Analyse probabiliste de la réduction des réseaux euclidiens cryptographiques

Mariya Georgieva

Université de Caen Basse-Normandie

École doctorale SIMEM

Thèse de doctorat

présentée et soutenue le : *09/12/2013*

par

Mariya Georgieva

pour obtenir le

Doctorat de l'Université de Caen Basse-Normandie

Spécialité : Informatique et applications

Analyse probabiliste de la réduction des réseaux euclidiens cryptographiques

Directrice de thèse : *Brigitte Vallée*

Jury

Valérie Berthé	Directrice de Recherche au CNRS	LIAFA (Paris)	(Rapporteure)
Guillaume Hanrot	Professeur à l'ENS Lyon	LIP (Lyon)	(Rapporteur)
Julien Clément	Chargé de recherche au CNRS	GREYC (Caen)	(Examinateur)
Pascal Paillier	Expert sécurité et Co-fondateur	CryptoExpert (Paris)	(Examinateur)
Emmanuel Prouff	Expert sécurité	ANSSI (Paris)	(Examinateur)
Brigitte Vallée	Directrice de Recherche au CNRS	GREYC (Caen)	(Directrice)
Fabien Laguillaumie	Professeur à l'ISFA et au ENS Lyon	LIP (Lyon)	(Invité)
Loïck Lhote	Maître de conférence à l'ENSICAEN	GREYC (Caen)	(Invité)

Table des matières

Notations

Nombres

$\mathbb{N}$	Ensemble des nombres entiers naturels
$\mathbb{Z}$	Ensemble des nombres entiers
$\mathbb{Q}$	Ensemble des nombres rationnels
$\mathbb{R}$	Ensemble des nombres réels
$\Re z$	Partie réelle du nombre complexe z
$\Im z$	Partie imaginaire du nombre complexe z
$\lvert x\rvert$	Valeur absolue de x
$\lfloor x \rfloor$	Partie entière de x
$\lceil x \rceil$	Entier immédiatement supérieur à x
$\lfloor x \rceil$	Entier le plus proche de x
$\langle \cdot \vert \cdot \rangle$	Produit scalaire euclidien
$\lVert \cdot \rVert$	Norme euclidienne

Réseaux euclidiens et paramètres géométriques.

$\mathcal{L}$	Réseau euclidien	déf. 1.1, page 10
d	Dimension du réseau $\mathcal{L}$	sec. 1.1.1, page 10
$\det \mathcal{L}$	Déterminant du réseau $\mathcal{L}$	déf. 1.9, page 14
$\lambda_i(\mathcal{L})$	i-ème minimum du réseau $\mathcal{L}$	déf. 1.13, page 16
$\gamma(\mathcal{L})$	Constante d'Hermite du réseau $\mathcal{L}$	déf. 1.16, page 17
γ_d	Constante d'Hermite de dimension d	déf. 1.17, page 18

Base d'un réseau et paramètres géométriques d'une base.

$B = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$	Base d'un réseau de dimension d	déf. 1.2, page 10
$B^\star = (\mathbf{b}_1^\star, \ldots, \mathbf{b}_d^\star)$	Base orthogonalisée de la base B par le procédé de Gram Schmidt	déf. 1.4, page 12
M	$\max \lVert b_i \rVert$	sec. 2.1.1, page 26
$\mathcal{P}$	Matrice de passage de Gram Schmidt qui exprime $B^\star$ dans B	sec. 1.1.2, page 12
$m_{i,j}$	Coefficient générique de la matrice $\mathcal{P}$	sec. 1.1.2, page 12
ℓ_i	Longueur du i-ième vecteur orthogonalisé, $\ell_i := \lvert\lvert \mathbf{b}_i^\star \rvert\rvert$	déf. 2.1, page 27
r_i	i-ième rapport de Siegel, $r_i := \ell_{i+1}/\ell_i$	déf. 2.1, page 27
$\mu_i(B)$	i-ième défaut de longueur de la base B	déf. 2.2, page 28
$\gamma(B)$	Constante d'Hermite de la base B	déf. 2.2, page 28
$\delta(B)$	Défaut d'orthogonalité de la base B	déf. 2.2, page 28

Algorithmique de la réduction des réseaux

s	Paramètre de Siegel	sec. 2.1.6, page 30
t	Paramètre de Lovász	sec. 2.1.7, page 31
ρ	Facteur de décroissance dans la réduction de Gauss	sec. 2.3.2, page 36
α	Décrément, $\alpha := -\log_s \rho$	déf. 4.12, page 85
K	Nombre d'itérations	sec. 2.3.3, page 38
$\Delta(B)$	Potentiel d'une base B, $\Delta(B) := \prod_{i=1}^{d} \ell_i^{d-i}$	déf. 4.13, page 86
$D(B)$	Autre Potentiel d'une base B, $D(B) := \prod_{i=1}^{d} \ell_i^{2(d-i)}$	sec. 2.3.4, page 38

Probabilités

$\mathbb{P}[X = w]$	Probabilité que la variable aléatoire X soit égale à w
$\mathbb{E}[X]$	Espérance de la variable aléatoire X
$\mathbb{V}[X]$	Variance de la variable aléatoire X
$\sigma[X]$	Écart type de la variable aléatoire X

Modèles d'exécution

`M1`	Modèle d'exécution simplifié "Tas de sable"	sec. 4.7.1, page 90
`M2`	Modèle d'exécution semi-simplifié déterministe	sec. 4.7.2, page 90
`M3`	Modèle d'exécution semi-simplifié probabiliste	sec. 4.7.3, page 91
`M4`	Algorithme LLL avec la condition de Siegel	sec. 4.7.4, page 92
`M5`	Algorithme LLL avec la condition de Lovász	sec. 4.7.5, page 92

Tas de sable

$\mathbf{q} = (q_1, \ldots, q_d)$	Configuration tas de sable	déf. 5.1, page 98
$\mathcal{Q}_d(\mathbf{q}, H, h)$	Tas de Sable	déf. 5.1, page 98
$\underline{\mathcal{M}}(\mathbf{q})$	Masse du tas de sable $\mathbf{q}$, $\underline{\mathcal{M}}(\mathbf{q}) := \sum_{i=1}^{d} q_i$	déf. 5.11, page 103
$\underline{\mathcal{E}}(\mathbf{q})$	Énergie du tas de sable $\mathbf{q}$, $\underline{\mathcal{E}}(\mathbf{q}) := \sum_{i=1}^{d} i\, q_i$	déf. 5.12, page 103

Chip Firing Game

$\mathbf{c} = (c_1, \ldots, c_{d-1})$	Configuration chip firing game	déf. 5.2, page 99
$\mathcal{C}_d(\mathbf{c}, H, h)$	Chip Firing Game	déf. 5.2, page 99
$\mathcal{M}(\mathbf{c})$	Masse du cfg $\mathbf{c}$, $\mathcal{M}(\mathbf{c}) := \sum_{i=1}^{d-1} c_i$	déf. 5.11, page 103
$\mathcal{E}(\mathbf{c})$	Énergie du cfg $\mathbf{c}$, $\mathcal{E}(\mathbf{c}) := \sum_{i=1}^{d-1} i(d-i)\, c_i$	déf. 5.12, page 103
$\underline{\mathcal{E}}^{-}(\mathbf{c})$	Énergie gauche du cfg $\mathbf{c}$, $\underline{\mathcal{E}}^{-}(\mathbf{c}) := \sum_{i=1}^{d-1} (d-i)\, c_i$	déf. 5.12, page 103
$\underline{\mathcal{E}}^{+}(\mathbf{c})$	Énergie droite du cfg $\mathbf{c}$, $\underline{\mathcal{E}}^{+}(\mathbf{c}) := \sum_{i=1}^{d-1} i\, c_i$	déf. 5.12, page 103

Modèles d'entrées

$\mathcal{A}(\Upsilon, g, d)$	cfg de type Ajtai de dimension $d-1$, de masse totale Υ, et de distribution g	sec. 6.1.1, page 125
$\mathcal{U}(\Upsilon, d, \beta)$	cfg uni-tas de dimension $d-1$, de masse Υ en position βd	sec. 6.2.1, page 129
$\mathcal{K}(\Upsilon, d)$	cfg de type sac-à-dos de dimension $d-1$ et de masse Υ	sec. 6.2.4, page 131
$\mathcal{N}(\Upsilon, d)$	cfg de type NTRU de dimension $d-1$ et de masse Υ	sec. 6.2.4, page 132

Introduction

Le contexte de la thèse.

Les thèmes abordés dans cette thèse se situent aux interfaces de la cryptographie, de l'algorithmique et de l'analyse des algorithmes. Ils se concentrent sur un domaine particulier, celui de la géométrie des nombres.

La réduction des réseaux euclidiens. L'objet central de la géométrie des nombres est le réseau euclidien, qui se définit de manière informelle comme un arrangement régulier de points dans un espace euclidien. Ce domaine mathématique a débuté avec les travaux d'Hermite, puis Minkowski, à la fin du dix-neuvième siècle, a vraiment fondé le domaine.

Un réseau euclidien admet une infinité de bases. D'un point de vue algébrique, elles jouent toutes le même rôle. Mais, du point de vue de la structure euclidienne, ce n'est plus le cas, et il y a des bases qui sont "meilleures" que d'autres. On s'accorde à dire qu'une base est de bonne qualité – ou encore réduite – quand elle possède des vecteurs "assez courts" et "assez orthogonaux". Mais, cela reste une notion assez floue, et la géométrie des nombres a cherché, dès le début, à préciser cette notion de "base réduite". Minkowski a cherché par exemple à déterminer un majorant de la longueur d'un vecteur le plus court, en fonction du déterminant du réseau. Mais, jusqu'à l'avènement de l'algorithmique, la géométrie des nombres s'est posé ces questions de manière non constructive, même si elle a pris vite conscience du rôle que jouaient les bonnes bases, dans la théorie diophantienne par exemple. Elle a cherché à décrire plus précisément ce qu'on pouvait attendre d'une base réduite, sans s'intéresser au processus de réduction, qui vise à construire une base réduite.

La géométrie algorithmique des nombres a débuté vers 1980. On s'est posé à ce moment-là deux questions : quelle est la complexité des principaux problèmes liés à la réduction ? Comment construire des algorithmes efficaces de réduction ? Il existe plusieurs notions de réduction, des notions historiques, comme celles introduites par Hermite ou Minkowski, ou des plus récentes, comme celle introduite par Lovász. À chacune d'entre elles correspond une notion de qualité de la base réduite, et une complexité du calcul nécessaire pour l'obtenir. On a démontré – ou conjecturé– dès 1980, que les problèmes "de base" des réseaux sont NP–difficiles : c'est prouvé pour le problème $\mathcal{CVP}$ (Closest Point Problem : trouver un point du réseau le plus proche d'un point donné de l'espace ambiant) et conjecturé pour le problème $\mathcal{SVP}$ (Shortest Vector Problem : trouver un vecteur non nul le plus court d'un réseau). Comme l'obtention d'un vecteur le plus court est (sans doute) NP–difficile, il n'est pas envisageable de construire une très bonne base en temps polynomial. Compte-tenu du potentiel applicatif énorme du domaine, on cherche le plus souvent à opérer un compromis entre la qualité de la base obtenue et le temps mis à l'obtenir. Il existe des algorithmes qui réalisent ce compromis, comme l'algorithme LLL [65], proposé par Lenstra, Lenstra et Lovász en 1982, qui construit en temps polynomial une base dite "LLL-réduite", avec de bonnes propriétés euclidiennes : les normes des vecteurs de la base

de sortie et leur orthogonalité sont maitrisées, même si leur rapport à l'optimum croît de manière exponentielle avec la dimension.

L'analyse probabiliste de la réduction des réseaux. Analyser un algorithme, c'est étudier son comportement, et de manière plus précise, évaluer ses performances, en rapidité d'exécution, ou en réussite à la sortie. L'analyse d'un algorithme aide d'abord à mieux comprendre les sources éventuelles de son inefficacité, et cette meilleure compréhension permet souvent d'en proposer des versions plus efficaces. En comparant les performances d'un algorithme donné avec d'autres algorithmes de la même classe, on peut aussi opérer une classification entre ces algorithmes, et décider celui qu'il faut employer.

Souvent, lorsqu'on parle de complexité d'algorithme, il s'agit de complexité dans le pire des cas. Les analyses classiques de complexité effectuées sur les algorithmes de réduction sont le plus souvent des analyses dans le pire des cas. Mais le pire des cas peut être atteint rarement, et ne donne pas lieu à une mesure de complexité "générique". Dans le cadre de l'analyse probabiliste, on définit une distribution sur les entrées, et les principaux paramètres de l'algorithme (nombre d'itérations, géométrie de la configuration de sortie) deviennent alors des variables aléatoires ; l'analyse probabiliste de l'algorithme est l'analyse de ces variables aléatoires, effectuée de manière asymptotique, quand la taille des entrées devient grande. L'analyse en moyenne s'intéresse à la valeur moyenne –ou l'espérance– de ces paramètres.

Il y a peu d'études qui concernent l'analyse probabiliste de la réduction. L'algorithme LLL, par exemple, a une structure complexe. En petite dimension (algorithme d'Euclide pour la dimension 1, algorithme de Gauss pour la dimension 2), les analyses utilisent fortement le système dynamique sous-jacent, et montrent clairement comment la distribution des données peut avoir une influence sur la valeur moyenne du nombre d'itérations, par exemple. Mais, malheureusement, en plus grande dimension, même si on peut modéliser l'exécution de l'algorithme par un système dynamique, celui-ci s'avère très complexe à étudier.

Devant la difficulté d'une modélisation exacte de l'exécution, il est légitime d'opérer une modélisation "simplifiée". Madritsch et Vallée [71] ont proposé une modélisation simplifiée de l'algorithme LLL, fondée sur les "tas de sable", qui est un système dynamique discret très classique, et facile à analyser (nombre d'itérations, configuration de sortie, influence de la stratégie). L'hypothèse principale, qui valide ce modèle simplifié, suppose que qu'un certain facteur de décroissance est constant lors de l'exécution, mais peut dépendre bien sûr de la base d'entrée du réseau. Cette modélisation très (trop ?) simplifiée donne une bonne idée qualitative du déroulement de l'algorithme, et permet aussi de développer des heuristiques pour établir une algorithmique plus performante.

La cryptologie et les réseaux. Dès l'avènement de la cryptologie, les réseaux et les algorithmes de réduction se sont révélés incontournables en cryptologie. Il y a trois raisons à ce phénomène. D'abord, les réseaux ont un grand pouvoir de modélisation, et ils sont sous-jacents à la plupart des problèmes linéaires discrets, et à beaucoup d'autres problèmes, non linéaires, après une linéarisation préalable. La solution à de tels problèmes est souvent reliée à un vecteur court du réseau (le plus court vecteur ou seulement un vecteur "assez court") . Ensuite, la géométrie algorithmique des nombres contient des problèmes algorithmiquement "difficiles" , qui sont pourtant résolubles de manière "approchée" . La difficulté de ces problèmes va être utilisée pour construire des protocoles avec de bonnes propriétés de sécurité, tandis que l'existence de bons algorithmes approchés va permettre de "casser" les cryptosystèmes. Historiquement, les réseaux sont d'abord intervenus en cryptanalyse, pour "casser" les cryptosystèmes, puis, il se sont avérés utiles pour construire des protocoles cryptographiques fondés sur un problème de

réseau difficile.

En cryptanalyse, à chaque fois qu'un système est fondé sur un problème linéaire, ou facilement linéarisable, les réseaux sont un instrument de cryptanalyse redoutable : on transforme l'instance cryptographique en une instance de réseau et la cryptanalyse est fondée sur la possibilité de trouver un petit vecteur du réseau. C'est ainsi qu'on peut "casser" beaucoup de protocoles de type "sac-à-dos" et certaines instances de protocoles de type RSA.

Ensuite, les cryptosystèmes à base des réseaux sont "à priori" résistants aux ordinateurs quantiques. Ce n'est pas le cas des systèmes fondés sur le problème de la factorisation, car Shor [104] a proposé, au milieu des années 1990, un algorithme de factorisation dont la complexité quantique est polynomiale. Si, donc, un jour, il y a des ordinateurs quantiques, tous les protocoles fondés sur la difficulté de la factorisation (comme RSA) deviendront inutilisables. Depuis, on a aussi cherché à construire des algorithmes pour les problèmes difficiles des réseaux, qui soient de complexité quantique polynomiale, mais, pour le moment, on n'a pas réussi (voir Micciancio et Regev [83] pour plus de détails). C'est pour cela que les protocoles à base des réseaux sont une bonne alternative cryptographique dans un monde quantique.

Les réseaux sont devenus aussi très centraux en cryptographie "positive", car on peut utiliser la difficulté des problèmes $\mathcal{SVP}$ et $\mathcal{CVP}$, pour construire des cryptosystèmes résistants, qui jouissent de fonctionnalités cryptographiques intéressantes. Tout d'abord, on peut relier, pour certains problèmes du domaine, la complexité dans le pire des cas et la complexité en moyenne, comme Ajtai [3] l'a fait en 1996. Cette connexion n'existe pas dans d'autres cadres. Or, en cryptographie, on ne peut guère se contenter d'une difficulté dans le pire des cas, car on souhaite que le cryptosystème reste sûr, sur la plupart de ses clés, et pas seulement pour un petit ensemble de clés difficiles à retrouver. D'autre part, ces protocoles permettent de faire du chiffrement homomorphe, comme l'a introduit Gentry [37]. Il y a un certain nombre de tels cryptosystèmes, comme ceux de Ajtai-Dwork, `GGH, NTRU` [4, 43, 49], ou celui de Regev basé du problème dit `LWE`.

L'analyse probabiliste de la réduction des réseaux cryptographiques. Les réseaux qui interviennent dans ces cadres liés à la cryptographie sont appelés par la suite des réseaux "cryptographiques". Ces réseaux sont donnés par des bases à la forme très particulière, et, il n'est pas légitime de les considérer comme "génériques" vis-à-vis de la réduction. Pour mieux cerner la sécurité de tels cryptosystèmes, il faut donc mieux évaluer la difficulté de la réduction sur les réseaux particuliers que sont les réseaux cryptographiques. Est-ce plus facile, ou plus difficile de réduire un réseau cryptographique qu'un réseau "générique" ? Le vecteur le plus court d'un tel réseau est-il vraiment très court ? Et, si l'on veut aborder la sécurité cryptographique de manière réaliste, on doit se poser ces questions de manière probabiliste, le modèle probabiliste étant lié à la caractéristique géométrique des réseaux cryptographiques.

Ces réseaux cryptographiques sont de nature très diverse, selon leur provenance. Par exemple, la méthode de Coppersmith [21], qui permet de trouver des "petites" racines de polynômes modulaires, univariés ou multivariés, a montré son extrême efficacité pour la cryptanalyse de plusieurs variantes de RSA [12, 31]. Plus généralement, cette technique s'avère extrêmement efficace, dès que l'attaque se ramène à la résolution d'équations polynomiales. Les réseaux sous-jacents, dits de Coppersmith, sont très particuliers, mais aussi très différents d'autres réseaux cryptographiques, comme ceux qui sont liés au système NTRU ou aux systèmes sac-à-dos, par exemple.

L'étude spécifique des réseaux cryptographiques (construits lors des cryptanalyses, ou fondements même des cryptosystèmes) pose des questions fondamentales, sur le comportement des algorithmes de réduction, comme sur les caractéristiques géométriques des réseaux. Et ces ques-

tions doivent être posées de manière probabiliste. La réponse à ces questions, pour ces classes probabilistes de réseaux cryptographiques, peut permettre d'expliquer les comportements génériques observés en pratique, et peut aussi conduire à des améliorations algorithmiques prenant en compte la spécificité géométrique de ces réseaux.

L'approche de la thèse

Cette thèse vise à analyser l'efficacité des cryptanalyses et à mieux comprendre la sécurité des systèmes cryptographiques basés sur les réseaux. Il s'agit d'aborder le sujet avec le point de vue probabiliste, en rupture avec l'approche du pire des cas universellement adopté par la communauté.

La modélisation simplifiée de l'algorithme LLL, proposée par Madritsch et Vallée est fondée sur les "tas de sable" Ce modèle est un système dynamique discret très classique, qui agit sur une suite de piles de sable, et cherche à les égaliser en prenant à chaque instant une quantité de sable, désignée par α dans la suite, sur une grande pile et en la répartissant de manière égale sur ses deux voisines immédiates. L'algorithme s'arrête lorsque les piles de sable ont toutes à peu près la même hauteur. Dans ce modèle, la quantité α ne varie pas au cours de l'exécution de l'algorithme – c'est l'hypothèse essentielle. Mais, l'algorithme LLL ne fonctionne pas aussi simplement, car cette quantité α n'est pas a priori constante au cours de l'exécution. C'est l'hypothèse de simplification qui suppose que α ne varie pas au cours de l'algorithme.

Dans cette thèse, nous nous posons trois types de questions à propos de cette modélisation simplifiée :

Modélisation de distributions cryptographiques. Le comportement probabiliste de l'algorithme LLL dépend du type des réseaux considérés en entrée, et cette thèse se concentre sur les réseaux cryptographiques. Puisque la modélisation de l'exécution de l'algorithme se fait via les tas de sable, nous avons étudié et modélisé les entrées particulières qui nous intéressent, liées aux réseaux cryptographiques, du point de vue des tas de sable. Nous avons ainsi étudié et modélisé trois familles de réseaux cryptographiques : tout d'abord, les réseaux dits réseaux d'Ajtai, qui interviennent dans la connexion "pire cas–cas moyen", donnent lieu à des tas "tous très pleins",– ensuite, les réseaux sac-à-dos ou réseaux NTRU, qui interviennent dans les cryptosystèmes de même nom, donnent lieu à des tas de sable "avec un seul tas" – et enfin les réseaux de Coppersmith, qui interviennent dans la méthode de Coppersmith pour rechercher des petites racines modulaires, donnent lieu à des tas de sable "avec des trous". Ces modélisations permettent de faire des expérimentations adaptées et de mieux comprendre le fonctionnement de l'algorithme LLL sur ces différentes familles.

Étude du décrément α. Nous avons effectué une campagne d'expérimentations (dont certaines ont été menées en commun avec Madritsch et Schneider) ; nous avons fait varier la dimension, la taille des vecteurs, et le type des réseaux d'entrée étudiés. Nous avons observé les variations du paramètre α lors de l'exécution de l'algorithme, selon la position de la pile, le moment où elle est modifiée, et aussi l'ordre dans lequel les piles sont considérées (la stratégie). Nous avons progressé dans la réponse à la question "comment adapter la valeur de α à la classe des réseaux d'entrée", même si cette réponse reste partielle.

Étude d'un modèle semi-simplifié. Nous avons aussi proposé une modélisation semi-simplifiée de l'algorithme, qui s'avère beaucoup plus réaliste : on suppose que la hauteur α

de sable qu'on retire d'une pile dépend de la hauteur de la pile. C'est beaucoup plus proche de ce qui se passe dans la réalité de l'algorithme LLL, même si on néglige encore dans ce modèle d'autres facteurs, qui semblent moins importants à prendre en compte. Ce modèle donne lieu à un système dynamique général, qui n'est plus un tas de sable, et qui s'avère donc plus complexe à étudier. Nous avons commencé à étudier ce système dynamique (en dimensions 1 et 2), ce qui modélise l'algorithme LLL en dimensions 2 et 3. La dimension 3 est déjà une dimension où l'algorithme LLL a une dynamique complexe, et la dynamique "simplifiée" que nous obtenons est, là encore très instructive, et permet de mieux comprendre l'algorithme LLL en dimension 3. Nous avons obtenu, dans ce modèle semi-simplifié, des résultats prouvés de même nature de ceux qui sont prouvés en dimension 2 sur le "vrai" algorithme de Gauss. Nous avons aussi des pistes pour étudier ce système dynamique "semi-simplifié" en dimension quelconque.

Plan de la thèse

Chapitre 1. Nous présentons dans ce chapitre toutes les notions fondamentales du domaine des réseaux euclidiens (base, déterminant, minima successifs, défaut et constante d'Hermite) et plusieurs résultats importants du domaine (les deux théorèmes de Minkowski). Nous adoptons ensuite un point de vue plus algorithmique, et discutons les principaux problèmes algorithmiques du domaine, en nous concentrant sur l'étude de leur "difficulté".

Chapitre 2. Ce chapitre est consacré à un problème algorithmique central dans le domaine, celui de la réduction d'un réseau. Réduire un réseau, c'est construire une base du réseau, avec de "bonnes" propriétés euclidiennes. Il existe plusieurs notions de réduction, qui dépendent des propriétés précises qu'on attend d'une bonne base. Ce chapitre décrit dans un premier temps les principaux processus de réduction, qui recherchent tous un compromis entre la qualité de la base obtenue et le temps passé à l'obtenir, en tentant de généraliser ce qui est connu en petite dimension. C'est pourquoi nous nous concentrons ensuite sur les algorithmes connus en ces petites dimensions, où les processus de réduction, en qualité et en complexité, sont en quelque sorte optimaux : c'est l'algorithme d'Euclide en dimension 1, et l'algorithme de Gauss en dimension 2, qui peut être considéré lui-même comme une extension à la dimension 2 de l'algorithme d'Euclide. Puis, nous revenons à une dimension quelconque, et présentons en détail l'algorithme LLL, qui est un des acteurs principaux de cette thèse. Nous décrivons les principaux paramètres de cet algorithme (nombre d'itérations, configuration de sortie), dans le cadre usuel, qui est celui du "pire des cas". Nous concluons en décrivant des applications diverses de la réduction des réseaux (programmation linéaire entière, approximations diophantiennes simultanées, calcul de racines n-ème modulaires, factorisation entière).

Chapitre 3. Au chapitre précédent, nous avons présenté plusieurs applications de la réduction des réseaux dans des domaines variés ; ici, nous nous centrons sur les applications des réseaux liées à la cryptologie. Nous commençons par quelques rappels sur la cryptographie, puis nous développons les interactions entre les deux domaines – réseaux euclidiens et cryptologie. Dans un premier temps, les réseaux sont intervenus essentiellement dans le "cassage" de protocoles ; puis, au milieu des années 90 et les travaux d'Ajtai, les réseaux sont devenus une brique de base en cryptologie. Nous décrivons au cours de ce chapitre plusieurs types de réseaux qui seront utilisés dans la suite, en particulier dans la modélisation ultérieure du chapitre 7.

Chapitre 4. Ce chapitre vise d'abord à décrire les principes généraux de l'analyse probabiliste d'un algorithme. On revient ensuite au cas d'étude de la thèse, les algorithmes de réduction de réseaux, et on décrit les principaux paramètres de ces algorithmes, liés à leur exécution comme

à leur configuration de sortie. On revient ensuite sur les analyses probabilistes précises qui ont déjà été effectuées : d'abord, en dimension générale dans des modèles probabilistes particuliers, appelés modèles sphériques, puis, en petite dimension (en dimension 1, pour l'algorithme d'Euclide, et en dimension 2, pour l'algorithme de Gauss), en utilisant les systèmes dynamiques sous-jacents.
On explique alors, informellement, pourquoi cette étude probabiliste de la réduction des réseaux est sans doute très difficile à mener dans le cas général, quand on veut travailler à la fois en dimension quelconque, et dans des modèles probabilistes proches de la réalité cryptographique. On introduit finalement le cadre de la thèse, qui vise à effectuer une analyse probabiliste "fine", mais dans un modélisation simplifiée – à la fois de l'exécution et des modèles probabilistes d'entrées. On termine en proposant des hypothèses simplificatrices et en décrivant les modèles d'exécutions simplifiés, au nombre de cinq, qui vont du plus simple au plus réaliste.

Chapitre 5. Le premier de ces modèles simplifiés –et le plus simplifié– est le tas de sable ou le "chip firing game", qui simule différents phénomènes physiques, inspirés par des observations dans la nature comme les formations de tas de sable ou les avalanches de neige. Madritsch et Vallée ont proposé il y a quelques années de modéliser les algorithmes de réduction de manière simplifiée par l'écoulement d'un tas de sable. Le point de vue additif qu'ils adoptent les conduit à étudier l'exécution de l'algorithme en suivant la trace des longueurs des orthogonalisés (les ℓ_i), ou mieux, la trace des rapports de Siegel (les r_i). C'est ainsi que la modélisation de l'exécution via les tas de sable (pour les variables ℓ_i) ou le chip firing game –ou cfg– (via les variables r_i) s'impose naturellement. Comme ce modèle des tas de sable est très classique, il peut aider à mieux comprendre les algorithmes de réduction. Nous revenons donc, comme Madritsch et Vallée l'ont déjà fait avant nous, sur les principales caractéristiques de ce modèle. Nous expliquons aussi pourquoi il faut adapter ce modèle, car les tas de sable qui apparaissent naturellement en liaison avec la réduction des réseaux ne sont pas ceux qui apparaissent habituellement. De plus, c'est vraiment le modèle "chip firing game" qui apparaît le plus adapté ici, car il modélise directement l'évolution des rapports r_i de Siegel. Or, c'est sous l'aspect "tas de sable" que ce modèle est le plus souvent décrit. Nous traduisons donc tous les résultats classiques dans le cadre des *cfg* qui nous sera utile par la suite.

Chapitre 6. Dans ce chapitre, nous nous concentrons sur le modèle le plus simple, celui du *cfg*, où nous cherchons à faire une analyse probabiliste, qui soit à la fois suffisamment générique, mais qui soit aussi applicable à la réalité cryptographique. Puisque nous nous intéressons à l'analyse probabiliste de la réduction, il est essentiel que les entrées soient aussi modélisées dans le même cadre, celui des tas de sable, ou, mieux encore celui des *cfg*. Comme nous cherchons une modélisation générique des entrées, qui puisse être aussi appliquée à la "réalité cryptographique", nous nous posons ici deux questions principales. Quels sont les principaux types de *cfg* choisis en entrée qui vont permettre de modéliser différents types d'exécution ? Ces types se rencontrent-ils naturellement en cryptographie ?
Nous introduisons ici trois types de cfg : les cfg "grands tas", formés de piles, toutes de grande hauteur positive – les cfg "uni-tas" constitués d'une seule grande pile positive, les autres piles étant de hauteur nulle – enfin les cfg "tas à trous" constitués d'alternance régulière de piles de hauteur positive et de piles de hauteur négative, qu'on appelle les trous. Il est clair que ces trois classes de cfg constituent des briques de base naturelles qui permettent de construire des cfg généraux par concaténation. Ces types de cfg sont déjà étudiés dans le chapitre précédent, car ils se présentent naturellement dans la problématique interne des tas de sable et des cfg. Mais, ici, nous montrons aussi que ces types de cfg se présentent naturellement dans les applications cryptographiques, que nous avons décrites au Chapitre 3. Les réseaux introduits par Ajtai donnent lieu naturellement à des entrées "grands tas" ; les cfg "uni-tas" qui semblent pourtant des cfg

très particuliers sont liés à des réseaux typiques de la cryptanalyse (sac-à -dos, factorisation, NTRU) ; les cfg “tas à trous” modélisent des cas particuliers de la méthode de Coppersmith pour trouver de petites racines modulaires, et posent la question de l'indépendance des blocs, question déjà abordée au chapitre précédent, sur laquelle nous revenons ici.

Chapitre 7. Le chapitre 4 a introduit une classe de modèles simplifiés de l'exécution de l'algorithme de réduction. Dans les chapitres 5 et 6, nous avons étudié le plus simplifié de tous ces modèles, le modèle du *cfg* dit encore modèle `M1`. Ici, nous nous tournons vers un modèle semi–simplifié, le modèle dit `M2`. C'est un modèle plus réaliste que le modèle du *cfg* puisqu'il suppose que la hauteur de sable retirée d'une pile (de sable) à chaque étape n'est plus constante mais dépend de la hauteur de la pile. Dans l'algorithme LLL, la hauteur de sable retirée dépend aussi des coefficients sous-diagonaux de la matrice d'orthogonalisation de Gram-Schmidt mais le modèle `M2` considère toujours cette quantité comme constante.
Le système dynamique obtenu n'est plus un tas de sable mais un système dynamique général, à trou. Dans ce chapitre, nous proposons une première analyse de ce système, et nous commençons par les petites dimensions : la dimension 1 (qui correspond à la modélisation semi–simplifiée de l'algorithme de Gauss), puis la dimension 2 (qui correspond à la modélisation semi–simplifiée de l'algorithme LLL en dimension 3). Le comportement de l'algorithme LLL en dimension 3 est déjà mal compris, et c'est donc un cas important à traiter. Nous faisons une première analyse de ce système dynamique, quand c'est la stratégie dite “gloutonne” qui est utilisée. Nous donnons quelques angles d'attaque pour l'étude du modèle `M2` en dimension quelconque.

Chapitre 8. C'est un chapitre expérimental qui vise à confronter nos modélisations simplifiées avec la réalité algorithmique. Dans le cadre du modèle `M1`, nous avons prouvé que la géométrie de la configuration de sortie est largement indépendante du modèle d'entrée, pourvu qu'il donne lieu à un cfg d'énergie initiale suffisamment grande. Par contre, l'exécution de l'algorithme (et en particulier son nombre d'itérations) dépend largement du modèle d'entrée. Les expériences de Gama, Nguyen et Stehlé montrent que ces phénomènes apparaissent aussi dans la réalité algorithmique, qui travaille avec le modèle de l'algorithme LLL dit modèle `M5`. Ces travaux fournissent une preuve expérimentale indirecte de la légitimité de nos modèles simplifiés. Mais nous nous posons ici ces questions de “légitimité” directement. Les hypothèses de simplification que nous avons faites dépendent du comportement du facteur de décroissance ρ, de sa contrepartie additive, qui est le décrément α. Ce chapitre est une étude expérimentale de ce décrément.

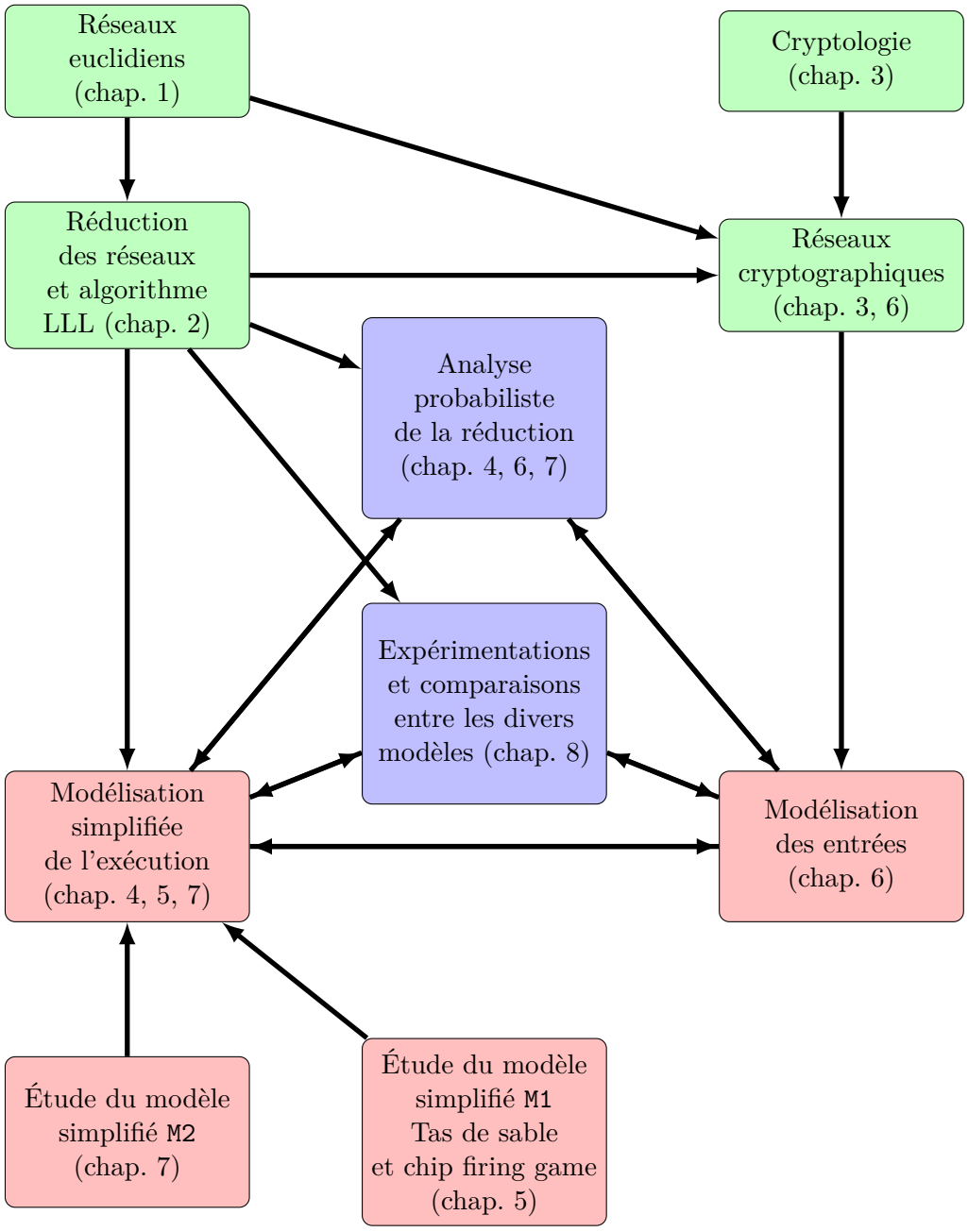


FIGURE 1: L'articulation entre les différents chapitres de la thèse

Chapitre 1

Les réseaux euclidiens

Sommaire

La géométrie des nombres est une branche de la théorie des nombres qui a pour origine les travaux de Minkowski, publiés en 1896 dans l'édition *Geometrie der Zahlen* [87]. Le contexte initial est d'étudier la réduction des formes quadratiques introduites dans les travaux de Lagrange [63], Gauss [35], Hermite [48] et Korkine et Zolotareff [58]. Dans ses travaux, Minkowski donne une interprétation géométrique à la fois plus naturelle et plus intuitive des résultats d'Hermite [48] sur les formes quadratiques positives.

Du point de vue algorithmique, la géométrie des nombres a connu un nouveau développement au début des années 1980 avec la publication de l'algorithme LLL [65], du nom de ses auteurs Arjen K. Lenstra, Hendrik W. Lenstra et László Lovász. Initialement présenté dans le cadre de la factorisation de polynômes, l'algorithme LLL admet de nombreuses applications en programmation linéaire entière, approximation diophantienne simultanée, cryptanalyse, cryptographie, transmission d'informations, arithmétique des ordinateurs, etc.

L'algorithme LLL est au cœur de cette thèse et l'objet mathématique fondamental qu'il manipule est le réseau euclidien. Un réseau se définit de manière informelle comme un arrangement régulier de points dans un espace euclidien. Précisément, un réseau est l'ensemble des combinaisons linéaires entières de vecteurs linéairement indépendants et appelés base du réseau. Un

réseau admet une infinité de bases et réduire un réseau signifie construire une "bonne base", avec des vecteurs assez courts et assez orthogonaux à partir d'une base quelconque. L'algorithme LLL est un algorithme de réduction de réseaux ayant une complexité polynomiale. Mais il en existe d'autres [99, 101, 33] effectuant des réductions plus ou moins fortes avec des complexités plus ou moins importantes.

Avant d'aborder aux chapitres 2 et 3 les applications de l'algorithme LLL et des réseaux euclidiens, nous présentons dans ce chapitre toutes les notions fondamentales et quelques résultats importants liés aux réseaux. Cette présentation n'est pas exhaustive et nous renvoyons le lecteur vers des ouvrages beaucoup plus complets comme [73, 105] pour les propriétés mathématiques et pour une approche algorithmique vers [18, 69, 92].

Plan. Les réseaux et l'orthogonalisation de Gram-Schmidt sont définis à la section 1.1. La section 1.2 est consacrée au déterminant et aux minima successifs qui sont des invariants du réseau. Les théorèmes de Minkowski et l'invariant d'Hermite sont présentés à la section 1.3. Finalement après un bref rappel de plusieurs notions de complexité à la section 1.4, la section 1.5 décrit les principaux problèmes faciles et difficiles sur les réseaux (SVP, CVP...) et qui sont omniprésents dans les applications.

1.1 Principales définitions sur les réseaux

1.1.1 Réseaux euclidiens

Intuitivement, les réseaux euclidiens sont des arrangements réguliers de points dans l'espace euclidien $\mathbb{R}^n$. L'exemple le plus simple est $\mathbb{Z}^n$ formé par tout les points avec des coefficients entiers, mais tous les sous groupes de $\mathbb{Z}^n$ sont aussi des réseaux. Une définition possible d'un réseau euclidien est la suivante.

Définition 1.1. *Un réseau euclidien $\mathcal{L}$ est un sous groupe additif discret non vide de $\mathbb{R}^n$ ($n \geq 0$).*

Si le réseau n'est pas trivial, alors il est de cardinalité infinie et forme un module de type fini sur $\mathbb{Z}$. Ainsi, tout élément peut s'écrire comme la combinaison linéaire à coefficients entiers d'une certaine base. La proposition suivante caractérise les réseaux et introduit la notion de base d'un réseau.

Proposition 1.2. *Soit $\mathcal{L}$ un réseau euclidien. Il existe des vecteurs $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$ linéairement indépendants de $\mathbb{R}^n$ tels que $\mathcal{L}$ est l'ensemble des combinaisons linéaires à coefficients entiers de $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$. Autrement dit, $\mathcal{L}$ est de la forme $\mathcal{L} := \{\mathbf{x} \in \mathbb{R}^n; \quad \mathbf{x} = \sum_{i=1}^{d} x_i \mathbf{b}_i, \qquad x_i \in \mathbb{Z}\}$. Nous dirons que $B := (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$ forme une base du réseau $\mathcal{L}$.*

Toutes les bases d'un réseau $\mathcal{L}$ ont le même nombre d'éléments d appelé rang ou dimension du réseau. Souvent, un réseau $\mathcal{L}$ est représenté par une de ses bases B écrite sous forme matricielle. Dans toute cette thèse nous adoptons la convention suivante : les lignes de la matrice sont les coordonnées des vecteurs de la base. Ainsi avec un réseau de rang d, la matrice obtenue contient d lignes et n colonnes et appartient à $\mathbb{R}^{d \times n}$. Cette représentation n'est pas unique, mais elle est finie. La proposition qui suit montre le lien entre deux bases d'un même réseau.

Proposition 1.3. *Soit $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$ et $(\mathbf{b}'_1, \mathbf{b}'_2, \ldots, \mathbf{b}'_d)$ deux bases de $\mathcal{L}$ et B et B' les matrices $(d \times n)$ respectives dans la base canonique de $\mathbb{R}^n$. Alors la matrice de changement de base est une matrice inversible unimodulaire U (matrice avec des coefficients entiers et un déterminant qui vaut ± 1) vérifiant $B' = UB$.*

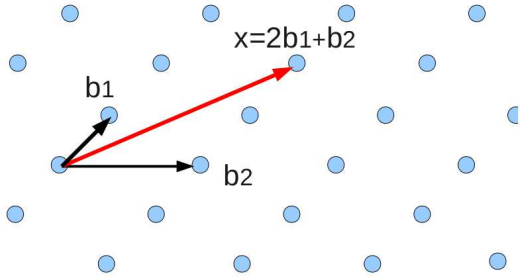


FIGURE 1.1: Un réseau de dimension 2 et un vecteur $\mathbf{x}$ représenté sous la forme d'une combinaison linéaires des vecteurs $(\mathbf{b}_1, \mathbf{b}_2)$.

Preuve. Comme B et B' sont deux bases du même réseau, il existe deux matrices $U \in \mathbb{Z}^{d\times d}$ et $U' \in \mathbb{Z}^{d\times d}$ telles que $B = U'B'$ et $B' = UB$. Alors $B = U'UB$ et en multipliant les deux cotés par ${}^tB$, nous obtenons $B^tB = U'UB^tB$. La matrice B^tB est inversible, donc $U'U = Id$ et $\det(U').\det(U) = 1$. D'un autre coté, les deux matrices ont des coefficients entiers et leur déterminant respectif est entier ce qui implique que $|\det U| = |\det U'| = \pm 1$. □

La proposition précédente montre que deux bases d'un réseau sont liées par une matrice unimodulaire. En fait, si B est une base et U une matrice unimodulaire, alors UB est aussi une base du réseau. Puisqu'il existe une infinité de matrices unimodulaires de dimension $d \geq 2$, un réseau $\mathcal{L}$ possède une infinité de bases. Pour tester si deux bases engendrent le même réseau, il suffit de calculer la matrice de passage d'une base à l'autre et de vérifier si elle est unimodulaire.

Notons que si un réseau est donné par un système générateur de vecteurs (pas nécessairement linéairement indépendants), alors il est possible de retrouver en temps polynomial une base du réseau en calculant la forme normale d'Hermite [78].

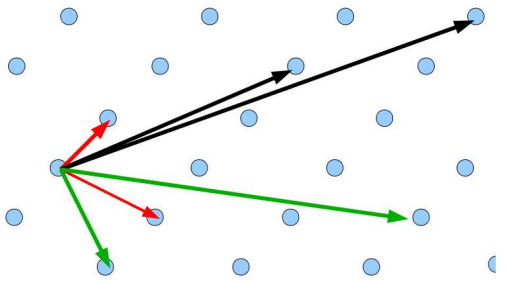


FIGURE 1.2: Un réseau de dimension 2 et trois bases équivalentes

Nous verrons que dans la plupart des applications introduites aux chapitres 2 et 3, trouver une base du réseau qui possède de bonnes propriétés euclidiennes, avec des vecteurs assez courts et assez orthogonaux est essentiel. Par exemple en cryptanalyse, après linéarisation et construction d'un réseau adéquat, casser un système est souvent équivalent à trouver un vecteur court dans un réseau (cf. chapitre 3). Nous introduisons maintenant l'orthogonalisation de Gram-Schmidt qui permet en un certain sens de mesurer la qualité de l'orthogonalité d'une base d'un réseau.

1.1.2 Orthogonalisation de Gram-Schmidt

Une des propriétés d'une "bonne base" est son orthogonalité. Malheureusement, un réseau $\mathcal{L}$ ne possède pas forcement de base orthogonale. Pour mesurer l'orthogonalité d'une base, il est utile d'avoir une base orthogonale de référence, comme la base orthogonale de Gram-Schmidt.

Définition 1.4. *(Orthogonalisation de Gram-Schmidt) Soit* $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$ *des vecteurs linéairement indépendants. La base orthogonalisée de Gram-Schmidt de* $B = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$ *est la famille de vecteurs linéairement indépendants* $B^\star = (\mathbf{b}_1^\star, \mathbf{b}_2^\star, \ldots, \mathbf{b}_d^\star)$ *définie par*

$$\mathbf{b}_1^\star = \mathbf{b}_1, \qquad \mathbf{b}_i^\star = \mathbf{b}_i - \sum_{j=1}^{i} m_{i,j}.\mathbf{b}_j^\star \quad \textit{pour } i \geq 2, \qquad \textit{où} \qquad m_{i,j} = \frac{\langle \mathbf{b}_i | \mathbf{b}_j^\star \rangle}{\|\mathbf{b}_j^\star\|^2}$$

On note $\mathcal{P}$ la matrice de passage de B à $B^\star$ ($B = \mathcal{P}B^\star$) et la quantité $\ell_i := \|\mathbf{b}_i^\star\|$ désigne la norme du i-ème vecteur orthogonalisé, appelée encore longueur de Siegel. La matrice de passage est une matrice triangulaire inférieure avec des 1 sur la diagonale et est donnée par

$$\mathcal{P} := \begin{array}{c} \\ \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_i \\ \mathbf{b}_{i+1} \\ \vdots \\ \mathbf{b}_d \end{array} \begin{array}{c} \begin{array}{ccccccccc} \mathbf{b}_1^\star & \mathbf{b}_2^\star & \ldots & \mathbf{b}_{i-1}^\star & \mathbf{b}_i^\star & \mathbf{b}_{i+1}^\star & \ldots & \mathbf{b}_d^\star \end{array} \\ \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\ m_{2,1} & 1 & \ldots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{i,1} & m_{i,2} & \ldots & m_{i,i-1} & 1 & 0 & 0 & 0 \\ m_{i+1,1} & m_{i+1,2} & \ldots & m_{i+1,i-1} & m_{i+1,i} & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{d,1} & m_{d,2} & \ldots & m_{d,i-1} & m_{d,i} & m_{d,i+1} & \ldots & 1 \end{pmatrix} \end{array}$$

La figure suivante représente une base orthogonale de Gram-Schmidt en dimension 2.

