

4 Résumé de la thèse (2 pages maxi)

Mon thème de recherche principal est l’algorithmique de la géométrie des nombres, c’est-à-dire la conception et l’étude d’algorithmes calculant sur les réseaux euclidiens. J’étudie les réseaux en grande partie dans l’objectif de les utiliser dans diverses applications comme la cryptographie par le biais de la théorie algorithmique des nombres, et l’arithmétique des ordinateurs.

Les réseaux euclidiens sont les arrangements réguliers de points dans l’espace, ou, plus précisément, les sous-groupes discrets de \mathbb{R}^n pour un certain entier n . La théorie des réseaux euclidiens est née dans les années 1890, quand, à la suite de travaux de Hermite, Minkowski a publié son ouvrage *Geometrie der Zahlen*. Il s’agit d’une branche de la théorie des nombres, dont les applications sont diverses (approximation diophantienne, théorie algébrique des nombres, ...). Un réseau est en pratique représenté par une base constituée de vecteurs linéairement indépendants $\mathbf{b}_1, \dots, \mathbf{b}_d$ (voir la Figure 1) : le réseau est l’ensemble des combinaisons linéaires entières des \mathbf{b}_i . Dès que la dimension d est au moins 2, un réseau donné admet une infinité de bases, liées entre elles par des relations unimodulaires, c’est-à-dire des matrices carrées à coefficients entiers et de déterminant ± 1 . Cependant, un réseau a un certain nombre d’invariants, ne dépendant pas de la base choisie. Un invariant simple à calculer est le volume du réseau, c’est-à-dire le volume d -dimensionnel du paralléloèdre engendré par les d vecteurs de n’importe quelle base (il s’agit d’un calcul de déterminant). D’un point de vue algorithmique, l’invariant le plus intéressant est le premier minimum du réseau, c’est-à-dire la norme euclidienne d’un vecteur non nul le plus court. Intuitivement, l’objectif algorithmique principal est de transformer des bases à partir desquelles il est difficile d’obtenir de l’information sur le premier minimum en de meilleures bases : ce procédé est appelé réduction de réseau. La date charnière en géométrie des nombres algorithmique se situe au début des années 1980, lorsque Lenstra, Lenstra et Lovász ont mis au point un algorithme de réduction de réseaux de complexité polynomiale, permettant d’obtenir des bases dont le vecteur le plus court est de longueur au plus $(\sqrt{4/3} + \epsilon)^d$ fois le premier minimum. Cet algorithme, que l’on appelle LLL, a constitué une véritable révolution. Les trois applications historiques sont la factorisation des polynômes à coefficients entiers, les approximations rationnelles simultanées et la programmation entière en dimension fixée. Cet algorithme a aussi reçu un succès immédiat dans le domaine de la cryptanalyse. Il a en particulier été utilisé pour casser de nombreuses variantes du cryptosystème de type sac-à-dos proposé par Merkle et Hellmann, le premier concurrent de RSA. L’algorithme de Lenstra, Lenstra et Lovász reste aujourd’hui un outil très utilisé en cryptanalyse de cryptosystèmes à clé publique, comme par exemple certaines variantes rapides de RSA.

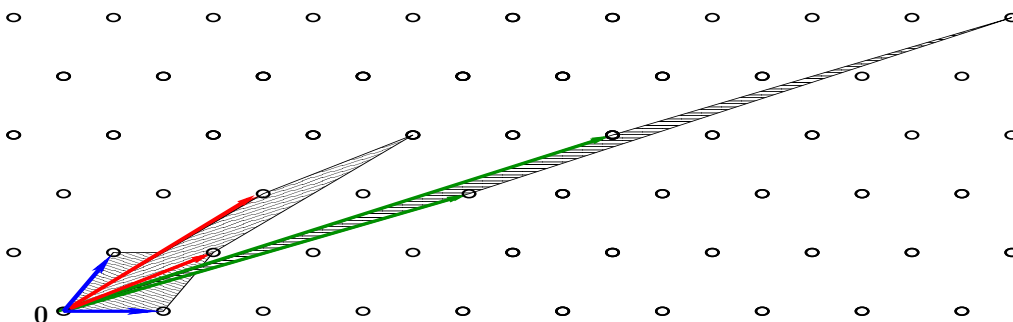


FIG. 1 – Trois bases d’un réseau de dimension 2.

L’objet de mes travaux est double : d’une part j’introduis de nouveaux algorithmes de réduction, de la dimension 1 $[G]^2$ (on a alors un problème de pgcd), à la dimension quelconque $[E]$ (comme l’algorithme LLL), en passant par la petite dimension $[H]$ (dimension inférieure à 4); d’autre part, je décris de nouvelles applications de la réduction des réseaux, dans le domaine de l’arithmétique des ordinateurs $[A,D,I,F]$.

En ce qui concerne la réduction des réseaux, j’ai construit trois principaux algorithmes : l’algorithme rapide de calcul de pgcd par les bits de poids faibles (avec Paul Zimmermann), l’algorithme glouton de réduction (avec Phong Nguyễn) qui généralise naturellement l’algorithme de Gauss-Lagrange jusqu’en di-

²Les références avec des lettres correspondent à mes publications, les autres étant données avec des chiffres.

mension 4, et l'algorithme L^2 (avec Phong Nguyễn) qui calcule très efficacement des bases de la même qualité que celles renvoyées par l'algorithme LLL, en dimension quelconque.

