq \sum_{j\leqq i} a_j$.

On peut l'utiliser dans un système de cryptographie dont la clé publique est le système des a_i : étant donné un message formé du mot X, on le code en M. Alors, de deux choses l'une :

— si la suite n'est pas super-croissante, personne ne peut décoder;

— si elle l'est, tout le monde pourra décoder!

On utilise une suite super-croissante, dont on cache la super-croissance en lui appliquant une transformation $a \to va \bmod u$: le couple (u, v^{-1}) sera alors la clé secrète qui permettra le décodage.

Cependant, ce système n'est pas sûr : Shamir [18] et Lagarias-Odlyzko [12] ont montré qu'on pouvait briser le code, en utilisant un vecteur assez court d'un réseau bien choisi.

4.8. La prédictibilité de la suite des bits produits par le générateur congruentiel linéaire

Le générateur pseudo-aléatoire le plus célèbre est dans doute le générateur linéaire congruentiel : on choisit un module m et un multiplicateur a, premier avec m, et une donnée x_1 de départ; puis on considère la suite (x_i) définie par

$$x_{i+1} = ax_i \bmod m.$$

Stern [23] a montré, en améliorant les résultats de Frieze [2] que, même si aucun des paramètres n'est connu, la suite y_i formée par une proportion

« assez grande » des bits dominants des x_i est prédictible et donc que le générateur n'est pas cryptographiquement sûr. On travaille dans les réseaux X et Y engendrés respectivement par les vecteurs

$$u_i = \begin{pmatrix} x_{i+1}-x_i \\ x_{i+2}-x_{i+1} \\ x_{i+3}-x_{i+2} \end{pmatrix} \quad \text{et} \quad v_i = \begin{pmatrix} y_{i+1}-y_i \\ y_{i+2}-y_{i+1} \\ y_{i+3}-y_{i+2} \end{pmatrix}.$$

Les k premiers vecteurs v_i étant donnés, on trouve, par l'algorithme 3.10 une relation entière courte entre eux de la forme

$$\sum_{i=1}^{k} \lambda_i v_i = 0$$

On déduit de cela que le vecteur $\sum_{i=1}^{k} \lambda_i u_i$ est un vecteur si court du réseau X... qu'il est nul.

En effet, si le réseau X est suffisamment « régulier » géométriquement — ce qui est presque toujours le cas —, il ne possède aucun vecteur non nul qui soit très court.
Si k est bien choisi en fonction de la taille présumée des données, on construit ainsi un polynôme P défini par $P(t) = \sum_{i=1}^{k} \lambda_i t^i$ et vérifiant $P(a) \equiv 0 \bmod m$.

Si on réitère cette construction, on détermine ainsi une suite de l polynômes P_j qui appartiennent tous à un réseau L de base

$$q_0(t) = m \quad \text{et} \quad q_i(t) = t^i - a^i \quad \text{pour } i, \quad 1 \leqq i \leqq k.$$

Le réseau L a comme déterminant le nombre m cherché. On peut, par l'algorithme 3.9, trouver le déterminant $\hat{m}$ du réseau engendré par les P_j. $\hat{m}$ est un multiple de m qui décroît rapidement quand l augmente; on déduit donc la valeur de m puis ensuite une valeur très probable de a obtenue en cherchant un polynôme du premier degré dans le réseau L.

4.9. L'étude des racines l-ièmes modulo n : le cassage des cryptosystèmes d'Okamoto [26]

Le problème général s'énonce de la manière suivante :

Soient deux entiers n et $l \geqq 2$. Étant donnés deux points x_0 et y_0, un voisinage I de x_0, un voisinage J de y_0 qui contiennent respectivement un point x et un point y vérifiant $x^l \equiv y\,[n]$, on veut trouver x et y.