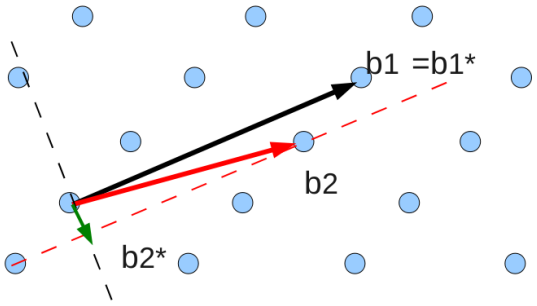


FIGURE 1.3: Deux vecteurs et leur orthogonalisation de Gram-Schmidt

Les vecteurs de $B^\star$ sont par construction des vecteurs orthogonaux et pour $i = 1 \ldots d$, les vecteurs $(\mathbf{b}_1^\star, \ldots, \mathbf{b}_i^\star)$ engendrent le même espace vectoriel que les vecteurs $(\mathbf{b}_1, \ldots, \mathbf{b}_i)$. Cependant, $B^\star$ n'est généralement pas une base du réseau. Pour le réseau donné par la Figure 1.3 engendré par les vecteurs $(\mathbf{b}_1, \mathbf{b}_2)$, la base orthogonale $(\mathbf{b}_1^\star, \mathbf{b}_2^\star)$ n'est pas une base du réseau. En outre, l'orthogonalisation de Gram-Schmidt dépend de l'ordre des vecteurs de la base B. Cette propriété est fondamentale pour l'algorithme LLL qui utilise des échanges entre deux vecteurs successifs de B pour "améliorer" son orthogonalité. La figure 1.4 montre l'effet d'un échange sur la base orthogonale $B^\star$. On voit que la norme du vecteur $\mathbf{b}_2^\star$ augmente (ou diminue selon le sens de l'échange) ce qui donne une base plus ou moins "plate".

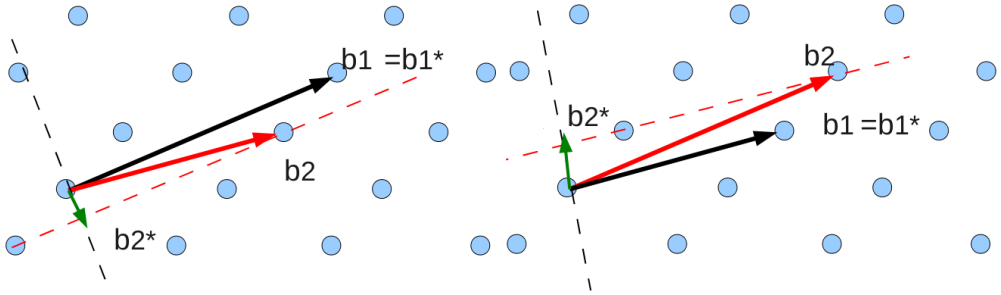


FIGURE 1.4: Échange de deux vecteurs

Du fait du lien entre la base B et son orthogonalisé de Gram-Schmidt, il est possible de retrouver plusieurs propriétés de la base B à partir de $B^\star$. La proposition suivante fait le lien entre un plus petit vecteur non-nul de B et les longueurs des vecteurs de la base orthogonale.

Proposition 1.5. *Soit un réseau $\mathcal{L}$ engendré par une base B. Soit $B^\star$ la base orthogonalisée de Gram Schmidt de la base B et $\mathbf{v} \in \mathcal{L}(B)/\ \{0\}$ alors :*

$$\|\mathbf{v}\| \geq \min_{i\in[1..d]} \|\mathbf{b}_i^\star\|.$$

Preuve. Si $\mathbf{v} \in \mathcal{L}(B)/\{0\}$ alors $\mathbf{v}$ s'écrit comme combinaison linéaire des vecteurs de la base $B = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$. Nous posons k le plus petit indice tel que $\mathbf{v}$ appartient au sous-espace engendré par $(\mathbf{b}_1, \ldots, \mathbf{b}_k)$. En particulier,

$$\mathbf{v} = v_1\mathbf{b}_1 + v_2\mathbf{b}_2 + \ldots + v_k\mathbf{b}_k$$

avec $v_k \neq 0$ soit

$$v = \sum_{i=1}^{k} v_i \left(\sum_{j=1}^{i} m_{i,j}\mathbf{b}_j^\star \right) = v_k\mathbf{b}_k^\star + \sum_{i=1}^{k-1} v_i \left(\sum_{j=1}^{i} m_{i,j}\mathbf{b}_j^\star \right).$$

On en déduit que $\|\mathbf{v}\| \geq |v_k| \cdot \|\mathbf{b}_k^\star\| \geq \|\mathbf{b}_k^\star\| \geq \min_{i\in[1..d]} \|\mathbf{b}_i^\star\|$ car $v_k \neq 0$. Ceci complète la preuve. □

1.2 Invariants des réseaux

Nous avons vu que le rang d d'un réseau ne dépend pas de la base choisie pour le représenter. Le rang est un exemple d'invariant de réseau.

Définition 1.6. *Une quantité est un invariant d'un réseau $\mathcal{L}$ si celle-ci ne dépend pas de la base choisie B représentant $\mathcal{L}$.*

Dans cette section, nous présentons d'autres invariants : le déterminant et les minima successifs.

1.2.1 Déterminant d'un réseau

Tout d'abord, nous définissons la matrice de Gram pour une famille de vecteurs.

Définition 1.7. *(Matrice de Gram) La matrice de Gram $G(B)$ d'une famille de vecteurs $B = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$ est la matrice carrée de $\mathbb{R}^{d\times d}$ formée par tous les produits scalaires $\langle \mathbf{b}_i | \mathbf{b}_j \rangle$.*

Remarquons que si la matrice B a pour lignes les vecteurs $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$, alors la matrice de Gram s'exprime comme $G(B) = B \cdot {}^t B$ et $G(B)$ est inversible si et seulement si les vecteurs $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$ sont linéairement indépendants.

Proposition 1.8. *Soient B et B' deux bases du même réseau $\mathcal{L}$, alors*

$$\det G(B) = \det G(B').$$

Autrement dit, $\det G(B)$ est un invariant du réseau $\mathcal{L}(B)$.

Preuve. Soit U et U' les matrices de changement de base entre B et B', telles que $B = U'B'$ et $B' = UB$. Comme U et U' sont unimodulaires, leur déterminant vaut ± 1 soit

$$\det G(B) = \det B^t B = \det(U'B'.{}^tB'.{}^tU') = \det U'. \det G(B'). \det U' = \det G(B').$$

□

Nous pouvons maintenant définir le déterminant d'un réseau $\mathcal{L}$.

Définition 1.9. *Soit B une base d'un réseau $\mathcal{L}$. Le déterminant du réseau $\mathcal{L}$ est défini par*

$$\det \mathcal{L} = \sqrt{\det G(B)} = \sqrt{\det B.{}^tB}.$$

Cette quantité correspond aussi au volume du parallélogramme engendré par les vecteurs $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$, appelé domaine fondamental du réseau $\mathcal{L}$ pour la base B et qui correspond à l'ensemble

$$\sum_{i=1}^{d} x_i \mathbf{b}_i : (x_1, \ldots, x_d) \in [0, 1[^d.$$

Comme le déterminant est un invariant, tous les domaines fondamentaux on le même volume qu'on appelle indifféremment volume ou déterminant du réseau $\mathcal{L}$. La figure 1.5 représente plusieurs domaines fondamentaux d'un même réseau. Notons que pour $n = d$, nous avons $\det \mathcal{L} = \det B$.

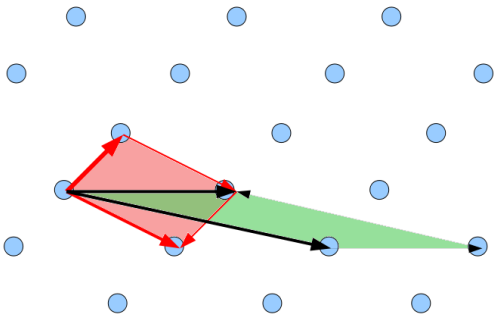


FIGURE 1.5: Un réseau de dim 2 et interprétation géométrique du déterminant

Nous relions maintenant le déterminant d'un réseau avec les bases orthogonales de Gram-Schmidt.

Proposition 1.10. $\det \mathcal{L}^2 = \det G(B) = \prod_{i=1}^{d} \|\mathbf{b}_i^\star\|^2$

Preuve. Soit $B^\star$ la matrice qui contient les vecteurs orthogonalisés de Gram-Schmidt. Alors $B = \mathcal{P}B^\star$ et puisque $\mathcal{P}$ est une matrice carrée de déterminant 1,

$$\det G(B) = \det \mathcal{P}B^\star.{}^tB^\star.{}^t\mathcal{P} = (\det \mathcal{P})^2.\det G(B^\star) = \det G(B^\star) = \prod_{i=1}^{d} \|\mathbf{b}_i^\star\|^2.$$

La dernière égalité vient du fait que $B^\star$ est une base orthogonale et que $G(B^\star)$ est une matrice diagonale. □

On peut utiliser la Proposition 1.10 pour donner une borne supérieure du déterminant d'un réseau. Géométriquement, l'inégalité suivante exprime que le volume du parallélépipède engendré par les vecteurs $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$ est maximal s'il est rectangle.

Proposition 1.11. *(Inégalité de Hadamard)*

$$\det(\mathcal{L}) \leq \prod_{i=1}^{d} \|\mathbf{b}_i\|.$$

Preuve. Comme les $\mathbf{b}_i^\star$ sont orthogonaux, nous avons pour tout $i = 1 \ldots d$,

$$\|\mathbf{b}_i\|^2 = \|\mathbf{b}_i^\star + \sum_{j=1}^{i-1} m_{i,j}\mathbf{b}_j^\star\|^2 = \|\mathbf{b}_i^\star\|^2 + \sum_{j=1}^{i-1} m_{i,j}^2 \|\mathbf{b}_j^\star\|^2 \geq \|\mathbf{b}_i^\star\|^2.$$

Donc pour tout i, $\|\mathbf{b}_i\| \geq \|\mathbf{b}_i^\star\|$. La proposition 1.10 donne alors le résultat. □

1.2.2 Minima successifs

Les points d'un réseau ne dépendent pas de la manière dont le réseau est représenté. Cette remarque "évidente" s'applique aux plus petits vecteurs non-nuls du réseau également appelés "plus courts vecteurs" et qui forment des invariants. Nous notons $\mathcal{B}(\mathbf{0}, r)$ la boule fermée de centre $\mathbf{0}$ et de rayon $r \geq 0$.

Définition 1.12. *Le premier minimum du réseau $\mathcal{L}$ est défini par :*

$$\lambda_1(\mathcal{L}) = \min(r, \mathcal{B}(\mathbf{0}, r) \cap \mathcal{L} \neq \{\mathbf{0}\}).$$

Il existe toujours un vecteur du réseau de norme $\lambda_1(\mathcal{L})$. En fait le premier minimum désigne bien la norme d'un plus court vecteur non nul de $\mathcal{L}$. C'est aussi la distance minimale entre deux points distincts de $\mathcal{L}$.
Plus généralement, on peut définir le i-ème minimum d'un réseau $\mathcal{L}$.

Définition 1.13. *Les minima successifs d'un réseau $\mathcal{L}$ sont les quantités positives $\lambda_1(\mathcal{L}), \lambda_2(\mathcal{L}) \ldots \lambda_d(\mathcal{L})$, où $\lambda_i(\mathcal{L})$ est le plus petit rayon d'une boule centrée en $\mathbf{0}$ et contenant i vecteurs linéairement indépendants de $\mathcal{L}$.*

$$\lambda_i(\mathcal{L}) = \min(r, \dim(\mathcal{B}(\mathbf{0}, r) \cap \mathcal{L}) = i).$$

Les d nombres $\lambda_i(\mathcal{L})$ sont bien définis et ne dépendent pas de la manière dont est représenté $\mathcal{L}$. Ce sont des invariants du réseau. Les minima successifs vérifient $\lambda_1(\mathcal{L}) \leq \lambda_2(\mathcal{L}) \leq \ldots \leq \lambda_d(\mathcal{L})$. Ces valeurs sont toujours atteintes (il existe toujours des vecteurs du réseau de normes égales aux minima successifs). En revanche dès que $d \geq 4$, ces vecteurs ne forment pas nécessairement une base. Un exemple est le réseau de Korkine-Zolotarev [58] engendré par les lignes de la matrice

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Les minima successifs sont tous égaux à 2 mais il n'existe aucune base de cinq vecteurs linéairement indépendants de norme 2 qui engendre le réseau donné. Contrairement au cas $d \leq 3$, pour $d \geq 4$ il n'est pas toujours possible d'avoir une base composée des "plus courts vecteurs" et ayant comme norme les minima successifs . La notion de base courte ne se généralise pas naturellement en dimension supérieure et nous verrons que plusieurs définitions coexistent.

En pratique, les valeurs $\lambda_i(\mathcal{L})$ ne sont pas nécessairement connues et les calculer est un problème difficile. Le résultat suivant donne une minoration du i-ème minimum successif en fonction des longueurs des vecteurs de la base orthogonalisée de Gram-Schmidt.

Proposition 1.14. *Soit $B^\star = (\mathbf{b}_1^\star, \mathbf{b}_2^\star, \ldots, \mathbf{b}_d^\star)$ la base orthogonalisée de Gram-Schmidt d'une base B du réseau $\mathcal{L}$. Alors,*

$$\forall i = 1 \ldots d, \qquad \lambda_i(\mathcal{L}) \geq \min_{j \in [i..d]} \|\mathbf{b}_j^\star\|.$$

Preuve. Supposons que $(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_i)$ est un système de i vecteurs non nuls linéairement indépendants du réseau $\mathcal{L}$ dont la norme est au plus $\lambda_i(\mathcal{L})$. Pour chaque $j \in [1 \,..\, i]$, $\mathbf{v}_j$ s'exprime dans la base $B = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$. En particulier, nous posons $k(j)$ le plus petit indice k tel que $\mathbf{v}_j$ appartient au sous-espace engendré par $(\mathbf{b}_1, \ldots, \mathbf{b}_k)$. Comme le système $(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_i)$ est linéairement indépendant, il existe au moins un vecteur $\mathbf{v}_j$ tel que $k(j) \geq i$. En reprenant la preuve de la proposition 1.5, nous obtenons $\|v_j\| \geq \|\mathbf{b}_{k(j)}^\star\|$ dont nous pouvons déduire que $\lambda_i(\mathcal{L}) \geq \|v_j\| \geq \min_{k \in [i..d]} \|\mathbf{b}_k^\star\|$.

□

La proposition 1.14 est une généralisation de la proposition 1.5 qui donne cette minoration dans le cas du premier minimum.

1.3 Théorèmes de Minkowski et invariant d'Hermite

La proposition 1.14 donne une minoration des minima successifs en fonction d'une base orthogonalisée de Gram-Schmidt. Pour les problèmes de réduction des réseaux et la recherche des plus courts vecteurs, une majoration est utilisée pour vérifier si une base donnée peut être améliorée ou si elle est déjà "très réduite". Les deux théorèmes de Minkowski présentés dans cette section permettent de majorer simplement les minima successifs d'un réseau.

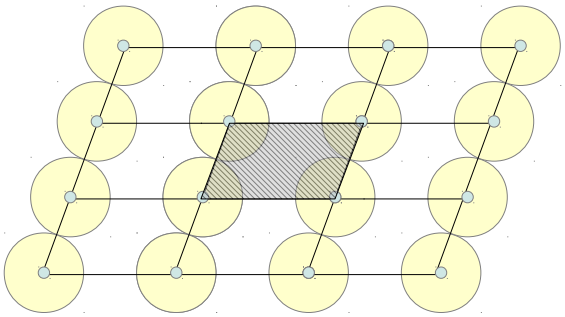


FIGURE 1.6: Densité du remplissage

La figure 1.6 représente un réseau $\mathcal{L}$ sur lequel nous avons dessiné autour de chaque point les boules euclidiennes de diamètre $\lambda_1(\mathcal{L})$. Nous avons aussi dessiné le domaine fondamental du réseau. Il est clair que les boules sont disjointes et leur volume est inférieur au volume du domaine fondamental donné par $\det \mathcal{L}$. Sans être une preuve rigoureuse, ceci nous donne le premier théorème de Minkowski.

Théorème 1.15. *(Théorème 1 de Minkowski) : Soit $\mathcal{L}$ un réseau de dimension d. Alors, le premier minimum $\lambda_1(\mathcal{L})$ du réseau est majoré par*

$$\lambda_1(\mathcal{L}) \leq \rho_d \cdot (\det \mathcal{L})^{(1/d)}$$

où ρ_d est le diamètre de la boule euclidienne de volume 1.

Preuve. Nous utilisons le fait que le volume de la boule euclidienne de diamètre λ_1 est égal à $\left(\frac{\lambda_1}{\rho_d}\right)^d$ et le volume du domaine fondamental est égal à $\det \mathcal{L}$. □

La densité de remplissage d'un réseau est définie comme le rapport du volume d'une boule euclidienne de diamètre $\lambda_1(\mathcal{L})$ avec le volume du réseau $\mathcal{L}$. C'est la densité maximum qui peut être obtenue en remplissant l'espace par des boules disjointes centrées en les points du réseaux. La figure 1.7 montre un réseau dont la densité est maximale.

Comme le volume de la boule euclidienne de diamètre $\lambda_1(\mathcal{L})$ s'écrit comme $\lambda_1(\mathcal{L})^d \cdot V_d$ avec V_d le volume de la boule unité dans $\mathbb{R}^d$, l'invariant d'Hermite défini ci-après est très lié à la racine d-ième de la densité.

Définition 1.16. *(Invariant d'Hermite) L'invariant d'Hermite d'un réseau $\mathcal{L}$ de dimension d est défini par*

$$\gamma(\mathcal{L}) = \left(\frac{\lambda_1(\mathcal{L})}{(\det \mathcal{L})^{(1/d)}}\right)^2.$$

Le premier théorème de Minkowski montre que l'invariant d'Hermite est borné par ρ_d^2 avec $\rho_d \underset{d\to\infty}{\sim} \sqrt{\frac{2d}{\pi e}}$. Ce résultat montre l'existence de la constante d'Hermite que nous introduisons maintenant.

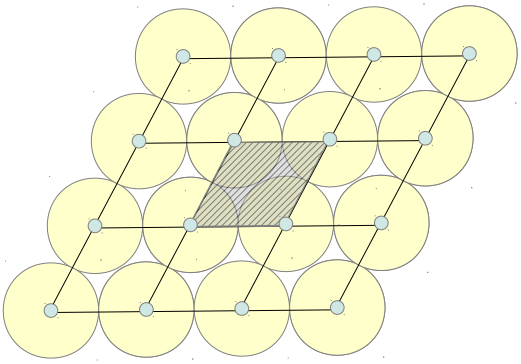


FIGURE 1.7: Empilement des sphères pour une densité du remplissage maximale

Définition 1.17. *(Constante de Hermite) La constante d'Hermite γ_d de rang d est définie par*

$$\gamma_d = \sup\{\gamma(\mathcal{L}) \ : \ \dim(\mathcal{L}) = d\}.$$

Le premier théorème de Minkowski montre que $\gamma_d \leq \rho_d^2$. La constante d'Hermite représente la densité de remplissage maximale d'un empilement régulier de sphères identiques dans l'espace de dimension d disposés sur un réseau (voir Figure 1.7). Notons que le sup de la définition est en fait un maximum et que γ_d^d est un rationnel. En revanche, cette quantité semble difficile à calculer et les valeurs exactes de γ_d sont connues pour $1 \leq d \leq 8$ [73] et $d = 24$ [19] et récapitulées dans le tableau suivant.

d	2	3	4	5	6	7	8	24
γ_d	$\frac{2}{\sqrt{3}}$	$2^{1/3}$	$\sqrt{2}$	$8^{1/5}$	$\left(\frac{64}{3}\right)^{1/6}$	$64^{1/7}$	2	4

Pour les autres valeurs de d, le meilleur encadrement connu est donné par [86] avec

$$\frac{d}{2\pi e} + \frac{\log(\pi d)}{2\pi e} + o(1) \leq \gamma_d \leq \frac{1.744d}{2\pi e}(1 + o(1)).$$

Le second théorème de Minkowski généralise le premier et montre que la moyenne géométrique de tous les minima d'un réseau de dimension d est majorée par une fonction de γ_d et du déterminant du réseau.

Théorème 1.18. *(Théorème 2 de Minkowski) Pour tout réseau $\mathcal{L}$ de dimension d,*

$$\left(\prod_{i=1}^{d} \lambda_i(\mathcal{L})\right)^{\frac{1}{d}} \leq \sqrt{\gamma_d}.\det(\mathcal{L})^{\frac{1}{d}}.$$

Preuve. L'idée de preuve est d'utiliser à la place de la boule euclidienne de diamètre λ_1, des ellipsoïdes disjointes de diamètre $\lambda_1, \lambda_2, \ldots, \lambda_d$ centrées sur les points du réseau.

□

Le second théorème de Minkowski montre qu'une base dont le produit des normes des vecteurs est de l'ordre du volume du réseau est une "bonne base". En pratique, les algorithmes cherchent souvent une base dont le produit des vecteurs est à une constante multiplicative près le volume du réseau. Par exemple, l'algorithme LLL calcule une "bonne base" avec un facteur exponentiel en d.

1.4 Rappels de complexité

Pour discuter de la difficulté des problèmes qui se posent sur les réseaux euclidiens, nous rappelons d'abord quelques définitions de la théorie de la complexité.

1.4.1 Quelques définitions.

Un algorithme $\mathcal{A}$ est une application $\mathcal{A} : \Omega \to \mathcal{S}$, où Ω est l'ensemble des entrées de l'algorithme et $\mathcal{S}$ l'ensemble de ses sorties.

Taille. On définit sur l'ensemble Ω une notion de taille $\tau : \Omega \to \mathbb{N}$, qui formalise (en la simplifiant) la place $\tau(\omega)$ qu'occupe en mémoire une entrée ω de l'algorithme. Ainsi, pour un entier $\omega \in \mathbb{N}$, la "vraie" taille est la taille (binaire) de l'entier, c'est-à-dire le nombre de ses chiffres en base 2. On définira la taille d'un entier ω comme son logarithme, en négligeant la base du logarithme, et la taille d'un rationnel p/q, qui est représenté en machine par le couple (p, q) comme la somme des tailles de son numérateur et de son dénominateur.

Paramètres d'étude. Analyser un algorithme, c'est étudier des paramètres de cet algorithme, qui sont donc des fonctions définies sur l'ensemble Ω des entrées, à valeurs réelles. Par exemple, le nombre d'itérations $c : \Omega \to \mathbb{N}$ de l'algorithme $\mathcal{A}$, ou bien une caractéristique géométrique $g : \Omega \to \mathbb{R}$ de la sortie de l'algorithme $\mathcal{A}$ quand il fonctionne sur l'entrée ω. Ainsi $c(\omega)$ est le nombre d'itérations sur l'entrée ω, et $g(\omega)$ est la caractéristique de la sortie quand l'entrée est ω. Pour les algorithmes de réduction de cette thèse, des caractéristiques géométriques importantes de la sortie sont par exemple le défaut d'orthogonalité, ou les rapports de Siegel.

Analyse. Analyser un algorithme, c'est étudier des paramètres de cet algorithme, en fonction de la taille de l'entrée. Ayant donc défini une notion de taille sur l'ensemble Ω, on classe les entrées par taille, et on considère Ω_N l'ensemble des entrées de taille N. Très souvent, l'ensemble Ω_N est fini (même si l'ensemble Ω de toutes les entrées possibles est, lui, très souvent infini) et pour tout paramètre X, on étudie la restriction X_N sur l'ensemble Ω_N, de manière asymptotique, quand la taille N de l'entrée devient grande.

Analyse dans le pire des cas. La complexité dans le pire des cas est définie par la suite

$$M_N = \max_{\omega \in \Omega_N} X_N(\omega).$$

La complexité d'un algorithme est très souvent décrite dans le pire des cas. Les analyses classiques rappelées dans le chapitre 2 sont toutes des analyses dans le pire des cas.
Les deux paramètres les plus étudiés sont certainement la complexité en temps (le nombre d'étapes élémentaires de calcul avant d'arriver à l'état final) et la complexité en espace (la mémoire utilisée pendant l'exécution de l'algorithme). On s'intéressera ici essentiellement à la complexité en temps. On dit alors qu'un algorithme est polynomial si sa complexité en temps

est bornée par un polynôme en la taille de ses entrées. On s'accorde à dire que les problèmes faciles se résolvent avec des algorithmes polynomiaux alors que les problèmes difficiles sont ceux pour lesquels on ne dispose pas d'algorithme polynomial.

1.4.2 Classes de complexité.

Un problème $P : \Omega \to \mathcal{S}$ est dit polynomial (en temps) s'il existe un algorithme $\mathcal{A} : \Omega \to \mathcal{S}$ qui le résout avec une complexité (en temps) polynomiale. La théorie de la complexité classe les problèmes selon leur "difficulté", et se limite le plus souvent à des problèmes de décision où la réponse attendue est booléenne (oui ou non). Un problème de décision a donc pour ensemble de sortie l'ensemble $\mathcal{S} := \{0, 1\}$. Voici les principales classes de complexité classiques :

- $\mathcal{P}$: la classe des problèmes de décision de complexité en temps polynomiale pour lesquels il existe un algorithme déterministe de complexité polynomiale en temps.
- $\mathcal{NP}$: la classe des problèmes de décision pour lesquels il existe un algorithme de complexité polynomiale qui vérifie une réponse positive "oui" en temps polynomial.
- co$-\mathcal{NP}$: la classe des problèmes de décision pour lesquels il existe un algorithme de complexité polynomiale qui vérifie une réponse négative "non" en temps polynomial

Il existe aussi des classes de complexité probabilistes, que nous décrivons plus informellement encore :

- $\mathcal{BPP}$: la classe des problèmes de décision pour lesquels il existe un algorithme probabiliste polynomial qui, si la solution est "oui", répond "oui" avec une probabilité au moins égale à 2/3, et qui, si la solution est "non" , répond "non" avec une probabilité au moins égale à 2/3.
- $\mathcal{RP}$: la classe des problèmes de décision pour lesquels il existe un algorithme probabiliste polynomial qui, si la solution est "oui", répond "oui" avec une probabilité au moins égale à 1/2, et qui, si la solution est "non" , répond "non" avec une probabilité égale à 1.
- $\mathcal{ZPP}$: la classe des problèmes de décision pour lesquels il existe un algorithme polynomial qui donne une réponse "oui" ou "non" avec une probabilité au moins égale à 1/2, et c'est dans ce cas toujours la bonne réponse. Mais l'algorithme peut répondre aussi "Je ne sais pas" avec une probabilité au plus égale à 1/2.

Il est clair que $\mathcal{P} \subseteq \mathcal{NP}$. Les conjectures les plus célèbres en théorie de la complexité sont :

(i) $\mathcal{P} \neq \mathcal{NP}$
(ii) $\mathcal{P} \neq \mathcal{NP} \cap$ co$-\mathcal{NP}$
(iii) $\mathcal{NP} \cap$ co$-\mathcal{NP} \neq \mathcal{NP}$

On sait aussi que $\mathcal{P} \subseteq \mathcal{BPP}$, mais on ne sait pas si $\mathcal{BPP} \subseteq \mathcal{NP}$. Les problèmes de la classe $\mathcal{BPP}$ sont souvent considérés comme des problèmes résolubles efficacement en pratique.

1.4.3 Réduction entre problèmes.

On dit qu'un problème A se réduit polynomialement à un problème B si, il existe un algorithme polynomial qui peut résoudre le problème A en se servant d'un oracle qui résout le problème B. Un problème de décision A se réduit aléatoirement à un problème B, s'il existe un algorithme polynomial probabiliste, qui se sert d'un oracle qui résout B, et qui donne les réponses suivantes : si la solution est "oui", l'algorithme répond "oui" avec une probabilité au moins égale à 2/3, si la solution est "non", l'algorithme répond "non" avec une probabilité au moins égale à 2/3.

Un problème de décision est dit $\mathcal{NP}$-complet s'il appartient à la classe $\mathcal{NP}$ et si n'importe quel autre problème $\mathcal{NP}$ s'y réduit polynomialement. Pour les problèmes calculatoires, on utilise

la notion de $\mathcal{NP}$-difficile. Un problème est dit $\mathcal{NP}$-difficile si tout problème $\mathcal{NP}$ lui est réductible polynomialement.
Un problème est dit $\mathcal{NP}$- difficile sous des réductions randomisées si tout autre problème de la classe $\mathcal{NP}$ lui est réductible aléatoirement.

L'intérêt de la première conjecture est que aucun problème $\mathcal{NP}$-difficile ou $\mathcal{NP}$-complet ne peut être résolu en temps polynomial. Sous cette conjecture, si un problème est démontré $\mathcal{NP}$-difficile, alors il est dur à résoudre en un temps "réaliste". Cette notion nous sera utile pour la suite pour quantifier la sécurité de différents protocoles cryptographiques.

1.5 Problèmes sur les réseaux euclidiens

Dans cette section, nous nous intéressons à quelques problèmes liés à la géométrie des nombres. Une partie des problèmes sont classés comme algorithmiquement faciles car résolubles en temps polynomial (voir la section précédente 1.4). L'autre partie contient des problèmes difficiles ($\mathcal{NP}$-complets et $\mathcal{NP}$-difficiles) comme le problème $\mathcal{SVP}$ (trouver un plus court vecteur dans un réseau) ou encore $\mathcal{CVP}$ (trouver le vecteur plus proche d'une cible donnée). Nous nous intéressons aussi à des problèmes d'approximation dont certains sont résolubles en temps polynomial pour des facteurs d'approximation donnés.

Les nombres manipulés par les ordinateurs sont des rationnels. Nous ne considérons que des problèmes liés à des réseaux qui sont représentés par une base B rationnelle. Quitte à multiplier tous les coefficients de la matrice par le ppcm de tous les dénominateurs, nous pouvons toujours nous ramener à une matrice à coefficients entiers de $\mathbb{Z}^{d\times n}$. La taille d'un réseau $\mathcal{L}(B)$, représenté par une base $B = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$, est alors donnée par le triplet $(n, d, \log M)$ où n est la dimension de l'espace ambiant, d la dimension du réseau et $M = \max_{i\leq d} \|\mathbf{b}_i\|$. Notons que $\log M$ est l'ordre de grandeur du nombre de bits du plus grand vecteur de la base B. La complexité des algorithmes sera toujours mesurée en fonction du triplet $(n, d, \log M)$.

1.5.1 Problèmes polynomiaux

Nous présentons d'abord quelques problèmes faciles. Ces problèmes se résolvent souvent avec des outils classiques d'algèbre linéaire dont la complexité est polynomiale.

Problème 1.19. (base) *Soit un réseau $\mathcal{L}$ défini par un système de vecteurs non nécessairement indépendants. Donner une base de $\mathcal{L}$*

Il suffit de calculer la forme normale d'Hermite du système donné en entrée qui s'obtient en temps polynomial [78] à partir d'un algorithme du type "pivot de Gauss". Une base $B = (b_{i,j}) \in \mathbb{R}^{d\times n}$ est sous forme d'Hermite si B est triangulaire inférieure, si les éléments de la diagonale sont strictement positifs et si pour tout i et j avec $j < i$, $0 \leq b_{j,j} \leq b_{i,i}$). Si $d \neq n$, les dernières $n - d$ colonnes sont nulles.

Problème 1.20. (appartenance) *Soit un réseau $\mathcal{L}$ donné par une base $B \in \mathbb{Z}^{d\times n}$ et un vecteur $\mathbf{v} \in \mathbb{Z}^n$. Décider si $\mathbf{v}$ appartient au réseau $\mathcal{L}$.*

Il suffit de résoudre le système à d équations et d inconnues suivant

$$\mathbf{x}B = \mathbf{v}$$

et de vérifier si la solution $\mathbf{x} = (x_1, \ldots, x_2)$ est bien entière. Pour résoudre le système, on peut utiliser la méthode du pivot de Gauss de complexité cubique en la dimension.

Problème 1.21. (équivalence) *Soit $B \in \mathbb{Z}^{d\times n}$ et $B' \in \mathbb{Z}^{d\times n}$ deux bases. Décider si ces deux bases engendrent le même réseau.*

Il suffit de calculer la matrice de passage $\mathcal{P}$ et de vérifier si celle-ci est bien unimodulaire.

1.5.2 Problèmes difficiles

Problèmes $\mathcal{CVP}$. Un problème d'une difficulté surprenante consiste à trouver le vecteur d'un réseau le plus proche d'un vecteur cible donné pour la norme euclidienne. Ce problème, appelé $\mathcal{CVP}$ pour *Closest Vector Problem*, existe en deux variantes, selon que l'on souhaite construire ou seulement décider.

Problème 1.22. ($\mathcal{CVP}$) *Étant donnés une base $B \in \mathbb{Z}^{d\times n}$ d'un réseau $\mathcal{L}$ et un vecteur $\mathbf{t} \in \mathbb{Q}^d$, trouver le vecteur de $\mathcal{L}$ le plus proche de $\mathbf{t}$ pour la norme euclidienne.*

Problème 1.23. ($\mathcal{CVP}$ **décisionnel**) *Étant donnés une base $B \in \mathbb{Z}^{d\times n}$ d'un réseau $\mathcal{L}$, un vecteur $\mathbf{t} \in \mathbb{Q}^d$ et une quantité $K \in \mathbb{Q}$, existe-t-il un point $\mathbf{x} \in \mathcal{L}$ vérifiant $\|\mathbf{t} - \mathbf{x}\| \leq K$?*

Le problème $\mathcal{CVP}$ calculatoire se réduit polynomialement au problème $\mathcal{CVP}$ décisionnel [82].

Problèmes $\mathcal{SVP}$. Le problème $\mathcal{SVP}$ pour Shortest Vector Problem, est une variante du problème précédent et consiste à calculer un vecteur du réseau non nul de plus petite norme. Comme le problème précédent, il existe deux versions du problème selon que l'on souhaite construire ou seulement décider.

Problème 1.24. ($\mathcal{SVP}$) *Étant donnée une base $B \in \mathbb{Z}^{d\times n}$ d'un réseau $\mathcal{L}$, déterminer un plus court vecteur non nul du réseau $\mathcal{L}$ pour la norme euclidienne.*

Problème 1.25. ($\mathcal{SVP}$ **décisionnel**) *Étant donnés une base $B \in \mathbb{Z}^{n\times d}$ d'un réseau $\mathcal{L}$ et une quantité $K \in \mathbb{N}^*$, existe-t-il un vecteur $\mathbf{x} \in \mathcal{L}$ non nul, tel que $\|\mathbf{x}\| \leq K$?*

Encore une fois, le $\mathcal{SVP}$ calculatoire se réduit polynomialement au $\mathcal{SVP}$ décisionnel [82]. Le problème $\mathcal{CVP}$ est une généralisation du problème $\mathcal{SVP}$. En effet, le problème $\mathcal{SVP}$ pour une base $B = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$ se ramène à la résolution de d problèmes $\mathcal{CVP}(B_i, \mathbf{i})$ avec $B_i = (\mathbf{b}_1, \ldots, 2\mathbf{b}_i, \ldots, \mathbf{b}_d)$. L'une des solutions retournées est un vecteur le plus court.

Solutions algorithmiques pour $\mathcal{CVP}, \mathcal{SVP}$. Les solutions algorithmiques de ces problèmes se ramènent souvent à la recherche exhaustive de points entiers dans des boules bien choisies. Pour résoudre le problème $\mathcal{SVP}$, il suffit par exemple d'énumérer tous les vecteurs de $\mathcal{L}$ de norme euclidienne inférieure au plus court vecteur de la base donnée. Malheureusement, le nombre de points entiers dans une boule croit exponentiellement avec la dimension du réseau et de manière polynomiale avec le rayon de la boule. La figure 1.8 décrit plusieurs méthodes pour résoudre les problèmes $\mathcal{SVP}$ et $\mathcal{CVP}$. La complexité pour ces algorithmes est exprimée en nombre d'opérations arithmétiques. Chacune de ces opérations a un coût polynomial en la taille en bits du plus grand vecteur de la base à savoir $\log \max_{i\leq d} \|\mathbf{b}_i\|$.

Remarquons que la complexité du meilleur algorithme déterministe est super-exponentielle. L'idée principale de l'algorithme de Kannan, Fincke et Pohst est simple. Considérons une base $B = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau $\mathcal{L}$. Alors il existe un plus court vecteur $\mathcal{L}$ de la forme $\mathbf{x} = x_1\mathbf{b}_1 + \ldots + x_d\mathbf{b}_d$, où les x_i sont entiers et $x_d \geq 0$. On utilise des encadrements pour les x_i en fonction des longueurs des vecteurs $\mathbf{b}_i$ et $\mathbf{b}_i^\star$ et on fait une recherche exhaustive sur $(x_1, \ldots, x_d)$.

Algorithme	Complexité	Description
AKS [5] (Ajtai-Kumar-Sivamumar)	$2^{O(d)}$	Algorithme probabiliste théoriquement le plus efficace. Mais il utilise un espace 2^d : pas utilisable en pratique.
KFP [52],[32] (Kannan-Fincke-Pohst)	$2^{O(d^2)}$	Algorithme appliqué à une base LLL-réduite. L'avantage : n'utilise qu'un espace polynomial.
L'algorithme de Kannan [52],[45]	$d^{\frac{d}{2}+O(d)}$	Algorithme déterministe, utilise KFP récursivement. Moins efficace en pratique que KFP pour les dimensions actuellement utilisées.

FIGURE 1.8: Algorithmes pour résoudre les problèmes $\mathcal{SVP}$ et $\mathcal{CVP}$.

Si la base B est déjà réduite par l'algorithme LLL (voir Chapitre 2), le cardinal de l'ensemble parcouru est majoré par $2^{O(d^2)}$. Une version récursive de Kannan ramène le nombre d'opérations effectuées à $d^{\frac{d}{2}+O(d)}$.

1.5.3 Problèmes approchés

Comme nous le verrons ans la section suivante 1.5.4, les problèmes $\mathcal{SVP}$ et $\mathcal{CVP}$ sont difficiles à résoudre de manière exacte. On s'intéresse donc ici aux problèmes d'approximation associés.

Problème 1.26. ($\mathcal{CVP}_\alpha$) *Soit un réseau $\mathcal{L}$, un facteur d'approximation $\alpha \geq 0$ et un vecteur cible $\mathbf{t}$. Trouver un vecteur $\mathbf{v} \in \mathcal{L}$ tel que pour la norme euclidienne, $\|\mathbf{t}-\mathbf{v}\| \leq \alpha\|\mathbf{t}-\mathbf{w}\|$, pour tout vecteur $\mathbf{w} \in \mathcal{L}$.*

Problème 1.27. ($\mathcal{SVP}_\alpha$) *Soit un réseau $\mathcal{L}$ et un facteur d'approximation $\alpha \in \mathbb{R}^+$. Trouver un vecteur non nul $\mathbf{v} \in \mathcal{L}$ t.q. $\|\mathbf{v}\| \leq \alpha \cdot K$ (K représente souvent le premier minimum $\lambda_1(\mathcal{L})$ mais pas tout le temps).*

Les problèmes $\mathcal{SVP}_\alpha$ et $\mathcal{CVP}_\alpha$ peuvent être aussi définis pour la norme infinie, et nous les noterons alors $\mathcal{SVP}_\alpha^\infty$ et $\mathcal{CVP}_\alpha^\infty$.

Pour s'approcher du vecteur le plus court du réseau, il est naturel de choisir pour K le premier minimum $\lambda_1(\mathcal{L})$. Mais il peut s'avérer difficile de vérifier la solution de ce problème, car $\lambda_1(\mathcal{L})$ n'est pas toujours connu exactement. Hermite introduit une autre définition des problèmes approchés en considérant $K = \det^{1/d}(\mathcal{L})$. En fait ces deux problèmes sont presque équivalents. Si le problème peut être résolu pour $K = \lambda_1(\mathcal{L})$ avec un facteur α, le second problème avec $K = \det^{1/d}(\mathcal{L})$ peut être résolu avec un facteur $\alpha\sqrt{d}$. Réciproquement, chaque algorithme qui résout le deuxième problème avec un facteur α, peut être utilisé plusieurs fois linéairement pour résoudre le premier problème avec un facteur α^2 en temps polynomial [69].
L'algorithme LLL décrit dans le prochain chapitre résout le problème $\mathcal{SVP}_\alpha$ pour $K = \det^{1/d}(\mathcal{L})$ en temps polynomial pour un facteur d'approximation exponentiel en la dimension du réseau.

1.5.4 Difficulté des problèmes de réseaux euclidiens.

En 1981 van Emde Boas [114] démontre le théorème suivant.

Théorème 1.28. *Le problème $\mathcal{CVP}$ décisionnel du plus proche vecteur est $\mathcal{NP}$–complet.*

Dans le même papier, l'auteur a prouvé que $\mathcal{SVP}^{\infty}$ (pour la norme infinie) est $\mathcal{NP}$–difficile et il a conjecturé que le problème $\mathcal{SVP}$ pour n'importe quelle norme est également $\mathcal{NP}$-difficile. La conjecture a été prouvée pour la norme euclidienne en 1998 par Ajtai [3] mais avec des réductions randomisées.

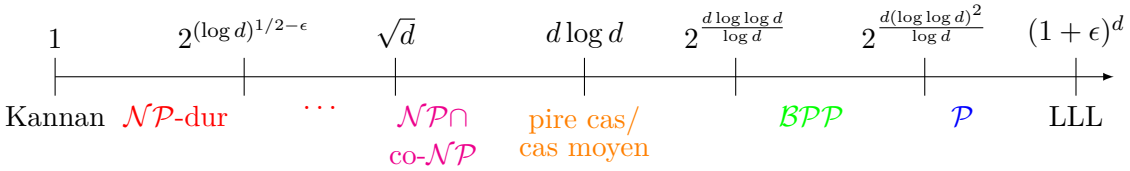
Théorème 1.29. *Le problème $\mathcal{SVP}$ décisionnel est $\mathcal{NP}$-complet sous des réductions randomisées. [3]*

En 2001, Micciancio [79] a amélioré ce résultat en démontrant que le problème approché $\mathcal{SVP}_\alpha$ reste $\mathcal{NP}$-difficile sous des réductions randomisées pour un facteur approximation α inférieur à $\sqrt{2}$. En 2004, Khot [56] a généralisé le résultat pour n'importe quelle constante. Il prouve aussi que $\mathcal{SVP}_\alpha$ reste $\mathcal{NP}$-difficile sous des réductions randomisées pour un facteur quasi-polynomial $2^{(\log d)^{1/2-\epsilon}}$ avec $\epsilon > 0$.

En augmentant le facteur d'approximation, le problème devient moins difficile. Goldreich et Goldwasser [41] ont montré que $\mathcal{SVP}_\alpha$ et $\mathcal{CVP}_\alpha$ ne sont plus des problèmes $\mathcal{NP}$-difficiles pour $\alpha = \sqrt{d}$, à moins que la hiérarchie polynomiale ne s'effondre. Ensuite Regev [1] a démontré que le problème devient $\mathcal{NP} \cap$ co-$\mathcal{NP}$ pour $\alpha = \sqrt{d}$. Pour la norme infinie, les problèmes ne sont plus $\mathcal{NP}$-difficiles pour un facteur $d/O(\log d)$. Enfin le problème devient facile pour des valeurs de α proches de 2^d. En fait Ajtai, Kumar et Sivakumar [5] ont démontré que pour $\alpha = 2^{d \log\log d/(\log d)}$, les problèmes $\mathcal{SVP}_\alpha$ et $\mathcal{CVP}_\alpha$ peuvent être résolus en temps probabiliste polynomial et Schnorr [99] a ensuite donné un algorithme déterministe polynomial pour $\alpha = 2^{d(\log\log d)^2/(\log d)}$.

La sécurité des protocoles cryptographiques repose sur la difficulté en moyenne de problèmes algorithmiques et non sur leur complexité dans le pire des cas. En effet, le protocole doit rester sûr avec la plupart de ses clés (i.e., en moyenne) et non sur quelques clés seulement (autrement dit le pire des cas). En 1996 Ajtai [2] est le premier a montré qu'il existe des réductions pire cas vers le cas moyen en utilisant des problèmes liés aux réseaux. Il a démontré que si le problème $\mathcal{SVP}$ peut être résolu en moyenne dans une vaste classe de réseaux, alors on peut résoudre le problème approché avec un facteur d'approximation $d \log d$. Nous y reviendrons en détail au chapitre 3 (section 3.2).

Nous résumons tous les résultats précédents liés à la complexité du problème $\mathcal{SVP}_\alpha$ en fonction du facteur d'approximation α dans le schéma suivant.



L'algorithme LLL [65] et ses variantes par blocs [99, 101, 33] résolvent le problème $\mathcal{SVP}_\alpha$ pour un facteur d'approximation exponentiel en la dimension de la forme $\alpha = (1+\epsilon)^d$, le tout en temps polynomial. La valeur de la constante ϵ dépend de l'algorithme et des paramètres choisis. Dans le prochain chapitre (chapitre 2), nous présentons en détail l'algorithme LLL (voir section 2.3).

Chapitre 2

Réduction de réseaux

Sommaire

Ce chapitre est consacré à un problème algorithmique central dans le domaine des réseaux euclidiens, celui de la réduction. Réduire un réseau, c'est construire une base du réseau, avec de "bonnes" propriétés euclidiennes. Il existe plusieurs notions de réduction, qui dépendent des propriétés précises qu'on attend d'une bonne base. Ce chapitre décrit dans un premier temps les principaux processus de réduction, qui recherchent tous un compromis entre la qualité de la base obtenue et le temps passé à l'obtenir, en tentant de généraliser ce qui est connu en petite dimension. C'est pourquoi nous revenons aux algorithmes en ces petites dimensions, où les processus de réduction, en qualité et en complexité, sont en quelque sorte optimaux : c'est l'algorithme d'Euclide en dimension 1(Section 2.2.1), et l'algorithme de Gauss en dimension 2 (Section 2.2.2), qui peut être considéré lui-même comme une extension à la dimension 2 de

l'algorithme d'Euclide. Puis, dans la Section 2.3, nous revenons à une dimension quelconque, et présentons en détail l'algorithme LLL, qui est un des acteurs principaux de cette thèse. Nous décrivons les principaux paramètres de cet algorithme (nombre d'itérations, configuration de sortie), dans le cadre usuel, qui est celui du "pire des cas". Nous concluons en décrivant des applications diverses de la réduction des réseaux.

