Calculs de relations entre constantes numériques par l'algorithme « LLL »

Sujet proposé par Frédéric Chyzak
(*)

frederic.chyzak[at]inria.fr

1 Objectif

Il s'agit pour ce sujet d'implanter un algorithme de réduction de réseau, connu sous le nom de « LLL », d'après le nom de ses inventeurs, A. K. Lenstra, H. W. Lenstra et L. Lovász. Parmi les applications possibles de cet algorithme, on notera :

- la factorisation de polynômes d'une variable en temps polynomial ;
- le cassage d'un cryptosystème (maintenant abandonné ;-)) basé sur le problème du sac à dos ;
- la mise en évidence de la faiblesse de toute une classe de générateurs de nombres (pseudo-)aléatoires ;
- la recherche de relations de dépendance linéaire à coefficients entiers entre plusieurs nombres réels donnés par des approximations numériques ;
- la détermination du polynôme minimal d'un nombre algébrique donné par une approximation numérique.

On s'intéressera particulièrement dans ce sujet aux deux dernières applications ; la première est très importante par ses conséquences mais demanderait un trop gros effort de programmation.

1.1 Dépendances linéaires entres constantes numériques

Étant donnés n nombres réels non nuls, $r_1, \ldots, r_n$, une relation de dépendance linéaire à coefficients entiers entre ces réels est une relation de la forme

$$c_1 r_1 + \cdots + c_n r_n = 0$$

pour des coefficients c_i entiers non tous nuls. Lorsqu'on se donne une approximation rationnelle de chaque réel r_i, ou mieux, la possibilité d'obtenir autant de chiffres décimaux que souhaité, une approche possible pour la détermination de relations de dépendance linéaire est de rechercher un vecteur « court », c'est-à-dire de norme petite en un sens à préciser, parmi les combinaisons linéaires à coefficients entiers des vecteurs lignes de la matrice

$$F = \begin{bmatrix} 1 & 0 & \ldots & 0 & a_1 \\ 0 & 1 & \ldots & 0 & a_2 \\ & & \ddots & & \vdots \\ 0 & \ldots & 0 & 1 & a_{n-1} \\ 0 & \ldots & 0 & 0 & a_n \end{bmatrix},$$

où chaque a_i est une troncature entière de Nr_i pour un grand entier N. (Par exemple, N est une grande puissance de 10.)

Un vecteur court est alors de la forme

$$v = (c_1, \dots, c_{n-1}, c_1 a_1 + \cdots + c_n a_n) \simeq \big(c_1, \dots, c_{n-1}, N(c_1 r_1 + \cdots + c_n r_n)\big).$$

En particulier,

$$|c_1 r_1 + \cdots + c_n r_n| \simeq \big|N^{-1}(c_1 a_1 + \cdots + c_n a_n)\big| \leq N^{-1}|v|$$

se doit d'être petit, ce qui fait de

$$c_1 r_1 + \cdots + c_n r_n = 0$$

un bon candidat pour une relation de dépendance entre les r_i. Bien qu'un tel argument ne constitue pas une preuve, la preuve formelle de relation entre constantes a dans un bon nombre de cas été grandement facilitée une fois que la bonne relation a pu être découverte expérimentalement par la méthode proposée ci-dessus. Dans un certain nombre de cas, même, des identités n'ont pu être prouvées que parce qu'elles avaient été découvertes heuristiquement en utilisant l'algorithme LLL.

Donnons un exemple. Des arguments théoriques donnent la certitude que le nombre

$$V = \int_0^\infty \frac{\sqrt{x}\,\ln^5 x}{(1-x)^5}\,\mathrm{d}x$$

peut se représenter comme l'évaluation en π d'une fonction polynomiale à coefficients rationnels. Il s'agit donc de trouver une dépendance linéaire entre

$$V, 1, \pi, \dots, \pi^d$$

pour un degré de polynôme d à déterminer. À peu près tout système de calcul formel généraliste connait des approximations pour π et permet de calculer aisément une approximation numérique de V. On prend donc par exemple $a_1 = N = 10^{25}$, $a_2 = 31415926535897932384626434 \simeq N\pi$, …, $a_9 = 94885310160705740071285755038 \simeq N\pi^8$, $a_{10} = -166994737192290704961872433 \simeq NV$ pour construire la matrice F. Les normes des vecteurs lignes de cette matrice sont toutes supérieures à $N = 10^{25}$. Par l'algorithme LLL, on trouve une base du même réseau de vecteurs, dont le premier vecteur ligne,