Plus précisément, on veut deviner un triplet (u_1, u_2, v) de « petits » entiers, solution de l'équation

$$(u_1 x_0 + u_2)^l \equiv y_0 + v\,[n]$$

qui se développe en

$$x_0^l u_1^l + C_l^1 x_0^{l-1} u_1^{l-1} u_2 + \ldots + C_l^i x_0^{l-i} u_1^{l-i} u_2^i + \ldots + C_l^1 x_0 u_1 u_2^{l-1} + u_2^l - v \equiv y_0\,[n].$$

Posant

$$w_i = u_2^i u_1^{l-i} \quad \text{pour tout } i, \qquad 0 \leqq i \leqq l-1 \quad \text{et aussi } w_l = y_0 + v - u_2^l$$

et, travaillant dans le réseau L des vecteurs $w = (w_0, w_1, \ldots, w_l)$ de $\mathbf{Z}^{l+1}$ vérifiant

$$\sum_{i=0}^{l-1} C_l^i x_0^{l-i} w_i - w_l \equiv 0\,[n],$$

on cherche un point w du réseau L qui est proche — en un sens à préciser — du point $(0, 0, \ldots, y_0)$.

Ce réseau L de déterminant n et de rang $l+1$ a pour matrice

$$\begin{pmatrix} 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0 \\ x_0^l & C_l^1 x_0^{l-1} & C_l^2 x_0^{l-2} & \ldots & C_l^{l-1} x_0 & n \end{pmatrix}$$

Si ce réseau est suffisamment « régulier », son premier minimum $\Lambda_1(L)$ sera proche de la moyenne géométrique des minima successifs, de l'ordre de $n^{2/(l+1)}$. Or, on peut montrer que la plupart des réseaux de ce type sont

« réguliers »; dans ce cas, l'unicité du point le plus proche permet d'affirmer que le point w trouvé par l'algorithme 4.2 donnera naissance au triplet (u_1, u_2, v) cherché.

4.10. Quelques autres applications...

Kaltofen [8] a utilisé l'algorithme en dimension 4 pour écrire un algorithme qui détermine le p.g.c.d. de deux nombres d'un corps quadratique principal mais non euclidien.

Landau et Miller [14] ont utilisé l'algorithme pour résoudre algorithmiquement la solvabilité par radicaux d'une équation polynomiale.

Vallée [24] a utilisé des idées de réduction de réseaux en dimension 2 pour décrire précisément la répartition des éléments dont les carrés modulo n sont inférieurs à $O(n^{2/3})$. *Elle en déduit un algorithme de factorisation entière qui a la meilleure borne de complexité prouvée à ce jour parmi les algorithmes probabilistes de factorisation entière.*

BIBLIOGRAPHIE

1. L. Babai, *On Lovász's Lattice Reduction and the Nearest Lattice Point Problem*, Combinatorica, vol. 5, 1985.
2. A. Frieze, J. Hastad, R. Kannan, J. C. Lagarias et A. Shamir, *Reconstructing Truncated Integer Variables Satisfying Linear Congruences*, in S.I.A.M. Journal on computing (to appear).
3. A. Dupré, Journal de Mathématiques, vol. 11, 1846, p. 41-64.
4. C. F. Gauss, *Recherches Arithmétiques*, Paris, 1807, réimprimé par Blanchard, Paris, 1953.
4. J. Hastad, B. Just, J. C. Lagarias et C. P. Schnorr, *Polynomial Time Algorithms for Finding Integer Relations Among Real Numbers*, Proceedings of S.T.A.C.S., Lecture Notes in Computer Science, 1986.
6. B. Helfrich, *Algorithms to Construct Minkowski and Hermite Reduced Bases*, Theoretical Computer Science, vol. 41, 1985, p. 125-139.
7. J. W. S. Cassels, *Rational Quadratic Forms*, Academic Press, 1978.
8. E. Kaltofen et H. Rolletschek, *Arithmetic in Quadratic Fields with Unique Factorization*, Comptes rendus de EUROCAL'85, Lectures notes in Computer Science, 204, Springer-Verlag.
9. R. Kannan, *Improved Algorithms for Integer programming and Related Lattice Problem*, J.A.C.M., 1983, p. 193-206.
10. R. Kannan, H. W. Lenstra et L. Lovász, *Polynomial Factorization and Bits of Algebraic and Some Transcendental Numbers*, Mathematics of Computation, vol. 50, n° 181, 1988, p. 235-250.
11. J. C. Lagarias, *Computational Complexity of Simultaneous Diophantine Approximation Problem*, 23rd I.E.E.E. Symp. F.O.C.S., 1982.