2.1 Le problème de la réduction

Un réseau possède une infinité des bases, qui sont toutes équivalentes d'un point de vue algébrique. Mais d'un point de vue euclidien, ce n'est plus le cas, et certaines de ces bases ont des propriétés euclidiennes plus intéressantes. L'objectif de la réduction est de trouver en un temps "raisonnable" une base avec d'"assez bonnes" propriétés euclidiennes, formée de vecteurs assez orthogonaux, et suffisamment courts pour donner des approximations pour les minima successifs. Mais, comme on l'a déjà vu, et à partir de la dimension 5, les minima successifs ne forment pas nécessairement une base du réseau. Il est donc difficile de trouver un critère absolu qui définisse ce qu'est une bonne base. Plusieurs notions de réduction existent, et chacune correspond à une notion de qualité de la base réduite. Évidemment, le temps passé à obtenir une telle base dépend lui aussi de la notion de réduction. Les principales réductions sont : la réduction au sens de Korkine et Zolotarev [58, 48], la réduction au sens de Lenstra, Lenstra et Lovász (LLL) [65], la réduction par blocs de Schnorr [102, 101]. Une notion de réduction opère un compromis entre la qualité de la réduction et la complexité pour l'obtenir. Par exemple, la réduction au sens de Korkine et Zolotarev produit une base dont la qualité est bien supérieure à celle qui est produite par la réduction au sens de LLL, mais le temps de calcul pour l'obtenir est bien plus important.

2.1.1 Orthogonalisation de Gram-Schmidt

Nous revenons d'abord sur l'orthogonalisation de Gram-Schmidt, (OGS en abrégé) et nous donnons plus de détails sur l'algorithme (voir figure 2.1).

Soit une suite $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$ de vecteurs linéairement indépendants de $\mathbb{R}^n$. Le procédé d'orthogonalisation de Gram-Schmidt (OGS) construit la suite $(\mathbf{b}_1^\star, \mathbf{b}_2^\star, \ldots, \mathbf{b}_d^\star)$, où $\mathbf{b}_i^\star$ est la projection orthogonale de $\mathbf{b}_i$ orthogonalement au sous-espace vectoriel engendré par $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{i-1})$. Notons que ce sous-espace est le même que le sous-espace vectoriel engendré par les vecteurs orthogonaux $(\mathbf{b}_1^\star, \mathbf{b}_2^\star, \ldots, \mathbf{b}_{i-1}^\star)$. De plus, la norme $\|\mathbf{b}_i^\star\|$ est la distance entre $\mathbf{b}_i$ et ce sous-espace. Les vecteurs $\mathbf{b}_i^\star$ et les $m_{i,j}$ se calculent facilement par récurrence sur i (voir chapitre 1, section 1.1.2).

Cet algorithme calcule a la fois la base orthogonalisée $B^\star$ et la matrice de passage $\mathcal{P} := (m_{i,j})$. Lorsque la base B est entière de "longueur" $M := \max \|\mathbf{b}_i\|$, le coût total de l'algorithme en nombres d'opérations binaires est en $O(d^4 n \log^2 M)$. La taille binaire de l'entrée est $O(d \log M)$, la taille des rationnels utilisés est en $O(d \log M)^2$ il y a d^2 boucles imbriquées, et le facteur n provient du calcul du produit scalaire n.

L'orthogonalisation de Gram-Schmidt permet de trianguliariser la représentation matricielle d'un réseau. En effet, la famille $(\mathbf{b}_1^\star, \mathbf{b}_2^\star, \ldots, \mathbf{b}_d^\star)$ est une base orthogonale de $\mathbb{R}^d$ et la matrice $\mathcal{P}$ qui représente la base B en fonction de la base orthogonale $B^\star$ est une matrice triangulaire inférieure, qui possède des 1 sur la diagonale.

Entrées : Une base B
Résultat : La base $B^\star$ orthogonalisée de B
$\mathbf{b}_1^\star = \mathbf{b}_1$;
pour i *de* 2 *à* d **faire**
  pour j *de* 1 *à* $i-1$ **faire**
    $m_{i,j} \leftarrow \dfrac{\langle \mathbf{b}_i | \mathbf{b}_j^\star \rangle}{\|\mathbf{b}_j^\star\|^2}$;
  finpour
  $\mathbf{b}_i^\star \leftarrow \mathbf{b}_i - \sum_{j=1}^{i} m_{i,j}\, \mathbf{b}_j^\star$;
finpour

FIGURE 2.1: L'algorithme pour calculer l' orthogonalisée de Gram-Schmidt (OGS)

$$\mathcal{P} := \begin{array}{c} \\ \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_d \end{array} \begin{array}{c} \begin{array}{cccc} \mathbf{b}_1^\star & \mathbf{b}_2^\star & \dots & \mathbf{b}_d^\star \end{array} \\ \begin{pmatrix} 1 & 0 & \dots & 0 \\ m_{2,1} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ m_{d,1} & m_{d,2} & \dots & 1 \end{pmatrix} \end{array}.$$

Dans toute la suite, la suite des longueurs $(\|\mathbf{b}_1^\star\|, \|\mathbf{b}_2^\star\|, \dots, \|\mathbf{b}_d^\star\|)$ jouera un rôle essentiel, et on introduit les notations suivantes

$$\ell_i := \|\mathbf{b}_i^\star\|, \qquad r_i := \frac{\ell_{i+1}}{\ell_i}. \tag{2.1}$$

Il y a une relation simple entre la longueur du vecteur $\mathbf{b}_i$ et les longueurs ℓ_j pour $j \leq i$,

$$\|\mathbf{b}_i\|^2 = \|\mathbf{b}_i^\star\|^2 + \sum_{j=1}^{i-1} m_{i,j}^2 \|\mathbf{b}_j^\star\|^2 = \ell_i^2 + \sum_{j=1}^{i-1} m_{i,j}^2\ \ell_j^2. \tag{2.2}$$

2.1.2 La propreté d'une base

Comme un réseau n'admet pas en général de base orthogonale, la base orthogonale de Gram-Schmidt sert de référence. La première idée est donc de rapprocher le vecteur $\mathbf{b}_i$ du vecteur $\mathbf{b}_i^\star$, en diminuant la valeur absolue des coefficients $m_{i,j}$ de la matrice $\mathcal{P}$, et ce, sans que ni le réseau $\mathcal{L}$ ni la base $B^\star$ ne soient modifiés. Cela justifie la définition suivante :

Définition 2.1. *On dit qu'une base $B = (\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_d)$ est propre si la matrice de passage $\mathcal{P}$ de B à son orthogonalisée $B^\star$ a tous ses coefficients $m_{i,j}$ non diagonaux (pour tout couple (i,j) vérifiant $1 \leq j < i \leq d$) qui sont en valeur absolue au plus égaux à 1/2.*

Il est possible de transformer une base quelconque d'un réseau en une base propre du même réseau en utilisant l'algorithme décrit dans la Figure 2.2. Cette notion de réduction, appelée encore réduction faible, a été introduite par Hermite [48].

Entrées : une base $B = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$
Résultat : une base Propre
Calculer la base OGS;
pour *i de* 2 *à* d **faire**
  pour *j de* $i-1$ *à* 1 **faire**
    $\mathbf{b}_i \leftarrow \mathbf{b}_i - \lfloor m_{i,j} \rceil \mathbf{b}_j$;
    [$\lfloor m_{i,j} \rceil$ l'entier le plus proche de $m_{i,j}$];
    pour *k de* 1 *à* j **faire**
      $m_{i,k} \leftarrow m_{i,k} - \lfloor m_{i,j} \rceil m_{j,k}$;
      [mettre à jour la matrice $\mathcal{P}$];
    finpour
  finpour
finpour

FIGURE 2.2: L'algorithme `Propre`

L'algorithme translate $\mathbf{b}_i$ parallèlement à chaque vecteur $\mathbf{b}_j$ pour $j < i$ et ne modifie donc ni $\mathbf{b}_i^\star$ ni le réseau $\mathcal{L}$. il ne modifie que les coefficients $m_{i,j}$ en leur soustrayant leur entier le plus proche. A la fin de cet algorithme, pour la nouvelle base rendue propre, tous les coefficients $m_{i,j}$ (pour $i > j$) sont dans l'intervalle $[-1/2, 1/2]$. L'algorithme est bien polynomial en la taille de la donnée, de complexité en $O(d^4 n \log^2 M)$.

2.1.3 Défaut d'orthogonalité et défaut de longueur d'une base

La procédure `Propre` permet de raccourcir "au mieux" et "simplement" les vecteurs $\mathbf{b}_i$ en les rapprochant de $\mathbf{b}_i^\star$. Mais cette procédure n'est pas suffisante pour rendre la base de "bonne qualité". Une base propre n'est pas nécessairement "presque orthogonale", car la procédure `Propre` assure seulement que la projection de chaque $\mathbf{b}_i$ sur le sous-espace vectoriel H_{i-1} engendré par $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$ est assez petite. Il faut une autre condition qui minore l'angle qui forme le vecteur $\mathbf{b}_i$ avec l'hyperplan H_{i-1}. Pour minorer cet angle, il est utile de minorer la longueur de $\mathbf{b}_i^\star$. La figure 2.6 illustre cette situation en dimension 3.

La qualité d'une base peut être mesurée par les trois paramètres définis comme suit :

Définition 2.2. *Soit $B = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$ une base d'un réseau $\mathcal{L}$. Les paramètres suivants, le défaut d'orthogonalité $\delta(B)$, les défauts de longueur $\mu_i(B)$ et le défaut d'Hermite $\gamma(B)$, mesurent la qualité de la base,*

$$\delta(B) = \prod_{i=1}^{d} \frac{\|\mathbf{b}_i\|}{\|\mathbf{b}_i^\star\|}, \qquad \mu_i(B) = \frac{\|\mathbf{b}_i\|}{\lambda_i(\mathcal{L})}, \qquad \gamma(B) = \left(\frac{\|\mathbf{b}_1\|}{(\det B)^{1/d}}\right)^2. \tag{2.3}$$

Le défaut d'orthogonalité est toujours supérieur à 1, et vaut 1 si et seulement si la base est orthogonale. La base sera "presque" orthogonale si $\delta(B)$ est proche de 1. Une base sera "presque" minimale si tous ses défauts de longueur $\mu_i(B)$ sont proches de 1.

Dans un réseau de dimension $d \leq 4$, il existe une base qui "réalise" les minima successifs, appelée base minimale du réseau. C'est alors la "meilleure base" du réseau. Mais ce n'est pas vrai pour des dimensions supérieures ou égales à 5. Il existe plusieurs notions de réductions,

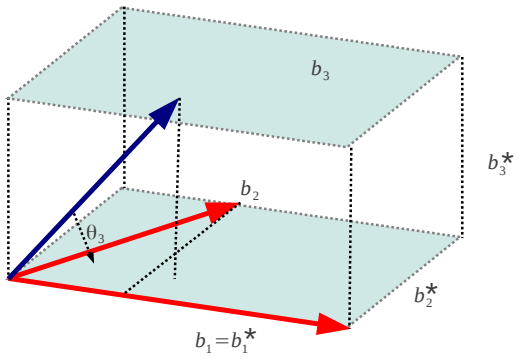


FIGURE 2.3: Base orthogonale pour $d = 3$.

celui de Minkowski qui cherche à minimiser les défauts de longueurs qui est la plus naturelle, et encore trois autres de Korkine-Zolotarev, de Siegel et de Lovász qui cherchent à minimiser les $\mathbf{b}_i^\star$ et $\mathbf{b}_i$ en même temps.

2.1.4 Réduction au sens de Minkowski

Une base $B = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau $\mathcal{L}$ est réduite au sens de Minkowski si les conditions suivantes sont vérifiées :

(i) le vecteur b_1 est un plus court vecteur du réseau $\mathcal{L}$

(ii) le vecteur b_{i+1} est un vecteur le plus court parmi tous les vecteurs indépendants des vecteurs $(\mathbf{b}_1, \ldots, \mathbf{b}_i)$, de telle sorte que $(\mathbf{b}_1, \ldots, \mathbf{b}_{i+1})$ peut être étendu à une base de $\mathcal{L}$.

De manière équivalente, une base est réduite au sens de Minkowski si les inégalités suivantes sont satisfaites :

$$\forall i \leq d, \qquad \|x_1\mathbf{b}_1 + \ldots + x_d\mathbf{b}_d\| \geq \|\mathbf{b}_i\| \tag{2.4}$$

pour tous les d-uplets d'entiers $(x_1, \ldots, x_d)$ formés par des entiers $x_i, \ldots, x_d$ premiers entre eux.

Le choix de $\mathbf{b}_i$ dépend a priori des vecteurs $\mathbf{b}_j$ qui ont été choisis précédemment pour $j \leq i$. La construction nécessite la vérification d'un ensemble de conditions, dont le cardinal explose très rapidement. Comme une base réduite au sens de Minkowski atteint toujours ses quatre premiers minima successifs, cette réduction est optimale jusqu'à la dimension 4. Une base de réseau de dimension d qui atteint ses d minima successifs est réduite au sens de Minkowski, mais l'inverse n'est pas toujours vrai. Le défaut d'orthogonalité d'une telle base est borné par une constante exponentielle en d^2 et le i-e défaut $\mu_i(B)$ est exponentiel en i.

Le défaut d'orthogonalité et les défauts de longueur d'une base réduite au sens de Minkowski vérifient :

$$\mu_i(B) \leq \left(\sqrt{\frac{4}{3}}\right)^i, \qquad \delta(\widehat{B}) \leq \left[d \cdot \left(\frac{5}{4}\right)^d\right]^{d/2}. \tag{2.5}$$

On ne connaît pas d'algorithme qui permet de réduire une base au sens de Minkowski, en un temps polynomial.

2.1.5 Réduction HKZ [Hermite, Korkine, Zolotarev]

Une autre réduction classique est celle de Hermite-Korkine-Zolotarev [HKZ]. Une base $B = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau $\mathcal{L}$ est réduite au sens de HKZ si les conditions suivantes sont vérifiées :

(i) la base est propre

(ii) le vecteur b_1 est un plus court vecteur du réseau $\mathcal{L}$

(iii) le système $(\mathbf{b}'_2, \ldots, \mathbf{b}'_d)$, formée par les projections des vecteurs $\mathbf{b}_i$ orthogonalement au vecteur $\mathbf{b}_1$ est une base HKZ-réduite du réseau $\mathcal{L}'$ engendré par les projections v' des vecteurs $v \in \mathcal{L}$ orthogonalement au vecteur $\mathbf{b}_1$.

Une base HKZ-réduite fournit une très bonne approximation des minima successifs [72, 61], puisqu'on connaît l'encadrement suivant pour les défauts de longueur :

$$\frac{4}{i+3} \leq \mu_i^2(B) := \frac{\|\mathbf{b}_i\|^2}{\lambda_i(\mathcal{L})^2} \leq \frac{i+3}{4}, \qquad \text{pour } 1 < i \leq d. \tag{2.6}$$

La borne supérieure est essentiellement due à Mahler [72] et la borne inférieure est démontrée dans [61]. Dans le même article, les auteurs montrent que ces bornes sont "presque" atteintes. Cette réduction apparaît être la meilleure tant pour le défaut d'orthogonalité que les défauts de longueurs. En revanche, comme la réduction de Minkowski, on ne connaît pas d'algorithme qui permet de réduire une base au sens de HKZ en temps polynomial.

On peut effectuer la réduction HKZ comme suit : On commence par obtenir le vecteur $\mathbf{b}_1$ à l'aide des algorithmes de plus court vecteur, soit l'algorithme déterministe de Kannan [52], de complexité super exponentielle, soit l'algorithme probabiliste d'Ajtai, Kumar et Sivakumar [5] de complexité exponentielle. On complète ensuite $\mathbf{b}_1$ en une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ de $\mathcal{L}$, qu'on projette en une base $(\mathbf{b}'_2, \ldots, \mathbf{b}'_d)$ orthogonalement au vecteur $\mathbf{b}_1$. On détermine alors un plus court vecteur $\mathbf{b}'_2$ du réseau projeté, qu'on relève en un vecteur $\mathbf{b}_2$ de $\mathcal{L}$, en ajoutant un multiple approprié de $\mathbf{b}_1$ de manière à ce que $(\mathbf{b}_1, \mathbf{b}_2)$ soit propre. On complète $(\mathbf{b}_1, \mathbf{b}_2)$ en une base et on répète le processus.

Les deux réductions suivantes ont l'avantage d'être obtenues au bout d'un temps polynomial, et, même si la qualité de la base obtenue n'est pas aussi bonne que dans le cas de Minkowski ou de HKZ, elle reste raisonnable et suffisante pour résoudre des problèmes de type très divers.

2.1.6 Réduction au sens de Siegel

Pour obtenir un défaut d'orthogonalité assez faible, on peut, comme on l'a déjà mentionné, minorer la projection de $\mathbf{b}_i$ orthogonalement à l'hyperplan H_{i-1} engendré par le système $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$ (voir Figure 2.6). La condition suivante fait intervenir les longueurs ℓ_i et les rapports r_i, appelés rapports de Siegel, et définis en (2.1). C'est Ali Akhavi, dans sa thèse, qui a donné le nom de Siegel à cette condition.

Soit s un réel, qui vérifie $s \geq 2/\sqrt{3}$. Une base $B = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau $\mathcal{L}$ est réduite au sens de s–Siegel si les conditions suivantes sont vérifiées :

(i) la base est propre

(ii) la base $B^\star$ vérifie la condition de Siegel, i.e., elle satisfait, pour tout $i \in [1\,..\,d-1]$,

$$\mathcal{S}_s(i) : \qquad \|\mathbf{b}^\star_{i+1}\| \geq \frac{1}{s}\,\|\mathbf{b}^\star_i\|, \qquad \text{ou encore} \quad \ell_{i+1} \geq \frac{1}{s}\,\ell_i \qquad \text{ou encore} \quad r_i \geq \frac{1}{s}. \tag{2.7}$$

Ces conditions assurent une bonne qualité de la base et permettent de majorer les défauts de longueurs, d'orthogonalité et le défaut d'Hermite.

Proposition 2.3. *Soit s un réel qui vérifie $s \geq 2/\sqrt{3}$. Une base B s-réduite au sens de Siegel vérifie :*

$$\delta(B) \leq s^{\frac{d(d-1)}{2}}, \quad \gamma(B) = \frac{\|\mathbf{b}_1\|^2}{\det(\mathcal{L})^{2/d}} \leq s^{d-1}, \quad \mu_i(B) \leq s^{d-1} \quad \textit{pour } 1 \leq i \leq d. \tag{2.8}$$

Preuve. Si la base vérifie la condition de Siegel pour le paramètre s, alors,

$$\|\mathbf{b}_j^\star\| \leq s^{i-j}\|\mathbf{b}_i^\star\| \qquad \text{pour} \qquad 1 \leq j \leq i \leq d,$$

et le vecteur $\mathbf{b}_i$ satisfait donc

$$\|\mathbf{b}_i\|^2 = \left|\sum_{j=1}^{i} m_{i,j}\mathbf{b}_j^\star\right|^2 \leq \|\mathbf{b}_i^\star\|^2 \left[\frac{1}{4}\sum_{j=1}^{i-1} s^{2(i-j)} + 1\right] = \|\mathbf{b}_i^\star\|^2 \left[\frac{s^2}{4}\frac{s^{2(i-1)}-1}{s^2-1} + 1\right].$$

Quand s vérifie la condition $s \geq 2/\sqrt{3}$, on déduit les inégalités suivantes

$$\|\mathbf{b}_i\|^2 \leq s^{2(i-1)}\|\mathbf{b}_i^\star\|^2,$$

puis la majoration pour le défaut d'orthogonalité,

$$\delta(B) = \prod_{i=1}^{d} \frac{\|\mathbf{b}_i\|}{\|\mathbf{b}_i^\star\|} \leq \prod_{i=1}^{d} s^{i-1} = s^{d(d-1)/2}.$$

Si k est un indice pour lequel $\|\mathbf{b}_k^\star\| = \min_{j\in[1..d]} \|\mathbf{b}_j^\star\|$, alors le défaut de longueur vérifie

$$\mu_i(\widehat{B}) = \frac{\|\mathbf{b}_1^\star\|}{\lambda_1(\mathcal{L})} \leq \frac{\|\mathbf{b}_1^\star\|}{\|\mathbf{b}_k^\star\|} \leq s^{k-1} \leq s^{d-1}.$$

Enfin, pour le défaut d'Hermite, on obtient

$$\gamma(B) = \frac{\|\mathbf{b}_1\|^2}{\det(\mathcal{L})^{2/d}} = \|\mathbf{b}_1\|^2 \left(\prod_{i=1}^{d} \frac{1}{\|\mathbf{b}_i^\star\|}\right)^{2/d} \leq \|\mathbf{b}_1\|^2 \left(\prod_{i=1}^{d} \frac{s^{i-1}}{\|\mathbf{b}_1^\star\|}\right)^{2/d} = s^{d-1}.$$

□

2.1.7 Réduction au sens de Lovász

On se pose maintenant la question : Comment construire une base qui vérifie la condition de Siegel ? En dimension 2, l'algorithme de t-Gauss (voir 2.11) permet d'assurer une condition un peu plus forte que la condition de Siegel. L'idée de Lovász a été de généraliser cette condition sur la "boîte locale" B_i. La boîte locale B_i est le système $(\mathbf{u}_i, \mathbf{v}_i)$ formé par les projections $\mathbf{u}_i, \mathbf{v}_i$ de $\mathbf{b}_i$ et $\mathbf{b}_{i+1}$ sur l'orthogonal de l'espace vectoriel H_{i-1} engendré par $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$ dans l'espace vectoriel H_{i+1} engendré par $(\mathbf{b}_1, \ldots, \mathbf{b}_{i+1})$.

Soit t un réel, qui vérifie $t > 1$. Une base $B = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau $\mathcal{L}$ est réduite au sens de t– Lovász si les conditions suivantes sont vérifiées :

(*i*) la base est propre

(*ii*) elle vérifie les conditions de t-Lovász, i.e., pour tout $i \in [1\,..\,d-1]$,

$$\mathcal{L}_t(i) : \qquad \|\mathbf{v}_i\|^2 \geq \frac{1}{t}\|\mathbf{u}_i\| \qquad \text{ou encore} \quad \|\mathbf{b}_{i+1}^\star\|^2 + m_{i+1,i}^2\|\mathbf{b}_i^\star\|^2 \geq \frac{1}{t^2}\|\mathbf{b}_i^\star\|^2. \tag{2.9}$$

La condition de Lovász se traduit sur les rapports de Siegel et les coefficients sous-diagonaux $m_{i+1,i}$ par les conditions : $r_i^2 \geq (1/t^2) - m_{i+1,i}^2$. Quand les deux paramètres de Siegel et Lovász s et t sont reliés par l'égalité suivante,

$$\frac{1}{s^2} = \frac{1}{t^2} - \frac{1}{4}, \tag{2.10}$$

alors une base qui est t-réduite au sens de Lovász est s-réduite au sens de Siegel. En effet, les deux conditions (condition de propreté sur le coefficient $m_{i+1,i}$, et condition de Lovász) entrainent la condition de Siegel. Il existe un algorithme qui construit une base réduite au sens de t-Lovász, et qui fonctionne en temps polynomial pour $t > 1$. C'est l'algorithme dû à Lenstra, Lenstra et Lovász [65], qui est décrit et étudié dans la Section 2.3.

2.1.8 Réduction par blocs BKZ

Les bases réduites au sens de Hermite-Korkine-Zolotarev sont de bonne qualité, mais elles se calculent en temps super-exponentiel. En revanche, les bases LLL-réduites s'obtiennent en temps polynomial, mais les minima successifs peuvent être exponentiellement éloignés des vecteurs de la base. En 1987, Schnorr [99] propose un nouveau type de réduction, dit "par blocs", qui est une réduction intermédiaire entre la réduction LLL et la réduction HKZ. La taille des blocs β est fixée dans l'algorithme, mais peut varier, et, selon le choix de la taille de bloc, on retrouve la notion de LLL-réduction pour $\beta = 2$ ou celle de HKZ pour $\beta = d$.

Soit $\beta \in [2\,..\,d]$. On dit qu'une base B est β-BKZ-réduite si les conditions suivantes sont vérifiées :

(i) La base est propre

(ii) Pour tout $i \in [\beta + 1\,..\,d]$, la boîte de dimension β formée par les projections $\mathbf{b}'_j$ des vecteurs $\mathbf{b}_j$ (pour $j \in [i - \beta + 1\,..\,i]$) orthogonalement au sous espace vectoriel $H_{i-\beta}$ engendré par les vecteurs $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-\beta})$, est HKZ-réduite

Une base β-BKZ-réduite approche le premier minimum avec un facteur d'approximation $\beta^{O(\frac{d-1}{\beta-1})}$. Schnorr propose un algorithme qui calcule une base β-BKZ-réduite en temps $\beta^{O(\beta)}$ en utilisant les algorithmes déterministes de Kannan et Helfrich [52] et en temps $2^{O(\beta)}$ avec l'algorithme probabiliste d'Ajtai, Kumar et Sivakumar [5].

2.2 Algorithmes de réduction de réseaux en petites dimensions

Nous présentons d'abord les algorithmes de réduction en dimension 1 et en dimension 2. En dimension 1, le premier minimum du réseau engendré par un couple (u, v) d' entiers est leur plus grand commun diviseur et il fournit une base pour ce réseau unidimensionnel. L'algorithme qui calcule le pgcd des deux nombres est l'algorithme dû à Euclide. Il est connu depuis 300 av. J.-C., et c'est le plus ancien algorithme connu. En dimension 2, c'est l'algorithme de Gauss [36] qui résout exactement le problème de la réduction de réseaux. Il généralise la variante centrée de l'algorithme d'Euclide, décrite dans la section suivante. Et enfin l'algorithme LLL en dimension $d \geq 3$ peut être vu comme une suite de deux types d'opérations : rendre la base propre et réduction de Gauss sur des bases locales.

2.2.1 Dimension 1 : L'algorithme d'Euclide

Un réseau de dimension 1 inclus dans $\mathbb{Z}$ peut s'écrire sous la forme $\mathbb{Z}x$ avec x entier positif. Les bases de ce réseau sont alors x et $-x$. Quand le réseau est donné sous la forme $\mathcal{L}(u, v) := \mathbb{Z}u + \mathbb{Z}v$, le plus grand commun diviseur des deux entiers u et v fournit une base de ce réseau. En effet,

Entrées : Une paire d'entiers (u, v) positifs.
Résultat : Le pgcd du couple (u, v)
si $u \leq v$ **alors**
| échanger u et v;
finsi
tant que $v \neq 0$ **faire**
| $m \leftarrow \lfloor u/v \rceil, \epsilon \leftarrow \text{Signe}(u/v - m)$;
| $r \leftarrow \epsilon(u - mv)$;
| $(u, v) \leftarrow (v, r)$;
fintq
Renvoyer u ;

FIGURE 2.4: L'algorithme d'Euclide centré

si x est le plus petit entier strictement positif de $\mathcal{L}(u, v)$, alors tout élément $w \in \mathcal{L}$ est multiple de mx, et donc x est un diviseur commun à tous les éléments du réseau, et en particulier un diviseur commun à u et v. Et c'est le plus petit.

L'algorithme d'Euclide centré effectue une suite de divisions euclidiennes centrées. La division euclidienne centrée de u par v, pour deux nombres positifs u et v vérifiant $v \leq u$ est définie par :

$$u = mv + \epsilon r \quad \text{avec} \quad m = \left\lfloor \frac{u}{v} \right\rceil, \quad \epsilon = \text{Signe}\left(\frac{u}{v} - m\right) \quad \text{et} \quad r \in [0, v/2].$$

Ici, $\lfloor a \rceil$ désigne l'entier le plus proche de $a \in \mathbb{R}$. Au couple (u, v), on associe donc une paire (m, ϵ) et un reste r.

L'algorithme d'Euclide centré prend en entrée un couple (u, v) et calcule la suite des divisions successives :

$$v_0 = u, \quad v_1 = v, \ldots, v_{i-1} = m_i v_i + \epsilon_i v_{i+1}, \ldots, v_{p-2} = m_{p-1} v_{p-1} + \epsilon_{p-1} v_p, \quad v_{p-1} = m_p v_p + 0.$$

Le pgcd est donné par le dernier reste non nul v_p obtenu par l'algorithme. Le nombre d'étapes est égal à l'indice p.

A chaque étape i de cet algorithme, on a $0 \leq v_i \leq v_{i-1}/2$, autrement dit, le plus petit des entiers perd au moins 1 bit à chaque itération, le nombre total d'itérations est donc borné par :

$$1 \leq v_{k-1} \leq \frac{v_1}{2^{k-2}} \leq \frac{v}{2^{k-2}}, \qquad \text{et donc} \quad k \leq 2 + \log_2(v).$$

C'est une borne peu fine sur le nombre d'itérations de l'algorithme, mais reste linéaire en la taille de l'entrée. La meilleure borne possible qui a été obtenue par [30] est $\log_{1+\sqrt{2}}(v)$.

2.2.2 Dimension 2 : L'algorithme de Gauss

L'algorithme de Gauss résout complètement le problème de la réduction des réseaux en dimension 2. Il trouve une base minimale, c'est-à-dire un couple de vecteurs dont les longueurs atteignent les deux minima du réseau. L'algorithme de Gauss est très similaire à l'algorithme centré d'Euclide. La division centrée est remplacée par une "division vectorielle centrée".
L'écriture $\mathbf{v} = m\mathbf{u} + \epsilon\mathbf{r}$ est une généralisation de la division euclidienne centrée, le quotient m est l'entier le plus proche du rapport $(\mathbf{v}|\mathbf{u})/\|\mathbf{u}\|^2$, et le signe ϵ est le signe de la différence entre

m et le rapport $(\mathbf{v}|\mathbf{u})/\|\mathbf{u}\|^2$. Le vecteur r est ainsi le vecteur du réseau de longueur minimale parmi les vecteurs de l'ensemble $\{m\mathbf{u} \pm \mathbf{v}; m \in \mathbb{Z}\}$: il peut être considéré comme un reste de cette "division vectorielle".

L'algorithme de Gauss calcule successivement la suite des vecteurs $\mathbf{u}_i$, avec

$$\mathbf{u}_0 = \mathbf{b}_1, \qquad \mathbf{u}_1 = \mathbf{b}_2, \qquad \text{et} \quad \mathbf{u}_{i-1} = m_i\,\mathbf{u}_i + \epsilon_i\,\mathbf{u}_{i+1}$$

et s'arrête dès que la condition $\|\mathbf{u}_i\| \geq \|\mathbf{u}_{i-1}\|$ est satisfaite. On montre alors que les vecteurs de sortie forment une base minimale du réseau. L'algorithme de t-Gauss, défini pour $t > 1$, a une condition d'arrêt plus faible, puisqu'il s'arrête dès que la condition $\|\mathbf{u}_i\| \geq (1/t)\|\mathbf{u}_{i-1}\|$ est vérifiée ; cependant, pour $1 < t \leq \sqrt{3}$, sa configuration de sortie est satisfaisante, puisque le triangle construit sur la base de sortie contient les deux minima du réseau. Cet algorithme est utilisé comme brique principale de l'algorithme LLL pour $d \geq 3$.

Entrées : une base $B = (\mathbf{b}_1, \mathbf{b}_2)$
Résultat : Une base réduite
si $\|\mathbf{b}_1\| > \|\mathbf{b}_2\|$ **alors**
  échanger $\mathbf{b}_1$ et $\mathbf{b}_2$;
finsi
répéter
  $m_{2,1} \leftarrow \langle \mathbf{b}_2|\mathbf{b}_1\rangle/\|\mathbf{b}_1\|^2$;
  $\epsilon \leftarrow \mathrm{Sign}(\langle \mathbf{b}_2|\mathbf{b}_1\rangle/\|\mathbf{b}_1\|^2 - m_{2,1})$;
  $\mathbf{r} \leftarrow \epsilon(\mathbf{b}_2 - \lfloor m_{2,1}\rceil \mathbf{b}_1)$;
  $\mathbf{b}_2 \leftarrow \mathbf{b}_1$;
  $\mathbf{b}_1 \leftarrow \mathbf{r}$;
jusqu'à $\|\mathbf{b}_2\| \geq \|\mathbf{b}_1\|$;
Renvoyer $(\mathbf{b}_1, \mathbf{b}_2)$

Entrées : une base $B = (\mathbf{b}_1, \mathbf{b}_2)$
Résultat : Une base réduite
si $\|\mathbf{b}_1\| > \|\mathbf{b}_2\|$ **alors**
  échanger $\mathbf{b}_1$ et $\mathbf{b}_2$;
finsi
répéter
  $m_{2,1} \leftarrow \langle \mathbf{b}_2|\mathbf{b}_1\rangle/\|\mathbf{b}_1\|^2$;
  $\epsilon \leftarrow \mathrm{Sign}(\langle \mathbf{b}_2|\mathbf{b}_1\rangle/\|\mathbf{b}_1\|^2 - m_{2,1})$;
  $\mathbf{r} \leftarrow \epsilon(\mathbf{b}_2 - \lfloor m_{2,1}\rceil \mathbf{b}_1)$;
  $\mathbf{b}_2 \leftarrow \mathbf{b}_1$;
  $\mathbf{b}_1 \leftarrow \mathbf{r}$;
jusqu'à $\|\mathbf{b}_2\| \geq (1/t)\|\mathbf{b}_1\|$;
Renvoyer $(\mathbf{b}_1, \mathbf{b}_2)$

FIGURE 2.5: A gauche, l'algorithme de Gauss. A droite, l'algorithme de t-Gauss

À la sortie de l'algorithme de t-Gauss, la base de sortie $\widehat{B} = (\widehat{\mathbf{b}}_1, \widehat{\mathbf{b}}_2)$ vérifie les deux conditions suivantes :

$$\|\widehat{\mathbf{b}}_2\| \geq \frac{1}{t}\|\widehat{\mathbf{b}}_1\| \qquad 0 \leq \langle \widehat{\mathbf{b}}_2|\widehat{\mathbf{b}}_1\rangle \leq \frac{1}{2}\|\widehat{\mathbf{b}}_1\|^2. \tag{2.11}$$

Ces conditions suffisent à assurer la minimalité de la base de sortie :

Proposition 2.4. *La configuration de sortie de l'algorithme de Gauss est une base minimale.*

Preuve. A partir de la base $B = (\mathbf{b}_1, \mathbf{b}_2)$ de sortie, on construit l'orthogonalisé $\mathbf{b}_2^\star$ qui vérifie

$$\mathbf{b}_2 = m_{2,1}\mathbf{b}_1 + \mathbf{b}_2^\star, \qquad \text{et donc} \quad \|\mathbf{b}_2^\star\|^2 = \|\mathbf{b}_2\|^2 - m_{2,1}^2\|\mathbf{b}_1\|^2 \geq \left(\frac{1}{t^2} - \frac{1}{4}\right)\|\mathbf{b}_1\|^2.$$

Quand s vérifie la relation $s = \sqrt{4t^2/(4-t^2)}$, on obtient la condition de Siegel,

$$\|\mathbf{b}_2^\star\| \geq \frac{1}{s}\|\mathbf{b}_1\| \qquad \text{et pour } t = 1 \quad \|\mathbf{b}_2^\star\| \geq \frac{\sqrt{3}}{2}\|\mathbf{b}_1\|.$$

Soit maintenant un vecteur $\mathbf{v}$ non nul de réseau, qui s'écrit dans la base $B = (\mathbf{b}_1, \mathbf{b}_2)$ sous la forme : $\mathbf{v} = x_1\mathbf{b}_1 + x_2\mathbf{b}_2$ avec $(x_1, x_2) \in \mathbb{Z}^2$ et $(x_1, x_2) \neq (0, 0)$. Il y a trois possibilités pour l'entier x_2 :

(1) si $x_2 = 0$, alors $\|\mathbf{v}\| = |x_1|\|\mathbf{b_1}\| \geq \|\mathbf{b_1}\|$, et donc b_1 est un vecteur plus court non nul du réseau
(2) si $|x_2| = 1$, alors $\mathbf{b}_2$ est par définition le plus court vecteur du réseau sur l'ensemble des deux droites $x_2 = \pm 1$, et donc $\|\mathbf{v}\| \geq \|\mathbf{b}_2\|$
(3) si $|x_2| \geq 2$, alors $\|\mathbf{v}\| \geq |x_1|\ \mathbf{b}_2^\star\| \geq \sqrt{3}\|\mathbf{b}_2\| > \|\mathbf{b}_2\|$.

On peut donc déduire dans tous les cas que la base $B = (\mathbf{b}_1, \mathbf{b}_2)$ est minimale. □

On montre maintenant que l'algorithme de Gauss effectue un nombre d'itérations linéaire en la taille des entrées.

Proposition 2.5. *Sur une base $B = (\mathbf{b}_1, \mathbf{b}_2)$ avec $M = max(\|b_1\|, \|b_2\|)$, le nombre K d'itérations de l'algorithme de Gauss satisfait l'inégalité : $K \leq \log_{\sqrt{3}} M + 2$.*

Preuve. On procède en deux temps : on calcule d'abord le nombre d'itérations de l'algorithme t-Gauss et on montre ensuite que l'algorithme de Gauss fait au plus une itération supplémentaire par rapport à l'algorithme t-Gauss, pour $t < \sqrt{3}$.
Quand l'algorithme t-Gauss effectue p itérations, alors

$$\|\mathbf{u}_{p-1}\| < \frac{1}{t^{p-1}}\|\mathbf{u}_0\|, \quad \text{et donc} \quad p \leq 1 + \log_t \frac{\|\mathbf{u}_0\|}{\|\mathbf{u}_{p-1}\|} \leq 1 + \log_t M,$$

la dernière majoration étant due aux inégalités $\|\mathbf{u}_{p-1}\| \geq 1$ et $\|\mathbf{u}_0\| \leq M$. À la sortie de l'algorithme de t-Gauss, la base $B = (\mathbf{b}_1, \mathbf{b}_2)$ vérifie les conditions (2.11),

$$\|\mathbf{b}_2\| \geq \frac{1}{t}\|\mathbf{b}_1\|, \qquad 0 \leq \langle \mathbf{b}_2 | \mathbf{b}_1 \rangle \leq \frac{1}{2}\|\mathbf{b}_1\|^2.$$

Si cette base n'est pas 1-réduite, l'inégalité $\|\mathbf{b}_2\| < \|\mathbf{b}_1\|$ est vérifiée et on effectue une étape de l'algorithme de Gauss, qui échange les vecteurs $\mathbf{b}_1$ et $\mathbf{b}_2$, et le coefficient sous-diagonal vérifie

$$m_{2,1} = \frac{\langle \mathbf{b}_2 | \mathbf{b}_1 \rangle}{\|\mathbf{b}_2\|^2} = \frac{\langle \mathbf{b}_2 | \mathbf{b}_1 \rangle}{\|\mathbf{b}_1\|^2} \frac{\|\mathbf{b}_1\|^2}{\|\mathbf{b}_2\|^2} \leq \frac{t^2}{2}, \qquad \text{et donc, pour } t < \sqrt{3}, \quad \lfloor m_{1,2} \rceil = 1.$$

On remarque que la norme $\|\mathbf{b}_1 \pm \mathbf{b}_2\|$ vérifie

$$\|\mathbf{b}_1 \pm \mathbf{b}_2\|^2 \geq \|\mathbf{b}_1\|^2 - \langle \mathbf{b}_2 | \mathbf{b}_1 \rangle + \|\mathbf{b}_2\|^2 \geq \|\mathbf{b}_2\|^2.$$

La nouvelle base $(\mathbf{b}_1 \pm \mathbf{b}_2, \mathbf{b}_2)$ est alors réduite au bout d'une itération et l'algorithme de Gauss fait au plus $\log_{\sqrt{3}} M + 2$ itérations. □

La meilleure borne possible est $\log_{1+\sqrt{2}} M$, comme le montre Vallée dans [108].

2.3 L'algorithme de Lenstra-Lenstra-Lovász

L'algorithme LLL construit en temps polynomial une base dite LLL-réduite où la norme des vecteurs et leur orthogonalité sont bien maîtrisées. L'algorithme LLL a été inventé en 1982 par H.W. Lenstra, A.K. Lenstra et L. Lovász. Cet algorithme, initialement présenté dans le cadre de la factorisation des polynômes, a trouvé de nombreuses applications en théorie des nombres (conjecture de Mersenne, approximations diophantiennes...), en programmation linéaire entière (résolution d'inéquations linéaires), en algèbre (factorisation de polynômes ou d'entiers), etc. Depuis une trentaine d'années, les réseaux et l'algorithme LLL ont connu un véritable essor en cryptologie : cryptanalyse de protocoles de type RSA ou sac-à-dos, conception de cryptosystèmes (Ajtai-Dwork, NTRU, GGH...), preuves de sécurité, conception de protocoles homomorphes...

2.3.1 Principes généraux de l'algorithme

On rappelle qu'une base $B = (\mathbf{b}_1, \ldots, \mathbf{b}_d) \in \mathbb{R}^{d \times n}$ est t-LLL-réduite si elle satisfait les deux conditions suivantes :

$$|m_{i,j}| \leq \frac{1}{2} \quad \text{pour } 1 \leq j < i \leq d, \qquad \|\mathbf{b}^\star_{i+1} + m_{i+1,i}\mathbf{b}^\star_i\| \geq \frac{1}{t}\|\mathbf{b}^\star_i\|, \quad \text{pour } 1 \leq i < d, \tag{2.12}$$

qui se lisent sur la matrice $\mathcal{P}$ et la boîte B_i.

$$\mathcal{P} := \begin{array}{c} \\ \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_d \end{array} \begin{array}{c} \begin{array}{cccc} \mathbf{b}^\star_1 & \mathbf{b}^\star_2 & \ldots & \mathbf{b}^\star_d \end{array} \\ \begin{pmatrix} 1 & 0 & \ldots & 0 \\ m_{2,1} & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ m_{d,1} & m_{d,2} & \ldots & 1 \end{pmatrix} \end{array}, \qquad B_i := \begin{array}{c} \\ \mathbf{u}_i \\ \mathbf{v}_i \end{array} \begin{array}{c} \begin{array}{cc} \mathbf{b}^\star_i & \mathbf{b}^\star_{i+1} \end{array} \\ \begin{pmatrix} 1 & 0 \\ m_{i+1,i} & 1 \end{pmatrix} \end{array}.$$

La i-ème base locale $B_i = (\mathbf{u}_i, \mathbf{v}_i)$ est formée par les projections de $\mathbf{b}_i$ et $\mathbf{b}_{i+1}$ sur l'orthogonal de l'espace vectoriel H_{i-1} engendré par $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$ dans l'espace vectoriel H_{i+1} engendré par $(\mathbf{b}_1, \ldots, \mathbf{b}_{i+1})$. La deuxième condition (condition de t-Lovász) relie les longueurs des vecteurs $\mathbf{u}_i, \mathbf{v}_i$ via l'inégalité $\|\mathbf{v}_i\| \geq (1/t)\|\mathbf{u}_i\|$.

L'algorithme LLL effectue deux types d'opérations : – des translations pour réduire en taille des coefficients $m_{i,j}$ (procédure `Propre`) et des échanges pour "améliorer" la qualité de la boîte B_i (réduction de t-Gauss). La réduction en taille effectue des translations du type

$$\mathbf{b}_{i+1} := \mathbf{b}_{i+1} - \lfloor m_{i+1,j} \rceil \mathbf{b}_j,$$

qui rendent propres les coefficients $m_{i+1,j}$. Ces translations ne modifient pas la valeur des ℓ_i.

2.3.2 Étapes d'échange

Quand la condition de Lovász n'est pas satisfaite, la réduction de Gauss effectue un échange entre $\mathbf{b}_i$ et $\mathbf{b}_{i+1}$, qui modifie les longueurs ℓ_{i+1} et ℓ_i des orthogonalisés $\mathbf{b}^\star_{i+1}$ et $\mathbf{b}^\star_i$, et les rapports de Siegel

$$r_i = \frac{\ell_{i+1}}{\ell_i}.$$

Le nouveau $\check{\ell}_i$ satisfait

$$\check{\ell}_i^2 := \ell_{i+1}^2 + m_{i+1,i}^2\,\ell_i^2, \qquad \text{et donc} \quad \check{\ell}_i = \rho\,\ell_i \qquad \text{avec} \quad \rho^2 = \frac{\ell_{i+1}^2}{\ell_i^2} + m_{i+1,i}^2.$$

L'invariance du déterminant assure l'égalité $\check{\ell}_i\,\check{\ell}_{i+1} = \ell_i\,\ell_{i+1}$ et donc la relation $\check{\ell}_{i+1} = (1/\rho)\,\ell_{i+1}$. Si la condition de Lovász (ou Siegel) n'est pas vérifiée, ce rapport ρ vérifie l'inégalité

$$\rho \leq \rho_0(t) \qquad \text{avec} \quad \rho_0^2(t) = \frac{1}{t^2} = \frac{1}{s^2} + \frac{1}{4} < 1.$$

On appelle ρ facteur de décroissance de cette réduction. Il s'ensuit que les trois rapports de Siegel r_{i-1}, r_i, r_{i+1} sont modifiés et vérifient :

$$\check{r}_{i-1} = \rho r_{i-1}, \quad \check{r}_i = \frac{1}{\rho^2} r_i, \quad \check{r}_{i+1} = \rho r_{i+1}. \tag{2.13}$$

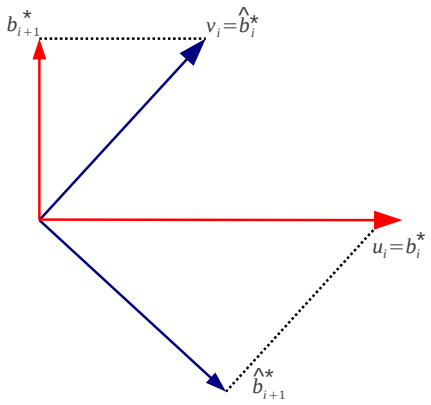


FIGURE 2.6: Échange entre les deux vecteurs $\mathbf{b}_i$ et $\mathbf{b}_{i+1}$.

Un tel échange a pour effet d'augmenter la valeur du rapport r_i (ce qu'on veut), en diminuant la valeur des rapports r_{i-1} et r_{i+1}. A la fin, tous les rapports r_i seront minorés. Les deux quantités

$$a := \min\{\ell_i : 1 \le i \le d\}, \qquad A := \max\{\ell_i : 1 \le i \le d\}, \tag{2.14}$$

jouent un rôle important dans l'algorithme :

Proposition 2.6. *Lors de chaque étape d'échange de l'algorithme LLL, la quantité a croît et la quantité A décroît.*

Preuve. On a vu qu'un échange sur la boîte B_i modifie les longueurs ℓ_{i+1} et ℓ_i des orthogonalisés, mais ne modifie pas les autres longueurs ℓ_j. Plus précisément, un échange entre $\mathbf{b}_i$ et $\mathbf{b}_{i+1}$ transforme (ℓ_i, ℓ_{i+1}) en $(\check{\ell}_i, \check{\ell}_{i+1})$, et il suffit de montrer les deux inégalités

$$\min(\check{\ell}_i, \check{\ell}_{i+1}) \ge \min(\ell_i, \ell_{i+1}), \qquad \max(\check{\ell}_i, \check{\ell}_{i+1}) \le \max(\ell_i, \ell_{i+1}).$$

On utilise les inégalités

$$\ell_{i+1}^2 \le \check{\ell}_i^2 = \ell_{i+1}^2 + m_{i+1,i}^2 \ell_i^2 \le \frac{1}{t^2}\ell_i^2 \le \ell_i^2, \qquad \text{et donc} \quad \ell_{i+1} \le \check{\ell}_i \le \ell_i.$$

L'invariance du déterminant $\ell_i \ell_{i+1} = \check{\ell}_i \check{\ell}_{i+1}$, permet alors d'en déduire les inégalités $\ell_{i+1} \le \check{\ell}_{i+1}$, et $\ell_i \ge \check{\ell}_{i+1}$ ce qui permet de conclure que

$$\ell_{i+1} \le \min(\check{\ell}_{i+1}, \check{\ell}_i), \qquad \max(\check{\ell}_{i+1}, \check{\ell}_i) \le \ell_i.$$

□

Proposition 2.7. *Le premier minimum vérifie les inégalités : $a \le \lambda_1 \le A\sqrt{d}$*

Preuve. En remarquant que le déterminant du réseau est majoré par A^d, l'inégalité de Minkowski prouve la majoration. La minoration est prouvée dans le Chapitre 1 (1.5). □

Entrées : Une base $B = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$ d'un réseau $\mathcal{L}$ de dimension d de $\mathbb{R}^n$ et $t > 1$
Résultat : Une base $\widehat{B} = (\widehat{\mathbf{b}}_1, \widehat{\mathbf{b}}_2, \ldots, \widehat{\mathbf{b}}_d)$ de $\mathcal{L}$ propre, et t-réduite.