$$v = (c_1, \dots, c_{n-1}, c_1 a_1 + \cdots + c_n a_n) = (0, 0, 120, 0, 140, 0, -15, 0, 0, 33),$$

est de norme inférieure à 200, tous les autres vecteurs de base étant de norme supérieure à 400. Les a_i étant connus, on trouve

$$(c_1, \dots, c_n) = (0, 0, 120, 0, 140, 0, -15, 0, 0, 24),$$

d'où la relation linéaire

$$V = \int_0^\infty \frac{\sqrt{x}\,\ln^5 x}{(1-x)^5}\,\mathrm{d}x = \frac{5\pi^2}{24}(3\pi^4 - 28\pi^2 - 24)$$

qui résout le problème posé.

1.2 Polynôme minimal de nombres algébriques

Un nombre complexe α est dit algébrique (sur les nombres rationnels) s'il est solution d'un polynôme P à coefficients rationnels

$$P(\alpha) = p_0 + \cdots + p_d \alpha^d = 0.$$

Pour pouvoir utiliser l'approche de la section précédente, on se limite au cas de nombres réels.

Étant donnée une approximation numérique (réelle) d'un nombre α qu'on a de bonnes raisons de penser être algébrique, la détermination heuristique de P peut se voir comme la recherche d'une relation de dépendance linéaire sur des approximations numériques rationnelles de

$$1, \alpha, \ldots, \alpha^d.$$

Dès lors qu'on a une borne supérieure sur le degré d de P, on peut donc employer la méthode de la section précédente.

Par exemple, soit à déterminer si un nombre

$$r \simeq 0.26625264629019611453024776557584454817650128610395\ldots$$

est algébrique. Par la méthode esquissée, en testant jusqu'à $d = 6$ et pour $N = 10^{28}$, on trouve (à partir de vecteurs lignes de norme supérieure à 10^{20}), le vecteur court

$$v = (-1, 0, 0, 54, 0, 0, -10),$$

de norme inférieure à 55, tous les autres vecteurs calculés étant de norme supérieure à 5000. En poursuivant avec la méthode, on trouve

$$(c_1, \ldots, c_7) = (-1, 0, 0, 54, 0, 0, -54),$$

soit

$$P = 54X^6 - 54X^3 + 1.$$

En effet, le nombre r s'avère être une approximation du nombre algébrique

$$\alpha = \sqrt[3]{\frac{1}{2} - \frac{5\sqrt{3}}{18}}.$$

2 Algorithme LLL pour la réduction de base

Étant donnés n vecteurs lignes à entrées rationnelles,

$$a_i = (a_{i,1}, \ldots, a_{i,n})$$

pour $1 \leq i \leq n$, l'algorithme LLL recherche des combinaisons linéaires

$$v = c_1 a_1 + \cdots + c_n a_n$$

à coefficients c_l entiers tels que les vecteurs lignes v soient courts, au sens de la norme euclidienne donnée par

$$|v|^2 = v_1^2 + \cdots + v_n^2.$$

L'ensemble des combinaisons linéaires des vecteurs a_i est appelé réseau engendré par les a_i. Sous l'hypothèse que les a_i soient linéairement indépendants sur l'anneau $\mathbb{Z}$ des entiers, les a_i constituent une base du réseau, et l'algorithme LLL calcule en réalité une base du même réseau constituée exclusivement de vecteurs courts.

La stratégie adoptée par l'algorithme est de construire la base courte F en en maintenant en permanence une variante orthogonale F^* obtenue par le procédé d'orthogonalisation de Gram–Schmidt.

2.1 Orthogonalisation par la méthode de Gram–Schmidt

On dote l'espace $\mathbb{R}^n$ du produit scalaire scalaire euclidien donné par

$$(u \mid v) = u_1 v_1 + \cdots + u_n v_n$$

et de la norme euclidienne donnée par

$$|u| = \sqrt{(u \mid u)} = \sqrt{u_1^2 + \cdots + u_n^2}.$$

Étant donnée une base quelconque $F = (f_1, \ldots, f_n)$ de $\mathbb{R}^n$, le procédé de Gram–Schmidt calcule une base orthogonale en calculant successivement, pour i de 1 à n

$$f_i^* = f_i - \sum_{1 \leq j < i} \mu_{i,j} f_j^* \qquad \text{où} \qquad \mu_{i,j} = \frac{|f_j^*|}{(f_i \mid f_j^*)}.$$

En vue de ce qui suit, il conviendra de pouvoir conserver les trois matrices F, $M = (\mu_{i,j})$ et $F^* = (f_1^*, \ldots, f_n^*)$, et de remarquer la relation $F = MF^*$.