L'algorithme de calcul de pgcd par les bits de poids faibles est une variante de l'algorithme de Knuth, qui quant à lui travaille par les poids forts. L'inconvénient de réduire les tailles des entiers dont on cherche le pgcd par les poids forts est que l'on va à l'encontre de la propagation des retenues. Cela rend nécessaire une étape de réparation dans l'algorithme de Knuth : cette étape est laborieuse et rend l'implantation délicate. L'idée sous-jacente à notre algorithme est de réduire les tailles par les poids faibles, pour aller dans le même sens que les retenues et supprimer par là l'étape de réparation. Notre algorithme est significativement plus simple, mais semble légèrement plus lent en moyenne.

L'algorithme glouton de réduction généralise et permet d'unifier les algorithmes de Gauss-Lagrange (en dimension 2), de Vallée et Semaev (en dimension 3). Nous montrons que, jusqu'en dimension 4, l'algorithme termine en temps quadratique en la taille de l'entrée et renvoie une base de qualité optimale. Bien que l'algorithme soit naturel et simple à décrire, son analyse est étonnamment complexe. Nous utilisons des considérations fines sur la géométrie des réseaux de petite dimension (et en particulier sur leurs cellules de Voronoï), ainsi qu'une analyse amortie pour sommer finement les coûts des étapes successives.

En dimension quelconque, nous introduisons l'algorithme L^2 , qui LLL-réduit très rapidement des réseaux. Étant donnée une base à coefficients entiers d'un réseau de dimension d avec des vecteurs de normes $\leq B$, l'algorithme LLL finit en temps $O(d^6 \log^3 B)$, en utilisant des opérations sur des entiers de taille $O(d \log B)$. Cette complexité est trop élevée pour réduire des réseaux de taille ne serait-ce que modérée. Ainsi, l'algorithme LLL n'est presque jamais utilisé. À la place, on se sert de variantes flottantes de LLL, dans lesquelles l'arithmétique entière utilisée dans l'orthogonalisation de Gram-Schmidt (central dans LLL) est remplacée par de l'arithmétique flottante. Il existait un algorithme flottant de LLL-réduction dû à Schnorr, mais ce dernier n'est que théorique car sa mise en œuvre est laborieuse, et probablement finalement trop coûteuse. En pratique, on utilise des variantes heuristiques, qui peuvent boucler et renvoyer des réponses incorrectes. Notre algorithme repose sur de l'arithmétique flottante, est prouvé et admet une complexité quadratique en $\log B$ en dimension fixée (d'où son nom) : sa complexité est plus précisément bornée par $O(d^5(d + \log B) \log B)$. Il s'agit du premier algorithme de LLL-réduction avec ce type de complexité (les autres étant cubiques en dimension fixée), ce qui le rend très naturel, puisque cette complexité généralise celle de l'algorithme d'Euclide, qui peut être interprété comme une réduction de réseau en dimension 1, et des algorithmes de Gauss-Lagrange, Semaev et glouton (jusqu'en dimension 4).

L'arithmétique des ordinateurs consiste en majeure partie à étudier l'implantation des nombres réels. L'alternative la plus répandue est l'arithmétique flottante. Celle-ci est standardisée pour un certain nombre de précisions et pour les opérations de base (addition, soustraction, multiplication, division et racine carrée) par la norme IEEE-754. Dans ma thèse, j'utilise la réduction de réseaux (*via* la méthode de Coppersmith pour trouver les petites racines de polynômes modulaires) pour répondre à certaines difficultés liées à l'implantation des fonctions mathématiques élémentaires. Les deux travaux que je décris ci-dessous donnent des arguments supplémentaires aux partisans de la standardisation des fonctions élémentaires.

Nous construisons un nouvel algorithme pour résoudre le dilemme du fabricant de tables $[A, D, I]$ (la première version de l'algorithme a été mise au point avec Vincent Lefèvre et Paul Zimmermann, je l'ai améliorée significativement par la suite). Apporter une réponse à ce dilemme revient à déterminer à quel point l'image par la fonction donnée d'un nombre représentable peut être proche d'un nombre représentable ou du milieu de deux nombres représentables consécutifs : ces nombres ont des images difficiles à arrondir puisqu'une très fine approximation de la valeur exacte est requise pour décider du sens d'arrondi. La version la plus aboutie de ma méthode permet de déterminer dans quelle mesure la suite des bits de l'image par la fonction donnée d'un nombre représentable est aléatoire : si x est représentable sur n bits et que l'on se donne les n^2 premiers bits de $f(x)$ sauf les n premiers, alors mon algorithme retrouve x en temps heuristique polynomial en n (à condition que f soit suffisamment régulière). Par ailleurs nous améliorons la méthode des tables de Gal $[F]$ (avec Paul Zimmermann), qui est une technique classique pour implanter certaines fonctions en précision fixe. Nous décrivons le lien qui existe entre les valeurs de ces tables et les pires cas pour l'arrondi de la fonction à implanter. Pour certaines (par exemple \sin), il s'agit de trouver les pires cas simultanés pour deux fonctions. Nous adaptons l'algorithme de recherche des pires cas à cette situation.