Calculer la base orthogonale $B^\star$ et la matrice de passage $\mathcal{P}$;
$i \leftarrow 1$;
tant que $i < d$ **faire**
  `Translater` $\mathbf{b}_{i+1}$ parallèlement à $\mathbf{b}_i$ afin d'assurer $|m_{i+1,i}| \leq 1/2$;
  `Tester` la `réduction` de la boîte $B_i := (\mathbf{u}_i, \mathbf{v}_i)$ au sens de t-Gauss :
    A-t-on $\|\mathbf{v}_i\| > (1/t)\,\|\mathbf{u}_i\|$?
      Si oui
        `Rendre propre.` Pour k allant de $i-1$ à 1, translater $\mathbf{b}_{i+1}$ parallèlement à $\mathbf{b}_k$ afin d'assurer $|m_{k,i}| \leq 1/2$;
        $i \leftarrow i + 1$;
      Sinon
        `Échanger` $\mathbf{b}_i$ et $\mathbf{b}_{i+1}$;
        `Recalculer` $(B^\star, \mathcal{P})$;
        `Si` $i \neq 1$ `alors` $i \leftarrow i - 1$;
fintq

FIGURE 2.7: Description de l'algorithme LLL(t)

2.3.3 Description de l'algorithme

L'indice i des bases locales varie dans l'intervalle $[1\,..\,d-1]$. C'est le plus grand indice pour lequel le système $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_i)$ est t-réduit au sens de Lovász (Siegel). On effectue des opérations de translation sur la i-ième ligne de $\mathcal{P}$ et d'échange sur la boîte B_i. L'indice est incrémenté ou décrémente à chaque étape selon le résultat du test de t-Gauss.

L'exécution de l'algorithme définit une marche aléatoire qui exprime l'indice $i \in [1\,..\,d-1]$ de la boîte B_i considérée en fonction du temps discret $j \in [1, K]$ où K est le nombre d'itérations de l'exécution. La longueur de la marche est le nombre K d'itérations lors de cette exécution.

On ne peut pas montrer que l'algorithme LLL effectue un nombre polynomial d'itérations quand le paramètre t est égal à 1. On choisit une valeur $t > 1$ et la valeur usuellement choisie, notamment pour simplifier les calculs, est $t = 2/\sqrt{3}$ qui correspond à $s = \sqrt{2}$. Dans la littérature, l'algorithme LLL est souvent décrit avec cette valeur-là du paramètre t. Mais nous nous intéressons ici à une valeur quelconque du paramètre t pourvu que t satisfasse $t > 1$.

2.3.4 Étude des paramètres de l'algorithme

On étudie ici les paramètres géométriques qui définissent la configuration de sortie, et la complexité en nombres d'opérations binaires de l'algorithme.

Configuration de sortie de l'algorithme. Comme une base t-réduite au sens de Lovász et aussi s-réduite au sens de Siegel, alors pour une base LLL t-réduite au sens de LLL les défauts de longueurs et Hermite sont bornés par une fonction exponentielle en la dimension d, et le défaut d'orthogonalité est borné par une fonction exponentielle en la carré de la dimension de d.

Proposition 2.8. *Une base B t-réduite au sens de Lovász a les caractéristiques suivantes, qui s'expriment en fonction de $s = 2t/\sqrt{4-t^2}$,*

$$\delta(B) = \prod_{i=1}^{d} \frac{\|\widehat{\mathbf{b}}_i\|}{\|\widehat{\mathbf{b}}_i^\star\|} \leq s^{\frac{d(d-1)}{2}}, \qquad \mu_i(B) = \frac{\|\mathbf{b}_i\|}{\lambda_i(\mathcal{L})} \leq s^{d-1}, \qquad \gamma(B) = \frac{\|\mathbf{b}_1\|^2}{\det(\mathcal{L})^{2/d}} \leq s^{d-1}.$$

Preuve. C'est la même preuve que celle de la Proposition 2.3, car une base t-réduite au sens de Lovász est s-réduite au sens de Siegel. □

Nombre d'itérations. L'algorithme LLL effectue un nombre d'itérations quadratique en la dimension d. Plus précisément :

Théorème 2.9. *Sur une base B d'un réseau $\mathcal{L} \in \mathbb{R}^n$ de dimension d, le nombre d'itérations K de l'algorithme LLL(t) vérifie les inégalités suivantes :*
(i) Si $a = \min\{\ell_i\}$ et $A = \max\{\ell_i\}$ sont les longueurs minimales et maximales des vecteurs de la base orthogonalisée $B^\star$, alors

$$K \leq (d-1)(1 + d \log_t \frac{A}{a}). \tag{2.15}$$

(ii) Si le réseau est entier et les vecteurs de la base initiale B sont de longueur au plus M, alors

$$K \leq (d-1)(1 + d \log_t M). \tag{2.16}$$

(iii) Si les vecteurs de la base initiale B sont de longueur au plus M, et si le réseau $\mathcal{L}$ engendré par B a un premier minimum $\lambda(\mathcal{L})$, alors

$$K \leq d^2 \log_t \frac{M\sqrt{d}}{\lambda(\mathcal{L})}. \tag{2.17}$$

Preuve. Soient $\mathcal{L}_k$ le réseau engendré par les k premiers vecteurs de B, et d_k^2 le déterminant de Gram associé. On pose

$$D(B) = \prod_{i=1}^{d-1} d_i^2 = \prod_{k=1}^{d-1} \|\mathbf{b}_k^\star\|^{2(d-k)}. \tag{2.18}$$

La quantité $D(B)$ est souvent appelée potentiel de la base B. L'inégalité d'Hadamard permet d'affirmer que :

$$d_k \leq \prod_{i=1}^{k} \|\mathbf{b}_i\| \leq M^k.$$

L'indice k croit si la condition de Lovász est vérifiée et décroît si ce n'est pas le cas. On note K_- le nombre de fois où le test de Lovász est négatif et K_+ le nombre de tests positifs. Nous allons les majorer en fonction de t, M et d. D'abord, le nombre d'itérations K_+ vérifie $K_+ \leq K_- + d - 1$, et donc, le nombre total K d'itérations de l'algorithme vérifie

$$K = K_+ + K_- \leq d - 1 + 2K_-. \tag{2.19}$$

Chaque passage dans la k-ème boîte locale, avec un test négatif, ne modifie pas les sous-réseaux $\mathcal{L}_j$ pour $j \neq k$. Seul, le réseau $\mathcal{L}_k$ est modifié et son déterminant également. On avait précédemment

$$d_k = \prod_{i=1}^{k} \|\mathbf{b}_i^\star\| = \|\mathbf{b}_k^\star\| \prod_{i=1}^{k-1} \|\mathbf{b}_i^\star\|$$

Après le test négatif :

$$\check{d}_k = \prod_{i=1}^{k} \|\check{\mathbf{b}}_i^\star\| = \|\check{\mathbf{b}}_k^\star\| \prod_{i=1}^{k-1} \|\mathbf{b}_i^\star\|, \qquad \text{avec} \quad \|\check{\mathbf{b}}_k^\star\| \leq \frac{1}{t}\|\mathbf{b}_k^\star\|.$$

On en déduit donc que $\check{d}_k \leq (1/t)\, d_k$ et donc $D(\check{B}) \leq (1/t)^2\, D(B)$. Autrement dit le potentiel $D(B)$ décroît d'un facteur $(1/t)^2$. D'autre part, lors de chaque passage avec un test positif, aucun des réseaux $\mathcal{L}_k$ n'est modifiés. Donc $D(B)$ reste inchangé lors de cette étape. Finalement, $D(B)$ décroît au long de l'algorithme, et décroît d'un facteur $(1/t)^2$ lors d'un test négatif. Ce qui montre l'inégalité suivante entre le potentiel $D(B)$ du début et le potentiel $D(\widehat{B})$ de la fin,

$$D(\widehat{B}) \leq \left(\frac{1}{t^2}\right)^{K_-} D(B), \qquad \text{et donc} \quad 2K_- \leq \log_t \left(\frac{D(B)}{D(\widehat{B})}\right). \tag{2.20}$$

La proposition 2.6 et les relations (2.18) entraînent les inégalités

$$D(\widehat{B}) \geq a^{d(d-1)}, \quad D(B) \leq A^{d(d-1)},$$

ce qui montre, avec (2.20) la première inégalité du théorème. Pour un réseau entier, on a, au départ, l'inégalité $D(B) \leq M^{d(d-1)}$, et, à l'arrivée, on a l'inégalité $D(\widehat{B}) \geq 1$. On obtient donc, avec (2.20) la majoration $2K_- \leq d(d-1)\log_t M$, et finalement la deuxième inégalité $K \leq (d-1)(1 + d\log_t M)$. On peut aussi faire intervenir le premier minimum de réseau $\lambda(\mathcal{L})$. La définition de la constante d'Hermite (voir 1.17) donne l'inégalité :

$$\det(\mathcal{L}_j) \geq \frac{\lambda(\mathcal{L}_j)^j}{\gamma_j^{j/2}}.$$

Puisque le réseau $\mathcal{L}_j$ est un sous-réseau de $\mathcal{L}$, l'inégalité $\lambda(\mathcal{L}_j) \geq \lambda(\mathcal{L})$ est vraie, et donc,

$$D(\widehat{B}) = \prod_{j=1}^{d-1} \frac{\lambda(\mathcal{L})^j}{\gamma_j^j} = \lambda(\mathcal{L})^{d(d-1)} \prod_{j=1}^{d-1} \gamma_j^{-j}.$$

Alors, la majoration $D(B) \leq M^{d(d-1)}$ entraîne, avec (2.20), l'inégalité

$$2K_- \leq d(d-1)\log_t \frac{M}{\lambda(\mathcal{L})} + \sum_{j=1}^{d-1} \log_t(\gamma_j^j).$$

La majoration de la constante d'Hermite $\gamma_j \leq j$ entraîne :

$$(d-1) + \sum_{j=1}^{d-1} \log_t(\gamma_j^j) \leq \frac{d^2}{2}\log_t d,$$

et l'inégalité (2.19) permettent de conclure dans le cas (*iii*).

□

Complexité binaire de l'algorithme. Le résultat final est le suivant :

Théorème 2.10. *L'algorithme LLL, associé au paramètre t, construit, à partir d'une base B d'un réseau $\mathcal{L} \in \mathbb{Z}^n$ de dimension d, une base $\widehat{B}$ (t-réduite au sens de LLL) en temps polynomial en la taille $(d, n, \log M = \log \max(\|b_i\|))$; la borne plus précise est en $O(d^5 n \log_t^3 M)$: le nombre d'opérations arithmétiques effectuées dans cet algorithme est $O(d^3 n \log_t M)$ et les entiers sur lesquels il opère sont de taille $O(d \log M)$.*

Pour montrer le théorème 2.10, il reste à borner la taille des nombres apparaissant dans l'algorithme, en fonction de la taille de la donnée $(d, n, \log M)$. Il faut vérifier que la croissance des tailles des entiers manipulés lors de l'exécution reste polynomiale en la taille de l'entrée. Les ℓ_i^2 sont des rationnels dont numérateur et dénominateur décroissent tout au long de l'algorithme. Par contre, ce n'est pas le cas pour les coefficients rationnels $m_{i,j}$ et les entiers $\|\mathbf{b}_i\|^2$. Dans [65] les auteurs montrent que les numérateurs et dénominateurs des rationnels $m_{i,j}$, tout comme les longueurs entières $\|\mathbf{b}_i\|^2$, ont des tailles qui restent majorées par un terme d'ordre $O(d \log M)$. Comme il y a un nombre d'opérations arithmétiques de l'ordre de $O(d^3 n \log_t M)$, cela termine la preuve du théorème 2.10.

Cas du paramètre $t = 1$. Même si on ne peut pas montrer que l'algorithme LLL(1) effectue un nombre polynomial d'itérations en $(d, \log M)$, on parvient à montrer que l'algorithme effectue un nombre d'itérations polynomial en $\log M$, et est donc un algorithme polynomial à dimension d fixée. Pour cela, on compare l'algorithme LLL(1) avec l'algorithme LLL(t) pour t suffisamment proche de 1, et on montre que la dernière phase de l'algorithme, pour passer d'une base réduite au sens de t-Lovász, à une base réduite au sens de 1-Lovász n'effectue qu'un nombre polynomial d'étapes en $\log M$, dont on ne parvient pas à montrer qu'il est polynomial en la dimension d. C'est la démarche adoptée par Lenstra [67] ou Akhavi [7]. Nous étudierons une variante de l'algorithme LLL associée à $t = 1$ dans le Chapitre 7, et nous procéderons de manière analogue.

2.3.5 Algorithmes du plus court vecteur

Tout d'abord, ce qui précède montre que l'algorithme LLL est un algorithme d'approximation pour le problème $\mathcal{SVP}$. Même si le premier vecteur renvoyé par LLL n'est pas nécessairement un plus court vecteur du réseau, on sait que sa norme peut se comparer au premier minimum, avec toutefois un facteur exponentiel en la dimension d.

Théorème 2.11. *Pour un réseau $\mathcal{L}$ de dimension $d \leq 2$, l'algorithme LLL(t) est un algorithme polynomial d'approximation qui fournit un vecteur dont la longueur est au plus égale à $s^{d-1}\lambda_1(\mathcal{L})$, avec s et t reliés par la relation (2.10.)*

La base de sortie de l'algorithme LLL est utilisée comme base d'entrée dans les algorithmes dus à Pohst [95], Kannan [52] et Finch-Pohst [32] qui résolvent exactement le problème du plus court vecteur. Ce sont des algorithmes d'énumération, fondés sur une idée simple : Considérons une base $B = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau $\mathcal{L}$. Un plus court vecteur du réseau $\mathcal{L}$ est donc de la forme

$$\mathbf{x} = x_1\mathbf{b}_1 + \ldots + x_d\mathbf{b}_d \qquad \text{avec des } x_i \text{ entiers et } x_d \geq 0.$$

On cherche des encadrements pour les entiers x_i en fonction des longueurs des vecteurs $\mathbf{b}_i$ et des orthogonalisés $\mathbf{b}_i^\star$. On a :

$$\mathbf{x} = \sum_{i=1}^{d} x_i\mathbf{b}_i = \sum_{i=1}^{d} x_i \left(\mathbf{b}_i^\star + \sum_{j=1}^{i-1} m_{i,j}\mathbf{b}_j^\star \right) = \sum_{j=1}^{d} \left(x_j + \sum_{i=j+1}^{d} m_{i,j}x_i \right) \mathbf{b}_j^\star,$$

et donc

$$\|\mathbf{x}\|^2 = \sum_{j=1}^{d} \left(x_j + \sum_{i=j+1}^{d} m_{i,j} x_i \right)^2 \|\mathbf{b}_j^\star\|^2.$$

Si $\mathbf{x}$ est un plus court vecteur, il est donc plus court que $\mathbf{b}_1$. On va faire une recherche exhaustive sur les d-uplets $(x_1, \ldots, x_d)$ d'entiers vérifiant ces conditions, en commençant par x_d, puis par x_{d-1}, etc. On a d'abord $0 \leq x_d \leq \|\mathbf{b}_1\|/\|\mathbf{b}_d^\star\|$. Et pour $j \in [1 \,..\, d-1]$,

$$\left(x_j + \sum_{i=j+1}^{d} m_{i,j} x_i \right)^2 \|\mathbf{b}_j^\star\|^2 < \|\mathbf{b}_1\|^2 - \sum_{k=j+1}^{d} \left(x_k + \sum_{i=k+1}^{d} m_{i,k} x_i \right)^2 \|\mathbf{b}_k^\star\|^2.$$

On fait donc une recherche exhaustive sur un ensemble de cardinal

$$\frac{\|\mathbf{b}_1\|}{\|\mathbf{b}_d^\star\|} \prod_{i=2}^{d-1} \left(\left\lfloor \frac{2\|\mathbf{b}_1\|}{\|\mathbf{b}_j^\star\|} \right\rfloor + 1 \right).$$

Si la base B est déjà LLL-réduite, le cardinal de l'ensemble est majoré par $2^{O(d^2)}$. Kannan [52] a remarqué qu'en appliquant récursivement l'algorithme d'énumération, on peut réduire le nombre d'itérations à $2^{O(d \log d)}$ opérations polynomiales. Mais les opérations polynomiales dans la version récursive sont plus coûteuses que dans la version classique et il n'est pas clair que l'algorithme de Kannan ait un intérêt pratique pour les dimensions actuellement utilisées.

2.4 Applications diverses de l'algorithme LLL

Dans cette section, nous présentons un certain nombre de problèmes classiques qu'on peut modéliser par des réseaux. Chacun de ces problèmes donne lieu à une base dont la forme est très liée au problème de départ. Et la base réduite, résultat de l'algorithme de réduction, permet de trouver une solution au problème de départ.

2.4.1 La programmation linéaire en nombre entiers

Le problème de la programmation linéaire se décrit comme suit :
Étant donné un polytope P de $\mathbb{R}^d$ de volume non nul, déterminer les points de coordonnées entières situés à l'intérieur de ce polytope.

Ce problème peut se transformer en un problème de réseaux. L'idée de Lenstra [66] est de coincer le polytope P entre deux ellipsoïdes. Alors si f est la transformation linéaire qui transforme un ellipsoïde P en une sphère $\mathcal{S}$ et $\mathcal{L}$ est le réseau obtenu par la transformation de $\mathbb{Z}^d$ par f, alors le problème précédent devient :
Énumérer les vecteurs d'un réseau $\mathcal{L}$ situés à l'intérieur d'une sphère $\mathcal{S}$.

Pour résoudre ce dernier problème, on commence par réduire le réseau $\mathcal{L}$ avec l'algorithme LLL et obtient une base réduite $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d)$. Pour une base quelconque, les longueurs $\|\mathbf{b}_i^\star\|$ mesurent des espacements entre sous-espaces vectoriels. En particulier, la longueur $\|\mathbf{b}_d^\star\|$ mesure l'espacement entre deux hyperplans parallèles à l'hyperplan H_{d-1} engendré par les vecteurs $\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_{d-1}$. Or dans une base LLL réduite, la longueur $\|\mathbf{b}_d^\star\|$ est minorée, et il n'y a donc pas trop d'hyperplans parallèles à l'hyperplan H_{d-1} qui rencontrent l'intérieur de $\mathcal{S}$. Puis, on procède de manière récursive, en utilisant l'argument d'espacement des hyperplans parallèles à l'hyperplan généré par $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{j-1})$, qui permet de borner le nombre de ces hyperplans (voir la figure 2.8).

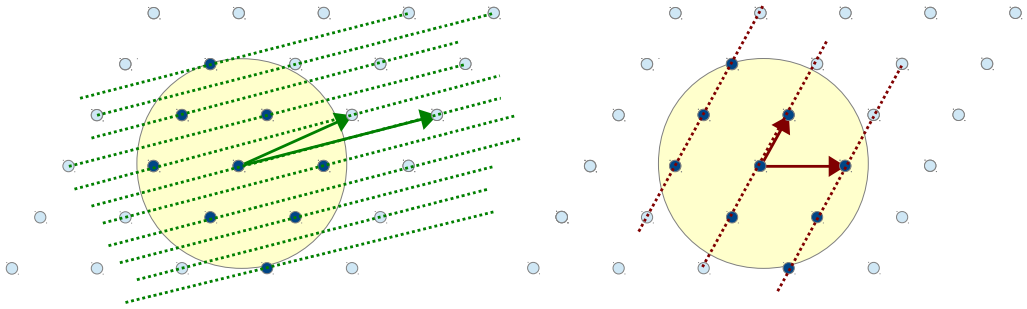


FIGURE 2.8: L'espacement des hyperplans parallèles pour une base non-réduite (à gauche) et une base réduite (à droite)

2.4.2 Les approximations diophantiennes simultanées

Ce problème consiste à trouver une famille de rationnels de même dénominateur qui approxime une famille de nombres réels, et se décrit comme suit :

Étant donnée une suite $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ *de* n *nombres réels, trouver* n *nombres entiers* $(p_1, p_2, \ldots, p_n)$ *et un nombre entier* q *tels que la suite des* n *nombres rationnels* $(p_1/q, p_2/q, \ldots, p_n/q)$ *fournisse une bonne approximation de la suite* $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ *donnée.*

Une réponse, non constructive à cette question, fondée sur le théorème de Minkowski, a été donnée par Dirichlet :

Théorème 2.12. [Dirichlet]. *Pour tout* n*, pour toute suite* $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ *et pour tout couple* (ϵ, Q) *vérifiant* $\epsilon > 0$ *et* $Q \geq \epsilon^{-n}$*, on peut construire en temps polynomial en* $(n, \log \epsilon)$ *des entiers* $(p_1, p_2, \ldots, p_n)$ *et un entier* q *qui vérifient*

$$0 < q \leq Q \qquad et \quad |q\alpha_i - p_i| < \epsilon \qquad pour\ tout \quad i, 1 \leq i \leq n.$$

Puis, Lagarias [60] a donné une version approchée mais constructive à ce théorème. Il considère le réseau $\mathcal{L}$ engendré par les lignes $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$ de la matrice

$$B := \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n & \epsilon/Q \end{pmatrix}$$

Le premier vecteur $\mathbf{v}$ de la base B réduite par l'algorithme LLL se décompose dans la base B avec des composantes $(p'_1, p'_2, \ldots, p'_n, q')$ sous la forme

$$\mathbf{v} = \sum_{i=1}^{n} p'_i \, \mathbf{b}_i + q' \, \mathbf{b}_{n+1} = (p'_1 + \alpha_1 q', \ldots, p'_n + \alpha_n q', \frac{\epsilon}{Q} q').$$

Si le vecteur $\mathbf{v}$ a sa norme $\|\mathbf{v}\|_\infty$ qui vérifie $\|\mathbf{v}\|_\infty < \epsilon$, alors on aura les inégalités $|p'_i + \alpha_i q'| < \epsilon$ et $q' < Q$. La possibilité de trouver un vecteur si court avec l'algorithme LLL est liée au choix de Q. D'après le Théorème 2.10, l'algorithme LLL construit un vecteur court qui vérifie :

$$\|\mathbf{v}\| < s^{n/2} \left(\det \mathcal{L}\right)^{1/n+1},$$

donc, pour assurer la condition $\|v\|_\infty < \epsilon$, l'entier Q doit vérifier $Q \geq s^{n(n+1)/2}\epsilon^{-n}$. On peut donc préciser le théorème constructif :

Théorème 2.13. *Pour tout n, pour toute suite $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ et pour tout couple (ϵ, Q) vérifiant $\epsilon > 0$ et $Q \geq s^{n(n+1)/2}\epsilon^{-n}$, on peut construire des entiers $(p_1, p_2, \ldots, p_n)$ et un entier q vérifiant*

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

2.4.3 Calcul de racines n-ièmes modulo m

On sait que le calcul de la racine n-ième modulo un nombre m de factorisation connue est polynomial en $\log m$. Mais, que se passe-t-il quand la factorisation de m est inconnue, et que l'on s'intéresse seulement à un problème approché ? Ce problème "approché" se décrit comme suit :
Soient deux entiers m et $n \geq 2$. Étant donnés des nombres entiers x_0 et y_0, un voisinage $I(x_0)$ de x_0 et un voisinage $J(y_0)$ de y_0 qui contiennent respectivement un point x et un point y vérifiant $x^n = y \mod m$, on veut trouver x et y.

On veut donc trouver un couple (u, v) de petits entiers, solution de l'équation $(x_0 + u)^n = y_0 + v \mod m$ qui se développe en

$$x_0^n + \binom{n}{1} x_0^{n-1} u + \ldots + \binom{n}{i} x_0^{n-i} u^i + \ldots + \binom{n}{n-1} x_0 u^{n-1} + u^n - v = y_0 \mod m.$$

On pose $w_i = u^i$ pour tout i, $0 \leq i \leq n-1$ et aussi $w_n = y_0 + v - u^n$, et on travaille dans le réseau $\mathcal{L}$ des vecteurs $\mathbf{w} = (w_0, w_1, \ldots, w_n)$ de $\mathbb{Z}^{n+1}$ vérifiant :

$$\sum_{i=0}^{n-1} \binom{n}{i} x_0^{n-i} w_i - w_n = 0 \mod m,$$

qui a pour matrice

$$B := \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ x_0^n & \binom{n}{1} x_0^{n-1} & \cdots & \binom{n}{n-1} x_0 & m \end{pmatrix}.$$

On cherche un vecteur $\mathbf{w}$ du réseau $\mathcal{L}$ proche du point $(0, 0, \ldots, y_0)$. Si le réseau $\mathcal{L}$ est suffisamment "régulier", ses minima successifs seront proches les uns des autres, et son premier minimum $\lambda_1(\mathcal{L})$ sera proche de la moyenne géométrique des minima successifs qui est de l'ordre de $m^{2/n+1}$, d'après le Théorème de Minkowski 1.18. Or, on peut montrer que la plupart des réseaux de ce type sont "réguliers", et, dans ce cas, un bon choix des paramètres des voisinages $I(x_0)$ et $J(x_0)$ permet d'affirmer que le point $\mathbf{w}$ trouvé par un algorithme du point le plus proche donnera, avec

grande probabilité, le couple (u, v) cherché, et donc le couple (x, y) cherché. Le bon choix des paramètres est

$$I(x_0) := \{x, |x - x_0| \mod n \leq m^{2/n(n+1)}\}, \qquad I(y_0) := \{y, |x - y_0| \mod n \leq m^{2/n(n-1)}\}.$$

Pour $n = 2$, on peut deviner un tiers des bits sur x, et deux tiers des bits sur y. Mais le nombres de bits qu'on peut deviner décroît quadratiquement avec le degré n de l'équation. Ce problème s'est posé de manière naturelle en liaison avec le cryptosystème d'Okamoto, et cette méthode proposée par Vallée, Girault, Toffin [111] peut être considérée comme un premier pas vers ce qui donnera lieu à la méthode de Coppersmith.

2.4.4 Factorisation de Schnorr

Le problème de la factorisation entière se décrit comme suit :
Soit N un grand entier ; on veut obtenir la décomposition de N en facteurs premiers.

Il existe un algorithme de factorisation, dû à Schnorr [100], fondé sur la méthode générale de la congruence des carrés, qui utilise, après linéarisation, les réseaux euclidiens. On cherche un couple $(x, y) \in \mathbb{Z}^2$, vérifiant

$$x^2 = y^2 \mod N, \qquad \text{avec } x \neq \pm y \mod N.$$

Dans ce cas, $\gcd(x \pm y, N)$ est un diviseur non trivial de N. Un nombre est dit p-lisse s'il ne contient pas de diviseur premiers supérieurs à p. On désigne par $p_1, p_2, \ldots, p_i$ les i premiers nombres premiers et on suppose que N n'a pas de diviseurs premiers inférieurs ou égaux à p_d pour un d fixé. Les nombres x et y cherchés sont construits comme des produits de nombres entiers p_d-lisses, de la forme u et $u - vN$, avec u est premier avec v.

A une constante $a > 1$, on associe l'ensemble $\mathcal{P} := \{p_1, \ldots, p_d\}$ formé des d premiers nombres premiers avec $p_d = (\log N)^a$ et $p_0 = -1$. La méthode de Schnorr se décrit comme suit :

(*i*) On cherche les diviseurs de N dans $\mathcal{P}$; pour chaque tel diviseur trouvé, on divise N par ce diviseur ;

(*ii*) On construit une liste de $t+2$ couples (u, v), vérifiant les deux conditions : u est p_d-lisse et $|u - vN| \leq p_d^\sigma$, pour une constante $\sigma > 1$ petite. Alors, il est très probable que $|u - vN|$ soit aussi p_d lisse.

(*iii*) Chaque couple $(u^{(i)}, v^{(i)})$ s'écrit

$$u^{(i)} = \prod_{a_j > 0} p_j^{a_j^{(i)}}, \qquad |u - vN| = \prod_{j=0}^{d} p_j^{b_j^{(i)}},$$

et à chaque couple $(u^{(i)}, v^{(i)})$, on associe le couple de vecteurs $(\mathbf{a}^{(i)}, \mathbf{b}^{(i)})$

$$\mathbf{a}^{(i)} = (a_0^{(i)}, a_1^{(i)}, \ldots, a_d^{(i)}), \quad \mathbf{b}^{(i)} = (b_0^{(i)}, b_1^{(i)}, \ldots, b_d^{(i)}) \qquad \text{avec } a_0^{(i)} = 0.$$

(*iv*) Avec $d+2$ congruences de ce type, on peut, via une élimination modulo 2, construire le couple (x, y) qui fournira un diviseur non trivial de N avec une probabilité au moins égale à $1/2$: on calcule en effet le vecteur $\mathbf{c} = (c^{(1)}, \ldots, c^{(d+2)}) \in \{0, 1\}^{d+2}$ vérifiant

$$\text{pour tout } j \in [0 \,..\, t+2], \quad d_j := \sum_{i=1}^{d+2} c^{(i)}(a_j^{(i)} + b_j^{(i)}) = 0 \mod 2,$$

et on pose

$$x = \prod p_j^{d_j/2} \mod N, \quad y = \prod p_j^{e_j} \mod N, \quad \text{avec} \quad e_j := \sum_{i=1}^{d+2} c^{(i)} a_j^{(i)}.$$

(v) Si $x \neq \pm y \mod N$, on renvoie $\gcd(x \pm y, N)$

Pour trouver les couples (u, v), Schnorr introduit un réseau $\mathcal{L}(\mathcal{B})$ dans lequel il cherche des vecteurs courts. Le réseau $\mathcal{L}(\mathcal{B})$ considéré est engendré par les lignes de la matrice suivante, où c est une constante positive suffisamment grande,

$$\mathcal{B} = \left(\begin{array}{c|cccc} N^c \log N & 0 & 0 & \cdots & 0 \\ \hline N^c \log p_1 & \log p_1 & 0 & \cdots & 0 \\ N^c \log p_2 & 0 & \log p_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ N^c \log p_{d-1} & 0 & 0 & \cdots & \log p_{d-1} \end{array}\right).$$

A partir d'un vecteur court $\mathbf{z}$ du réseau $\mathcal{L}(\mathcal{B})$, on associe une paire (u, v) définie par

$$u = \prod_{z_i > 0} p_i^{z_i}, \qquad v = \prod_{z_i < 0} p_i^{-z_i},$$

et Schnorr montre que, si le vecteur est suffisamment court, alors l'entier $u - vN$ vérifie $|u - vN| \leq p_d^{\sigma}$ pour une certaine constante σ petite.

Finalement, l'algorithme de Schnorr travaille avec un réel $c > 0$ et une dimension d pour laquelle le nombre premier p_d soit de l'ordre de $(\log N)^a$; Et donc la dimension d est de l'ordre approximatif $(\log N)^a$, avec a souvent proche de 2.

Dans le chapitre suivant, on présente les applications de la réduction des réseaux dans le domaine de la cryptologie.

Chapitre 3

Réseaux et Cryptologie

Sommaire

Au chapitre précédent, nous avons présenté plusieurs applications de la réduction des réseaux à la programmation linéaire entière [66], aux approximations diophantiennes simultanées [60], au calcul de racines n-ème modulaires [111] ou à la factorisation [100]. Dans ce chapitre, nous nous intéressons aux applications liées à la cryptologie.

La cryptologie regroupe deux grands types d'activité : la cryptographie et la cryptanalyse. La cryptographie vise à concevoir des protocoles, à prouver leur sécurité ou encore à améliorer l'algorithmique sous-jacente. La cryptanalyse consiste à mettre en défaut les protocoles ou les problèmes algorithmiques liés aux preuves de sécurité. Dans un premier temps, les réseaux sont intervenus essentiellement dans la cryptanalyse de protocoles ; puis, au milieu des années 90, les réseaux sont devenus une brique de base en cryptologie. Depuis les travaux de Ajtai [2], les réseaux sont utilisés dans la conception de protocoles "prouvés sûrs dans le pire des cas". Ils interviennent aussi dans la conception de protocoles complètement homomorphes [37], et permettent de lever ainsi des verrous scientifiques importants.

Dans ce chapitre, nous commençons par une introduction très générale à la cryptographie avec un bref historique, les objectifs de sécurité, la cryptographie symétrique ou asymétrique

(section 3.1). La section 3.2 est consacrée aux liens entre la cryptologie et les réseaux euclidiens : nous donnons des exemples de cryptanalyse et de construction de protocoles. Pour plus de détails sur la cryptologie à base des réseaux, nous renvoyons le lecteur vers les ouvrages [84, 90, 97, 80, 81]. Pour terminer, nous présentons en détail la méthode de Coppersmith qui constitue la base de nombreuses attaques sur le protocole RSA (section 3.3).

3.1 Introduction à la cryptologie

3.1.1 Bref historique de la cryptologie

La cryptologie se définit comme étant la science du secret. Son origine étymologique provient des mots grecs anciens “kruptos” qui signifie “secret” ou encore “caché” et “logos” pour le “discours” ou la “raison”. Les origines de la cryptologie remontent à plus de 4000 ans et l'on distingue généralement trois périodes. La première, appelée âge artisanal, s'étend des origines de la cryptologie jusqu'à la première guerre mondiale. Pendant cette période, les cryptosystèmes étaient adaptés aux capacités humaines avec des substitutions et des permutations relativement simples. Le plus vieux document chiffré est une tablette d'argile retrouvée en Irak et datant de -1600 avant notre ère mais les égyptiens pratiquaient déjà des permutations dans certains hiéroglyphes. Citons encore la méthode de César, le carré de Polybe ou le chiffrement Vigenère qui sont probablement les systèmes les plus connus de cette période.

L'âge technique s'étend de la première guerre mondiale au milieu des années 70. Les deux premières guerres mondiales ont entraîné un essor considérable de la cryptologie à la fois dans la conception mais aussi dans la cryptanalyse des protocoles. Les systèmes et les attaques s'appuient de plus en plus sur la technologie [1]. L'exemple le plus connu est probablement la machine Enigma utilisée par les allemands pendant la seconde guerre mondiale. Cette machine électromécanique a fait l'objet de nombreuses attaques par les alliés avec la construction de machines dédiées à sa cryptanalyse, plus connues sous le nom de “bombes”. L'âge technique marque aussi le début d'une standardisation des protocoles cryptographiques. En 1973, le bureau des standards américain lance un appel pour la création d'un protocole de chiffrement symétrique. Le processus aboutit en 1977 par l'adoption du protocole DES.

Jusqu'en 1976, deux personnes qui souhaitaient communiquer de manière sécurisée étaient au préalable obligées de se mettre d'accord sur une clé secrète commune. En 1976, Diffie et Hellman initie une nouvelle forme de cryptographie, dite asymétrique et proposent un protocole d'échange de clés sur une ligne non sécurisée. Ce qui est échangé sur la ligne est public mais à partir de ces éléments publics, il est difficile de retrouver la clé secrète partagée. C'est le début de l'âge paradoxal. Deux ans plus tard, Rivest, Shamir et Adleman créent le premier protocole de chiffrement asymétrique, le RSA, où chaque entité dispose d'une clé privée, gardée secrète, et d'une clé publique que tout le monde peut consulter. Les deux clés sont liées par une relation mathématique et comme précédemment, il est difficile de retrouver la clé privée à partir de la clé publique. Autrefois réservée aux activités militaires ou diplomatiques, la cryptographie asymétrique a permis la démocratisation des moyens de communication sécurisés. La cryptographie est omniprésente dans les communications par internet, les paiements dématérialisés, les signatures électroniques (qui ne sont pas possibles en cryptographie symétrique), les cartes à puces, le vote électronique, les puces RFID, etc. L'âge paradoxal correspond à une véritable innovation dans le domaine de la cryptologie et l'intense activité scientifique du domaine permet aux protocoles actuels de répondre à des objectifs de sécurité de plus en plus ambitieux.

1. Exception faite du code Navajo utilisé par les américains pendant la seconde guerre mondiale et qui utilisait le langage du peuple Navajo

3.1.2 Principaux objectifs de sécurité de la cryptologie

Pendant l'âge artisanal, la cryptologie avait essentiellement pour objectif de rendre confidentielles les informations transmises. Avec le développement des sciences et de la cryptographie asymétrique, les objectifs de sécurité se sont multipliés correspondant à chaque fois à des contextes d'utilisation particuliers. Les trois objectifs les plus couramment exigés sont la confidentialité, l'authenticité et l'intégrité.

La confidentialité garantit que le contenu d'une communication n'est pas accessible à tout autre personne que le destinataire légitime. Le chiffrement de l'information permet d'atteindre cet objectif. De manière générale, la personne qui souhaite envoyer un message applique une fonction dite de chiffrement et transmet le chiffré au destinataire. Ce dernier retrouve le message original en utilisant une fonction de déchiffrement.

L'intégrité des données garantit que le contenu d'une communication n'a pas été modifié de façon malveillante. Les fonctions de hachage "à sens unique" et "résistantes aux collisions" sont des outils de base pour assurer l'intégrité des messages. Le sens unique assure qu'il est difficile de changer la donnée associée à une empreinte (empreinte=valeur de la fonction de hachage) et la résistance aux collisions assure qu'il y a peu de chance de trouver deux messages avec la même empreinte.

L'authenticité garantit l'identité d'une entité donnée (identification) ou l'origine d'une communication (authentification). Les problèmes d'authentification et d'identification sont résolus à l'aide des signatures numériques, dont le principe a été introduit par Diffie et Hellman [29]. Le signataire possède un secret avec lequel il est le seul à pouvoir générer une signature valide. Les autres entités, utilisent la clé publique pour vérifier la validité de la signature sans que le signataire révèle son secret.

D'autres objectifs dépendent du contexte d'utilisation. La non-répudiation garantit qu'une personne ne peut pas nier avoir commis une action. C'est le cas lors d'une signature électronique où le signataire ne doit pas pouvoir nier avoir fait la signature.

Garantir l'anonymat est un objectif de plus en plus répandu. L'anonymat consiste à réaliser certaines actions sensibles sans qu'il soit possible de remonter à la source. On comprend l'intérêt de cet objectif dans des situations de crise politique mais il se montre aussi très utile pour éviter la traçabilité des activités des internautes.

Lors d'un vote électronique, la machine doit certifier le vote d'un citoyen sans en connaître le contenu puisque le vote doit rester secret. Le processus s'appuie sur la notion de signature aveugle que l'on retrouve aussi dans les systèmes de porte-monnaie électronique. Il s'agit de signer un document sans en connaître le contenu.

En outre, une machine à voter doit pouvoir effectuer le bilan des votes qui ne sont pas conservés en clair pour des raisons de "vote confidentiel". Le calcul à partir de données chiffrées se fait à l'aide de protocoles dits homomorphes.

Bien sûr, cette liste n'est pas exhaustive.

3.1.3 Cryptographie symétrique

La cryptographie symétrique ou à clés secrètes regroupe toutes les techniques de chiffrement qui utilisent une clé secrète pour chiffrer et soit une clé identique soit une clé déductible de la clé secrète pour déchiffrer. Les protocoles des deux premiers âges font partie de cette catégorie ainsi que le protocole AES, dernier standard de chiffrement symétrique par bloc. Tant que l'expéditeur et le destinataire connaissent la clé secrète, ils peuvent chiffrer et déchiffrer tous les messages qui utilisent cette clé.

On distingue deux types de chiffrement symétrique : les chiffrements par bloc et les chiffrements par flot. Les chiffrements par bloc agissent sur des données de taille fixée. Par exemple, le DES chiffre des blocs de 64 bits avec des clés de 56 bits [76] et son successeur, l'AES, chiffre des blocs de 128 bits ou 192 bits avec des clés ayant au plus 256 bits [24]. Pour les messages très longs (dépassant la taille d'un bloc), ces protocoles sont couplés avec des modes opératoires (CBC, ECB, OCB, etc) qui découpent en plusieurs morceaux le message d'origine et combinent ces morceaux avec la méthode de chiffrement pour obtenir un chiffré final.

Les chiffrements par flot peuvent traiter des données de longueur quelconque. Le message est considéré comme un flot de bits qui sont combinés un par un (en utilisant un ou exclusif) à un autre flot de bits pseudo-aléatoires. Le générateur pseudo-aléatoire (GPA) qui émet le deuxième flot de bits est initialisé avec la clé secrète. La difficulté majeure dans ces protocoles est de concevoir des GPA à la fois sûrs et efficaces. Étant par construction très rapides, on retrouve ces générateurs dans plusieurs applications de la vie quotidienne : WIFI, bluetooth, protocole SSL, etc.

De manière générale, les protocoles symétriques utilisent des opérations simples sur les bits et leur principal atout est leur grande rapidité. En revanche la mise en place de tels systèmes souffre de deux inconvénients majeurs. Le premier est que le secret doit être partagé. Ce n'est donc plus un secret pour tout le monde... L'autre défaut est l'explosion du nombre de clés nécessaires pour que plusieurs personnes communiquent de manière sécurisée. Pour n personnes, il faut $n(n-1)/2$ clés secrètes ce qui est impossible à mettre en place à l'échelle d'internet par exemple.

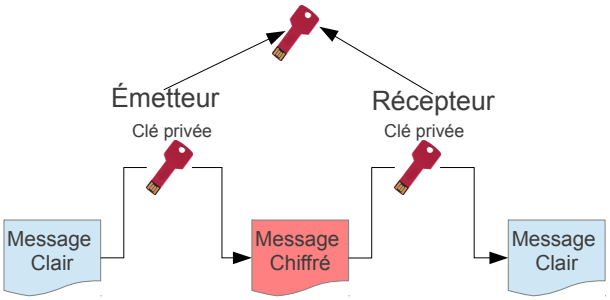


FIGURE 3.1: Principe d'un chiffrement symétrique

3.1.4 Principes de la cryptographie asymétrique

Les protocoles asymétriques sont basés sur des opérations mathématiques (exponentiations modulaires, additions sur des courbes elliptiques, etc.) qui sont beaucoup plus coûteuses en temps que les manipulations sur les bits des protocoles symétriques. Ce sont donc des protocoles "lents" mais qui ne souffrent pas des défauts des protocoles symétriques.

Un chiffrement asymétrique fonctionne avec une paire de clés : une clé publique qui est librement accessible par toute entité qui souhaite envoyer un message, et une seconde clé dite privée qui est gardée secrète. Les messages sont chiffrés à l'aide de la clé publique et peuvent uniquement être déchiffrés par celui qui possède la clé privée correspondante. Si n personnes souhaitent communiquer ensemble, il suffit de n couples de clés générés indépendamment les uns des autres, ce qui rend possible la mise en œuvre à grande échelle.

Le premier protocole asymétrique est un protocole d'échange de clés inventé par Diffie et

Hellman en 1976 [29]. Deux ans plus tard, Rivest, Shamir et Adleman proposent le premier protocole de chiffrement asymétrique, le RSA [98] que l'on retrouve dans les cartes à puces ou encore dans le protocole SSL. Nous détaillerons plus loin le protocole RSA dont de nombreuses cryptanalyses sont basées sur les réseaux et la méthode de Coppersmith.

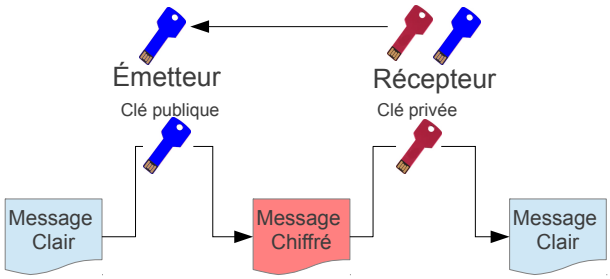


FIGURE 3.2: Principe d'un chiffrement asymétrique.

Dans la pratique, on peut combiner les avantages de la cryptographie symétrique et asymétrique. La cryptographie asymétrique est utilisée pour initier une communication et se mettre d'accord sur une clé de session qui sera ensuite utilisée au sein d'un protocole symétrique. La combinaison des deux familles permet des communications rapides avec un grand nombre d'entités.

3.1.5 Protocole RSA

De nombreux protocoles asymétriques ont été proposés depuis 1976. Le protocole Diffie-Hellman, le cryptosystème Elgamal [35] ou encore les protocoles basés sur les courbes elliptiques reposent tous sur la difficulté de résoudre le problème du logarithme discret. Dans cette thèse, nous nous intéressons à la réduction des réseaux et à leurs applications en cryptographie. En pratique, la réduction des réseaux, en lien avec la méthode de Coppersmith, se révèle très efficace pour attaquer les protocoles dont la sécurité repose non pas sur le problème du logarithme discret mais plutôt sur le problème de la factorisation. C'est le cas du protocole RSA que nous décrivons en détail dans cette section et qui interviendra à plusieurs reprises dans ce manuscrit.

Le système RSA a été inventé par Ron Rivest, Adi Shamir et Len Adleman [98] et fut présenté pour la première fois dans la chronique mathématique de Martin Gardner en août 1977. La sécurité du système RSA repose sur la difficulté de la factorisation d'un grand entier N qui est le produit de deux nombres premiers p et q. Nous décrivons maintenant les trois étapes principales du protocole : la génération des clés, le chiffrement et le déchiffrement.

On appelle e l'exposant de chiffrement et d l'exposant de déchiffrement. Bien entendu, seul la personne qui dispose de la clé privée d est capable de déchiffrer le message. Retrouver d à partir de e et N nécessite (naïvement) de calculer $\phi(N)$ ce qui est équivalent à factoriser N. Pour le voir, il suffit de remarquer que p et q sont les racines du polynôme

$$(X-p)(X-q) = X^2 - X(N+1-\phi(N)) + N.$$

Génération des clés

- choisir deux grands nombres premiers p et q et calculer $N = pq$.
- choisir e un entier premier avec

$$\phi(N) = (p-1)(q-1) = (N+1) - (p+q) \qquad \text{(indicateur d'Euler)},$$

- calculer l'inverse modulaire $d = e^{-1} \mod \phi(N)$.
- la paire (N, e) est la clé publique
- les entiers (p, q, d) forment la clé privée

Chiffrement

Pour chiffrer un message m avec $m < N$ et $\text{PGCD}(m, N) = 1$, on calcule :

$$c = m^e \mod N.$$

Déchiffrement

Pour déchiffrer un message c avec $c < N$ et $\text{PGCD}(c, N) = 1$, on calcule :

$$m = c^d \mod N.$$

Le protocole RSA est à la fois le plus ancien protocole de chiffrement asymétrique et l'un des protocoles les plus utilisés au monde. Ses variantes font l'objet d'une intense activité scientifique mais jusqu'à aujourd'hui, le RSA a résisté à toutes les attaques connues. La recherche dans le domaine de la cryptographie asymétrique ne s'arrête heureusement pas au RSA et la conception de systèmes cryptographiques à clés publiques de plus en plus performants et sûrs a mené les spécialistes du domaine à l'utilisation d'outils mathématiques de plus en plus sophistiqués. Une des raisons pour chercher des cryptosystèmes alternatifs est que les clés RSA deviennent de plus en plus longues. En effet selon la loi de Moore sur l'évolution de la puissance des ordinateurs, la taille minimale recommandée des clés augmente assez rapidement. Mais l'augmentation de la taille des clés ralentit le chiffrement et le déchiffrement et il se peut qu'un jour des protocoles plus efficaces à base de courbes elliptiques, de codes correcteurs d'erreurs ou de réseaux soient préférés. La recherche de remplaçants au protocole RSA est également motivée par l'algorithme de factorisation de Shor [104] qui sera efficace dès que l'on disposera des ordinateurs quantiques.

3.1.6 Cryptographie asymétrique : logarithme discret, courbes elliptiques et autres approches

Les protocoles Diffie-Hellman et RSA sont de natures très différentes. La sécurité du dernier repose sur la difficulté de la factorisation de grands entiers alors que celle du premier repose sur la difficulté du problème du logarithme discret décrit ci-dessous.