2.2 Réduction de base

Le cœur de l'algorithme LLL est le suivant : étant donnée une base $F = (f_1, \ldots, f_n)$ d'un réseau de $\mathbb{R}^n$,

1. calculer l'orthogonalisation de Gram–Schmidt de F ;
2. en partant de $i = 2$ et tant que $i \leq n$, faire :
   (a) pour j décroissant de $i - 1$ à 1 remplacer f_i par $f_i - \lfloor \mu_{i,j} + 1/2 \rfloor f_j$ et mettre à jour l'orthogonalisation de Gram–Schmidt de F ;
   (b) si $i \neq 1$ et $|f_{i-1}^*|^2 > 2|f_i^*|^2$, échanger f_i et f_{i-1} et mettre à jour l'orthogonalisation de Gram–Schmidt de F, puis faire $i = i - 1$; sinon faire $i = i + 1$;
3. renvoyer $(f_1, \ldots, f_n)$.

($\lfloor x \rfloor$ représente le plus grand entier inférieur ou égal à x, si bien que $\lfloor x + 1/2 \rfloor$ représente l'entier le plus proche de x.)

3 Considérations théoriques

On montre que si $A = \max |f_i|$, l'algorithme LLL effectue $O(n^4 \ln A)$ opérations arithmétiques sur des entiers de taille $O(n \ln A)$.

On pourra s'apercevoir que, même sur des exemples simples tels ceux donnés dans la section d'exemples, des entiers de plus de 64 bits (les `long` de Java) seront nécessaires. C'est pourquoi des entiers en précision arbitraire devront être utilisés. Leur l'implantation n'est pas demandée : elle justifierait bien plus qu'un projet à elle seule. Mais fort heureusement la classe `java.math.BigInteger` de Java fournit l'accès à de grands entiers.

4 Travail demandé

Le travail minimum demandé est le suivant :

1. Programmer les opérations nécessaires sur les nombres rationnels en définissant une représentation de nombres rationnels à partir d'entiers en précision arbitraire. En Java, il conviendra d'utiliser la classe `java.math.BigInteger` qui fournit de tels entiers. (La documentation est disponible sur les pages WEB de l'X.)

2. Implanter le procédé d'orthogonalisation de Gram–Schmidt.

3. Implanter l'algorithme LLL et le tester sur les exemples de la section qui suit. On cherchera à comprendre quelle est la précision des entrées nécessaire pour obtenir un résultat crédible.

5 Autres exemples

Voici quelques autres exemples pour vous permettre de mettre au point votre programme. (Tous ces exemples sont tirés du livre *Modern Computer Algebra*, cité en référence.)

Un tout petit exemple est donné par les vecteurs $(12, 2)$ et $(13, 4)$, dont le réseau engendré contient $(1, 2)$ et $(11, 0)$.

Dans des exemples de l'application à la factorisation de polynômes, on doit montrer que le réseau engendré par $(1, -5136, 0)$, $(0, 1, -5136)$ et $(0, 0, 15625)$ contient le vecteur $(3, 1, 1)$, ou que le réseau engendré par $(1, -34967, 0)$, $(0, 1, -34967)$ et $(0, 0, 117649)$ contient le vecteur $(1, 1, 1)$.

Une autre application, liée au cassage du cryptosystème mentionné dans l'introduction, demande de déterminer si l'entier 1215 peut s'exprimer comme somme (éventuellement avec répétition, mais un petit nombre de répétitions) des entiers 366, 385, 392, 401, 422 et 437. Par l'algorithme naïf, on doit tester un nombre exponentiel de sommes ; par un algorithme de réduction de réseau, on se ramène à une complexité polynomiale.

Bien d'autres exemples sont donnés sur le site donné en référence.

6 Références

(Me contacter si ces références sont difficiles à trouver.)

- Description originale de l'algorithme : Factoring Polynomials with Rational Coefficients, *Mathematische Annalen* 261, pages 515–534, de A. K. Lenstra, H. W. Lenstra et L. Lovász (1982).

- Les chapitres 16 et 17 du livre *Modern Computer Algebra*, de Joachim von zur Gathen et Jürgen Gerhard, Cambridge University Press (1999). Voir aussi l'URL `http://www-math.uni-paderborn.de/mca/`.

- Un site entier est dédié à la recherche de relations entre constantes numériques à l'URL `http://www.cecm.sfu.ca/projects/IntegerRelations/`.