12. J. C. LAGARIAS et A. ODLYZKO, *Solving Low-Density Subset Sum Problems*, 24th I.E.E.E. Symp. F.O.C.S., 1983.

13. J. C. LAGARIAS, H. W. LENSTRA et C. P. SCHNORR, *Korkine-Zolotarev Bases and Successive Minima of a Lattice and its Reciprocal Lattice*, Technical Report, M.S.R.I. 07718-86, Mathematical Sciences Research Institute, Berkeley.

14. S. LANDAU et G. L. MILLER, *Solvability by Radicals is in Polynomial Time*, 15th Annual A.C.M. Symposium on Theory of Computing, 1983.

15. A. K. LENSTRA, H. W. LENSTRA et L. LOVASZ, *Factoring Polynomial with Rational Coefficients*, Math. Annalen, vol. 261, 1982, p. 513-534.

16. H. W. LENSTRA, *Integer Programming with a Fixed Number of Variables*, Mathematics of Operations Research, vol. 8, n° 4, nov. 1983.

17. L. LOVASZ, *An Algorithmic Theory of Numbers, Graphs and Convexity*, Technical Report, Universitat Bonn.

18. A. SHAMIR, *A Polynomial Time Algorithm for Breaking the Merkle-Hellman Cryptosystem*, 23rd I.E.E.E. Symp. F.O.C.S., 1982.

19. A. SCHONHAGE, *Factorization of Univariate Integer Polynomial by Diophantine Approximation and by an Improved Basis Reduction Algorithm*, Proceedings of the 11th I.C.A.L.P., Antwerpen, 1984, Lecture Notes in Computer Science, Vol. 172, Springer, 1984.

20. C. P. SCHNORR, *A More efficient Algorithm for Lattice Basis Reduction*, Proceedings of the 13th I.C.A.L.P., Rennes, 1986, Lecture Notes in Computer Science, vol. 226, Springer, 1986, dans Journal of Algorithms,, 1987 (à paraître).

21. C. P. SCHNORR, *A Hierarchy of Polynomial Tume Lattice Basis Reduction Algorithms*, Theoretical Computer Science, vol. 53, 1987, p. 201-224.

22. J. STERN, Lectures Notes, University of Singapore, 1986.

23. J. STERN, *Secret Linear Congruential Generators are not Cryptographically Secure*, 28th I.E.E.E. Symp. F.O.C.S., 1987.

24. B. VALLÉE, Provably fast integor factoring algorithm with quasi-uniform small quadratic residues, ACM. STOC 89, p. 98-106.

25. B. VALLÉE, *Une approche géométrique de la réduction des réseaux en petite dimension*, Thèse de doctorat de l'Université de Caen (1986), résumé paru dans le Séminaire de Théorie des Nombres de Bordeaux (1986), et dans Proceedings of EUROCAL'87, Lecture notes in Computer Science, Springer (à paraître).

26. B. VALLÉE, M. GIRAULT et Ph. TOFFIN, *How to Guess l-th Roots Modulo n by Reducing Lattice Bases*, Prépublications de l'Université de Caen, 1988, First International Joint Conference of I.S.S.A.C.-88 and A.A.E.C.C.-6, juillet 1988 (soumis).

27. P. VAN EMDE BOAS, *Another NP-Complete Partition Problem and the Complexity of Computing Short Vectors in a Lattice*, Rep. MI, U.V.A. 81-04, Amsterdam, 1981.