Logarithme discret : *Soit $(\mathbb{G}, \times)$ un groupe cyclique d'ordre q et g un générateur de ce groupe. Soit x un entier avec $x \in [1, q-1]$. Posons $y = g^x$. Le problème du logarithme discret consiste à retrouver x à partir de $(\mathbb{G}, \times)$, g et y.*

Dès qu'une structure de groupe cyclique existe, il est possible de définir un protocole d'échange de clés inspiré de Diffie-Hellman ou des protocoles de chiffrement à la Elgamal [35].

Parmi les groupes utilisés, nous pouvons citer le groupe des éléments inversibles d'un corps fini ou bien le groupe des points d'une courbe elliptique.

Cryptosystèmes avec des petites clés : La cryptographie sur les courbes elliptiques (ECC) [85, 57] permet de réduire significativement la taille des clés utilisées par rapport au RSA. A titre de comparaison, des clés de 160 et 256 bits sur les courbes elliptiques sont équivalentes à des clés de 1024 et 3072 bits pour le RSA. Même si les opérations de base (des additions sur une courbe elliptique) sont plus coûteuses que les opérations de base du RSA (des multiplications modulaires), les courbes elliptiques se montrent très compétitives en pratique.

Cryptosystèmes avec des opérations rapides : L'un des désavantages de la cryptographie asymétrique "classique" est l'utilisation d'arithmétique modulaire qui est bien plus lente que les opérations élémentaires des protocoles symétriques. L'idée est de concevoir des protocoles asymétriques utilisant une arithmétique binaire simple. La cryptographie à base de codes, du type McEliece [75], en est un exemple mais la taille des clés les rend actuellement inutilisables. La cryptologie à base de réseaux entre dans cette catégorie et fait l'objet de la prochaine section.

3.2 Cryptologie et réseaux euclidiens

Depuis une trentaine d'années, l'utilisation des réseaux et des algorithmes de réduction s'est largement développée en cryptologie. Dans un premier temps, les réseaux intervenaient surtout à travers la cryptanalyse de protocoles de type sac-à-dos ou RSA. Depuis le milieu des années 90, les réseaux sont à la base de plusieurs cryptosystèmes (Ajtai-Dwork [4], NTRU [49], GGH [43]...), de preuves de sécurité [2] et ont permis des avancées importantes dans la conception de protocoles complètement homomorphes [37].

Il s'avère que de manière un peu inattendue, beaucoup de systèmes cryptographiques sont liés aux réseaux euclidiens, soit parce que le système est directement fondé sur des problèmes de réseaux algorithmiquement difficiles (comme dans le cryptosystème d'Ajtai-Dwork [4] ou celui de Peikert [93]), soit parce que la cryptanalyse du système se ramène à un problème de réseau. En particulier, à chaque fois qu'un système est fondé sur un problème linéaire, ou facilement linéarisable, les réseaux sont un instrument de cryptanalyse redoutable : on transforme l'instance cryptographique en une instance de réseau et la cryptanalyse est fondée sur la possibilité de trouver un petit vecteur du réseau.

Bien que la taille des clés des protocoles basés sur les réseaux soit encore beaucoup trop grande pour les rendre compétitifs, l'étude des réseaux euclidiens cryptographiques présente un double intérêt. Tout d'abord les cryptosystèmes à base de réseaux sont "à priori" résistants aux ordinateurs quantiques. Depuis la découverte de l'algorithme de Shor pour la factorisation [104], beaucoup de tentatives visant à résoudre les problèmes difficiles des réseaux ont été faites, mais pour le moment aucune n'a abouti (voir Micciancio et Regev [83] pour plus de détails). La cryptologie à base de réseaux est donc une alternative crédible à l'apparition des ordinateurs quantiques.

Les réductions pire cas/cas moyen forment l'autre intérêt majeur de la cryptologie à base de réseaux. En cryptographie, un problème difficile dans le pire des cas n'est pas utilisable. Il faut que le problème reste difficile quand les clés sont choisies de manière aléatoire. La notion de problème "difficile en moyenne" est en fait très "naturelle" en cryptographie. A l'opposé, la notion de problème "difficile dans le pire des cas" est bien maîtrisée par la communauté informatique. Les réseaux permettent de relier ces deux notions avec des connexions "pire des

cas /cas moyen' qui n'existent pas pour les protocoles classiques. Ces connexions admettent la forme générale suivante :
Si le problème A est difficile dans le pire des cas, alors le problème B est difficile en moyenne.

En 1996, Ajtai [2] a démontré une première connexion "pire des cas/cas moyen" qui affirme : s'il n'existe pas d'algorithme polynomial qui résout pour tout réseau (même pour l'instance la plus difficile à réduire) le problème $\mathcal{SVP}$ décisionnel approximé à un facteur polynomial près, alors le problème $\mathcal{SVP}$ calculatoire exact est au moins aussi dur à résoudre pour des réseaux choisis aléatoirement. Dans sa preuve, Ajtai effectue une réduction entre les problèmes $\mathcal{SIVP}_\alpha$ et $\mathcal{SIS}$ (`Short Integer Solution`). Le premier problème consiste à trouver n vecteurs linéairement indépendants d'un réseau de longueur bornée par $\alpha\lambda_n(\mathcal{L})$. Le second, pour des entiers positifs n, m et q et pour toute matrice A à n lignes et m colonnes à coefficients dans $\mathbb{Z}_q$, consiste à trouver une petite solution $\mathbf{z} \in \mathbb{Z}^m$ non triviale du système linéaire $A\mathbf{z} = 0 \mod q$. Les bases qui interviennent sont dites de type $\mathcal{SIS}$ ou encore q-aires. Elles ont une forme particulière et sont représentées par la matrice

$$\mathcal{Q} = \left(\begin{array}{c|c} q \cdot I_{m-n} & 0 \\ \hline A' & I_n \end{array}\right).$$

où A' est reliée à A via l'écriture $A = (A'|I_n)$. Ajtai prouve qu'avec les paramètres $q \approx 2^n$ et $m = O(n^2)$, le problème de trouver le plus court vecteur de norme $\sqrt{m}$ dans un réseau uniformément choisi dans l'ensemble des réseaux de type $\mathcal{SIS}$ est au moins aussi difficile que de résoudre le problème $\mathcal{SIVP}_\alpha$ en dimension n avec $\alpha = O(n^3)$ dans le pire des cas. Le choix des paramètres a été ensuit raffiné ($q = \text{Poly}(n)$, $m = O(n \log n)$ et $\alpha = O(n \log n)$) en utilisant des distributions gaussiennes (voir [83, 94, 40]).

En 2005, Regev [96] obtient une connexion quantique entre les problèmes $\mathcal{SIVP}_\alpha$ et le problème LWE (`Learning With Errors`). Considérons n et q des entiers positifs. Étant donné un nombre arbitraire de paires indépendantes $(\mathbf{a}, \frac{1}{q}\langle \mathbf{a}|\mathbf{s}\rangle + e)$, le problème LWE consiste à trouver $\mathbf{s} \in \mathbb{Z}_q^n$. Le vecteur $\mathbf{a}$ est choisi uniformément et aléatoirement dans $\mathbb{Z}_q^n$ et e est un petit entier (le bruit) comparé à 1. Une connexion classique (non quantique) avec le problème LWE a également été obtenue par les auteurs de [15].

Les réductions Pire cas/cas moyen ont été l'occasion de créer des protocoles cryptographiques dont la preuve de sécurité repose sur des problèmes difficiles dans le pire des cas. Citons par exemple le protocole de Ajtai-Dwork [4] ou plus récemment ceux de Regev basés sur le problème LWE et Ring-LWE [96, 70]. Des protocoles comme NTRU, GGH, Merkle et Hellman [49, 43, 77] ne s'appuient pas sur une connexion pire cas/cas moyen mais leur conception repose sur la difficulté du problème $\mathcal{CVP}$. Récemment une variante de NTRU avec une preuve de sécurité a été proposé par Stehlé et Steinfeld [106]. Enfin, les réseaux ont permis de lever un verrou scientifique important avec la création d'un protocole complètement homomorphe proposé par Gentry [37] et amélioré ensuite par [39, 14, 38, 16, 17].

Les prochaines sections sont consacrées à la description de protocoles basés sur les réseaux ou dont la cryptanalyse utilise les réseaux. Ensuite, la méthode de Coppersmith est présentée en détail avec un exemple de cryptanalyse de RSA.

3.2.1 Cryptosystème de Merkle et Hellman

Le cryptosystème de Merkle et Hellman [77] est le premier exemple de cryptanalyse d'un protocole asymétrique utilisant les réseaux. La sécurité du protocole reposait sur le problème du sac-à-dos qui est un des 21 problèmes $\mathcal{NP}$–complets démontrés par Karp en 1972 [53].

Problème 3.1. *Soit $\mathcal{A} := \{a_1, a_2, \ldots, a_n\}$ un ensemble de n entiers positifs et S un entier. Existe-t-il un sous ensemble de $\mathcal{A}$ tel que la somme de ses éléments soit égale à S ? si oui indiquer ses éléments.*

Autrement dit, le problème de sac-à-dos consiste en la recherche d'une solution $(x_1, \ldots, x_n) \in \{0,1\}^n$ de l'équation

$$a_1 x_1 + \ldots + a_n x_n = S.$$

Certaines instances de ce problème sont faciles à résoudre. Lorsque les a_i forment une suite super-croissante, c'est-à-dire, pour tout i avec $2 \leq i \leq n$,

$$\sum_{j=1}^{i-1} a_j < a_i,$$

il suffit d'enlever de S les a_i successivement dans l'ordre décroissant pour retrouver les x_i. Précisément si a_i peut être soustrait, alors $x_i = 1$, sinon $x_i = 0$.

L'idée principale du cryptosystème de Merkle et Hellman est de "transformer" une instance facile du problème (avec une suite super-croissante) en une instance "générique" du problème, à priori difficile. L'instance facile correspond à la clé privée alors que l'instance générique sera la clé publique. Bien entendu, le lien entre les deux reste privé. Ce principe se retrouve beaucoup dans la cryptographie à base de codes ou à base de réseaux.

Description. Nous décrivons maintenant le protocole de Merkle et Hellman en commençant par la génération des clés.

Génération des clés

1 On choisit une suite super croissante $(a_1, a_2, \ldots, a_n)$
2 On choisit un entier N tel que $N > \sum_{i=1}^{n} a_i$
3 On choisit un autre entier d tel que $pgcd(d, N) = 1$
4 On calcule $e_i = da_i \mod N$ pour $1 \leq i \leq n$
La clé publique sera l'ensemble $(e_1, \ldots, e_n)$ et la clé secrète $(N, d, (a_1, \ldots, a_n))$.

A la fin de ce processus, la clé privée est un instance facile du problème du sac-à-dos. Si d est choisi aléatoirement et assez grand, les entiers $(e_1, \ldots, e_n)$ sont des entiers aléatoires de l'intervalle $[1, N]$ et ils forment une instance générique du problème de sac-à-dos.

Chiffrement

Pour chiffrer un message M, on transforme le message en une suite binaire $(x_1, \ldots, x_n) \in \{0,1\}^n$ et on calcule le chiffré $S = \sum_{i=1}^{n} x_i e_i$.

Déchiffrer le message S à partir de la clé publique $(e_1, \ldots, e_n)$ revient à résoudre le problème du sac-à-dos qui est à priori difficile.

Déchiffrement

Pour déchiffrer, on utilise la clé secrète $(N, d, (a_1, \ldots, a_n))$ et
on calcule $S' = Sd^{-1} \mod N$
- on résout l'équation $S' = \sum_{i=1}^{n} \widetilde{x}_i a_i$ (possible car $(a_1, \ldots, a_n)$ est une suite super-croissante)
- on a $(x_1, \ldots, x_n) = (\widetilde{x}_1, \ldots, \widetilde{x}_n)$.

Le protocole fonctionne puisque

$$S' = Sd^{-1} \mod N = d^{-1}\sum_{i=1}^{n} x_i e_i \mod N = d^{-1}\sum_{i=1}^{n} x_i d a_i \mod N = \sum_{i=1}^{n} x_i a_i \mod N.$$

La suite $(a_1, a_2, \ldots, a_n)$ est super croissante et S' a une unique présentation dans cette base. Pour tout i, on a alors $x_i = \widetilde{x}_i$.

Afin de complexifier les attaques, il est possible d'appliquer une permutation aux éléments du sac-à-dos pour masquer ceux qui proviennent des plus petits éléments de la clé secrète.

Attaques par réduction de réseaux. Nous montrons maintenant comment les réseaux peuvent être utiles pour attaquer ce type de cryptosystèmes. L'attaque que nous présentons ne correspond pas à un "cassage total" du protocole, autrement dit, on ne retrouve pas la clé secrète à partir des éléments publics. En fait, il s'agit d'une attaque sur la fonction à sens unique de chiffrement. Précisément, il est possible de retrouver le message clair à partir du message chiffré sans forcément avoir la clé secrète.

La première attaque contre le protocole de Merkle et Hellman est due à Shamir en 1982 [103], qui s'appliquent sous certaines conditions sur la taille des éléments de la clé privée et de la clé publique. En 1985, Lagarias et Odlyzko [62] proposent une attaque "générique" basée sur les réseaux contre le problème du sac-à-dos.

Étant donné une instance $(S, (a_1, \ldots, a_n))$ du problème, on définit le réseau $\mathcal{L}(B)$ généré par les lignes de la matrice $B = (\mathbf{b}_1, \ldots, \mathbf{b}_{n+1})$ suivante

$$B := \begin{pmatrix} SC & 0 & \cdots & 0 \\ a_1 C & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_n C & 0 & \cdots & 1 \end{pmatrix}$$

avec C une constante positive qui, choisie suffisamment grande, garantira le succès de méthode. S'il existe un vecteur $\mathbf{x} := (x_1, x_2, \ldots, x_n)$ de $\{0,1\}^n$ tel que $S = \sum_{i=1}^{n} x_i a_i$, alors le vecteur

$$\mathbf{v} = (0, x_1, \ldots, x_n) = \sum_{i=2}^{n+1} x_{i-1}\mathbf{b}_i - \mathbf{b}_1$$

est un vecteur de $\mathcal{L}(\mathcal{B})$ de longueur $\|\mathbf{v}\| \leq \sqrt{n+1}$. Si C est choisi de la forme $C = 2^{n^2}$, l'inégalité de Minkowski montre que la plupart des vecteurs ont une norme exponentielle en n alors que celle de $\mathbf{v}$ est d'ordre $\sqrt{n}$. Le vecteur $\mathbf{v}$ est donc un vecteur *très* court du réseau $\mathcal{L}$ et l'algorithme LLL appliqué à la base B produira une bonne approximation de ce vecteur. Cependant, Lagarias et Odlyzko [62] sont allés plus loin. Ils ont montré que lorsque la densité du sac-à-dos définie par

$$\delta = \frac{n}{\max_i \log_2 a_i}$$

est suffisamment faible (i.e. $\delta \leq 2(\log 2)/(\log \pi e) \approx 0.646$), le vecteur cible $\mathbf{v}$ est avec une forte probabilité le plus court vecteur du réseau $\mathcal{L}$. De plus, si la densité est d'ordre au plus $1/n$, l'algorithme LLL trouve le vecteur. Il est alors possible de résoudre le problème de sac-à-dos dès que la densité est faible.

Une meilleure densité a été obtenue par Coster et al. [23] en considérant le réseau

$$\widetilde{\mathcal{B}} := \begin{pmatrix} SC & \frac{1}{2} & \cdots & \frac{1}{2} \\ a_1 C & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_n C & 0 & \cdots & 1 \end{pmatrix}.$$

Cette fois, le vecteur cible est $\widetilde{\mathbf{v}} = (x_1 - \frac{1}{2}, \ldots, x_n - \frac{1}{2}, 0)$. Il a été démontré lorsque la densité est inférieure à 0.9408..., le vecteur $\widetilde{\mathbf{v}}$ est avec une grande probabilité un plus court vecteur du réseau $\mathcal{L}(\widetilde{\mathcal{B}})$.

Même si, depuis, plusieurs évolutions du protocoles ont été proposées, elles ont été presque toutes cassées [91].

3.2.2 Cryptosystème d'Ajtai et Dwork

La section précédente montre que les réseaux sont des outils efficaces de cryptanalyse. Le cryptosystème d'Ajtai et Dwork [4], créé en 1997, est un exemple où les réseaux interviennent à la fois dans la conception, dans la preuve de sécurité et dans la cryptanalyse du protocole. En 1996, Ajtai [2] démontre la première connexion pire des cas/cas moyen. Un an plus tard, Ajtai et Dwork [4] proposent un cryptosystème inspiré par le travail d'Ajtai et prouvent que le protocole obtenu est sémantiquement sûr sous l'hypothèse que le problème unique-SVP est difficile dans le pire des cas. Le problème unique-SVP consiste à trouver un vecteur le plus court $\mathbf{v}$ tel qu'il est polynomialement plus court que n'importe quel autre élément du réseau non colinéaire à $\mathbf{v}$. Ce chiffrement probabiliste à une faible probabilité d'erreur pendant le déchiffrement, mais il existe une version qui corrige cette erreur [42].

Description. La description du protocole nécessite plusieurs définitions. Soit $(\mathbf{w}_1, \ldots, \mathbf{w}_n)$ n vecteurs linéairement indépendants. On appelle $P(\mathbf{w}_1, \ldots, \mathbf{w}_n)$ le parallélépipède fondamental engendré par les vecteurs $(\mathbf{w}_1, \ldots, \mathbf{w}_n)$, c'est-à-dire l'ensemble des combinaisons linéaires réelles de $(\mathbf{w}_1, \ldots, \mathbf{w}_n)$ à coefficients dans $[0, 1[$. L'épaisseur de $P(\mathbf{w}_1, \ldots, \mathbf{w}_n)$ correspond au minimum sur i de la distance euclidienne entre $\mathbf{w}_i$ et l'hyperplan engendré par les autres $\mathbf{w}_j$. On désigne par B_n l'hypercube de $\mathbb{R}^n$ de coté $\rho_n = 2^{n \log n}$, et par S_n la boule de $\mathbb{R}^n$ de rayon n^{-8} et $m = n^3$.

La clé publique est formée de $(m + n)$ vecteurs proches du réseau $\mathcal{L}_u = \{\mathbf{x} : \langle \mathbf{x} | \mathbf{u} \rangle \in \mathbb{Z}\}$. La taille de la clé publique peut correspondre en pratique à plusieurs gigaoctets !

Génération des clés

La clé privée est un vecteur aléatoire $\mathbf{u}$ de la boule unité. La clé publique est constituée de $(m + n)$ vecteurs $(\mathbf{w}_1, \ldots, \mathbf{w}_n, \mathbf{v}_1, \ldots, \mathbf{v}_m)$ tirés aléatoirement dans B_n de la manière suivante, avec la contrainte que l'épaisseur de $P(\mathbf{w}_1, \ldots, \mathbf{w}_n)$ soit au moins $n^{-2}\rho_n$:

1. Choisir un vecteur $\mathbf{a}$ uniformément dans $\{\mathbf{x} \in B_n : \langle \mathbf{x} | \mathbf{u} \rangle \in \mathbb{Z}\}$.
2. Choisir aléatoirement n vecteurs $(\delta_\mathbf{1}, \ldots, \delta_\mathbf{n})$ dans S_n.
3. Retourner le vecteur $\mathbf{x} = \mathbf{a} + \sum_{i=1}^{n} \delta_\mathbf{i}$.

Chiffrement

Le chiffrement s'effectue bit à bit. Un bit est chiffré sous la forme d'un vecteur de $\mathbb{R}^n$.

bit 1 : tirer aléatoirement un vecteur $\mathbf{c}$ dans le parallélépipède $P(\mathbf{w}_1, \ldots, \mathbf{w}_n)$

bit 0 :

(a) choisir uniformément $\mathbf{b}_1, \ldots, \mathbf{b}_m \in \{0, 1\}$

(b) calculer $\mathbf{c} = \sum_{i=1}^{m} \mathbf{b}_i \mathbf{v}_i \mod P(\mathbf{w}_1, \ldots \mathbf{w}_n)$.

(c) retourner le chiffré $\mathbf{c}$.

Intuitivement, si on tire aléatoirement un vecteur dans $P(\mathbf{w}_1, \ldots, \mathbf{w}_n)$, ce vecteur a peu de chance d'être proche du réseau $\mathcal{L}(\mathbf{w}_1, \ldots, \mathbf{w}_n)$ qui est lui-même proche du réseau $\mathcal{L}_u$. En revanche, lorsque le bit à chiffrer est 0, le vecteur construit est proche du réseau $\mathcal{L}_u$. Ceci justifie la procédure de déchiffrement suivante.

Déchiffrement

Pour déchiffrer $\mathbf{c}$ avec la clé privée $\mathbf{u}$, on calcule $\tau = \langle \mathbf{c}|\mathbf{u} \rangle$ et on vérifie si $|\tau - \lfloor \tau \rceil| \leq \frac{1}{n}$. Si la condition est vraie alors $\mathbf{c}$ est déchiffré en 0, sinon il est déchiffré en 1.

Un "mauvais aléa" pour le chiffrement d'un bit à 1 peut conduire à un déchiffrement en 0 mais il est possible de corriger cette erreur [42].

Attaques par réduction de réseaux. Ajtai et Dwork [4] ont prouvé que le protocole obtenu est sémantiquement sûr sous l'hypothèse que le problème unique-SVP est difficile dans le pire des cas. Pourtant, Nguyen et Stern [89] ont construit une attaque à base de réduction de réseaux. L'attaque ne remet pas en cause les travaux d'Ajtai et Dwork mais le paramètre n doit être choisi de l'ordre de plusieurs centaines. La clé publique obtenue ferait alors plusieurs gigaoctets ce qui rend inutilisable le protocole.

L'idée de Nguyen et Stern est de chercher des combinaisons linéaires très courtes entre les vecteurs de la clé publique. Plus précisément, les auteurs montrent qu'il est possible d'obtenir des informations sur les valeurs V_i des entiers les plus proches de $< \mathbf{v}_i, \mathbf{u} >$ en utilisant $\mathcal{L}(B)$, un réseau m-dimensionnel de $\mathbb{R}^{n+m}$ engendré par les lignes de la matrice

$$B := \begin{pmatrix} \beta\mathbf{v}_1 & 1 & 0 & \cdots & 0 \\ \beta\mathbf{v}_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \beta\mathbf{v}_m & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Les auteurs obtiennent le résultat suivant.

Théorème 3.2. *Soit* $\mathbf{x} = (\beta(\lambda_1\mathbf{v}_1 + \ldots + \lambda_m\mathbf{v}_m), \lambda_1, \ldots, \lambda_m)$ *un vecteur de* $\mathcal{L}(B)$*, les* λ_i *étant des entiers. Si*

$$n^7 \left\| \sum_{i=1}^{m} \lambda_i \mathbf{v}_i \right\| + \sum_{i=1}^{m} |\lambda_i| < n^7$$

alors

$$\sum_{i=1}^{m} \lambda_i V_i = 0$$

pour $\beta^2 \geq \frac{n^{14}}{2n^7-1}$ *et* $\|\mathbf{x}\| < \frac{n^4}{\sqrt{2n^7-1}}$.

On désigne par $\mathbf{V}$ le vecteur de $\mathbb{Z}^m$ qui a pour coordonnées les V_i et par $\mathcal{L}(\mathbf{V})^\perp = \{\mathbf{x} \in \mathbb{Z}^m \mid \forall \mathbf{v} \in \mathcal{L}(\mathbf{V}), < \mathbf{x}, \mathbf{v} >= 0\}$ le réseau orthogonal de $\mathbf{V}$. Le théorème montre qu'il est possible d'associer les vecteurs suffisamment courts de $\mathcal{L}(B)$ à des vecteurs du réseau orthogonal $\mathcal{L}(\mathbf{V})^\perp$. En outre, les auteurs montrent aussi que $\mathcal{L}(B)$ contient beaucoup de vecteurs suffisamment courts. Pour trouver ces vecteurs, on réduit le réseau (pour m trop grand, on ne garde que certaines lignes choisies aléatoirement). Si on connaît $m-1$ vecteurs de $\mathcal{L}(\mathbf{V})^\perp$ linéairement indépendants, alors on peut en déduire le vecteur $\mathbf{V}$ et donc la clé secrète $\mathbf{u}$.

Même si le protocole est cassé, le cryptosystème d'Ajtai et Dwork est historiquement très important puisqu'il est le premier dont la preuve de sécurité repose sur un problème dans le pire

des cas. Nous présentons maintenant un autre protocole, basé (indirectement) sur les réseaux, et qui a pour l'instant résisté aux attaques.

3.2.3 Cryptosystème NTRU

Le cryptosystème NTRU a été présenté pour la première fois à la conférence Crypto'96 et fut ensuite publié en 1998 [49]. Le système utilise l'arithmétique sur les anneaux de polynômes $\mathcal{P} = \mathbb{Z}[X]/(X^N - 1)$, mais il peut aussi être considéré comme un système à base de réseaux. Les paramètres principaux sont deux entiers p et q premiers entre eux et N un nombre entier. Le paramètre p peut aussi être un petit polynôme mais dans tous les cas, p (resp. q) est inversible dans $\mathcal{P} \mod q$ (resp. $\mathcal{P} \mod q$).

Les inverses de f dans $\mathcal{P} \mod q$ et $\mathcal{P} \mod p$ sont notés respectivement $f_p = f^{-1} \mod p$ et $f_q = f^{-1} \mod q$. La procédure de calcul des inverses dans les anneaux $\mathcal{P}_p = \mathbb{Z}_p[X]/(X^N - 1)$ et $\mathcal{P}_q = \mathbb{Z}_q[X]/(X^N - 1)$ est assez longue. La génération des clés utilisant ces inverses, la première phase du protocole est lente ce qui constitue un inconvénient important.

Description. Dans ce qui suit, $*$ est le produit de deux polynômes dans $\mathbb{Z}[X]/(X^N - 1)$.

Génération des clés

- Choisir deux polynômes f et g aléatoirement dans des ensembles prédéfinis de polynômes dont les coefficients sont petits.
- Calculer l'inverse f_q de f dans $\mathcal{P}_q$.
- Calculer l'inverse f_p de f dans $\mathcal{P}_p$.
- Calculer $h = g * f_q \mod q$ dans $\mathcal{P}_q$. La clé secrète est $(f,\ f_p)$ et la clé publique est (h, p, q).

Le polynôme g est un aléa qui permet de cacher la clé secrète f. Lors du chiffrement, le message m est un polynôme dont les coefficients sont petits (0 ou 1 par exemple). Dans la procédure décrite ci-dessous, le message joue le rôle d'une petite erreur qui a été ajoutée à un calcul "principal".

Chiffrement

Le message à chiffrer est représenté par un polynôme m. Choisir un petit polynôme aléatoire r et calculer ensuite

$$e = p * r * h + m \mod q \tag{3.1}$$

La procédure de déchiffrement est la suivante.

Déchiffrement

Pour déchiffrer le message e, calculer

$$a = f * e \mod q \tag{3.2}$$

suivi de

$$m = f_p * a \mod p \tag{3.3}$$

En faisant attention aux structures algébriques sous-jacentes, il est possible de vérifier que le protocole est correct. Nous allons voir que la clé privée correspond à un vecteur court d'un réseau alors que le problème de déchiffrement se ramène au problème $\mathcal{CVP}$.

Lien avec les réseaux et attaques connues. On considère le réseau $\mathcal{L}$ engendré par les lignes de la matrice $A(q,h)$ de $\mathbb{R}^{2N\times 2N}$. Cette matrice est constituée 4 blocs : une matrice identité multipliée par le coefficient q, une matrice nulle, une matrice circulante et une matrice identité. Les h_i sont les coefficients entiers du polynôme h.

$$\left(\begin{array}{cccc|cccc} q & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & q & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & q & 0 & 0 & \dots & 0 \\ \hline h_1 & h_2 & \dots & h_N & 1 & 0 & \dots & 0 \\ h_N & h_1 & \dots & h_{N-1} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ h_2 & h_3 & \dots & h_1 & 0 & 0 & \dots & 1 \end{array}\right) \tag{3.4}$$

Chaque ligne de la matrice circulante correspond aux coefficients du polynôme $X^i h \mod X^N-1$. Combiner ces lignes revient à calculer le produit de h par un autre polynôme qui peut se lire (après combinaison des lignes) sur la partie inférieure droite de la matrice (à la place de la matrice identité). Le premier bloc de lignes peut ensuite être utilisé pour effectuer le modulo q. Autrement dit, le réseau $\mathcal{L}$ est l'ensemble des vecteurs (u,v) tels que $h*v-u=0 \mod q$. Comme $h*f=g \mod q$, la clé privée (f,g) est un vecteur court de ce réseau dès lors que f et g ont de petits coefficients (par exemple -1, -0 et 1). En partant de cette construction, Coppersmith et Shamir [22] ont proposé une méthode qui retrouve la clé privée pour de petites valeurs du paramètre N en utilisant par exemple l'algorithme LLL.

Le problème de déchiffrement peut aussi être vu comme un problème de plus proche vecteur. En effet, le chiffré d'un petit message m est $e=p*r*h+m \mod q$, autrement dit, e est proche de $p*r*h \mod q$. Déchiffrer le message revient à chercher un vecteur (u,v) avec u proche de e puis à calculer $m=e-u \mod q$.

Même si la sécurité de NTRU repose sur les problèmes $\mathcal{CVP}$ et $\mathcal{SVP}$ dans une classe particulière de réseaux, le protocole ne possède pas de preuve de sécurité. Récemment Stehlé et Steinfeld [106] ont proposé une preuve de sécurité d'une variante de NTRU, qui remplace l'anneau $\mathbb{Z}[X]/(X^N-1)$ avec $\mathbb{Z}[X]/(X^N+1)$.

3.3 Méthode de Coppersmith et Cryptanalyse de RSA

En 1996, Coppersmith [20] propose une méthode générale qui utilise l'algorithme LLL pour trouver les petites racines d'un polynôme f univarié ou multivarié modulo un entier. Pour des paramètres $X_1,\dots,X_m$, un grand entier N et $b>N^\beta$ avec $\beta\in]0,1]$, les équations traitées sont de la forme

$$f(x_1,x_2,\dots,x_m)=0 \mod b \quad \text{avec} \quad |x_1|\le X_1, |x_2|\le X_2 \dots |x_m|\le X_m.$$

La méthode de Coppersmith remplace la résolution de l'équation modulaire par la résolution d'une équation polynomiale sur $\mathbb{Z}$ de la forme $g(x_1,\dots,x_m)=0$. Le polynôme g dépend du polynôme f et vérifie $g(x_1,\dots,x_m)=0$ dès que $(x_1,\dots,x_m)$ est assez petit et $f(x_1,\dots,x_m)=0 \mod b$. Coppersmith construit le polynôme g à l'aide d'une famille de polynômes $f_1,f_2,\dots,f_n$ dont les racines sont celles de f mais modulo b^m pour un m fixé. En combinant linéairement ces polynômes, l'algorithme LLL donne un polynôme g "petit" qui vérifie la relation $|g(\mathbf{x})|\le b^m$

pour $\mathbf{x}$ une racine de f. Par construction, on a aussi $g(\mathbf{x}) = 0 \mod b^m$ ce qui donne $g(\mathbf{x}) = 0$ dans $\mathbb{Z}$.

La méthode de Coppersmith avec $m = 1$ ou $m = 2$ est très utile pour la cryptanalyse de variantes "affaiblies" du protocole RSA. Dans cette section, nous commençons par décrire la méthode pour $m = 1$ et $m = 2$ en mettant l'accent sur les réseaux sous-jacents. Ensuite, nous présentons l'attaque de Boneh et Durfee [12] sur le cryptosystème RSA.

3.3.1 Méthode de Coppersmith univariée

Dans [20], puis dans [21], Coppersmith décrit une méthode générale qui trouve une petite racine d'un polynôme $f(x)$ univarié modulo un entier b dont la factorisation n'est pas connue. Il améliore ainsi grandement la méthode proposée en 1988 par Vallée, Girault et Toffin, décrite au chapitre 2, ou celle proposée par Hastad [46]. Par simplicité, il est supposé que f est unitaire et irréductible de degré n

$$f(x) = x^n + a_{n-1}x^{n-1} + \dots a_1 x + a_0.$$

Plus tard, la méthode a été améliorée par Howgrave-Graham [50]. Le lemme suivant donne deux critères pour choisir un polynôme g qui a comme racines sur $\mathbb{Z}$ les petites racines de f.

Lemme 3.3. *(Howgrave-Graham) Soit $g(x) \in \mathbb{Z}[x]$ un polynôme en une variable, somme d'au plus w monômes et m un entier positif. On suppose que*

(1) $g(x_0) = 0 \mod b^m$ *et* $|x_0| \leq X$

(2) $\|g(xX)\| \leq \frac{b^m}{\sqrt{w}}$.

Alors $g(x_0) = 0$.

Preuve. Les inégalités

$$|g(x_0)| = |\sum_i a_i x_0^i| \leq \sum_i |a_i x_0^i| \leq \sum_i |a_i| X^i \leq \sqrt{w}\|g(xX)\| < b^m$$

montrent que l'égalité $g(x_0) = 0 \mod b^m$ entraîne l'égalité $g(x_0) = 0$. □

L'objectif est maintenant de construire un polynôme satisfaisant les deux conditions du lemme avec x_0 une petite racine de f modulo b. Coppersmith utilise la famille redondante de polynômes suivante

$$g_{i,k}(x) = x^i f(x)^k b^{m-k} \quad \text{pour } i = 0, \dots, n-1 \quad \text{et} \quad k = 0, \dots, m.$$

Pour tout i et k, les égalités $g_{i,k}(x_0) = 0 \mod b^m$ sont vraies par construction même. En particulier, toute combinaison linéaire entière des $g_{i,k}$ satisfait la condition (1) du lemme. L'algorithme LLL intervient pour construire une "petite" combinaison linéaire qui satisfait la condition (2). Pour cela, il suffit de considérer le réseau $\mathcal{L}$ engendré par les vecteurs dont les composantes sont les coefficients des $g_{i,k}(xX)$. La matrice $\mathcal{M}$ obtenue est composée de $(m+1)$ blocs triangulaires inférieurs contenant n vecteurs. Par exemple, pour $m = 2$ et $n = 3$, nous obtenons

$$\mathcal{M} = \left(\begin{array}{ccccccccc} b^2 & & & & & & & & \\ - & b^2X & & & & & & & \\ - & - & b^2X^2 & & & & & & \\ \hline - & - & - & bX^3 & & & & & \\ - & - & - & - & bX^4 & & & & \\ - & - & - & - & - & bX^5 & & & \\ \hline - & - & - & - & - & - & X^6 & & \\ - & - & - & - & - & - & - & X^7 & \\ - & - & - & - & - & - & - & - & X^8 \end{array}\right).$$

Comme la matrice est triangulaire inférieure, le déterminant du réseau est facile à calculer et dépend de m, n, b et X. En appliquant la proposition 2.8 qui borne le premier vecteur en sortie de LLL en fonction du déterminant du réseau, une majoration de la norme du "petit" polynôme obtenu est connue et en réglant X et m de manière optimum, il est possible de satisfaire la condition 2 du lemme. Coppersmith obtient le résultat suivant.

Théorème 3.4 (Coppersmith)**.** *Soit f un polynôme unitaire, univarié et de degré n sur $\mathbb{Z}[x]$. Soit b un entier dont la factorisation est inconnue. En temps polynômial en $\ln b$ et n, il est possible de trouver toutes les solutions x_0 de $f(x) = 0 \mod b$ avec $|x_0| \leq b^{1/n}$.*

3.3.2 Méthode de Coppersmith à deux variables

La méthode de Coppersmith bivariée est très similaire au cas univarié. C'est une méthode très utilisée en cryptographie lorsqu'il est possible de relier un élément secret (clé ou message clair) à un élément public (clé ou message chiffré) à travers une équation polynomiale. La méthode trouve les petites racines d'une équation de la forme

$$f(x, y) = 0 \mod b,$$

où $f(x, y)$ est un polynôme bivarié. Comme dans le cas univarié, l'idée est de construire une famille de polynômes dont les racines satisfont la même équation mais modulo b^m. A partir de cette famille, deux "petits" polynômes bivariés $f_1(x, y)$ et $f_2(x, y)$ sont construits avec $f_1(x_0, y_0) = f_2(x_0, y_0) = 0$ lorsque (x_0, y_0) est une petite racine de f. Pour retrouver les racines, on peut se ramener au cas univarié dans $\mathbb{Z}$ par un calcul de résultant et en résolvant l'équation $\mathrm{Res}(f_1, f_2) = 0$.

Considérons un polynôme $h(x, y) = \sum_{i,j} a_{i,j} x^i y^j$. La norme de h est définie par $\|h(x, y)\|^2 = \sum_{i,j} |a_{i,j}^2|$. Le lemme suivant montre que si un polynôme $h(x, y)$ a une petite norme, alors chaque petite racine de $h(x, y)$ modulo un grand nombre est aussi une racine de $h(x, y)$ sur les entiers.

Lemme 3.5. [Howgrave - Graham] *Soit un polynôme $h(x, y) \in \mathbb{Z}[x, y]$, somme d'au plus w monômes, et qui vérifie :*

(1) *$h(x_0, y_0) = 0 \mod b^m$, pour un entier m avec $|x_0| < X$, $|y_0| < Y$*

(2) *$\|h(xX, yY)\| < \frac{b^m}{\sqrt{w}}$*

Alors $h(x_0, y_0) = 0$.

Preuve. Les inégalités

$$h(x_0, y_0) = \left|\sum_{i,j} a_{i,j} x_0^i y_0^j\right| = \left|\sum_{i,j} a_{i,j} X^i Y^j \left(\frac{x_o}{X}\right)^i \left(\frac{y_o}{Y}\right)^j\right|$$

$$\leq \sum_{i,j} \left| a_{i,j} X^i Y^j \left(\frac{x_o}{X}\right)^i \left(\frac{y_o}{Y}\right)^j \right| \leq \sum_{i,j} |a_{i,j} X^i Y^j| \leq \sqrt{w}\|h(xX, yY)\| < b^m$$

montrent que l'égalité $h(x_0, y_0) = 0 \mod b^m$, entraîne $h(x_0, y_0) = 0$. □

En s'inspirant du cas univarié, une famille de polynômes associée au polynôme initial $f(x, y)$ est donnée par les $g_{i,k}$ et $h_{i,k}$ suivants :

$$g_{i,k}(x, y) = x^i f^k(x, y) e^{m-k}, \tag{3.5}$$

$$h_{j,k}(x, y) = y^j f^k(x, y) e^{m-k}, \tag{3.6}$$

avec $k \in [0\,..\,m]$, $i \in [0\,..\,m-k]$ et $j \in [0\,..\,t]$ où t est un paramètre entier, fixé par la suite. Si (x_0, y_0) vérifie l'équation $f(x_0, y_0) = 0 \mod b$, alors pour tout i, j et k nous avons $g_{i,k}(x_0, y_0) = h_{j,k}(x_0, y_0) \mod b^m = 0 \mod b^m$. Les polynômes $g_{i,k}(x, y)$ sont appelés x-shifts et les polynômes $h_{j,k}(x, y)$ y-shifts.

Comme dans le cas univarié, toute combinaison linéaire entière de ces polynômes a pour racine (x_0, y_0) modulo b^m et l'idée est d'appliquer LLL pour trouver deux "petits" polynômes qui satisfont en plus l'hypothèse 2 du lemme d'Howgrave-Graham. Pour cela, nous introduisons le réseau $\mathcal{L}_{m,t}$ engendré par les vecteurs dont les composantes sont les coefficients des polynômes $g_{i,k}(xX, yY)$ et $h_{j,k}(xX, yY)$. Décrire le réseau obtenu pour une fonction f générale est fastidieux et n'est pas vraiment notre propos, en lien avec la cryptologie. Toutefois, la prochaine section présente la matrice obtenue lors de la cryptanalyse d'une variante de RSA.

La matrice $\mathcal{M}$ est triangulaire inférieure et le déterminant du réseau est facilement calculable. Une borne sur la norme des vecteurs en sortie de LLL est alors connue et en réglant X, Y de manière optimale, il est possible de satisfaire la condition (2) du lemme. La méthode reste heuristique car il n'est pas possible de s'assurer que le résultant, nécessaire pour se ramener à une équation dans $\mathbb{Z}$, donne des solutions non triviales.

3.3.3 Cryptanalyse de RSA

La méthode de Coppersmith [21, 12, 74] est à la base de plusieurs cryptanalyses contre le cryptosystème RSA lorsque l'exposant secret de déchiffrement est petit. L'exponentiation modulaire est une opération coûteuse et un petit exposant est très intéressant pour accélérer le déchiffrement. En 1990, Wiener [116] a montré en utilisant les développements en fractions continues qu'il est possible de retrouver l'exposant d si $d < N^{0.25}$. En 1997, Verheul et Tilborg [115] prouvent qu'il est possible de retrouver d plus vite que par recherche exhaustive si $d < N^{0.5}$. Finalement en 1999, Boneh et Durfee [12] montrent que si $d < N^{0.292}$, la méthode de Coppersmith et l'algorithme LLL donnent l'exposant secret en temps polynomial.

On rappelle que le cryptosystème RSA utilise comme clé publique un grand nombre $N = p.q$, produit de deux nombres premiers p et q de taille voisine. La clé publique de chiffrement est un entier e premier avec $\phi(N)$ et la clé privée est l'inverse de e dans $\mathbb{Z}_{\phi(N)}$. On suppose $\gcd(p-1, q-1) = 2$ et puisque e et d sont impairs, il vient

$$ed = 1 \mod \frac{\phi(N)}{2}.$$

Il existe alors un entier k tel que

$$ed + k\,(A + s) = 1, \qquad s = -\frac{p+q}{2}, \quad A = \frac{N+1}{2}, \tag{3.7}$$

autrement dit

$$k(A+s) = 1 \mod e. \tag{3.8}$$

On écrit $e = N^\alpha$ ($\alpha < 1$) et suppose que l'exposant de déchiffrement d est borné par N^δ ($\delta < 1$). D'après l'équation 3.7, nous avons

$$|k| < \frac{2de}{\phi(N)} \leq \frac{3de}{N} < 3e^{1+\frac{\delta+1}{\alpha}}. \tag{3.9}$$

De la même façon, si p et q ont le même nombre de bits, ils sont inférieurs à $2\sqrt{N}$ et s est majoré par

$$|s| < 2N^{0.5} = 2e^{\frac{1}{2\alpha}}. \tag{3.10}$$

En posant

$$f(x,y) = x(A+y) - 1,$$

l'équation 3.8 indique que le couple $(x,y) = (k,s)$ est solution de l'équation $f(x,y) = 0 \mod e$. Lorsque e est de l'ordre de N, α est proche de 1 et en ignorant les petites constantes des équations 3.9 et 3.10, nous sommes amenés à trouver les couples (x_0, y_0) tels que

$$f(x_0,y_0) = 0 \mod e \qquad \text{avec} \qquad |x_0| < e^\delta \quad \text{et} \quad |y_0| < e^{\frac{1}{2}}.$$

Étant donnés deux nombres m et t, on applique la méthode de Coppersmith. Le réseau $\mathcal{L}_{m,t}$ sous-jacent est donné par les vecteurs dont les composantes sont les coefficients des polynômes $g_{i,k}(xX, yY)$ et $h_{j,k}(xX, yY)$ pour $k = 0, \ldots, m$, $i = 0, \ldots, m-k$ et $j = 0, \ldots, t$. La matrice définie par ces vecteurs admet pour forme générale :

$$\begin{pmatrix} C_x & 0 \\ - & C_y \end{pmatrix} \qquad \text{avec}$$

$$C_x = \begin{pmatrix}
e^m & & & & & & & \\
- & e^m X & & & & & & \\
- & - & e^{m-1}XY & & & & & \\
- & - & \ddots & & & & & \\
- & - & - & - & e^m X^m & & & \\
- & - & - & - & - & e^{m-1}X^m Y & & \\
- & - & - & - & - & \ddots & & \\
- & - & - & - & - & - & X^m Y^m
\end{pmatrix}$$

$$C_y = \begin{pmatrix}
e^m Y & & & & & & & & & \\
- & e^m Y^2 & & & & & & & & \\
- & - & \ddots & e^m Y^t & & & & & & \\
- & - & - & - & e^{m-1}XY^2 & & & & & \\
- & - & - & - & - & e^{m-1}XY^3 & & & & \\
- & - & - & - & - & - & \ddots & e^{m-1}XY^{t+1} & & \\
 & & & & & & & \ddots & & \\
- & - & - & - & - & - & - & - & X^m Y^{m+1} & \\
- & - & - & - & - & - & - & - & \ddots & X^m Y^{m+t}
\end{pmatrix}.$$

C'est une matrice triangulaire dont le déterminant dépend de e, m, X, Y et t. La contribution des x-shifts au déterminant est donnée par

$$\det_x = \left(e^{m(m+1)(m+2)/3}\right)\left(X^{m(m+1)(m+2)/3}\right)\left(Y^{m(m+1)(m+2)/6)}\right)$$

alors que la contribution des y-shifts vérifie

$$\det_y = \left(e^{tm(m+1)/2}\right)\left(X^{tm(m+1)/2}\right)\left(Y^{t(m+1)(m+t-1)/2}\right).$$

Cependant, X et Y sont connus et donnés par $X = e^{\delta}$ et $Y = e^{0.5}$. En insérant ces valeurs dans les deux contributions, les déterminants s'écrivent

$$\det_x = \exp\left[m(m+1)(m+2)(5+4\delta)/12\right] = \exp\left[\frac{5+4\delta}{12}tm^2 + \frac{mt^2}{4} + o(tm^2)\right],$$

$$\det_y = \exp\left[\frac{tm(m+1)(1+\delta)}{2} + \frac{t(m+1)(m+t+1)}{2}\right] = \exp\left[\frac{3+2\delta}{4}tm^2 + \frac{mt^2}{4} + o(tm^2)\right].$$

La dimension de la sous-matrice correspondant aux x-shifts est $w_x = (m+1)(m+2)/2$ et celle correspondant aux y-shifts est $w_y = t(m+1)$. La dimension et le déterminant de la matrice entière vérifient alors

$$w = w_x + w_y = \frac{m^2}{2} + tm + o(m^2)$$

et

$$\det(\mathcal{L}_{m,t}) = \det_x.\det_y = \exp\left[\frac{5+4\delta}{12}m^3 + \frac{3+2\delta}{4}tm^2 + \frac{mt^2}{4} + o(m^3)\right].$$

Après application de l'algorithme LLL sur le réseau $\mathcal{L}_{m,t}$, deux vecteurs courts $\mathbf{b_1}$ et $\mathbf{b_2}$ sont obtenus. Pour appliquer lemme 3.5 aux polynômes définis par ces deux vecteurs, il faut montrer les inégalités

$$\|\mathbf{b_1}\| \leq \frac{e^m}{\sqrt{w}}, \qquad \|\mathbf{b_2}\| \leq \frac{e^m}{\sqrt{w}}. \tag{3.11}$$

Or les deux vecteurs plus courts retournés par l'algorithme LLL satisfont

$$\|\mathbf{b_1}\| \leq s^{w/2}\det \mathcal{L}_{m,t}{}^{1/w}, \qquad \|\mathbf{b_2}\| \leq s^{w/2}\det \mathcal{L}_{m,t}{}^{1/w-1} \tag{3.12}$$

L'algorithme LLL (historique) correspond à $s = 2$ et le lemme 3.5 donne les inégalités

$$\det(\mathcal{L}_{m,t}) < \frac{e^{mw}}{\gamma_1}, \qquad \text{où} \qquad \gamma_1 = (w2^w)^{w/2} = o(e^{mw}), \tag{3.13}$$

$$\det(\mathcal{L}_{m,t}) < \frac{e^{mw}}{\gamma_2}, \qquad \text{où} \qquad \gamma_2 = (w2^w)^{(w-1)/2} = o(e^{mw}). \tag{3.14}$$

Les constantes γ_i sont négligeables par rapport au terme e^{mw}. Nous cherchons δ tel que $\det(\mathcal{L}_{m,t}) \leq e^{mw}$ et en insérant la valeur de $\det(\mathcal{L}_{m,t})$, nous obtenons

$$\delta < \frac{7}{6} - \frac{1}{3}\sqrt{7} \approx 0.284.$$

Par conséquence, pour m assez grand, lorsque $d < N^{0.284-\epsilon}$ et $\epsilon > 0$ fixé, nous pouvons trouver deux polynômes bivariés f_1 et $f_2 \in \mathbb{Z}[x,y]$ linéairement indépendants, tels que

$$f_1(x_0, y_0) = 0,$$

$$f_2(x_0, y_0) = 0,$$

pour toute solution (x_0, y_0) de $f(x, y) = 0 \mod e$. Le calcul du résultant $r(y) = \mathrm{Res}_x(f_1, f_2)$ permet de se ramener à l'équation $r(y) = 0$ dans $\mathbb{Z}$ qui peut se résoudre efficacement. On trouve alors $y_0 = (p + q)/2$ ce qui permet de factoriser N. Un raffinement du choix de la famille de polynômes [12] permet de retrouver la clé secrète dès que $d \leq N^{0.292}$.

3.4 Conclusion

Dans ce chapitre, nous avons décrit plusieurs cryptosystèmes qui sont liés aux réseaux soit par construction, soit à travers une cryptanalyse. Par abus de langage, nous appellerons ces réseaux des réseaux "cryptographiques". On verra que les structures des réseaux de type sac-à-dos, des réseaux de type NTRU ou des réseaux de Coppersmith sont très différentes. Une étude spécifique des propriétés de ces réseaux vis-à-vis des algorithmes de réduction est ainsi un problème algorithmique très intéressant.

Une approche naturelle pour cette étude est l'analyse probabiliste d'algorithmes, introduite au chapitre suivant. En dimension 2, le comportement probabiliste de l'algorithme de Gauss a été étudié très précisément [112, 113]. En dimension supérieure, les analyses menées sur des classes de réseaux cryptographiques peuvent expliquer les comportements observés en pratique et conduire à des améliorations algorithmiques qui prennent en compte la spécificité de ces types de réseaux. Cependant, l'algorithme LLL a une dynamique très complexe et il est difficile d'imaginer qu'une analyse précise soit possible. C'est pourquoi nous introduisons au chapitre 4, des modèles simplifiés d'exécution. Le plus simple de ces modèles est étudié au chapitre 5. Nous introduisons aussi au chapitre 6 des modèles probabilistes adaptés à chacun des réseaux cryptographiques.

Chapitre 4

Analyse probabiliste de la réduction. Approches simplifiées.

Sommaire

Ce chapitre vise d'abord à décrire les principes généraux de l'analyse probabiliste d'un algorithme. On revient ensuite au cas d'étude de la thèse, les algorithmes de réduction de réseaux, et on décrit les principaux paramètres de l'algorithme que l'on veut étudier, tant pour l'exécution que pour la configuration de sortie. On décrit ensuite les analyses probabilistes qui ont déjà été effectuées : d'abord, en dimension générale dans des modèles probabilistes particuliers, appelés modèles sphériques, puis, en petite dimension (en dimension 1, pour l'algorithme d'Euclide, et en dimension 2, pour l'algorithme de Gauss), en utilisant les systèmes dynamiques sous-jacents.

On explique alors, informellement, pourquoi cette étude probabiliste de la réduction des réseaux est sans doute très difficile dans le cas général, quand on veut travailler à la fois en dimension quelconque, et dans des modèles probabilistes proches de la réalité cryptographique. On introduit finalement le cadre de la thèse, qui consiste à effectuer une analyse probabiliste "fine", mais dans un modélisation simplifiée – à la fois de l'exécution et des modèles probabilistes d'entrées. Les hypothèses simplificatrices sont présentées et les modèles d'exécutions simplifiés sont ensuite introduits.

4.1 Généralités sur l'analyse probabiliste d'un algorithme.

Analyser un algorithme, c'est étudier son comportement, et de manière plus précise, évaluer ses performances, en rapidité d'exécution, ou en réussite à la sortie. L'analyse d'un algorithme aide d'abord à mieux comprendre les sources éventuelles de son inefficacité, et cette meilleure compréhension permet souvent d'en proposer des versions plus efficaces. En comparant les performances d'un algorithme donné avec d'autres algorithmes de la même classe, on peut aussi opérer une classification entre ces algorithmes, et décider celui qu'il faut employer.

4.1.1 Notions générales.

Nous commençons par reprendre des notions très générales, déjà abordées dans cette thèse, au Chapitre 1.

Un algorithme $\mathcal{A}$ est une application $\mathcal{A} : \Omega \to \mathcal{S}$, où Ω est l'ensemble des entrées de l'algorithme et $\mathcal{S}$ l'ensemble de ses sorties.

Taille. On définit sur l'ensemble Ω une notion de taille $\tau : \Omega \to \mathbb{N}$, qui formalise (en la simplifiant) la place $\tau(\omega)$ qu'occupe en mémoire une entrée ω de l'algorithme. Ainsi, pour un entier $\omega \in \mathbb{N}$, la "vraie" taille est la taille (binaire) de l'entier, c'est-à-dire le nombre de ses chiffres en base 2. On définira la taille d'un entier ω comme son logarithme, en négligeant la base du logarithme, et la taille d'un rationnel p/q, qui est représenté en machine par le couple (p, q) comme la somme des tailles de son numérateur et de son dénominateur.

Paramètres d'étude. Analyser un algorithme, c'est étudier des paramètres de cet algorithme, qui sont donc des fonctions définies sur l'ensemble Ω des entrées, à valeurs réelles. Par exemple, le nombre d'itérations $c : \Omega \to \mathbb{N}$ de l'algorithme $\mathcal{A}$, ou bien une caractéristique géométrique $g : \Omega \to \mathbb{R}$ de la sortie de l'algorithme $\mathcal{A}$, quand il fonctionne sur l'entrée ω. Ainsi $c(\omega)$ est le nombre d'itérations sur l'entrée ω, et $g(\omega)$ est la caractéristique de la sortie quand l'entrée est ω.

Pour les algorithmes de réduction de cette thèse, des caractéristiques géométriques importantes de la sortie sont par exemple le défaut d'orthogonalité, ou les rapports de Siegel.

Analyse. Analyser un algorithme, c'est étudier des paramètres de cet algorithme, en fonction de la taille de l'entrée. Ayant donc défini une notion de taille sur l'ensemble Ω, on classe les entrées par taille, et on considère Ω_N l'ensemble des entrées de taille N. Très souvent, l'ensemble Ω_N est fini (même si l'ensemble Ω de toutes les entrées possibles est, lui, très souvent infini) et on étudie la restriction X_N du paramètre X sur l'ensemble Ω_N, de manière asymptotique, quand la taille N de l'entrée devient grande.

Analyse dans le pire des cas. La complexité dans le pire des cas est définie par la suite

$$M_N = \max_{\omega \in \Omega_N} X_N(\omega).$$

Souvent, lorsqu'on parle de complexité d'algorithme, il s'agit de complexité dans le pire des cas. Les analyses classiques rappelées dans le Chapitre 2 sont toutes des analyses dans le pire des cas.

4.1.2 Analyse probabiliste.

Mais le pire des cas peut être atteint rarement, et ne donne pas lieu à une mesure de complexité "générique". On se tourne donc vers une mesure qui rend mieux compte de la réalité. Pour chaque taille N, on définit une distribution de probabilité $\mathbb{P}_N$ sur l'ensemble Ω_N, qui correspond (idéalement) à une modélisation des entrées rencontrée dans la "réalité". Souvent, on commence par choisir la distribution uniforme sur l'ensemble fini Ω_N. Une fois que l'on a choisi cette distribution $\mathbb{P}_N$ sur l'ensemble Ω_N, les restrictions des paramètres intéressants à Ω_N deviennent des variables aléatoires, et l'analyse probabiliste de l'algorithme $\mathcal{A}$ sur l'ensemble Ω est l'analyse de ces variables aléatoires, effectuée de manière asymptotique, quand la taille N devient grande.

L'analyse en moyenne s'intéresse à la valeur moyenne –ou l'espérance– de ces paramètres, et étudie pour un paramètre X de l'algorithme $\mathcal{A}$ sur les données $(\Omega_N, \mathbb{P}_N)$ la suite

$$\mathbb{E}_N[X] = \sum_{\omega \in \Omega_N} \mathbb{P}_N(\omega) X(\omega).$$

Si la nature du problème le permet, on peut approfondir l'analyse avec l'étude de l'écart type

$$\sigma_N[X] = \sqrt{\mathbb{V}_N[X]} \quad \text{avec} \qquad \mathbb{V}_N[X] = \sum_{\omega \in \Omega_N} \mathbb{P}_N(\omega) \, [X(\omega) - \mathbb{E}_N[X]]^2 .$$

Le résultat le plus précis est fourni par l'analyse en distribution de la variable aléatoire, et l'obtention éventuelle de la loi limite pour la variable

$$\widetilde{X}_N := \frac{X_N - \mathbb{E}_N[X]}{\sigma_N[X]}.$$

Souvent, l'obtention des expressions exactes pour l'espérance $\mathbb{E}_N[X]$ ou la variance $\mathbb{V}_N[X]$ est hors d'atteinte et on recherche plutôt des équivalents asymptotiques de la moyenne et de la variance, et des distributions limites, quand la taille N des entrées devient grande.

4.2 Analyse probabiliste d'un algorithme de réduction

Nous reprenons ces notions et voyons maintenant comment elles s'appliquent à notre cas d'étude.

4.2.1 Entrées et sorties.

Un algorithme de réduction transforme une base en une autre. Les ensembles d'entrée et de sortie coïncident ici, (on a $\Omega = \mathcal{S}$) et une entrée, tout comme une sortie, est représentée, avec nos notations, par d vecteurs (linéairement indépendants) de $\mathbb{R}^n$, c'est-à-dire un matrice de $[\mathbb{R}^n]^d = \mathbb{R}^{n\times d}$. D'un point de vue algorithmique, on travaille avec des vecteurs à composantes entières (ou rationnelles) et les ensembles Ω et $\mathcal{S}$ coïncident avec $\mathbb{Z}^{n\times d}$ ou $\mathbb{Q}^{n\times d}$.

Mais on peut aussi travailler à homothétie près et remplacer un modèle discret par un modèle continu, où tous les vecteurs sont dans la boule unité $\mathcal{B}_n := \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| \leq 1\}$ de $\mathbb{R}^n$. L'ensemble Ω sera alors l'ensemble $\mathcal{B}_n^d$.

Nous travaillerons aussi dans la suite dans des modèles de tas de sable (tas de sable ou un chip firing game) et une entrée correspondante à une base de dimension d sera alors un élément de $\mathbb{R}^d$ ou $\mathbb{R}^{d-1}$ selon qu'on travaille avec un tas de sable ou un chip firing game.

Taille de l'entrée. Dans le cadre algorithmique de la réduction des réseaux, on travaille avec l'ensemble $\mathcal{B}_n^d(M)$, qui est l'ensemble des bases d-dimensionnelles dans l'espace ambiant de dimension n, où M est une borne supérieure de toutes les entrées-composantes des vecteurs de la base. Dans ce cas, la taille d'une entrée est donc égale à $n \cdot d \cdot \log M$. Dans le modèle continu, il n'y plus de notion de taille, mais, on peut "espérer" que lorsque la taille M tend vers l'infini, le modèle discret va "converger" en un sens à déterminer vers le modèle continu.

4.2.2 Modèles continus et modèles discrets.

L'algorithmique travaille avec des données entières. On considère par exemple les vecteurs $\mathbf{v} \in \mathbb{Z}^n$ dont toutes les composantes sont inférieures (en valeur absolue) à un entier M. Puisque les problèmes de réduction sont invariants par homothétie, on peut considérer à la place de $\mathbf{v}$ le vecteur $\mathbf{v}/M \in [\mathbb{Z}/M]^n$ qui a toutes ses composantes (en valeur absolue) inférieures à 1. Lorsque M tend vers l'infini, alors ce modèle discret va "tendre" vers un modèle continu. C'est plus facile d'expliquer la procédure inverse : considérons un domaine $\mathcal{X} \subset \mathbb{R}^n$ avec une frontière assez "lisse". Pour tout entier M, on peut "remplacer" une distribution définie sur $\mathcal{X}$ relative à une densité f de classe $\mathcal{C}^1$ par la distribution dans le domaine

$$\mathcal{X}_M := \mathcal{X} \cap \frac{\mathbb{Z}^n}{M},$$

définie par la restriction f_M de f à $\mathcal{X}_M$. Quand $M \to \infty$, la distribution relative à f_M tend vers la distribution définie par f, en vertu du Principe de Gauss, qui relie le volume d'un domaine $\mathcal{A} \subset \mathcal{X}$ (avec une frontière lisse $\partial\mathcal{A}$) et le nombre de points $\mathcal{A}_M := \mathcal{A} \cap \mathcal{X}_M$,

$$\frac{1}{M^n}\mathrm{card}(\mathcal{A}_M) = \mathrm{Vol}(\mathcal{A}) + O\left(\frac{1}{M}\right) \mathrm{Area}(\partial\mathcal{A}).$$

Nous pourrons appliquer ce principe à tous les modèles probabilistes de la thèse, définies comme des modèles discrets : nous les rendrons continus, ferons l'analyse dans ce modèle continu, et re-traduirons ces résultats dans le modèle discret (pour M tendant vers ∞). Ce retour peut s'avérer plus délicat en dimension 1, mais il est très robuste en toute dimension $d \geq 2$.

4.2.3 Principaux paramètres d'étude.

Paramètres géométriques de l'entrée. Dans un algorithme de réduction, il y a deux types d'entrée : l'entrée donnée (qui est la base de type (n, d)) et l'entrée calculée qui est celle sur laquelle on va commencer le calcul : c'est la matrice $\mathcal{P}$ qui exprime la base B dans la base B^*. C'est donc la distribution de la matrice $\mathcal{P}$, qui est de fait la matrice d'entrée, qui va jouer un rôle essentiel dans l'algorithme, via la distribution des longueurs ℓ_i des orthogonalisés, et la distribution des coefficients $m_{i,j}$. La distribution initiale des coefficients $m_{i,j}$ peut assez naturellement être supposée uniforme dans l'intervalle $[-1/2, +1/2]$. Mais ce sont les longueurs ℓ_i des orthogonalisés, ou mieux encore, les rapports de Siegel r_i au début de l'algorithme, qui sont importants. Si on veut pouvoir analyser (assez) facilement un algorithme de réduction, il faut pouvoir travailler avec un modèle d'entrées pour lequel il est facile de calculer la distribution (initiale) des rapports de Siegel.

Mais bien sûr, on veut aussi pouvoir travailler sur des entrées "naturelles". Elles peuvent être naturelles d'un point de vue "mathématique", mais ici, nous sommes plutôt intéressés par leur "réalisme" dans un cadre essentiel d'application de la réduction, qui est la cryptographie.

Paramètres liés à l'exécution. Pour les algorithmes de réduction considérés dans cette thèse, le paramètre essentiel est le nombre d'itérations, car nous ne traiterons pas ici la complexité binaire, que nous jugeons hors d'atteinte. Même si nous nous limitons au nombre d'itérations, un certain nombre de questions plus précises peuvent être intéressantes : combien y-a-t-il d'étapes qui se passent dans la boîte B_i d'indice i ? Ces étapes de réduction ont-elles plutôt lieu au début de l'exécution ? à la fin ? à quelle fraction de l'exécution ? En fait, c'est la marche aléatoire qui décrit l'exécution qui nous intéresse, et nous voudrions mieux comprendre ses caractéristiques.

Mais évidemment cette marche aléatoire dépend elle-même de la stratégie de l'algorithme : la stratégie de l'algorithme définit l'indice de la prochaine boîte à réduire. Nous avons décrit jusqu'à présent la stratégie naturelle de l'algorithme, qui réduit toujours la boîte non réduite de plus petit indice. Cette stratégie présente beaucoup d'avantages algorithmiques, puisqu'on peut calculer les orthogonalisés au moment où on en a besoin. Cependant, d'autres stratégies sont naturelles, et elles peuvent avoir une influence sur le nombre d'itérations de l'algorithme.

La preuve du nombre d'itérations effectuée au Chapitre 2 est vraiment élémentaire. On considère une grandeur donnée : connaissant sa valeur initiale Δ, sa valeur finale $\widehat{\Delta}$, et le facteur $\rho < 1$ dont elle diminue à chaque étape, supposé un instant constant lors de l'exécution, on obtient une estimation du nombre d'étapes, égal à

$$K = \log_\rho \left(\frac{\widehat{\Delta}}{\Delta} \right).$$

Cette estimation reste aussi exacte, même si le facteur de décroissance ne reste plus constant. Désignant en effet par $\rho^{(k)}$ le facteur de décroissance à l'étape k, et Δ_k la valeur de Δ au début de l'étape k, on

$$\rho^{(k)} := \frac{\Delta_{k+1}}{\Delta_k}, \quad \text{donc} \quad \frac{\widehat{\Delta}}{\Delta} = \prod_{k=1}^{K} \rho^{(k)}.$$

Si nous désignons par $\bar{\rho}$ la moyenne géométrique de ρ lors de l'exécution, on obtient plus généralement

$$K = \log_{\bar{\rho}} \left(\frac{\widehat{\Delta}}{\Delta} \right), \qquad \text{avec} \quad \bar{\rho} := \sqrt[K]{\prod_{k=1}^{K} \rho^{(k)}}. \tag{4.1}$$

Si nous voulons raffiner cette estimation, notamment de manière probabiliste, nous devons avoir des informations sur :

– la distribution d'entrées, déjà évoquée à l'instant ;
– la distribution de la sortie, qui sera évoquée dans le paragraphe suivant ;
– le paramètre ρ, ou sa moyenne géométrique $\bar{\rho}$ pendant l'exécution. C'est un objet d'étude important de notre travail, sur lequel nous nous concentrons maintenant.

Dans la preuve du chapitre 2, on utilise un majorant pour ρ, à savoir $\rho \leq 1/t$. Mais, évidemment, l'exécution dépend fondamentalement de ce paramètre, et il est important d'étudier la manière dont ρ évolue lors de l'exécution de l'algorithme, et lors donc de la marche aléatoire qui décrit l'exécution. A priori, ce facteur $\rho = \rho(i, j)$ dépend de l'étape (ou "instant") $j \in [1\,..\,K]$ de l'exécution et aussi de l'indice $i \in [1\,..\,d]$ de la boîte où la réduction est effectuée.

Paramètres liés à la configuration de sortie. A la sortie de l'algorithme, la base vérifie... les conditions de sortie ! Mais là aussi, des questions sur la distribution des rapports de Siegel sont intéressantes. Chacun des rapports $\widehat{r}_i$ (en sortie) est minoré par $1/s$, et donc les rapports $\widehat{r}_i$ appartiennent à l'intervalle $[1/s, +\infty[$. Comment sont-ils distribués dans cet intervalle $[1/s, \infty[$? Cette distribution est-elle très dépendante de l'indice i de la boîte ? Les longueurs $\widehat{\ell}_i$ des orthogonalisés à la sortie fournissent une estimation du premier minimum du réseau. Est-ce, en général, une estimation de bonne qualité ?

4.2.4 Résultats expérimentaux existants sur le comportement probabiliste de l'algorithme LLL

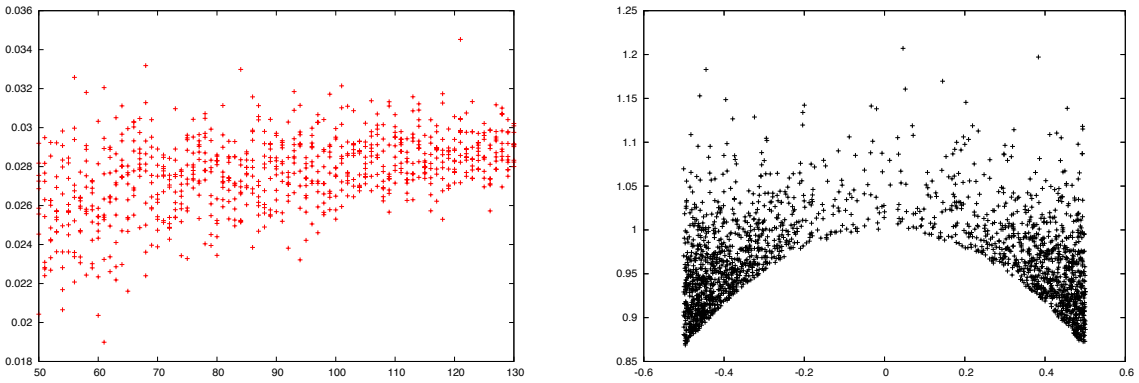


FIGURE 4.1: À gauche, la distribution de paramètre $\frac{1}{d-1}\log_s \gamma(\widehat{B})$ en fonction de la dimension d ; à droite la distribution de sortie des bases locales.

Les travaux de Gama, Nguyen et Stehlé [34, 88] fournissent des expérimentations intéressantes, qui indiquent que la géométrie de la sortie de l'algorithme semble être largement indépendante de la distribution des entrées, tandis que le nombre d'itérations est fortement dépendant de la distribution des entrées. Dans cette section, nous décrivons les résultats de l'article [88].

Les auteurs conduisent des expérimentations pour deux types de bases : les bases d'Ajtai et les bases de types sac-à-dos (ces types d'entrées seront introduites dans le chapitre 6). Ils montrent expérimentalement qu'il existe une valeur moyenne $\beta \approx 1.04$ telle que, pour la plupart des bases de sortie $\widehat{B}$, le défaut d'Hermite $\gamma(\widehat{B})$ est proche de β^{d-1} (voir figure 4.1).

Les auteurs observent aussi que les boîtes finales $\widehat{B}_i = (\mathbf{b}_i^\star, m_{i+1,i}\mathbf{b}_i^\star + \mathbf{b}_{i+1}^\star)$, ont des propriétés géométriques qui apparaissent être largement indépendantes de l'indice i (avec une distribution

Principaux paramètres	$\widehat{r}_i$	$\gamma(\widehat{B})$	K
Pire des cas	$1/s$	s^{d-1}	$\Theta(d^2 \log M)$
Bases d'Ajtai	$1/\beta$	β^{d-1}	$\Theta(d^2 \log M)$
Bases sac-à-dos	$1/\beta$	β^{d-1}	$\Theta(d \log M)$

TABLE 4.1: Comparaison entre les majorants prouvés et les moyennes empiriques des principaux paramètres.

qui donne un poids important aux «coins»). Dans la figure 4.1, chaque point (de coordonnées $m_{i+1,i}$ et $r_i = \ell_{i+1}/\ell_i$) correspond à une base locale. La valeur moyenne de $|m_{i+1,i}|$ est proche de 0.38 et $r_i \approx \frac{1}{\beta}$ avec la même valeur $\beta = 1.04$.

Les moyennes empiriques des rapports de Siegel et le défaut d'Hermite paraissent être de la même forme que les majorations prouvées (voir chapitre 2), avec un facteur β qui remplace le facteur s obtenu dans l'analyse du pire des cas. Dans le tableau 4.1 on cherche à comparer les moyennes empiriques de ces deux variables et les bornes supérieures obtenues dans le chapitre 2.

Concernant le nombre d'itérations, la situations est différente selon le type de la base. Dans le cas des bases d'Ajtai, la moyenne empirique du nombre d'itérations semble être expérimentalement quadratique en la dimension d, et donc du même ordre que la majoration prouvée. Pour les bases de type sac-à-dos, la valeur expérimentale est linéaire en la dimension d, donc d'un ordre de grandeur plus petit que la majoration prouvée. Un des objectifs de cette thèse est d'expliquer et de tenter de prouver ces phénomènes observés dans des modèles "simplifiés".

4.2.5 Des exemples de questions naturelles.

Les questions suivantes sont à la fois centrales et naturelles pour les algorithmes de réduction :

(i) Considérons un réel $s > 1$. Quelle est la probabilité $\pi_{d,n,s}$ qu'une base aléatoire $B_{d,n}$ soit déjà s-réduite au sens de Siegel (i.e., satisfasse les relations (2.7)) ou t-réduite au sens de Lovász (i.e., satisfasse les relations (2.9)) ?

(ii) Quelle est la valeur moyenne du premier minimum d'une base aléatoire $B_{d,(n)}$?

(iii) Considérons un réel $t > 1$. Quel est le nombre moyen d'itérations de l'algorithme LLL(t) sur une base aléatoire $B_{d,n}$?

Finalement, en résumé, il faut non seulement trouver un bon descriptif du modèle probabiliste des entrées de l'algorithme, mais aussi un bon descriptif de l'algorithme lui-même, c'est-à-dire une bonne modélisation de son exécution.

On passe maintenant une revue certains travaux existant à ce sujet,

– ce qu'on sait faire avec des modèles d'entrées (en dimension générale) ;
– ce qu'on sait faire avec des modélisations de l'exécution (en petites dimensions).

4.3 Analyse de la réduction dans les modèles sphériques

Dans cette section nous citons les principaux résultats de l'analyse probabiliste de l'algorithme LLL dans un modèle sphérique. On trouve les preuves de ces résultats dans [28, 6, 8].

Malheureusement, ce modèle n'apparaît que dans la programmation linéaire entière et pas dans les principales applications, notamment les applications cryptographiques.

4.3.1 Modèles sphériques

Une des modèles probabilistes continus les plus naturels pour les entrées d'un algorithme de réduction de réseaux est le modèle sphérique, défini comme suit :

- on considère une distribution v_n sur $\mathbb{R}^n$, invariante par rotation, qui vérifie $v_n(0) = 0$, qui définit ce qu'on appelle une "distribution sphérique simple" ;
- on choisit indépendamment d vecteurs selon cette distribution.

Les distributions sphérique simples les plus naturelles sont par exemple :

- la distribution uniforme dans la boule unité $\mathcal{B}_n = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| \leq 1\}$;
- la distribution uniforme sur la sphère unitaire $\mathcal{S}_n = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| = 1\}$;
- la distribution gaussienne, où les n composantes du vecteur $\mathbf{x}$ sont indépendantes et distribuées selon la loi normale standard.

Dans une distribution sphérique simple, la partie angulaire $\theta = \mathbf{x}/\|\mathbf{x}\|$ est uniformément distribuée sur la sphère unité $\mathcal{S}_n$ et la distribution est complètement déterminée par la distribution ρ_n de sa partie radiale $\rho = \|\mathbf{x}\|$, définie par

$$\rho_n(x) = \mathbb{P}[\rho \leq x].$$

Définition 4.1 (Modèle sphérique)**.** *Un système $B_{d,n}$ de d vecteurs de $\mathbb{R}^n$ est distribué selon une loi sphérique si ces vecteurs sont choisis indépendamment selon la même distribution sphérique simple v_n dans $\mathbb{R}^n$.*

Pour $d \leq n$, un tel système est presque sûrement linéairement indépendant. On s'intéresse à des distributions sphériques particulières qui vérifient la propriété de concentration suivante :

Définition 4.2 (Modèle sphérique concentré)**.** *Une distribution sphérique est concentrée s'il existe une suite $(a_n)_n$ et des constantes $d_1, d_2, \alpha > 0, \tau_0 \in (0,1)$ telles que, pour n et $\tau \in (0, \tau_0)$, la fonction distribution radiale ρ_n satisfait :*

$$\rho_n(a_n(1+\tau)) - \rho_n(a_n(1-\tau)) \geq 1 - d_1 e^{-n d_2 \tau^\alpha}.$$

La propriété de concentration est vérifiée dans le cas des trois exemples précédents.

4.3.2 Distribution des rapports de Siegel en entrée.

Tout d'abord, on montre que beaucoup de variables aléatoires reliées de près ou de loin aux longueurs ℓ_i des orthogonalisés admettent des lois *beta*.

On rappelle d'abord ce que c'est qu'une loi *beta*. Pour deux nombres a et b strictement positifs, la distribution *beta* de paramètres (a, b) est a une densité $\beta_{(a,b)}$ définie sur l'intervalle $[0, 1]$ par la relation

$$\beta_{(a,b)}(x) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1},$$

où la constante $B(a, b)$ s'exprime à l'aide de la fonction Γ d'Euler sous la forme suivante

$$B(a,b) := \int_0^1 x^{a-1}(1-x)^{b-1}dx = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} \qquad \text{avec} \quad \Gamma(s) = \int_0^\infty e^{-t}t^{s-1}dt.$$

Proposition 4.3. *Dans tout modèle sphérique dans $\mathbb{R}^n$:*

(i) les variables ℓ_j sont indépendants

(ii) sous le modèle uniforme de la boule unité, la variable ℓ_j^2 suit la loi beta $\beta_{(n-j+1),(j+1)/2}$

4.3.3 Probabilité qu'une base d'entrée soit déjà réduite au sens de Siegel.

Le résultat suivant est dû à Akhavi, Marchert et Rouault [8].

Théorème 4.4. *Considérons une base aléatoire $B_{d,n}$ d'un modèle sphérique concentré et posons $g = n - d$. Soit $s > 1$ un paramètre réel, et supposons que n tend vers l'infini. Alors,*

(a) Si g tend vers l'infini, alors la probabilité que $B_{d,n}$ soit s-réduite (au sens de Siegel) tend vers 1.

(b) Si g est constant, alors la probabilité que $B_{d,n}$ soit s-réduite (au sens de Siegel) converge vers une constante dans l'intervalle $(0,1)$ (dépendant que de s et g).

Ce résultat montre en particulier que, dans le modèle sphérique $B_{\theta n,n}$, avec $\theta < 1$, toutes les bases sont presque sûrement réduites au sens de s-Siegel lorsque la dimension n tend vers l'infini. Dans ce modèle, les algorithmes de réduction n'ont pas "beaucoup de travail à faire" dès qu'ils travaillent sur des systèmes de vecteurs qui sont loin d'être de dimension pleine.

4.3.4 Une première analyse probabiliste de l'algorithme LLL

Daudé et Vallée [28] travaillent dans le modèle de la boule unité, et, utilisant essentiellement que les longueurs initiales des orthogonalisés suivent des lois β, ils obtiennent

(a) une borne supérieure pour le nombre moyen d'itérations K.

(b) une borne inférieure pour la valeur moyenne du plus petit vecteur $\lambda(\mathcal{L})$ du réseau engendré par la base d'entrée.

L'étude est menée en dimension pleine $d = n$, mais la preuve peut être étendue à une dimension $d \leq n$ quelconque. Partant des relations qui relient les ℓ_i initiaux, le nombre d'itérations K et le premier minimum $\lambda(\mathcal{L})$ du réseau, démontrées au chapitre 2

$$\lambda(\mathcal{L}) \geq \min \ell_i, \qquad K \leq d - 1 + d(d-1) \log_t \frac{1}{\lambda(\mathcal{L})},$$

la loi des ℓ_i permet d'obtenir l'estimation suivante

$$\mathbb{P}[\ell_i \leq x] \leq (x\sqrt{n})^{n-i+1}.$$

On en déduit des informations sur la distribution de la variable aléatoire $a := \min \ell_i$

$$\mathbb{P}[a \leq x] \leq (2\sqrt{n})x^{n-d+1}, \qquad \mathbb{E}\left[\log\left(\frac{1}{a}\right)\right] \leq \frac{1}{n-d+1}\left(\frac{1}{2}\log n + 2\right).$$

Daudé et Vallée [28] obtiennent le résultat suivant :

Théorème 4.5. *[28] Dans le modèle de la boule unité, relatif à un espace ambiant de dimension n et une base de dimension d, le nombre d'itérations $K = K(t)$ de l'algorithme LLL(t) sur une base $B_{d,n}$ et le premier minimum du réseau engendré par la base $B_{d,n}$ vérifient*

$$\mathbb{E}_{d,n}[K] \leq d-1+\frac{d(d-1)}{n-d+1}\left(\frac{1}{\log t}\right)\left[\frac{1}{2}\log n + 2\right], \qquad \mathbb{E}_{d,n}[\lambda(\mathcal{L})] \geq \frac{n-d+1}{n-d+2}\left(\frac{1}{2\sqrt{n}}\right)^{1/(n-d+1)}.$$

En particulier, pour $d = cn$, avec $c < 1$, on a

$$\mathbb{E}_{d,n}[K] \leq \frac{cn}{1-c}\left(\frac{1}{\log t}\right)\left[\frac{1}{2}\log n + 2\right], \qquad \mathbb{E}_{d,n}[\lambda(\mathcal{L})] \geq \exp\left(\frac{-1}{2(1-c)n}\log(4n)\right).$$

Et lorsque $n - d$ est d'ordre $\Omega(n^a)$ pour un $a \in [0,1[$ arbitraire, alors

$$\mathbb{E}_{d,n}[K] = O(n^{2-a}\log n), \qquad \mathbb{E}_{d,n}[\lambda(\mathcal{L})] = \Omega\left(\exp\left(-n^{-a}\log n\right)\right).$$

Les mêmes auteurs donnent aussi une estimation de la distribution de la longueur du premier minimum $\lambda(\mathcal{L})$ et du nombre d'itérations K, que nous décrivons dans le cas de la dimension pleine ($d = n$),

$$\mathbb{P}[\lambda(\mathcal{L}) \leq u] \leq 2u\sqrt{n}, \quad \text{pour } u \in [0,1], \qquad \mathbb{P}[K \geq k] \leq 2\sqrt{n}t^{(n-k)/n^2}.$$

La distribution du nombre d'itérations admet donc une queue exponentielle "ressemble" à une loi géométrique. On verra que des lois géométriques interviennent fréquemment en dimension 2, comme on le montre dans la section suivante, ou en dimension 3, dans le modèle semi-simplifié du chapitre 7.

4.4 Analyse de l'exécution de l'algorithme en petites dimensions.

Nous expliquons les principales idées qui permettent d'obtenir une analyse très précise des algorithmes de réduction en petite dimension (Euclide pour la dimension 1, Gauss pour la dimension 2). Ce sont dans les deux cas des analyses qui utilisent fortement le système dynamique sous-jacent.

4.4.1 L'algorithme d'Euclide.

L'algorithme d'Euclide centré sur une entrée (v_0, v_1) effectue une suite de divisions euclidiennes de la forme (voir section 2.2.2)

$$v_0 = m_1 v_1 + \epsilon_1 v_2, v_1 = m_2 v_2 + \epsilon_2 v_3, v_{i-1} = m_i v_i + \epsilon_i v_{i+1}, \ldots, v_{p-1} = m_p v_p + 0, \tag{4.2}$$

où le reste v_{i+1} est le reste centré de la division de v_{i-1} par v_i. Dans ce cas, le quotient m_i est défini comme étant l'entier le plus proche du quotient v_{i-1}/v_i, et le signe ϵ_i est le signe de la différence entre m_i et v_{i-1}/v_i.

Système dynamique sous-jacent. Au lieu d'observer la suite de couples (v_{i-1}, v_i), on observe maintenant les rationnels qui sont quotients entre deux entiers successifs, de la forme $x_i := v_i/v_{i-1}$ et il y a alors équivalence entre les deux écritures

$$v_{i-1} = m_i v_i + \epsilon_i v_{i+1} \qquad \Longleftrightarrow \qquad \frac{1}{x_i} = \left\lfloor \frac{1}{x_i} \right\rceil + \epsilon_i x_{i+1}.$$

Considérons alors la fonction $U : [0, 1/2] \to [0, 1/2]$ définie par

$$U(x) = \epsilon\left(\frac{1}{x}\right)\left(\frac{1}{x} - \left\lfloor \frac{1}{x} \right\rceil\right), \qquad U(0) = 0, \tag{4.3}$$

où $\lfloor x \rceil$ désigne l'entier le plus proche de x et $\epsilon(x)$ le signe de $x - \lfloor x \rceil$. Alors l'exécution décrite par (4.2) s'écrit aussi $x_2 = U(x_1), x_3 = U(x_2), \ldots, U(x_p) = 0$. C'est donc la trajectoire de l'entrée x_1 sous l'action du "shift" U.

Le couple $([0, 1/2], U)$ définit un système dynamique qui "étend" l'algorithme d'Euclide. C'est un système dit "complet", dont les branches sont surjectives, où les branches inverses locales sont des homographies de la forme $z \mapsto h_{\langle m,\epsilon\rangle}(z)$ avec

$$h_{\langle m,\epsilon\rangle}(z) := \frac{1}{m + \epsilon z}. \tag{4.4}$$

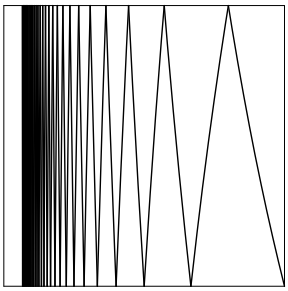


FIGURE 4.2: Système dynamique sous-jacents à l'algorithme d'Euclide centrée.

L'ensemble des branches inverses de cet algorithme est $\mathcal{H} := \{h_{\langle m,\epsilon\rangle};\quad (m,\epsilon) \geq (2,1)\}$. Une exécution de l'algorithme d'Euclide crée donc une fraction continue et le rationnel v_1/v_0 s'écrit

$$\frac{v_1}{v_0} = \cfrac{1}{m_1 + \cfrac{\epsilon_1}{m_2 + \cfrac{\epsilon_2}{\ddots + \cfrac{\epsilon_p}{m_p}}}} = h_{\langle m_1,\epsilon_1\rangle} \circ h_{\langle m_2,\epsilon_2\rangle} \circ \cdots \circ h_{\langle m_p,\epsilon_p\rangle}(0).$$

Ainsi, les couples (u,v) sur lesquels l'algorithme effectue exactement p divisions sont ceux qui s'écrivent $v_1/v_0 = h(0)$, avec une homographie $h \in \mathcal{H}^p$. Les trajectoires rationnelles du système dynamique arrivent ainsi au point 0 en un nombre fini d'itérations, et sont donc très particulières, car toutes les trajectoires commençant sur un nombre irrationnel sont infinies.

Opérateur de transfert. Pour étudier le comportement probabiliste des trajectoires d'un système dynamique, on utilise l'opérateur de transfert. C'est une généralisation de l'opérateur transformateur de densité, que nous décrivons maintenant. On considère une densité $f = f_0$ sur l'intervalle $[0,1/2]$. L'application du "shift" U modifie cette densité et la transforme en une densité f_1. Il est facile de voir que la nouvelle densité f_1 s'exprime en fonction de l'ancienne densité f_0 sous la forme $f_1 := \mathbf{H}[f_0]$ où l'opérateur $\mathbf{H}$, appelé transformateur de densité, s'écrit en fonction de l'ensemble $\mathcal{H}$ des branches inverses de l'application U,

$$\mathbf{H}[f](x) = \sum_{h\in\mathcal{H}} |h'(x)|\, f\circ h(x).$$

Il est souvent utile de "rajouter" en exposant un paramètre s, et l'opérateur de transfert $\mathbf{H}_s$ est défini par la relation

$$\mathbf{H}_s[f](x) = \sum_{h\in\mathcal{H}} |h'(x)|^s\, f\circ h(x). \tag{4.5}$$

Cet opérateur est très utile dans le cas de l'étude probabiliste des trajectoires génériques du système dynamique.

Modèle continu et modèle discret. L'algorithme d'Euclide calcule le pgcd de deux nombres entiers. Son entrée est soit un couple (u,v) d'entiers, soit un nombre rationnel u/v, mais il est par nature discret. Nous l'avons prolongé naturellement en un algorithme (l'algorithme des fractions continues) qui travaille sur tout l'intervalle $[0,1/2]$, et donc dans un modèle continu. Et,

comme souvent le continu est plus facile à appréhender que le discret, on commence par analyser l'algorithme dans ce modèle continu, et on utilise alors fortement l'opérateur de transfert et tous les outils de l'analyse (fonctionnelle, spectrale) associée.

Puis on cherche à faire un retour du continu vers le discret. Ce retour du continu au discret est délicat dans le cas de l'algorithme d'Euclide, car les algorithmes ne se comparent pas si facilement : en effet, l'algorithme des fractions continues ne termine jamais, sauf dans le cas où l'entrée est rationnelle, où il coïncide avec l'algorithme d'Euclide. Le cas des entrées rationnelles est donc extrêmement particulier, et il n'est pas du tout évident que l'algorithme se comporte sur les données rationnelles, comme il se comporte sur les données réelles. C'est grâce aux séries génératrices de Dirichlet que le discret va se relier au continu, c'est là l'idée principale de la méthode d'analyse dynamique des algorithmes initiée par B. Vallée autour des années 1995. Nous donnons ici l'exemple de l'analyse du nombre d'itérations. Mais il y a beaucoup d'autres exemples, qui montrent l'efficacité de ce point de vue.

Cette méthode permet d'analyser en moyenne des paramètres très variés : nombre de quotients prenant une certaine valeur – taille binaire totale occupée par la fraction continue – complexité binaire, égale au nombre total d'opérations sur les bits effectuées par l'algorithme. Cette méthode s'adapte aussi très bien à toute une classe d'algorithmes dits "euclidiens", c'est-à-dire des algorithmes de même type que l'algorithme d'Euclide, comme le célèbre algorithme binaire, celui du "lièvre et de la tortue" ou encore des versions de l'algorithme d'Euclide dites "interrompues". [9, 13, 25, 26, 107, 109].

L'analyse en distribution de l'algorithme d'Euclide est encore plus récente. En 1994, Hensley [47] a démontré que le nombre de divisions de l'algorithme d'Euclide sur les nombres entiers suit une distribution asymptotiquement gaussienne. Ensuite, en 2004, Baladi et Vallée [11] ont simplifié, généralisé et précisé cette analyse. Elles ont démontré la nature asymptotiquement gaussienne de toute une classe d'algorithmes euclidiens et de toute une classe de paramètres de coûts additifs et à croissance modérée. Puis Lhote et Vallée [68] ont montré que la complexité binaire de l'algorithme étendu d'Euclide (qui calcule les coefficients de Bezout) satisfait aussi une loi limite gaussienne, à la fois sur les entiers et les polynômes.

Nombre moyen d'étapes de l'algorithme d'Euclide. On veut étudier le comportement asymptotique de la variable K "nombre d'itérations" sur Ω_N,

$$\Omega_M := \{(u,v) \in \mathbb{N}^2; \quad 0 \leq u < v \leq M\},$$

lorsque $M \to \infty$. L'outil est donc la série de Dirichlet, et on introduit les deux séries

$$T(s) = \sum_{(u,v)\in\Omega} \frac{1}{v^{2s}} = \sum_{n\geq 1} \frac{a_n}{n^{2s}}, \qquad S(s) = \sum_{(u,v)\in\Omega} \frac{K(u,v)}{v^{2s}} = \sum_{n\geq 1} \frac{b_n}{n^{2s}},$$

et l'espérance de K sur Ω_M est égale à

$$\mathbb{E}_M[K] = \frac{\sum_{n\leq M} b_n}{\sum_{n\leq M} a_n}.$$

L'asymptotique des coefficients d'une série de Dirichlet est liée à la position et à la nature de sa singularité dominante. Pour trouver l'asymptotique de $\mathbb{E}_M[K]$ en utilisant la formule précédente, il faut donc connaître la singularité dominante de chacune de ces séries de Dirichlet. Or, chacune de ces séries de Dirichlet s'exprime en fonction de l'opérateur de transfert, et notamment du quasi-inverse $(I - \mathbf{H}_s)^{-1}$ de cet opérateur. Plus précisément, on a

$$T(s) \text{ est relié à } (I - \mathbf{H}_s)^{-1}, \qquad S(s) \text{ est relié à } (I - \mathbf{H}_s)^{-1} \circ \mathbf{H}_s \circ (I - \mathbf{H}_s)^{-1}.$$

Ces relations sont basées sur la remarque simple suivante : pour un couple (u, v) d'entiers premiers entre eux qui vérifie $u/v = h(0)$, avec $h \in \mathcal{H}^\star$, le dénominateur v s'exprime en fonction de $h'(0)$, grâce à l'égalité $|h'(0)| = 1/v^2$, due à une propriété de base des homographies.

Maintenant, comme $\mathbf{H} = \mathbf{H}_1$ est un transformateur de densité, il a, dans un espace fonctionnel adéquat, une valeur propre égale à 1, qui est dominante et séparée du reste du spectre par un saut spectral, ce qui démontre que le quasi-inverse $(I - \mathbf{H}_s)^{-1}$ y admet un pôle simple en $s = 1$. De plus, le résidu en $s = 1$ fait intervenir $1/\lambda'(1)$ où $\lambda(s)$ est la valeur propre dominante de $\mathbf{H}_s$ qui est bien définie et analytique sur un voisinage de $s = 1$. Enfin, la dérivée $|\lambda'(1)|$ coïncide avec l'entropie du système dynamique, égale à $\pi^2/(6 \log 2)$. Donc $T(s)$ a un pôle simple en $s = 1$, tandis que $S(s)$ a un pôle double en $s = 1$. En appliquant alors un théorème Taubérien, on montre que le numérateur et le dénominateur de l'espérance admettent les asymptotiques suivantes, pour un nombre $a > 0$,

$$\sum_{n \leq M} a_n \sim aM^2, \qquad \sum_{n \leq N} b_n \sim \frac{2a}{|\lambda'(1)|} \cdot M^2 \log N.$$

Et on obtient ainsi une preuve "dynamique" du résultat suivant, déjà établi dans les années 1970 par Heilbronn et Dixon, par des méthodes combinatoires (pour Heilbronn) ou probabilistes (pour Dixon) :

Théorème 4.6. [Heilbronn, Dixon, Vallée] *Sur l'ensemble Ω_M formé de deux entiers inférieurs ou égaux à M, l'algorithme d'Euclide effectue un nombre moyen d'itérations qui vérifie*

$$\mathbb{E}_M[K] \sim \frac{12 \log 2}{\pi^2} \log M.$$

4.4.2 L'algorithme de Gauss.

Après des premiers travaux de Laville et Vallée [64] qui ont étudié les caractéristiques probabilistes de la configuration de sortie, notamment le premier minimum et le défaut d'Hermite de la base de sortie, Daudé, Flajolet et Vallée [110, 27] ont effectué une analyse probabiliste très précise de l'exécution de l'algorithme de Gauss. Les auteurs ont travaillé dans un modèle uniforme et ont étudié le nombre d'itérations [en moyenne, et en distribution]. En 2007, Vallée et Vera [112, 113] ont repris les mêmes analyses dans un modèle probabiliste plus général et ont étudié la transition de l'algorithme de Gauss vers l'algorithme d'Euclide.

On rappelle que l'algorithme de Gauss effectue la suite de transformations de la forme (voir la section 2.2.2)

$$\mathbf{v}_0 = m_1 \mathbf{v}_1 + \epsilon_1 \mathbf{v}_2, \ldots, \mathbf{v}_{i-1} = m_i \mathbf{v}_i + \epsilon_i \mathbf{v}_{i+1}, \ldots, \mathbf{v}_{p-1} = m_p \mathbf{v}_p + \epsilon_p \mathbf{v}_{p+1}, \tag{4.6}$$

et se termine quand le couple $(\mathbf{v}_p, \mathbf{v}_{p+1})$ est une base minimale. Une telle base est caractérisée par les deux relations

$$\|\mathbf{v}_{p+1}\| \geq \|\mathbf{v}_p\|, \qquad 0 \leq \frac{\langle \mathbf{v}_{p+1} | \mathbf{v}_p \rangle}{\|\mathbf{v}_p\|^2} \leq \frac{1}{2}.$$

La transformation $\mathbf{v} = m\mathbf{u} + \epsilon\mathbf{r}$ est une généralisation de la division euclidienne et peut être considérée comme une "division vectorielle" ; le quotient m est l'entier le plus proche du rapport $\langle \mathbf{v} | \mathbf{u} \rangle / \|\mathbf{u}\|^2$, et le signe ϵ est le signe de la différence entre m et le rapport $\langle \mathbf{v} | \mathbf{u} \rangle / \|\mathbf{u}\|^2$.

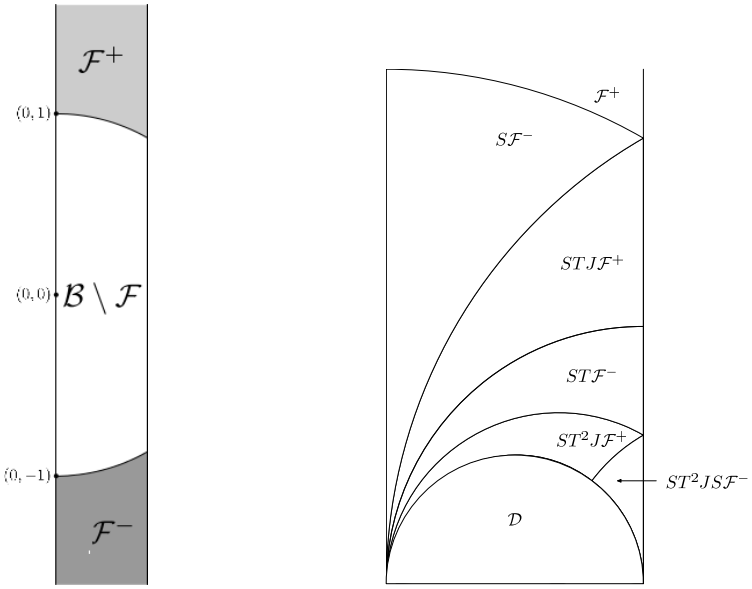


FIGURE 4.3: À gauche les domaines $\mathcal{F} = \mathcal{F}^- \cup \mathcal{F}^+$ et $\mathcal{B}$, à droite la décomposition de $(\mathcal{B} \setminus D)$.

Modélisation complexe. Daudé, Flajolet et Vallée [27] introduisent la modélisation complexe. Il est ici d'abord plus facile de considérer que les vecteurs sont des éléments du plan complexe $\mathbb{C}$, et il est important de remarquer que le quotient de deux nombres complexes $\mathbf{v}/\mathbf{u}$ a ses parties réelles et complexes qui vérifient

$$\Re\left(\frac{\mathbf{v}}{\mathbf{u}}\right) = \frac{\langle \mathbf{u}|\mathbf{v}\rangle}{|\mathbf{u}|^2}, \qquad \Im\left(\frac{\mathbf{v}}{\mathbf{u}}\right) = \frac{\det(\mathbf{u},\mathbf{v})}{|\mathbf{u}|^2}. \tag{4.7}$$

On va, comme on l'a fait pour l'algorithme d'Euclide, remplacer un couple de vecteurs $(\mathbf{u},\mathbf{v})$ par son quotient complexe $z = \mathbf{v}/\mathbf{u}$. Alors, les conditions pour une base minimale s'écrivent facilement sur le nombre z et l'ensemble des bases minimales correspond à l'ensemble

$$\mathcal{F} := \{z \in \mathbb{C}; \quad |z| \geq 1 \quad \text{et} \quad 0 \leq \Re(z) \leq 1/2\}. \tag{4.8}$$

Dans le plan complexe, un échange entre $\mathbf{u}$ et $\mathbf{v}$ correspond à la transformation $S : z \mapsto 1/z$, une translation de la forme $(\mathbf{v},\mathbf{u}) \mapsto (\mathbf{v} + m\mathbf{u},\mathbf{u})$ correspond à la transformation T^m, avec $T : z \mapsto z + 1$ et un changement possible de signe correspond à la transformation $J : z \mapsto -z$. L'algorithme termine quand z est dans l'ensemble $\mathcal{F}$ défini dans (4.8) (voir figure 4.3).

Système dynamique sous-jacent. Puis les mêmes auteurs remarquent que la transformation $(\mathbf{u},\mathbf{v}) \mapsto (\mathbf{u},\mathbf{r})$ associée à la division vectorielle $\mathbf{v} = m\mathbf{u} + \epsilon\mathbf{r}$, se traduit dans le plan complexe par la transformation $\underline{U}$ définie par

$$\underline{U}(z) = \epsilon\left(\frac{1}{z}\right)\left(\frac{1}{z} - \left\lfloor \Re\left(\frac{1}{z}\right)\right\rceil\right), \quad \text{avec} \quad \epsilon(z) = \text{signe}\left(\Re(z) - \lfloor \Re(z)\rceil\right). \tag{4.9}$$

Remarquons que, lorsque z est réel, alors on retrouve exactement l'expression donnée du shift U de l'algorithme d'Euclide donné en (4.3).

On considère la bande verticale $\mathcal{B} = \{z : 0 \leq \Re(z) \leq 1/2\}$, et, au début d'algorithme, le nombre complexe z est choisi dans $\mathcal{B} \setminus \mathcal{F}$. On peut montrer que le cœur de l'algorithme se passe lorsque z appartient au disque $\mathcal{D}$ de diamètre $[0, 1/2]$. En effet, lorsque le nombre z appartient à la bande $\mathcal{B}$ et n'appartient plus à $\mathcal{D}$, alors l'algorithme ne peut effectuer au plus que deux étapes avant de terminer, comme le montre la figure 4.3, à droite.

On considère donc l'algorithme *Cœur-Gauss* et le système dynamique $(\underline{U}, \mathcal{D})$. Contrairement à l'algorithme d'Euclide, l'algorithme de Gauss termine au bout d'un nombre fini d'itérations, sauf sur l'ensemble de mesure nulle constitué par les réels irrationnels, car les complexes z non réels finissent toujours par "sortir" du disque $\mathcal{D}$, et les réels rationnels arrivent en 0, sur le bord du disque $\mathcal{D}$. Pendant son exécution, l'algorithme débute sur une entrée $z = z_1 \in \mathcal{D}$ et produit la suite $z_1, z_2 = \underline{U}(z_1), \ldots, z_p$, obtenue en itérant la transformation $\underline{U}$ jusqu'au moment où le nombre z_p cesse d'appartenir au disque $\mathcal{D}$. Et les branches inverses locales utilisées sont alors de la forme $z \mapsto h_{\langle m,\epsilon\rangle}(z)$ avec

$$h_{\langle m,\epsilon\rangle}(z) := \frac{1}{m + \epsilon z}, \qquad \text{avec } (m, \epsilon) \geq (2, 1). \tag{4.10}$$

Considérant toujours l'ensemble $\mathcal{H} := \{h_{\langle m,\epsilon\rangle}; \quad (m, \epsilon) \geq (2, 1)\}$, une exécution de l'algorithme de *Cœur-Gauss* sur une entrée $z_1 = \mathbf{v}_1/\mathbf{v}_0 \in \mathcal{D}$ crée donc une fraction continue et le complexe $\mathbf{v}_1/\mathbf{v}_0$ s'écrit

$$z_1 = \frac{\mathbf{v}_1}{\mathbf{v}_0} = \cfrac{1}{m_1 + \cfrac{\epsilon_1}{m_2 + \cfrac{\epsilon_2}{\ddots \cfrac{}{m_p + \epsilon_p z_{p+1}}}}} = h_{\langle m_1,\epsilon_1\rangle} \circ h_{\langle m_2,\epsilon_2\rangle} \circ \ldots h_{\langle m_p,\epsilon_p\rangle}(z_{p+1}),$$

et l'algorithme s'arrête quand z_{p+1} est dans $\mathcal{B}\setminus\mathcal{D}$. Donc l'algorithme effectue au moins $p + 1$ itérations à l'intérieur de $\mathcal{D}$ si et seulement si le complexe z_p est encore dans le disque $\mathcal{D}$: cela signifie que l'entrée z_1 appartient à l'image inverse

$$\underline{U}^{-p}(\mathcal{D}) = \bigcup_{h \in \mathcal{H}^p} h(\mathcal{D}).$$

Nombre d'itérations de l'algorithme de Gauss. On a alors accès au nombre d'itérations K de l'algorithme de Gauss à l'intérieur du disque $\mathcal{D}$, car l'événement $[K \geq p+1]$ coïncide avec le domaine $\underline{U}^{-p}(\mathcal{D})$. Par conséquent, si on considère une distribution initiale uniforme sur le disque $\mathcal{D}$, et si on désigne par $\|\mathcal{X}\|$ l'aire du domaine $\mathcal{X}$, alors la probabilité que l'algorithme fasse au moins $p + 1$ itérations est :

$$\mathbb{P}[K \geq p + 1] = \frac{\|\underline{U}^{-p}(\mathcal{D})\|}{\|\mathcal{D}\|} = \frac{16}{\pi} \sum_{h \in \mathcal{H}^p} \|h(\mathcal{D})\|.$$

La figure 4.4 montre l'efficacité de l'algorithme et les domaines $[K = p]$, pour $p \geq 1$. Pour calculer cette probabilité, on introduit l'opérateur de transfert qui agit sur des fonctions à deux variables,

$$\underline{\mathbf{H}}_{2s}[F](z, u) = \sum_{h \in \mathcal{H}} h'(z)^s \, h'(u)^s \, F(h(z), h(u)), \tag{4.11}$$

qui est une extension de l'opérateur de transfert de l'algorithme d'Euclide, car lorsque z et u sont des nombres réels égaux, on retrouve l'opérateur défini en (4.5). Comme le Jacobien de la

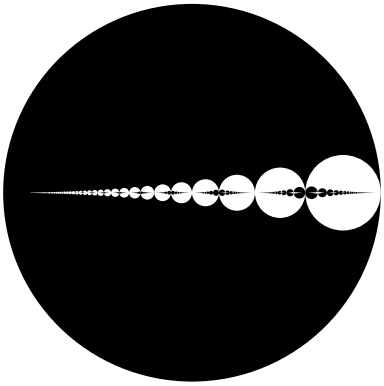


FIGURE 4.4: Les domaines $[K = p]$ pour $p \geq 1$, alternativement en noir et blanc.

transformation $(x, y) \mapsto (\Re h(x + iy), \Im h(x + iy))$ est égal à $h'(z) \cdot h'(\bar{z}) = |h'(z)|^2$, on obtient l'égalité :

$$\sum_{h \in \mathcal{H}^p} \|h(\mathcal{D})\| = \iint_{\mathcal{D}} \underline{\mathbf{H}}_2^p[1](z, \bar{z}) dx dy \qquad \text{avec} \quad z = x + iy.$$

On effectue alors de nouveau une analyse spectrale. L'opérateur $\underline{\mathbf{H}}_s$ a, sur un espace fonctionnel adéquat, et lorsque s est réel, une valeur propre dominante qui est la même que la valeur propre dominante de l'opérateur $\mathbf{H}_s$ du système dynamique d'Euclide, et que nous avions désignée par $\lambda(s)$. C'est ainsi que l'on montre le résultat suivant :

Théorème 4.7. [Daudé, Flajolet, Vallée] *Sur le disque $\mathcal{D}$ de diamètre $[0, 1/2]$ muni de la distribution uniforme, le nombre d'itérations K de l'algorithme* Cœur-Gauss *suit une loi asymptotiquement géométrique de raison $\lambda(2)$, où $\lambda(s)$ est la valeur propre dominante de l'opérateur de transfert $\mathbf{H}_s$ lié au système dynamique d'Euclide.*

Modèle probabiliste avec valuation. Quand on choisit le complexe $z := \mathbf{u}/\mathbf{v}$ uniformément dans le disque $\mathcal{D}$, on travaille avec des nombres complexes dont la partie imaginaire n'est pas très petite en général. Or la partie imaginaire de z, compte-tenu de l'expression (4.7), mesure le "degré d'aplatissement" de la base $(\mathbf{u}, \mathbf{v})$ et donc la difficulté à la réduire. Une manière de privilégier les bases "difficiles" est de modifier la distribution dans le disque $\mathcal{D}$ et de choisir une densité avec valuation.

Pour un réel $r > -1$, appelé par la suite valuation, nous considérons la distribution pour laquelle les bases $(\mathbf{u}, \mathbf{v})$ satisfont

$$\mathbb{P}\left[(\mathbf{u}, \mathbf{v}); \quad \frac{|\det(\mathbf{u}, \mathbf{v})|}{\max(\|\mathbf{u}\|, \|\mathbf{v}\|)^2} \leq y\right] = \Theta(y^{r+1}), \qquad \text{quand} \quad y \to 0.$$

Le modèle avec valuation définit ainsi une échelle de densités, pour lequel le poids accordé aux bases aplaties varie.

Dans le cadre complexe, une densité $f(x, y)$ sur $\mathcal{D}$ est de valuation r (avec $r > -1$) si elle est proportionnelle à $|y|^r$, avec un facteur de normalisation $A(r)$. Dans le cas $r = 0$, on retrouve la densité uniforme. On désigne par $\mu_r(\mathcal{X})$ la mesure d'un domaine $\mathcal{X} \subset \mathcal{D}$ pour la densité f de valuation r. On peut alors reprendre l'analyse précédente, et dans ce modèle, la probabilité que l'algorithme fasse au moins $p + 1$ itérations est :

$$\mathbb{P}_r[K \geq p + 1] = \mu_r(\underline{U}^{-p}(\mathcal{D})) = \sum_{h \in \mathcal{H}^p} \mu_r(h(\mathcal{D})).$$

Pour calculer cette probabilité, on peut encore utiliser l'opérateur de transfert défini en (4.11) car on a l'égalité

$$\sum_{h\in\mathcal{H}^p} \mu_r(h(\mathcal{D})) = \frac{1}{A(r)} \iint_{\mathcal{D}} \underline{\mathbf{H}}^p_{2+r}[1](z,\bar{z})dxdy \qquad \text{avec} \quad z = x+iy.$$

qui fait maintenant intervenir, en plus de la constante de normalisation $A(r)$, l'opérateur $\underline{\mathbf{H}}_s$ mais pour la valeur du paramètre $s = 2+r$. La même analyse spectrale s'effectue, et elle fait maintenant intervenir la valeur propre dominante $\lambda(2+r)$. C'est ainsi que l'on montre le résultat suivant :

Théorème 4.8. *[112] Sur le disque $\mathcal{D}$ de diamètre $[0,1/2]$ muni de la distribution de valuation $r > -1$, le nombre d'itérations K de l'algorithme* Cœur-Gauss *suit une loi asymptotiquement géométrique de raison $\lambda(2+r)$, où $\lambda(s)$ est la valeur propre dominante de l'opérateur de transfert $\mathbf{H}_s$ lié au système dynamique d'Euclide.*

Transition de l'algorithme de Gauss vers l'algorithme d'Euclide. Remarquons que, lorsque la valuation r tend vers -1, les bases s'aplatissent complètement, et ce modèle tend vers le modèle à une dimension où les vecteurs $\mathbf{u}$ et $\mathbf{v}$ sont colinéaires. Dans ce cas, on peut s'attendre à ce que l'algorithme de Gauss "tende" vers l'algorithme d'Euclide. C'est ce que le résultat montre car lorsque r tend vers -1, la raison de la loi géométrique tend vers $\lambda(1) = 1$ et la loi devient "de moins en moins" géométrique. On peut montrer que l'espérance dans le modèle de valuation r fait intervenir le quasi-inverse $(I - \underline{\mathbf{H}}_{2+r})^{-1}$, qui devient singulier quand r tend vers -1. Vallée et Vera [112, 113] déduisent de ces remarques une description précise de la transition de l'algorithme de Gauss vers l'algorithme d'Euclide.

4.5 Approches de la thèse

4.5.1 Difficultés de l'analyse probabiliste de la réduction.

Une analyse probabiliste d'algorithmes est réussie quand elle obtient des résultats *précis* dans des modèles *réalistes.*

Modèles probabilistes réalistes. La réduction des réseaux intervient de manière centrale en cryptographie, comme on l'a décrit dans le chapitre 3. Dans ce cas, le problème cryptographique se modélise par une base de réseau, souvent de forme très particulière, qui donne lieu à ce que nous appelons un réseau "cryptographique". Ces réseaux cryptographiques peuvent être de natures très diverses, selon leur provenance : réseaux d'Ajtai, réseaux sac-à-dos ou réseaux NTRU, et enfin réseaux de Coppersmith. Certaines de ces bases apparaissent "faciles" à réduire (Sac-à-dos, Coppersmith), d'autres apparaissent difficiles et sont d'ailleurs construites pour cela (bases d'Ajtai).

Il faut donc modéliser (de manière probabiliste) les distributions d'entrée auxquelles ces bases donnent lieu, afin de pouvoir ensuite analyser le processus de réduction dans chacun de ces modèles. Il s'agit donc de décrire un modèle probabiliste (le plus souvent de nature continue) où l'on puisse décrire la distribution des longueurs ℓ_i ou des rapports de Siegel r_i.

Modélisation de l'exécution. Nous avons donné un exemple de modélisation très fine de l'exécution de l'algorithme en petite dimension, qui se fonde sur les propriétés du système dynamique sous-jacent. Mais, malheureusement, en plus grande dimension, même si on peut modéliser l'exécution de l'algorithme par un système dynamique, celui-ci s'avère très complexe à étudier.

4.5.2 Approches simplifiées.

Nous nous attachons ici donc à une modélisation conjointe de l'exécution de l'algorithme et de ses entrées.

Modélisation simplifiée de l'exécution. Devant la difficulté d'une modélisation exacte de l'exécution, nous avons choisi d'adopter une modélisation "simplifiée", dont nous espérons cependant dégager des phénomènes probabilistes qui soient de même nature qualitative que ceux qui seraient obtenus par une modélisation exacte. Madritsch et Vallée ont proposé une modélisation simplifiée de l'algorithme LLL, fondée sur les "tas de sable", qui est un système dynamique discret très classique, et facile à analyser (nombre d'itérations, configuration de sortie, influence de la stratégie). L'hypothèse principale suppose que le facteur de décroissance $\rho = \rho(i, j)$ (défini précédemment page 71) est constant lors de l'exécution, mais peut dépendre bien sûr de la base d'entrée du réseau. Cette modélisation très (trop ?) simplifiée donne une bonne idée qualitative du déroulement de l'algorithme, et permet aussi de développer des heuristiques pour établir une algorithmique plus performante.

Mais évidemment, cette hypothèse n'est pas vraiment ... juste. C'est pourquoi nous avons effectué une campagne d'expérimentations (dont certaines ont été menées en commun avec Madritsch et Schneider), pour expliciter la manière dont le facteur de décroissance évolue au cours de l'algorithme et aussi la manière dont il dépend du type d'entrées et de la stratégie utilisée. Ces expérimentations sont présentées dans le chapitre 8.

Modélisation des entrées. Puisque la modélisation de l'exécution de l'algorithme se fait via les tas de sable, nous étudions et modélisons les entrées particulières qui nous intéressent, liées aux réseaux cryptographiques, du point de vue des tas de sable. Nous étudions ainsi trois familles de réseaux cryptographiques : réseaux d'Ajtai (qui donnent lieu à des tas très pleins), réseaux sac-à-dos ou réseaux NTRU (qui donnent lieu à des tas de sable uni-tas) et enfin réseaux de Coppersmith qui donnent lieu à des tas de sable "avec des trous". Ces modélisations permettent de faire des expérimentations adaptées et de mieux comprendre le fonctionnement de l'algorithme LLL sur ces différentes familles. Ces modèles seront présentés dans le chapitre spécifique 6.

Modélisation semi-simplifiée de l'exécution. Nous avons aussi proposé une modélisation de l'exécution, de nature intermédiaire, entre le tas de sable et le véritable algorithme de réduction, que nous avons étudiée en dimension 3, première dimension où une modélisation exacte apparaît hors d'atteinte.

Dans la suite de ce chapitre, nous nous concentrons sur la modélisation de l'exécution.

4.6 Vers une modélisation probabiliste d'un algorithme de réduction

Dans le chapitre 2, on a vu que l'algorithme LLL est une généralisation de l'algorithme de Gauss. Bien que le comportement probabiliste de l'algorithme de Gauss soit bien compris, les méthodes utilisées semblent difficilement généralisables en dimension supérieure. C'est pour cette raison que nous décrivons dans la section suivante des modélisations simplifiées de l'algorithme LLL, partant de modélisations très simplifiées, comme celle que Madritsch et Vallée [71] ont introduit, pour parvenir à des modélisations moins simplifiées, mais aussi plus complexes à analyser...

4.6.1 Une vue synthétique de l'algorithme LLL

Le schéma général de l'algorithme LLL est présenté dans le chapitre 2. Cet algorithme prend en entrée une matrice B, où les vecteurs de la base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ sont écrits en ligne et travaille avec la matrice de passage $\mathcal{P} = (m_{i,j})$ qui relie la matrice B à la matrice orthogonale $B^\star$ des orthogonalisés $(\mathbf{b}_1^\star, \ldots, \mathbf{b}_d^\star)$ (toujours écrite en ligne). On a ainsi $B = \mathcal{P}B^\star$.

Dans le cadre de cette thèse, nous nous concentrons sur des paramètres qui sont centraux pour décrire l'évolution de l'algorithme : ce sont *les normes des orthogonalisés* ou, mieux encore, *les rapports de Siegel*. Ce sont vraiment ces rapports de Siegel qui dirigent l'algorithme, puisque l'algorithme cherche à les faire "grandir". Ce sont eux qui mesurent la qualité de la base de sortie, tout comme la "difficulté" que la base d'entrée va avoir à se faire réduire.

Une vue « multiplicative ». On rappelle qu'on désigne par ℓ_i la longueur du i^{e} orthogonalisé $\mathbf{b}_i^\star$ pour $i \in [1\,..\,d]$, et que le i^{e} rapport de Siegel r_i est défini pour $i \in [1\,..\,d-1]$ par

$$r_i = \frac{\ell_{i+1}}{\ell_i}.$$

L'algorithme LLL vise à modifier ces rapports, de sorte qu'ils vérifient à la fin certaines conditions de minoration, comme celles de Siegel $\mathcal{S}_s$ ou de Lovász $\mathcal{L}_t$ (vues au chapitre 2 avec les définitions 2.7 et 2.9). A cette fin, comme on l'a vu au chapitre 2, l'algorithme LLL effectue deux types d'opérations : des translations et des échanges. Les translations ne modifient pas les longueurs ℓ_i, et ne nous intéressent pas fondamentalement ici. Ce sont les échanges qui vont jouer le rôle essentiel. En effet, comme nous l'avons décrit en Section 2.3.2, l'échange $\mathbf{b}_{i+1} \leftrightarrow \mathbf{b}_i$ modifie le rapport $r_i = \ell_{i+1}/\ell_i$, et se traduit par de nouvelles valeurs des rapports de Siegel $\check{r}_i$ et $r_{i+1}^{\check{}}$,

$$\check{r}_{i-1} = \rho r_{i-1}, \qquad \check{r}_i = \frac{1}{\rho^2} r_i, \qquad \check{r}_{i+1} = \rho r_{i+1}.$$

Cet échange n'est effectué que si le rapport r_i n'est pas réduit au sens de s-Siegel, et vérifie donc $r_i \leq (1/s)$. Dans ce cas, et si le paramètre de Siegel s vérifie $s > s_0$ avec $s_0 = 2\sqrt{3}$, le facteur

$$\rho = \left(r_i^2 + m_{i+1,i}^2\right)^{1/2}$$

est strictement inférieur à 1 : c'est le *facteur de décroissance* de l'algorithme LLL.

Une vue « additive ». Nous adoptons un point de vue additif en considérant les logarithmes des rapports de Siegel. On considère le paramètre de Siegel s, qui peut être choisi dans l'intervalle $]s_0, +\infty[$ avec $s_0 = 2/\sqrt{3}$, et on le choisit comme base des logarithmes considérés. La base courante de l'algorithme LLL est alors représentée par la configuration $\mathbf{c} := (c_1, \ldots, c_{d-1})$ avec pour $1 \leq i < d$

$$c_i = -\log_s r_i = \log_s \frac{\ell_i}{\ell_{i+1}}.$$

La condition de Siegel $\mathcal{S}_s$ se traduit dans ce cadre par l'inégalité $c_i \leq 1$. Une étape d'échange $\mathbf{b}_i \leftrightarrow \mathbf{b}_{i+1}$ opéré par l'algorithme LLL se traduit alors

$$\check{c}_i := c_i - 2\alpha, \quad \check{c}_{i-1} := c_{i-1} + \alpha, \qquad \check{c}_{i+1} := c_{i+1} + \alpha$$

avec

$$\alpha = -\log_s \rho = \frac{-1}{2\log s} \log\left(r_i^2 + m_{i+1,i}^2\right). \tag{4.12}$$

Cette quantité est *centrale* dans toute cette thèse. La borne sur le facteur de décroissance ρ, de la forme $\rho \leq \rho_0[s]$ dans le chapitre 2 se traduit en une borne sur le décrément α

$$\alpha \geq \alpha_0[s], \quad \text{avec} \quad \alpha_0[s] = \frac{-1}{2\log s} \log\left(\frac{1}{s^2} + \frac{1}{4}\right).$$

Remarquons que $\alpha_0[s]$ est positif ou nul, est nul pour la valeur limite $s_0 := 2/\sqrt{3}$, et est sinon strictement positif.

Nombre d'itérations. Dans le chapitre 2, nous avons déjà de fait utilisé ce cadre additif, dans l'étude du nombre K d'itérations. Le potentiel [1] $\Delta(B)$ d'une base B est la quantité :

$$\Delta(B) = \prod_{i=1}^{d-1} \ell_i^{d-i}. \tag{4.13}$$

On a obtenu au cours de la preuve du théorème 2.9 une majoration du nombre d'étapes où le test de Lovász (Siegel) est négatif, noté [2] K en fonction du potentiel $\Delta(B)$ de la base initiale et du potentiel $\Delta(\widehat{B})$ de la base finale $\widehat{B}$,

$$K \leq \log_t \left(\frac{\Delta(B)}{\Delta(\widehat{B})}\right) = \frac{1}{\alpha_0(s)} \log_s \left(\frac{\Delta(B)}{\Delta(\widehat{B})}\right).$$

Mais, si on désigne par $\alpha(j)$ la valeur du décrément α à l'itération j, et si $\bar{\alpha}$ est la valeur moyenne du décrément α sur une exécution,

$$\bar{\alpha} = \frac{1}{K} \sum_{j=1}^{K} \alpha(j), \qquad \bar{\alpha} \leq \alpha_0(s),$$

on remarque que K s'exprime exactement en fonction du potentiel $\Delta(B)$ de la base initiale, du potentiel $\Delta(\widehat{B})$ de la base finale et de la valeur $\bar{\alpha}$, (en correspondance avec (4.1) dans le cadre multiplicatif)

$$K(B) = \frac{1}{\bar{\alpha}(B)} \log_s \left(\frac{\Delta(B)}{\Delta(\widehat{B})}\right). \tag{4.14}$$

4.6.2 Évolution expérimentale du décrément pendant une exécution.

Les graphes de l'évolution du décrément α au cours d'une exécution de LLL (test de Siegel) pour des types de bases d'entrée très différents sont représentées sur la figure 4.5, et ces graphes semblent indiquer que le décrément varie de manière assez "chaotique", tout en restant la plupart du temps dans un intervalle borné, qui apparaît "relativement" indépendant de la base d'entrée.

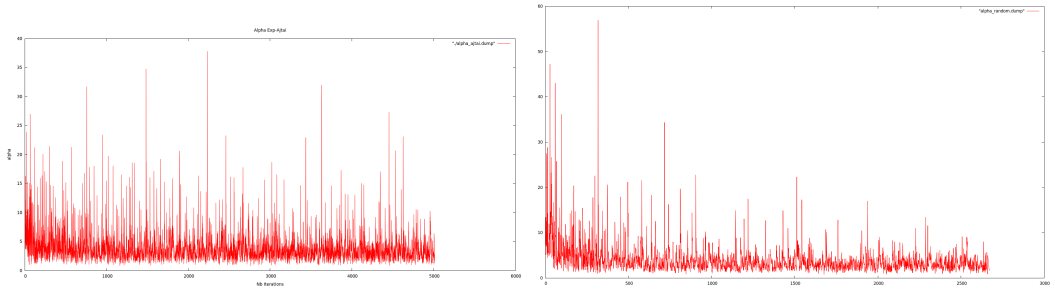
4.6.3 La variable aléatoire α_c.

La principale difficulté d'analyse de l'algorithme LLL est due au fait que, dans le cadre additif par exemple, le décrément α dépend à la fois de c_i et $m_{i+1,i}^2$, sous la forme

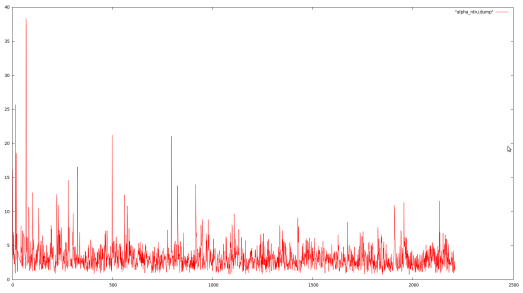
$$\alpha = -\frac{1}{2} \log(s^{-2c_i} + m_{i+1,i}^2).$$

1. Nous utilisons selon les cas les deux notions de potentiel $D(B)$ comme dans le chapitre 2, mais aussi sa racine carrée désignée par $\Delta(B)$.

2. Ici K correspond à K_- dans le théorème 2.9, mais du fait que l'on s'intéresse surtout au nombre d'étapes où le test est négatif, on utilisera par la suite K pour simplifier les notations .



(a) $\mathcal{A}$ [Ajtai] (b) $\mathcal{K}$ [Sac-à-dos]



(c) $\mathcal{N}$ [NTRU]

FIGURE 4.5: L'évolution du décrément α lors d'une exécution de l'algorithme de Siegel (modèle `M4`) dans trois modèles d'entrée très différents.

Coefficients sous-diagonaux $m_{i+1,i}$. Une première simplification consiste à considérer que les coefficients sous-diagonaux $m_{i+1,i}$ sont *indépendants* de c_i et qu'ils sont (à tout instant où ils sont considérés) *uniformément distribués* dans l'intervalle $[-1/2, +1/2]$. La valeur moyenne de $m_{i+1,i}^2$ est alors égale à $1/12$. Si l'on souhaite simplifier plus encore le modèle en considérant que les considérant comme constants, un choix cohérent avec la distribution uniforme est donc de les considérer toujours égaux à $1/12$.

C'est un parti pris de cette thèse de n'accorder qu'un rôle secondaire aux coefficients sous-diagonaux, et ce pour deux raisons. Tout d'abord, l'algorithme de réduction est dirigé par les rapports de Siegel r_i, et cherche prioritairement à les minorer. Les logarithmes c_i peuvent beaucoup varier au cours de l'algorithme, alors que les coefficients $m_{i+1,i}$, lorsqu'ils sont considérés dans le calcul de α, le sont via leur partie fractionnaire centrée, et "vivent" donc dans un intervalle fixe et borné. Ensuite, la modélisation de ces coefficients sous-diagonaux est certainement effroyablement difficile.

Étude du décrément. Dans ce contexte de modélisation probabiliste, le décrément α s'exprime en fonction de la "hauteur" $c > 1$ comme

$$\alpha = -\frac{1}{2}\log_s\left(s^{-2c} + \nu^2\right),$$

où ν est une variable aléatoire à valeur dans $[-1/2, 1/2]$ modélisant le coefficient sous-diagonal pendant l'exécution. Quand la hauteur c est fixée, et la variable ν est uniformément distribuée dans l'intervalle $[-1/2, +1/2]$, le décrément devient une variable aléatoire α_c étudiée en particulier dans la Figure 4.6.

On a facilement l'encadrement

$$-\frac{1}{2}\log_s\left(s^{-2c} + \frac{1}{4}\right) \leq \alpha_c \leq c.$$

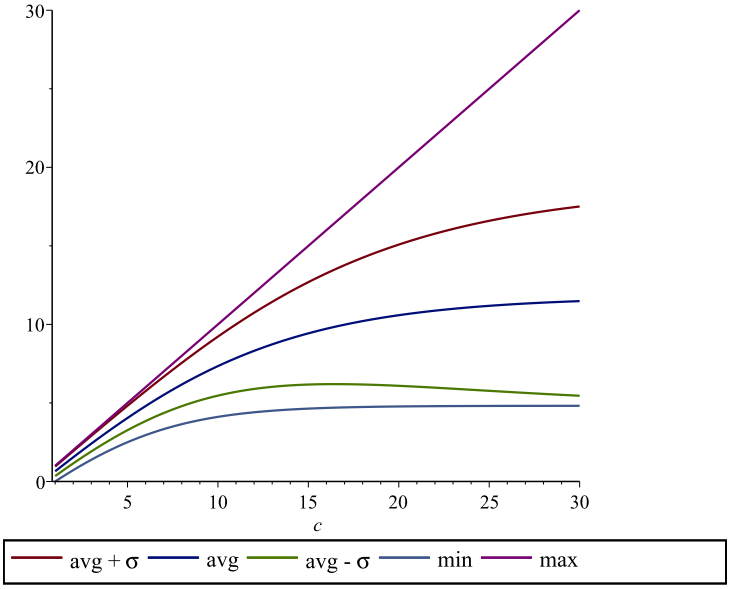


FIGURE 4.6: On trace sur une même figure les graphes de cinq fonctions relatives à la variable aléatoire α_c : la fonction de la valeur moyenne $c \mapsto \mathbb{E}[\alpha_c]$, les fonctions $c \mapsto \min \alpha_c$, $c \mapsto \max \alpha_c$, $\mathbb{E}_{[}\alpha_c] \pm \sigma_c$.

La valeur moyenne vérifie

$$\mathbb{E}[\alpha_c] = -\int_{-1/2}^{1/2} \frac{1}{2} \log_s \left(s^{-2c} + x^2 \right) dx$$

$$\mathbb{E}[\alpha_c] = -\frac{1}{2 \log s} \left[4 s^{-c} \arctan\left(\frac{1}{2} s^c \right) - 2 \log 2 + \log\left(1 + 4 s^{-2c} \right) - 2 \right].$$

La fonction $c \mapsto \mathbb{E}[\alpha_c]$ est une fonction croissante du paramètre c. Dans l'intervalle $[1, \infty[$, et pour la valeur limite du paramètre $s_0 = 2/\sqrt{3}$, elle satisfait

$$\text{pour } c \to \infty, \quad \lim_{c \to \infty} \mathbb{E}[\alpha_c] = \frac{1}{\log s_0} (\log 2 + 1) \approx 11.77,$$

$$\text{pour } c = 1, \quad \mathbb{E}[\alpha_c] = \frac{1}{\log s_0} \left[1 - \frac{\pi}{2\sqrt{3}} \right] \approx 0.65.$$

On peut aussi calculer l'écart type σ_c de cette variable aléatoire, et la figure 4.6 donne une idée du comportement du décrément α_c en fonction de la hauteur c. On observe notamment que le décrément est borné avec une grande probabilité et que le comportement est de « nature » presque linéaire pour des petites valeurs de c. En particulier, même quand $c \to 1$, le décrément reste strictement positif avec une grande probabilité.

4.6.4 Un cadre commun pour la modélisation de l'algorithme LLL

Pour modéliser un algorithme de réduction, on adopte le point de vue général des systèmes dynamiques, qui se décrit selon le schéma suivant : On considère

(*a*) l'ensemble $\mathcal{C}$ des configurations, et deux sous-ensembles de $\mathcal{C}$, l'ensemble $\mathcal{J}$ des configurations initiales et l'ensemble $\mathcal{O}$ des configurations finales ;

(*b*) Un sous-ensemble $\mathcal{T}$ de l'ensemble des applications $T : \mathcal{C} \to \mathcal{C}$ qui va former l'ensemble des transformations possibles.

Le processus est itératif : on part d'une configuration initiale $\mathbf{c} \in \mathcal{J}$, et on choisit à chaque instant une transformation $T \in \mathcal{T}$, avec laquelle on transforme la configuration $\mathbf{c}$ en $T(\mathbf{c})$. Ce choix peut être déterministe, non déterministe, peut dépendre ou non de la configuration courante. Ce processus est itéré jusqu'à ce que l'on atteigne une configuration appartenant à l'ensemble $\mathcal{O}$. On s'arrête alors.

On adapte ce cadre général à la modélisation de l'algorithme LLL en dimension d :

(a) L'ensemble $\mathcal{C}$ est un sous-ensemble de $\mathbb{R}^{d-1}$. La configuration initiale est définie à partir des rapports de Siegel initiaux r_i, et on pose

$$\mathbf{c} = (c_1, \ldots, c_i, \ldots, c_{d-1}) \in \mathbb{R}^{d-1} \quad \text{avec} \quad c_i := -\log_s r_i.$$

L'ensemble $\mathcal{O}$ des configurations finales s'écrit comme un produit cartésien $\mathcal{O} = \underline{\mathcal{O}}^{d-1}$ et l'ensemble $\underline{\mathcal{O}} \subset \mathbb{R}$ sera relié à la condition de Lovász ou de Siegel selon les cas.

(b) L'ensemble $\mathcal{T}$ sera

$$\mathcal{T} := \{T_\alpha^{(i)};\ \ i \in [1\,..\,d-1], \alpha \in \mathbb{R}^+\}.$$

L'indice i correspond à la position de la réduction, le réel α est le décrément, et la transformation $T_\alpha^{(i)} : \mathbf{c} = (c_1, \ldots, c_{d-1}) \mapsto \check{\mathbf{c}} = (\check{c}_1, \ldots, \check{c}_{d-1})$ est définie par

$$\check{c}_j = \begin{cases} c_j + \alpha & \text{si } j \in \{i-1, i+1\} \\ c_j - 2\alpha & \text{si } j = i \\ c_j & \text{sinon.} \end{cases}$$

Entrées : La configuration initiale $\mathbf{c} \in \mathcal{J} \subset \mathbb{R}^{d-1}$
Sorties : La configuration finale $\mathbf{c} \in \mathcal{O} = \underline{\mathcal{O}}^{d-1}$ (si l'algorithme termine)

tant que $\mathbf{c} \notin \mathcal{O}$ **faire**
  `Choisir` i `tel que` $c_i \notin \underline{\mathcal{O}}$;
  `Choisir` $\alpha \in \mathbb{R}^+$;
  $\mathbf{c} \leftarrow T_\alpha^{(i)}(\mathbf{c})$;
fintq
Renvoyer $\mathbf{c}$

FIGURE 4.7: Schéma général d'un système dynamique pour modéliser l'algorithme LLL.

Finalement, une modélisation probabiliste d'un algorithme de réduction sera complètement définie par les configurations initiales $\mathcal{J}$ et finales $\underline{\mathcal{O}}$, et les deux choix, celui de l'indice i, et celui du décrément α, comme cela est décrit dans la Figure 4.7.

4.6.5 La stratégie.

Le choix de l'indice i de la boîte à réduire, qui est effectué par la fonction `Choisir` i définit ce qu'on appelle la stratégie. Dans la version classique de l'algorithme LLL, la réduction commence à partir de l'indice $i = 1$ et, ensuite, l'indice i est incrémenté ou décrémenté à chaque étape. Cependant, la position i où la réduction a lieu peut très bien être choisie différemment.

Pour tout $j \in [1\,..\,K]$, on désigne par $\mathcal{NR}(j)$ l'ensemble des positions $i \in [1\,..\,n-1]$ pour laquelle $c_i \notin \underline{\mathcal{O}}$ au début de l'itération j. Les trois types de stratégie les plus naturels sont les suivants :

– *Standard.* C'est la stratégie standard de l'algorithme LLL et elle effectue la réduction de gauche à droite, c.-à-d., i sera le plus petit indice dans l'ensemble $\mathcal{NR}(j)$.
– *Glouton.* Cette stratégie commence par réduire la boîte la moins réduite. Donc l'indice i est choisi dans $\mathcal{NR}(j)$ tel que c_i soit maximal.
– *Aléatoire.* L'indice i est choisi aléatoirement dans l'ensemble $\mathcal{NR}(j)$.

Bien sûr, la stratégie peut avoir une influence importante à la fois sur le nombre d'itérations et sur la configuration de la sortie. Dans chacun des modèles qui vont être décrits dans la section suivante, on pourra effectuer l'analyse pour chacune des stratégies définie ci-dessus. Mais nous montrerons que, dans le premier modèle, le modèle dit `M1`, le comportement de l'algorithme (aussi bien pour son nombre d'itérations que sa configuration de sortie) ne dépend pas de la stratégie adoptée.

Nous considérerons la modélisation des configurations initiales dans le Chapitre 6. Nous nous concentrons ici sur la modélisation de l'exécution, et nous nous posons les questions suivantes
– Comment choisir α ?
– Comment définir la configuration de sortie $\underline{\mathcal{O}}$?
Nous introduisons maintenant cinq modèles pour l'exécution de l'algorithme LLL.

4.7 Les cinq modèles pour l'exécution de l'algorithme LLL

Nous proposons ici deux grands groupes de modèles. Dans le premier groupe on trouve les modèles apparentés à l'algorithme LLL pour la condition de Siegel (`M1`, `M2`, `M4`). En effet, dans ces trois modèles, la condition est fixe avec une modélisation de plus en plus précise des coefficients sous-diagonaux. Dans le deuxième groupe, on trouve `M3` et `M5` (qui n'est autre que le véritable LLL) où la condition de sortie varie au cours de l'exécution. Une perspective très intéressante consisterait à étudier plus en profondeur le modèle `M3`, ce qui n'est pas fait dans cette thèse.

4.7.1 Le modèle `M1` à base de CFG.

Le premier modèle est le plus simple. Il a été proposé par Madritsch et Vallée [71] qui l'ont aussi étudié. Ici :
– le décrément $\alpha = -\log_s \rho$ est fixé une fois pour toute, et est constant tout le long de l'exécution de l'algorithme
– le test de sortie est celui de Siegel, et donc l'ensemble $\underline{\mathcal{O}}$ est l'intervalle $]-\infty, +1]$. Notons que la condition de Lovász n'a pas d'intérêt ici puisque les coefficients sous-diagonaux ne sont jamais pris en compte.

Le modèle obtenu est un modèle très classique de l'algorithmique, appelé « Chip Firing Game » (*cfg* en abrégé) et introduit par Back, Tang et Wiesenfeld [10]. Ce modèle est le plus facile à étudier car il satisfait les deux propriétés suivantes : le nombre d'itérations et la configuration finale ne dépendent pas de la stratégie choisie. Dans l'article [71], les auteurs décrivent les principaux résultats sur le temps d'exécution et les propriétés de la configuration finale. Nous présenterons plus en détail ces résultats dans le chapitre 5.

Dans les modèles qui suivent, plus réalistes que le modèle `M1`, les hypothèses seront moins simplificatrices.

4.7.2 Le modèle `M2` : système dynamique déterministe.

On prend en compte les coefficients sous-diagonaux. On suppose que leurs carrés sont constants et égaux à μ_0. On peut choisir n'importe quelle valeur $\mu_0 \in [0, 1/4]$ et par exemple la

valeur moyenne $\mu_0 = 1/12$ obtenue pour une distribution uniforme sur $[-1/2, +1/2]$. La valeur du décrément α dépend alors uniquement de la hauteur c_i correspondant à la position i choisie,

$$\alpha = -\frac{1}{2}\log_s\left(s^{-2c_i} + \mu_0\right).$$

La condition d'arrêt, elle, ne varie pas au cours du temps, elle est définie par $\underline{\mathcal{O}} = [-\infty, H]$ avec une hauteur H qui peut être de type Siegel : $H = 1$, ou de type Lovász :

$$H = \frac{-1}{2}\log_s\left(\frac{1}{s^2} + \frac{1}{4} - \mu_0\right) = \frac{-1}{2}\log_s\left(\frac{1}{t^2} - \mu_0\right).$$

Remarquons que la condition de Lovász pour le paramètre $t = 1$ se retrouve avec $s = s_0 = 2/\sqrt{3}$ et s'écrit

$$H = \frac{-1}{2\log s_0}\log\left(1 - \mu_0\right),$$

et nous la considérerons en détail dans le chapitre 7. Le système dynamique obtenu est bien plus difficile à étudier car le décrément est maintenant variable. Dans ce modèle le nombre d'itérations et la configuration finale dépendent de la stratégie et on ne peut plus utiliser les méthodes utilisées pour étudier les « Chip Firing Game ». Néanmoins, ce système dynamique garde une propriété intéressante, car la condition d'arrêt ne varie pas au cours du temps.

Nous avons commencé à étudier ce modèle dans le chapitre 7. Il est plus facile d'adopter dans le modèle `M2` un point de vue multiplicatif. Nous nous sommes d'abord concentrés sur une dimension de réduction $d = 3$, qui est la première dimension difficile pour l'algorithme LLL. La dynamique que nous obtenons est très instructive et permet de mieux comprendre l'algorithme LLL en dimension 3, et la manière dont l'algorithme LLL(1) se compare à l'algorithme LLL(t). Nous avons aussi établi une conjecture pour une dimension générale d.

4.7.3 Le modèle `M3` : système dynamique probabiliste.

On peut penser à un modèle moins simplifié où l'on n'utilise pas toujours la même valeur pour le coefficient sous-diagonal. Ce coefficient sous-diagonal n'est plus supposé constant, mais uniformément distribué dans l'intervalle $[-1/2, 1/2]$. On obtient ainsi un système dynamique probabiliste : À chaque itération, on tire aléatoirement et indépendamment un nouveau coefficient ν uniformément dans l'intervalle $[-1/2, 1/2]$, on pose $\mu = \nu^2$ et le décrément α

$$\alpha = -\frac{1}{2}\log_s\left(s^{-2c} + \mu\right).$$

utilisé pour cette itération dépend à la fois de la hauteur c_i et de la valeur μ. C'est la condition de Lovász qui devient alors la plus naturelle,

$$c \leq H \qquad \text{avec} \quad H := \frac{-1}{2\log s}\log\left(\frac{1}{s^2} + \frac{1}{4} - \mu\right) = \frac{-1}{2}\log_s\left(\frac{1}{t^2} - \mu\right)$$

La hauteur H devient variable, car dépendant de la valeur μ qui vient d'être tirée. Remarquons que la condition de Lovász pour le paramètre $t = 1$ se retrouve avec $s = s_0 = 2/\sqrt{3}$ et fait intervenir la hauteur

$$H = \frac{-1}{2\log s_0}\log\left(1 - \mu\right).$$

Il faut noter que la variable aléatoire μ (ou ν) est indépendante de la "hauteur" c_i. Mais on peut aussi, et en particulier quand la hauteur c_i devient petite prendre en compte des distributions de la variable ν qui soient plus réalistes.

Ce système est encore plus difficile à étudier, et nous n'avons pas débuté son étude pendant cette thèse. Ce serait intéressant d'approfondir cette étude pour voir si les caractéristiques de cet algorithme « moins simplifié » sont plus proches de celles du véritable algorithme LLL.

4.7.4 Le modèle M4 dit de Siegel : algorithme LLL avec la condition de Siegel.

Dans ce modèle, à tout instant de l'exécution, le coefficient $m_{i+1,i}$ et la hauteur c_i sont *calculés* par l'algorithme LLL, et le décrément α dépend de ces deux variables. Mais la condition d'arrêt est celle de Siegel $c_i \leq 1$, elle ne dépend que de c_i. Elle reste simple et fixe tout au long de l'algorithme. C'est pourquoi cet algorithme dit de Siegel se laisse sans doute mieux approché par les modèles simplifiés.

4.7.5 Le modèle M5 dit de Lovász : algorithme LLL avec la condition de Lovász.

Le modèle M5 présente le "vrai" algorithme LLL, où le facteur de décroissance et la condition d'arrêt (la condition de Lovász) à l'indice i dépendent de c_i et $m_{i+1,i}$. Par rapport au modèle précédent, il y a une difficulté supplémentaire, puisque le domaine pour la condition d'arrêt change à chaque étape de l'algorithme.

4.8 Aide à la modélisation et la simulation : notre logiciel.

Afin de stimuler notre compréhension intuitive du comportement probabiliste d'un algorithme de réduction et de pouvoir simuler de manière interactive, nous avons réalisé un petit logiciel, avec l'aide de Julien Clément. Ce logiciel permet d'observer le déroulement de l'algorithme de réduction, dans ses modélisations simplifiées ou non simplifiées. Nous l'avions réalisé au départ pour un usage interne, mais Loïck Lhote et Brigitte Vallée l'ont aussi utilisé lors d'un cours à l'Ecole de Printemps d'Informatique Théorique lors d'un cours sur l'analyse probabiliste des algorithmes de réduction. C'est un bon instrument pédagogique, bien adapté aux publics qui découvrent pour la première fois

La première version permet de visualiser l'exécution des modèles M4 ou M5 : ce sont donc les "véritables" algorithmes de réduction, et on spécifier si on veut utiliser la version de Siegel (modèle M4)(voir figure 4.8) ou celle de Lovász (modèle M5) (voir figure 4.9). La simulation comporte six sous fenêtres.

Fenêtre 1-. (en haut à gauche) On observe l'évolution de la suite $(q_1, \ldots, q_n)$ des logarithmes des longueurs des orthogonalisés. C'est donc le point de vue des tas de Sable (voir chapitre suivant). Comme à la fin de l'exécution, les piles deviennent très petites, on peut à chaque instant changer d'échelle et redimensionner la taille des piles, par rapport à la dimension de la plus grand pile. Cela facilite grandement l'observation.

Fenêtre 2-. (en haut à droite) On observe l'évolution de la suite $(c_1, \ldots, c_{n-1})$ des logarithmes des rapports de Siegel. C'est donc le point de vue des chip firing games (voir chapitre suivant). La partie en orange, est la quantité avec laquelle le pile diminue pendant l'itération courante. En gris, on représente les quantités gagnées par les piles voisines.

Fenêtre 3-. (au milieu à gauche) On visualise la matrice de passage. Les coefficients sont proportionnels à l'aire des carrés. La couleur verte est utilisé pour les valeurs négatives et la couleur rouge pour les valeurs positives.

Fenêtre 4-. (au milieu à droite) On montre ici la marche aléatoire qui est le graphe de la fonction $j \mapsto P(j)$ qui associe au temps d'exécution j la position $i = P(j)$ où prend place la réduction.

Fenêtre 5-. (en bas à gauche) On visualise ici la distribution des bases locales de sortie à l'intérieur du domaine fondamental.

FIGURE 4.8: Fonctionnement de logiciel V1 avec la Condition de Siegel

Fenêtre 6-. (en bas à droite) On visualise l'évolution du décrément α pendant l'exécution. C'est le graphe de la fonction $j \mapsto \alpha(j)$ qui est représenté.

La version 2 (voir Figure 4.10) permet de visualiser l'exécution du modèle simplifié `M1` (tas de sable ou *cfg*). Pour cette version, on peut choisir la quantité h de sable enlevée à chaque itération et le seuil H. On dispose des six mêmes fenêtres.

4.9 Conclusion

Il nous faut maintenant étudier ces modèles simplifiés : le modèle `M1` sera étudié au Chapitre 5 et le modèle `M2` sera étudié au chapitre 7. Il nous faut aussi les comparer avec les vrais modèles `M4` et `M5`, en particulier pour des modélisations d'entrée qui prennent en compte la réalité cryptographique introduites au Chapitre 6.

Mais il nous faut aussi discuter le bien-fondé des hypothèses simplificatrices. Voici les principales questions qu'on peut se poser à ce sujet :

- L'hypothèse « le décrément α est constant » est-elle raisonnable ?
- L'hypothèse « les $m_{i+1,i}$ sont indépendants des c_i » est-elle valide ?
- Dans le cas du modèle `M1`, on peut chercher à déterminer la valeur "optimale" du décrément α, pour laquelle l'algorithme simplifié se « rapproche » de l'algorithme LLL. Il est également intéressant de savoir si cette valeur de α dépend du modèle probabiliste d'entrée (voir le chapitre 6).
- Est-ce que la marche aléatoire qui décrit l'évolution de la position i au cours de l'exécution dans ces modèles simplifiés est de même type que celle observée pour le vrai algorithme LLL ?

Dans le chapitre 8, nous avons fait une série d'expérimentations afin de comparer le comportement des algorithmes avec celui de leurs versions simplifiées. Puisque la modélisation de l'exécution de l'algorithme se fait *via* les tas de sable, nous avons étudié et modélisé les entrées

FIGURE 4.9: Fonctionnement de logiciel V1 avec la Condition de Lovász

FIGURE 4.10: Fonctionnement de logiciel V2

particulières qui nous intéressent, liées aux réseaux cryptographiques, du point de vue des tas de sable. Ces modélisations (présentées dans le chapitre 6) permettent de faire des expérimentations adaptées (voir le chapitre 8) et de mieux comprendre le fonctionnement de nos algorithmes –simplifiés ou non– sur ces différentes familles.

Chapitre 5

Modèle M1 : Tas de sable et chip firing game

Sommaire

Les modèles «Tas de sable» et «Chip firing game» ont été introduits par Bak, Tang et Wiesenfeld [10] pour simuler différents phénomènes physiques, comme les formations de tas de sable ou les avalanches de neige. Plusieurs de leurs caractéristiques, liés à leur dynamique (configuration d'équilibre et temps pour parvenir à cet équilibre), sont bien étudiées, [51]. Madritsch et Vallée [71] ont montré que ces modèles pouvaient être aussi utilisés pour modéliser de manière simplifiée des algorithmes de réduction de réseau, mais il n'est plus légitime de se limiter dans ce cas à des paramètres à valeurs entières et positives. Ces auteurs ont donc généralisé l'étude de ces modèles très classiques, pour prendre en compte des valeurs réelles de signe quelconque. Nous décrivons leurs résultats, mais en nous concentrons sur une présentation plus axée vers les "chip firing game" que sur les tas de sable, car ce sont les chip firing games (en abrégé *cfg*) qui sont naturellement attachés à l'objet central de la réduction des réseaux que sont les rapports de Siegel.

Nous commençons par une description générale, en introduisant les paramètres classiques de ces modèles et les différents types de configurations. Puis, nous donnons la preuve du résultat fondamental, qui affirme que à la fois la configuration finale et le nombre d'étapes pour l'obtenir sont indépendants de la stratégie. Puis nous nous concentrons sur trois types de *cfg* (ceux à une seule pile, ceux qui sont positifs, ou ceux qui sont de signe quelconque, que nous voyons comme des piles de sable avec des "trous"). Nous revenons à la fin aux réseaux euclidiens et traduisons dans ce cadre les résultats décrits dans ce chapitre.

5.1 Description générale

Nous introduisons les tas de sable et les chip firing games, qui présentent deux points de vue sur le même modèle, et nous analysons précisément leurs relations, puis nous décrivons les différents types de configurations initiales.

Ces modèles manipulent un ensemble de réels (q_i pour les tas de sable, ou c_i pour les chip firing games) qu'on peut imaginer comme des piles de sable. Dans chacun des modèles, les piles évoluent en "donnant" du sable à leur voisines.

Dans la suite, on considèrera deux réels $H \geq 0$ et $h > 0$.

5.1.1 Tas de sable

Définition 5.1 (Tas de sable)**.** *On appelle* tas de sable *le modèle $\mathcal{Q}_d(\mathbf{q}, H, h)$ qui décrit les évolutions possibles des configurations partant de la configuration initiale $\mathbf{q} = (q_1, \ldots, q_d) \in \mathbb{R}^d$ sous l'action de la famille des fonctions de transition $\mathcal{F}(H, h) = \{f_i\}_{i=1..d-1}$. Pour $i \in \{1, \ldots, d-1\}$ la fonction $f_i : \mathbf{q} \mapsto \check{\mathbf{q}}$ est définie par*

$$\check{q_j} = \begin{cases} q_j - h & \text{si } j = i \text{ et } q_i - q_{i+1} > H, \\ q_j + h & \text{si } j = i+1 \text{ et } q_i - q_{i+1} > H, \\ q_j & \text{sinon.} \end{cases}$$

Une fonction de transition est applicable à la position $i \in \{1, \ldots, d-1\}$ si deux positions successives ont une différence de taille supérieure à H, c.-à-d., $q_i - q_{i+1} > H$. Puis l'étape d'écroulement consiste à déplacer h grains de sable de la position i à la position $i+1$. La quantité de sable globale reste constante.

5.1.2 "Chip firing game"

De manière analogue on définit le modèle «Chip Firing Game», où la configuration $\mathbf{c} = (c_1, \ldots, c_{d-1})$ est définie à partir de la configuration $\mathbf{q}$ de la façon suivante

$$c_i = q_i - q_{i+1}, \qquad \text{pour } i \in [1\,..\,d-1].$$

Définition 5.2 (Chip Firing Game)**.** *On appelle* Chip Firing Game *(cfg) le modèle* $\mathcal{C}_d(\mathbf{c}, H, h)$ *qui décrit les évolutions possibles des configurations en partant d'une configuration initiale* $\mathbf{c} = (c_1, \ldots, c_{d-1}) \in \mathbb{R}^{d-1}$ *sous l'action de la famille des fonctions de transition* $\mathcal{G}(H, h) = \{g_i\}_{i=1..d-1}$*, où pour* $i \in \{1, \ldots, d-1\}$ *la fonction de transition* $g_i : \mathbf{c} \mapsto \check{\mathbf{c}}$ *est définie par*

$$\check{c}_j = \begin{cases} c_j - 2h & \text{si } j = i \text{ et } c_i > H, \\ c_j + h & \text{si } j \in \{i-1, i+1\} \text{ et } c_i > H, \\ c_j & \text{sinon.} \end{cases}$$

Les règles de ce modèle sont aussi simples : Lorsque la hauteur d'une pile c_i d'indice i est trop élevée, i.e., supérieure à une quantité H, une quantité de sable égale à $2h$ est retirée de cette pile et distribuée à parts égales de h entre ses deux voisines c_{i-1} et c_{i+1} (quand ils existent). Dans ce modèle, une fonction de transition ne modifie la quantité de sable totale que lorsque l'indice est à la position 1 ou $d-1$. Dans chacun de ces deux cas, une quantité de h grains est éliminée du système. L'évolution du système s'arrête quand toutes les piles ont une taille inférieure à la quantité H.

5.1.3 Relation entre tas de sable et chip firing game.

L'application ∇ réminiscente des différences finies

$$\nabla : \begin{cases} \mathbb{R}^d & \longrightarrow \mathbb{R}^{d-1} \\ \mathbf{u} = (u_1, \ldots, u_d) & \longmapsto \mathbf{v} = (v_1, \ldots, v_{d-1}) \end{cases}$$

avec $v_i = v_i - v_{i+1}$ pour $i \in [1\,..\,d-1]$ permet de transformer un tas de sable en un *cfg*. Plus précisément, il associe à un tas de sable $\mathcal{Q}_d(\mathbf{q}, H, h)$ avec une configuration initiale $\mathbf{q} = (q_1, \ldots, q_d)$, un *cfg* $\mathcal{C}_d(\mathbf{c}, H, h)$ avec configuration initiale $\mathbf{c} = (c_1, \ldots, c_{d-1})$ où $\mathbf{c} = \nabla\mathbf{q}$.

Notons que l'application ∇ n'est pas injective, et qu'on perd de l'information dans le passage d'un tas de sable au *cfg* correspondant, puisque l'ensemble $\nabla^{-1}(\mathbf{c})$ est égal à

$$\nabla^{-1}(\mathbf{c}) = \left\{ \mathbf{q} = (q_1, \ldots, q_d); \quad q_i = \sum_{k=i}^{d-1} c_k + q_d, \forall i \in [1\,..\,d] \right\}.$$

Ces configurations sont en quelque sorte toutes équivalentes à «translation près» de la hauteur de la dernière pile. On peut donc supposer sans perte de généralité que la dernière pile q_d de la configuration initiale d'un tas de sable est toujours égale à

$$q_d^* = \max\left(0, -\min_{k \le d-1}\left(\sum_{k} c_k\right)\right).$$

Avec ce choix, on est assurés que les q_i sont positifs pour $i \in [1\,..\,d-1]$ et que $\min_i q_i = 0$. Un tel tas de sable sera qualifié de "canonique". La fonction ∇ définit une bijection entre l'ensemble des tas de sable "canoniques" et l'ensemble des *cfg*.

On verra dans la suite que, selon les cas, il sera plus naturel d'utiliser l'un ou l'autre point de vue (tas de sable ou chip firing game) pour faire les preuves. Mais nous énoncerons toujours les résultats dans le modèle *cfg*, puisque c'est ainsi que nous les appliquerons.

Un exemple d'évolution d'un «Tas de sable» et de son «chip firing game» associé est représenté sur la figure 5.1.

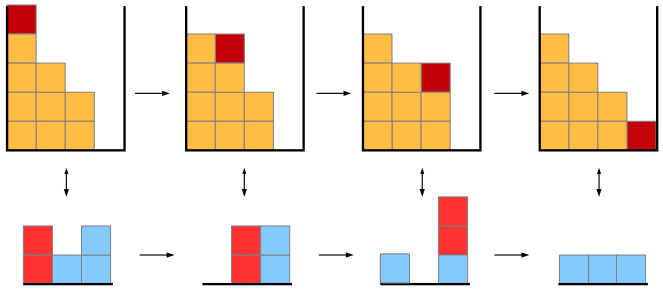


FIGURE 5.1: L'évolution d'un Tas de sable et *cfg* correspondant.

5.1.4 Différents types de tas de sable et *cfg* : cas de base et cas générique

Dans la littérature, on rencontre le plus souvent des tas de sables ou *cfg* dits «de base» :

Définition 5.3. *Un tas de sable $\mathcal{Q}_d(\mathbf{q}, H, h)$ est appelé* **tas de sable de base** *si $q_i \in \mathbb{N}$ pour $i \in \{1, \dots, d\}$ et $(H, h) = (1, 1)$.*
Un cfg $\mathcal{C}_d(\mathbf{c}, H, h)$ est appelé ***cfg* de base** *si $c_i \in \mathbb{Z}$ pour $i \in \{1, \dots, d-1\}$ et $(H, h) = (1, 1)$.*

Cependant, les *cfg* (ou le tas de sable) correspondant aux modélisations simplifiées de l'algorithme LLL conduisent au choix $(H, h) = (1, \alpha)$, avec $\alpha \in \mathbb{R}^+$. La fonction $\phi : \mathbf{c} \mapsto \mathbf{c}'$ définie par

$$c'_i := 1 - \left\lfloor \frac{H - c_i}{h} \right\rfloor, \tag{5.1}$$

transforme un *cfg* générique en un *cfg* de base. Et, de manière similaire, l'application $\psi : \mathbf{q} \mapsto \mathbf{q}'$ définie par

$$q'_d = \max\left(0, -\min_{k \le d-1}\left(\sum_{k}^{d-1} c'_k\right)\right), \qquad q'_i = \sum_{k=i}^{d-1} c'_k \text{ pour } i \in [1\,..\,d-1]$$

transforme un tas de sable générique en un tas de sable de base. De plus, on a la relation $\nabla \circ \psi = \phi \circ \nabla$.

5.1.5 Tas de sable décroissants et *cfg* positifs.

Dans la littérature, on rencontre le plus souvent des tas de sables décroissants et les *cfg* correspondants qui sont positifs. On donne une définition précise de ces types de configurations, et de leurs variantes :

Définition 5.4 (Signe d'un *cfg* de base)**.** *Un cfg de base $\mathcal{C}_d(\mathbf{c}, 1, 1)$ est dit*
(1) positif *si $c_i \ge 0$ pour chaque $i \in [1\,..\,d-1]$;*

(2) strictement positif *si* $c_i \geq 1$ *pour chaque* $i \in [1\,..\,d-1]$ *;*
(3) totalement non réduit *si* $c_i > 1$ *pour chaque* $i \in [1\,..\,d-1]$.
De manière équivalente, un tas de sable de base est dit
(1) décroissant, *i.e.,* $q_{i+1} \geq q_i$ *pour* $i \in [1\,..\,d-1]$ *;*
(2) strictement décroissant, *i.e.,* $q_{i+1} \geq q_i + 1$ *pour* $i \in [1\,..\,d-1]$ *;*
(3) totalement non réduit, *i.e.,* $q_{i+1} > q_i + 1$ *pour* $i \in [1\,..\,d-1]$.

Par la transformation ϕ qui associe un *cfg* quelconque à un *cfg* de base ayant le même graphe d'évolution, on étend ces définitions à tous les *cfg*.

Définition 5.5 (Signe d'un *cfg* quelconque)**.** *Un cfg* $\mathcal{C}_d(\mathbf{c}, H, h)$ *générique est dit*
(1) positif *si* $c_i > H - 2h$ *pour chaque* $i \in [1\,..\,d-1]$ *;*
(2) strictement positif *si* $c_i > H - h$ *pour chaque* $i \in [1\,..\,d-1]$ *;*
(3) totalement non réduit *si* $c_i > H$ *pour chaque* $i \in [1\,..\,d-1]$.

On remarque finalement que cette dernière définition est valable à la fois pour les *cfg* de base et pour pour les *cfg* quelconques. Mais les éléments c_i d'un *cfg* positif ne sont pas toujours positifs..., et les éléments c_i d'un *cfg* strictement positif ne sont pas toujours strictement positifs... Tout dépend de la position relative des réels $H, h, 2h$. Remarquons que la hauteur $H - 2h$ représente la hauteur "plancher" pour un *cfg* totalement non réduit.

5.2 Graphe des stratégies, masses, énergies.

5.2.1 Configurations accessibles et finales.

Définition 5.6. *La configuration* $\mathbf{c}'$ *(respectivement* $\mathbf{q}'$*) est accessible à partir de* $\mathbf{c}$ *(respectivement* $\mathbf{q}$*), s'il existe une suite d'indices* $i_1, \ldots, i_k \in [1\,..\,d-1]$ *telle que* $\mathbf{c}' = g_{i_k} \circ \cdots \circ g_{i_1}(\mathbf{c})$ *(*$\mathbf{q}' = f_{i_k} \circ \cdots \circ f_{i_1}(\mathbf{q})$*).*

Certaines configurations particulières jouent évidemment un rôle important, comme les configurations finales qui correspondent à des points fixes.

Définition 5.7 (Configuration finale)**.** *Une configuration* $\mathbf{c}$ *(respectivement* $\mathbf{q}$*) est* finale *si et seulement si une des deux conditions suivantes équivalentes est vérifiée :*
(1) $c_i = q_i - q_{i+1} \leq H$ *pour* $i \in [1\,..\,d-1]$,
(2) $g_i(\mathbf{c}) = \mathbf{c}$ *pour chaque* $i \in [1\,..\,d-1]$ *(*$f_i(\mathbf{q}) = \mathbf{q}$ *pour chaque* $i \in [1\,..\,d]$*).*
Tout tas de sable ou cfg ayant atteint une configuration finale est dit réduit [1]*.*

5.2.2 Graphes des stratégies.

Définition 5.8 (Graphe des stratégies)**.** *Le graphe des stratégies associé à un cfg* $\mathcal{C}_d(\mathbf{c}, H, h)$ *de configuration initiale* $\mathbf{c}$ *est le graphe orienté* $\mathcal{S} = (V, E)$ *dont*
- *l'ensemble des sommets* V *est l'ensemble des configurations accessibles à partir de* $\mathbf{c}$*,*
- *l'ensemble* $E \subset V \times V$ *des arêtes est l'ensemble des couples* $(\mathbf{u}, \mathbf{v})$ *tels qu'il existe une fonction* $g_i \in \mathcal{G}(H, h)$ *avec* $\mathbf{v} = g_i(\mathbf{u})$ *et* $\mathbf{u} \neq \mathbf{v}$*.*

On peut définir exactement de la même façon le graphe des stratégies associé à un tas de sable $\mathcal{Q}_d(\mathbf{q}, H, h)$. On observe facilement que les deux graphes de stratégies associés à $\mathcal{Q}_d(\mathbf{q}, H, h)$ et $\mathcal{C}_d(\mathbf{c}, H, h)$ sont isomorphes dès que les configurations $\mathbf{q}$ et $\mathbf{c}$ sont liées par la relation $\mathbf{c} = \nabla(\mathbf{q})$.

1. Par analogie avec la réduction des réseaux.

5.2.3 Isomorphisme entre graphes de stratégies.

De plus, le théorème suivant montre que le graphe de stratégie associé à un *cfg* quelconque est isomorphe à celui du *cfg* de base associé $\phi(\mathbf{c})$. La fonction ϕ est définie sur ses coordonnées via (5.1).

Théorème 5.9. *Soit $\mathbf{c} \in \mathbb{R}^{d-1}$ et soient h et H deux nombres positifs, alors les deux graphes de stratégies associés aux deux configurations $\mathcal{C}_d(\mathbf{c}, H, h)$ et $\mathcal{C}_d(\phi(\mathbf{c}), 1, 1)$ sont isomorphes.*

Démonstration. Pour ne pas trop alourdir les notations, on désigne par $\mathbf{c}^*$ la configuration initiale du *cfg* dans cette preuve. Utilisant la fonction ϕ définie dans la section précédente, on doit donc montrer que :

(1) il existe une bijection entre les sommets des graphes de stratégies de $\mathcal{C}_d(\mathbf{c}^*, H, h)$ et $\mathcal{C}'_d(\phi(\mathbf{c}^*), 1, 1)$,
(2) si $\mathbf{u}$ et $\mathbf{v}$ sont deux sommets (configurations) de $\mathcal{C}_d(\mathbf{c}^*, H, h)$, alors il existe une arête entre $\mathbf{u}$ et $\mathbf{v}$ si et seulement s'il existe une arête entre $\phi(\mathbf{u})$ et $\phi(\mathbf{v})$.

Tout d'abord par définition de ϕ on a bien $\mathbf{c}'^* = \phi(\mathbf{c}^*) \in \mathbb{Z}^{d-1}$. On note que pour deux configurations courantes, $\mathbf{c}$ et $\mathbf{c}' = \phi(\mathbf{c})$, les deux conditions $c_i > H$ et $c'_i > 1$ sont équivalentes. De plus si $g_i \in \mathcal{G}(H, h)$ et $g'_i \in \mathcal{G}(1, 1)$ sont les transformations associées à l'indice i pour les deux *cfg*, alors on a la relation

$$\phi \circ g_i = g'_i \circ \phi.$$

Notons $\mathbf{u}' = \phi(\mathbf{u})$, $\mathbf{v} = g_i(\mathbf{u})$, $\mathbf{v}' = \phi(\mathbf{v})$ et $\mathbf{v}'' = g'_i(\mathbf{u}')$. On vérifie facilement l'égalité $\mathbf{v}' = \mathbf{v}''$.

$$\begin{aligned}
v'_i &= 1 - \left\lfloor \frac{H - v_i}{h} \right\rfloor = 1 - \left\lfloor \frac{H - (u_i - 2h)}{h} \right\rfloor = 1 - \left\lfloor \frac{H - u_i}{h} \right\rfloor - 2 = v'_i - 2 = v''_i \\
v'_{i-1} &= 1 - \left\lfloor \frac{H - v_{i-1}}{h} \right\rfloor = 1 - \left\lfloor \frac{H - (u_{i-1} + h)}{h} \right\rfloor = 1 - \left\lfloor \frac{H - u_{i-1}}{h} \right\rfloor + 1 = v'_{i-1} + 1 = v''_{i-1} \\
v'_{i+1} &= 1 - \left\lfloor \frac{H - v_{i+1}}{h} \right\rfloor = 1 - \left\lfloor \frac{H - (u_{i+1} + h)}{h} \right\rfloor = 1 - \left\lfloor \frac{H - u_{i+1}}{h} \right\rfloor + 1 = v'_{i+1} + 1 = v''_{i+1},
\end{aligned}$$

et que pour $j \notin \{i-1, i, i+1\}$,

$$v'_j = 1 - \left\lfloor \frac{H - v_j}{h} \right\rfloor = 1 - \left\lfloor \frac{H - u_j}{h} \right\rfloor = u'_j = v''_j.$$

Ainsi la fonction ϕ envoie de manière bijective tout chemin $\mathbf{c}^* = \mathbf{c}^{(0)} \to \mathbf{c}^{(1)} \to \cdots \to \mathbf{c}^{(\ell)}$ de longueur $\ell \geq 0$ du graphe des stratégies associé à $\mathcal{C}_d(\mathbf{c}^*, H, h)$ sur un chemin $\phi(\mathbf{c}^*) = \phi(\mathbf{c}^{(0)}) \to \phi(\mathbf{c}^{(1)}) \to \cdots \to \phi(\mathbf{c}^{(\ell)})$ du graphe des stratégies de $\mathcal{C}'_d(\phi(\mathbf{c}^*), 1, 1)$. Il y a donc bien isomorphisme entre les deux graphes. □

On définit également l'application $\psi : \mathbf{q} \mapsto \mathbf{q}'$ par

$$q'_d = \max\left(0, -\min_{k \leq d-1}\left(\sum_{k}^{d-1} c'_k\right)\right), \qquad q'_i = \sum_{k=i}^{d-1} c'_k \text{ pour } i \in [1\,..\,d-1].$$

Alors $\nabla \circ \psi = \phi \circ \nabla$, et les graphes de stratégies correspondant à $\mathcal{Q}_d(\mathbf{q}, H, h)$ et $\mathcal{Q}_d(\psi(\mathbf{q}), 1, 1)$ sont isomorphes par l'application ψ.

Corollaire 5.10. *Soit $\mathbf{q} \in \mathbb{R}^d$ et deux réels $H \geq 0$ et $h > 0$. Alors il existe une configuration $\mathbf{q}' \in \mathbb{N}^d$, tel que les deux graphes $\mathcal{Q}_d(\mathbf{q}, H, h)$ et $\mathcal{Q}_d(\mathbf{q}', 1, 1)$ sont isomorphes.*

5.2.4 Masses.

Pour étudier la dynamique de ces modèles, nous introduisons diverses notions de *masse* et plusieurs types *d'énergies*.

Définition 5.11. [Masses.] *On appelle masse d'un tas de sable et masse d'un cfg les quantités suivantes :*

$$\underline{\mathcal{M}}(\mathbf{q}) = \sum_{i=1}^{d} q_i, \qquad \mathcal{M}(\mathbf{c}) = \sum_{i=1}^{d-1} c_i = q_1 - q_d. \tag{5.2}$$

On remarque que pour chaque $f_i \in \mathcal{F}_d(H,h)$, la masse du tas de sable reste invariante :

$$\underline{\mathcal{M}}(\mathbf{q}) = \underline{\mathcal{M}}(f_i(\mathbf{q})).$$

Pour les réseaux euclidiens vus comme des tas de sable, c'est le logarithme du déterminant, comme nous le verrons dans la section 5.7.
Contrairement à la masse $\underline{\mathcal{M}}$ d'un tas de sable, la masse $\mathcal{M}$ d'un *cfg* n'est pas constante, elle diminue de la quantité h lorsque l'on applique g_1 et g_{d-1}, mais n'est pas modifiée si on applique g_i pour $2 \leq i \leq d-2$.

5.2.5 Énergies.

On définit également des fonctions d'énergie $\underline{\mathcal{E}}$ et $\mathcal{E}$. Celles-ci nous permettront plus tard de mesurer le nombre d'étapes nécessaires à la réduction d'un tas de sable ou d'un *cfg*.

Définition 5.12. *On définit un type d'énergie pour une configuration* $\mathbf{q}$ *de tas de sable*

$$\underline{\mathcal{E}}(\mathbf{q}) := \sum_{i=1}^{d} i\, q_i, \tag{5.3}$$

et trois types d'énergie pour une configuration $\mathbf{c}$ *de cfg, l'énergie gauche* $\underline{\mathcal{E}}^-(\mathbf{c})$*, l'énergie droite* $\underline{\mathcal{E}}^+(\mathbf{c})$ *et l'énergie* $\mathcal{E}(\mathbf{c})$*,*

$$\underline{\mathcal{E}}^-(\mathbf{c}) := \sum_{i=1}^{d-1} i\, c_i, \qquad \underline{\mathcal{E}}^+(\mathbf{c}) := \sum_{i=1}^{d-1} (d-i)\, c_i, \qquad \mathcal{E}(\mathbf{c}) := \sum_{i=1}^{d-1} i(d-i) c_i. \tag{5.4}$$

Après chaque étape et pour chaque $f \in \mathcal{F}(H,h)$ modifiant le système, l'énergie $\underline{\mathcal{E}}$ satisfait la relation suivante :

$$\underline{\mathcal{E}}(f(\mathbf{q})) = \underline{\mathcal{E}}(\mathbf{q}) + h. \tag{5.5}$$

On a une relation analogue pour $\mathcal{E}$. Après chaque étape et pour chaque $g \in \mathcal{G}(H,h)$ modifiant le système, l'énergie $\mathcal{E}$ satisfait la relation suivante :

$$\mathcal{E}(g_i(\mathbf{c})) = \mathcal{E}(\mathbf{c}) - 2h. \tag{5.6}$$

Il est important pour la suite de noter que $\underline{\mathcal{E}}$ et $\mathcal{E}$ sont des fonctions monotones respectivement croissante et décroissante tout au long de l'évolution du système. Les énergies droites et gauches sont reliées à la masse de sable qui est exclue à droite et à gauche lors de la réduction, et joueront un rôle important dans la dernière section où nous étudierons les *cfg* à "trous".

5.2.6 Relations entre masses et énergies

Il y a une relation entre les deux énergies et la masse :

$$\mathcal{E}(\mathbf{c}) = (d+1)\underline{\mathcal{M}}(\mathbf{q}) - 2\underline{\mathcal{E}}(\mathbf{q}), \tag{5.7}$$

si la relation $\mathbf{c} = \nabla\mathbf{q}$ est vérifiée.

La masse du tas de sable $\underline{\mathcal{M}}(\mathbf{q})$ s'exprime en fonction de la hauteur q_d et de l'énergie gauche $\underline{\mathcal{E}}^-(\mathbf{c})$ du *cfg* correspondant,

$$\underline{\mathcal{M}}(\mathbf{q}) = \sum_{i=1}^{d} q_i = q_d + \sum_{i=1}^{d-1}\sum_{k=i}^{d-1}(c_k + q_d) = d\cdot q_d + \sum_{i=1}^{d-1} i\cdot c_i = d\cdot q_d + \underline{\mathcal{E}}^-(\mathbf{c}).$$

En particulier, si le tas de sable $\mathbf{q}$ est décroissant et canonique, alors la hauteur q_d est nulle. Dans ce cas, la masse du tas de sable est égale à l'énergie gauche $\underline{\mathcal{E}}^-(\mathbf{c})$. De manière analogue, la masse du tas de sable $\underline{\mathcal{M}}(\mathbf{q})$ s'exprime en fonction de la hauteur q_1, de l'énergie droite $\underline{\mathcal{E}}^+(\mathbf{c})$ et de la masse $\mathcal{M}(\mathbf{c})$ du *cfg* correspondant,

$$\underline{\mathcal{M}}(\mathbf{q}) = \sum_{i=1}^{d} q_i = dq_1 - \sum_{i=1}^{d-1}(d-i)\cdot c_i = d\cdot q_1 - \underline{\mathcal{E}}^+(\mathbf{c}) = d\,(q_1 - \mathcal{M}(\mathbf{c})) + \underline{\mathcal{E}}^-(\mathbf{c}).$$

5.2.7 Relations entre masse et énergies du *cfg* dans le cas générique et le cas de base associé.

Il nous sera utile de relier les masses et les diverses énergies d'un *cfg* $\mathcal{C}(\mathbf{c}, H, h)$ générique à celle de son *cfg* de base $\mathcal{C}(\mathbf{c}' = \phi(\mathbf{c}), 1, 1)$. Avec l'inégalité

$$\frac{c_i}{h} + \frac{h-H}{h} \le c'_i \le \frac{c_i}{h} + \frac{2h-H}{h},$$

on obtient les encadrements suivants

$$\frac{1}{h}\mathcal{E}(\mathbf{c}) + A(d)\left(\frac{h-H}{h}\right) \le \mathcal{E}(\mathbf{c}') \le \frac{1}{h}\mathcal{E}(\mathbf{c}) + A(d)\left(\frac{2h-H}{h}\right), \qquad A(d) = \sum_{i=1}^{d-1} i(d-i) = \frac{d(d^2-1)}{6},$$

$$\frac{1}{h}\mathcal{M}(\mathbf{c}) + d\left(\frac{h-H}{h}\right) \le \mathcal{M}(\mathbf{c}') \le \frac{1}{h}\mathcal{M}(\mathbf{c}) + d\left(\frac{2h-H}{h}\right),$$

$$\frac{1}{h}\underline{\mathcal{E}}^{\pm}(\mathbf{c}) + \frac{d(d-1)}{2}\left(\frac{h-H}{h}\right) \le \underline{\mathcal{E}}^{\pm}(\mathbf{c}') \le \frac{1}{h}\underline{\mathcal{E}}^{\pm}(\mathbf{c}) + \frac{d(d-1)}{2}\left(\frac{2h-H}{h}\right).$$

On en déduit les comportements asymptotiques suivants, quand la dimension $d \to \infty$

$$\frac{1}{h}\mathcal{E}(\mathbf{c}) - \mathcal{E}(\mathbf{c}') = O(d^3), \qquad \frac{1}{h}\underline{\mathcal{E}}^{\pm}(\mathbf{c}) - \underline{\mathcal{E}}^{\pm}(\mathbf{c}') = O(d^2), \qquad \frac{1}{h}\mathcal{M}(\mathbf{c}) - \mathcal{M}(\mathbf{c}') = O(d).$$

Remarquons que les termes de reste ont des ordres exacts $\Theta(d^3), \Theta(d^2), \Theta(d)$ quand les deux réels $2h-H$ et $h-H$ sont tous les deux de même signe et non nuls, c'est-à-dire pour $(2h-H)(h-H) > 0$.

5.3 Configuration finale et nombre d'itérations.

Le but de cette section est de montrer le théorème suivant, essentiel dans ce qui suit. Ce résultat a été montré d'abord par Goles et Kiwi [44] dans le cas d'un tas de sable de base partant d'une configuration initiale dite «décroissante» et ensuite généralisé par Madritsch et Vallée [71] pour un tas de sable de base quelconque. Nous l'exprimons dans le vocabulaire des *cfg*.

Théorème 5.13. *Le graphe $\mathcal{C}_d(\mathbf{c}, H, h)$ est fini et la configuration finale est unique et ne dépend pas de la stratégie.*
La longueur de tout chemin qui mène de la configuration initiale $\mathbf{c}$ à la configuration finale $\widehat{\mathbf{c}}$ est indépendante du chemin : c'est le nombre d'étapes de la réduction $K(\mathbf{c})$ du cfg $\mathbf{c}$ qui vérifie

$$K(\mathbf{c}) = \frac{1}{2h}\left[\mathcal{E}(\mathbf{c}) - \mathcal{E}(\widehat{\mathbf{c}})\right] = \frac{1}{2h}\sum_{i=1}^{d-1} i(d-i)(c_i - \widehat{c}_i).$$

Du fait de la correspondance entre les *cfg* génériques et *cfg* de base, il suffit de considérer le graphe de stratégies d'un tas de sable de base $\mathcal{C}_d(\mathbf{q}, 1, 1)$. On utilisera dans la preuve un des tas de sable dans l'ensemble $\nabla^{-1}(\mathbf{c})$ qu'on notera dans cette section $\mathcal{Q}_d(\mathbf{q})$.

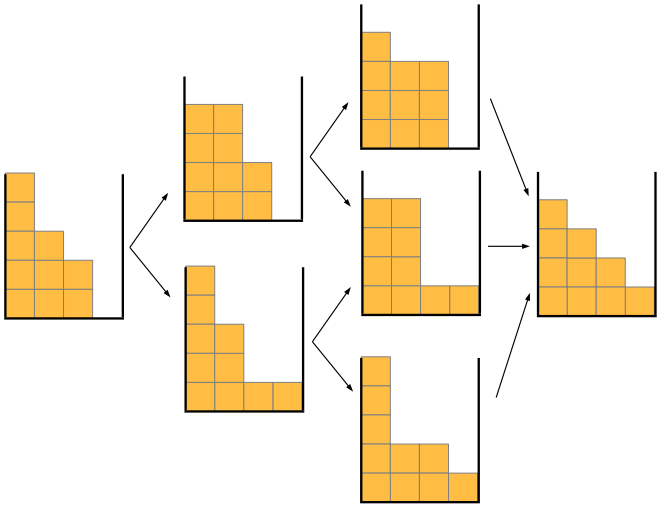


FIGURE 5.2: Graphe de stratégie d'un tas de sable.

La preuve de ce théorème procède en trois étapes. D'abord on montre que l'énergie est bornée. Ensuite, on définit un ordre sur les configurations ; une configuration est alors complètement décrite par son énergie et sa position pour cet ordre. Enfin, on montre que l'ensemble des états finaux est réduit à un seul élément.

5.3.1 Borne sur l'énergie.

Nous établissons d'abord une borne supérieure pour la fonction d'énergie.

Lemme 5.14. *Soit un tas de sable $\mathcal{Q}_d(\mathbf{q})$. Pour chaque configuration $\mathbf{u} \in \mathcal{Q}_d(\mathbf{q})$ on a les relations suivantes :*

$$\underline{\mathcal{E}}(\mathbf{q}) \leq \underline{\mathcal{E}}(\mathbf{u}) \leq \underline{\mathcal{E}}_{\max}(\mathbf{q}).$$

avec

$$\underline{\mathcal{E}}_{\max}(\mathbf{q}) = \frac{1}{2}\left((d+1)\underline{\mathcal{M}}(\mathbf{q}) - \sum_{i=1}^{d-1} i(d-i)\min\left(0, q_j - q_{j+1}\right)\right).$$

Démonstration. Pour chaque configuration $\mathbf{u} \in \mathcal{Q}_d(\mathbf{q})$, on commence par démontrer que :

$$u_i - u_{i+1} \geq \min\left(0, q_i - q_{i+1}\right).$$

Il y deux cas, par rapport la valeur de $\min(0, q_i - q_{i+1})$. Si $\min(0, q_i - q_{i+1}) = 0$ (c.-à-d., $q_i - q_{i+1} \geq 0$) quelle que soit l'évolution du système on aura toujours pour $\mathbf{u} \in \mathcal{Q}_d(\mathbf{q})$ l'inégalité $u_i - u_{i+1} \geq 0$. En effet pour un *cfg* de base, les règles empêchent une valeur positive ou nulle de devenir négative. Si $\min(0, q_i - q_{i+1}) = q_i - q_{i+1}$ (i.e., $q_i - q_{i+1} \leq 0$), au cours de l'évolution pour $\mathbf{u} \in \mathcal{Q}_d(\mathbf{q})$ on aura $u_i - u_{i+1} \geq q_i - q_{i+1}$ (toujours à cause des règles de transition : une valeur négative ne peut pas diminuer pour le *cfg* correspondant).

En conséquence pour $\mathbf{u} \in \mathcal{Q}_d(\mathbf{q})$,

$$\mathcal{E}(\mathbf{u}) = \sum_{i=1}^{d-1} i(d-i)(u_j - u_{j+1}) \geq \sum_{i=1}^{d-1} i(d-i)\min\left(0, q_j - q_{j+1}\right).$$

De la relation (5.7), on sait que

$$\begin{aligned}\underline{\mathcal{E}}(\mathbf{u}) &= \frac{1}{2}\left((d+1)\underline{\mathcal{M}}(\mathbf{u}) - \mathcal{E}(\mathbf{u})\right) \\ &\leq \frac{1}{2}\left((d+1)\underline{\mathcal{M}}(\mathbf{q}) - \sum_{i=1}^{d-1} i(d-i)\min\left(0, q_j - q_{j+1}\right)\right) =: \underline{\mathcal{E}}_{\max}(\mathbf{q}).\end{aligned}$$

Donc $\underline{\mathcal{E}}_{\max}(\mathbf{q}) \geq \underline{\mathcal{E}}(\mathbf{u}) \geq \underline{\mathcal{E}}(\mathbf{q})$ pour toute configuration $\mathbf{u} \in \mathcal{Q}_d(\mathbf{q})$. □

5.3.2 Ordre sur les configurations.

Maintenant, nous définissons un ordre partiel sur l'ensemble des configurations.

Définition 5.15. *On définit un ordre partiel avec la relation $\succcurlyeq$: pour tout couple $(\mathbf{q}, \mathbf{q}') \in \mathbb{Z}^d \times \mathbb{Z}^d$ on a*

$$\mathbf{q} \succcurlyeq \mathbf{q}' \Longleftrightarrow (\forall i \leq d) \quad \sum_{j \geq i} q_j \geq \sum_{j \geq i} q'_j.$$

Cette relation ordonne les configurations par rapport à leurs niveaux d'énergie. Le lemme suivant est une conséquence immédiate de la définition mais sera utile par la suite.

Lemme 5.16. *Soient deux configurations $\mathbf{q}$ et $\mathbf{q}'$ de $\mathbb{Z}^d$ ($d > 0$), alors*

$$\left(\underline{\mathcal{E}}(\mathbf{q}) \leq \underline{\mathcal{E}}(\mathbf{q}') \quad \textit{et} \quad \mathbf{q} \succcurlyeq \mathbf{q}'\right) \quad \Longrightarrow \quad \mathbf{q} = \mathbf{q}'.$$

Démonstration. On remarque que les inégalités $\underline{\mathcal{E}}(\mathbf{q}) \leq \underline{\mathcal{E}}(\mathbf{q}')$ et $\mathbf{q} \succcurlyeq \mathbf{q}'$ s'écrivent

$$\sum_{i \leq d}\sum_{j \geq i} q_j \leq \sum_{i \leq d}\sum_{j \geq i} q'_j, \qquad (\text{pour tout } i \leq d) \quad \sum_{j \geq i} q_j \geq \sum_{j \geq i} q'_j.$$

On en déduit

$$(\text{pour tout } i \leq d), \quad \sum_{j \geq i} q_j = \sum_{j \geq i} q'_j, \qquad \text{et donc} \quad (\forall j \leq d) \quad q'_j = q_j,$$

et on conclut donc l'égalité $\mathbf{q}' = \mathbf{q}$. □

Corollaire 5.17. *Si* $\mathbf{u} \in \mathcal{Q}_d(\mathbf{q})$*, alors* $\mathbf{u} \succcurlyeq \mathbf{q}$.

La preuve est une conséquence directe du Lemme 5.16 et de la relation (5.5) entre les énergies pour deux étapes consécutives dans l'évolution du système.

5.3.3 Etats finaux.

On définit ensuite $\mathrm{Fin}(\mathbf{q}) = \{\widehat{\mathbf{q}} \in \mathcal{Q}_d(\mathbf{q}) \mid \widehat{\mathbf{q}} \text{ est un état final}\}$. On montre le lemme suivant :

Lemme 5.18. *Pour un tas de sable de base* $\mathcal{Q}_d(\mathbf{q})$*, l'ensemble des états finaux est non vides, i.e.,* $\mathrm{Fin}(\mathbf{q}) \neq \varnothing$.

Démonstration. Supposons par l'absurde que $\mathrm{Fin}(\mathbf{q}) = \varnothing$. Il existe alors une suite infinie $(\mathbf{u}^{(0)} = \mathbf{q}, \mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots)$ de configurations distinctes de $\mathcal{Q}_n(\mathbf{q})$ correspondant à l'évolution du système. Alors la suite $\underline{\mathcal{E}}(\mathbf{u}^{(i)}) = \underline{\mathcal{E}}(\mathbf{u}^{(0)}) + i$ est strictement croissante et diverge donc vers $+\infty$. Ceci contredit le fait que l'énergie soit bornée par $\underline{\mathcal{E}}_{\max}(\mathbf{q})$. On a donc $\mathrm{Fin}(\mathbf{q}) \neq \varnothing$. □

Pour une configuration $\mathbf{u} \in \mathcal{Q}_d(\mathbf{q})$, on définit l'ensemble suivant :

$$S(\mathbf{u}) = \left\{\mathbf{v} \succcurlyeq \mathbf{u} \mid \text{Il existe } \mathbf{s} \in \mathbb{Z}^d \text{ tel que } \mathbf{v} \text{ est un état final pour le tas de sable de base } \mathcal{Q}_d(\mathbf{s})\right\}.$$

On remarque que $S(\mathbf{q})$ contient les éléments de $\mathrm{Fin}(\mathbf{q})$ (i.e., $\mathrm{Fin}(\mathbf{q}) \subseteq S(\mathbf{q})$).

Lemme 5.19. *Soit* $\mathbf{u} \in \mathcal{Q}_d(\mathbf{q})$ *et* $i \in [1\,..\,d-1]$ *tel que* $f_i(\mathbf{u}) \neq \mathbf{u}$*, alors :* $\mathbf{v} \in S(\mathbf{u}) \Rightarrow \mathbf{v} \succcurlyeq f_i(\mathbf{u})$.

Démonstration. On se place dans les hypothèses du lemme. Alors pour chaque $\ell \neq i+1$, on a

$$\sum_{j \geq \ell} (f_i(\mathbf{u}))_j = \sum_{j \geq \ell} u_j \leq \sum_{j \geq \ell} v_j.$$

Il nous reste à démontrer l'inégalité pour $\ell = i+1$, c.-à-d.,

$$\sum_{j \geq i+1} (f_i(\mathbf{u}))_j \leq \sum_{j \geq i+1} v_j.$$

Par l'absurde supposons que $\sum_{j\geq i+1} u_j \geq \sum_{j\geq i+1} v_j$. Puisque $\mathbf{v} \in S(\mathbf{u})$ on a aussi par définition $\mathbf{v} \succcurlyeq \mathbf{u}$ et donc

$$\sum_{j \geq i+1} u_j = \sum_{j \geq i+1} v_j.$$

De plus on a aussi

$$\sum_{j \geq i} u_j \leq \sum_{j \geq i} v_j, \qquad \text{et} \qquad \sum_{j \geq i+2} u_j \leq \sum_{j \geq i+2} v_j,$$

d'où l'on déduit que $u_i \leq v_i$, $u_{i+1} \geq v_{i+1}$, et donc $v_i - v_{i+1} \geq u_i - u_{i+1}$. On sait aussi que $f_i(\mathbf{u}) \neq \mathbf{u}$ et $f_i(\mathbf{v}) = \mathbf{v}$, donc $u_i - u_{i+1} > 1$ et $v_i - v_{i+1} \leq 1$. C'est une contradiction avec l'inégalité $v_i - v_{i+1} \geq u_i - u_{i+1}$. On a donc $\sum_{j\geq i+1} u_j < \sum_{j\geq i+1} v_j$ et

$$\sum_{j \geq i+1} (f_i(\mathbf{u}))_j = \left(\sum_{j \geq i+1} u_j\right) + 1 \leq \sum_{j \geq i+1} v_j.$$

□

On en déduit immédiatement le corollaire suivant :

Corollaire 5.20. *Pour chaque* $\mathbf{v} \in S(\mathbf{q})$ *et chaque* $\mathbf{w} \in \mathrm{Fin}(\mathbf{q})$*, on a* $\mathbf{v} \succcurlyeq \mathbf{w}$.

Démonstration. Soit $\mathbf{v} \in S(\mathbf{q})$ et $\mathbf{w} \in \mathrm{Fin}(\mathbf{q})$. Il existe une suite $(\mathbf{u}^{(0)} = \mathbf{q}, \mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots, \mathbf{u}^{(\ell)} = \mathbf{w})$ et une suite d'indices $i_1, i_2, \dots, i_\ell$ telles que $\mathbf{u}^{(j)} = f_{i_j}(\mathbf{u}^{(j-1)})$ et $\mathbf{u}^{(j)} \neq \mathbf{u}^{(j-1)}$ pour $1 \leq j \leq \ell$. D'après le lemme précédent, puisque $\mathbf{v} \in S(\mathbf{q})$ et $\mathbf{u}^{(1)} \neq \mathbf{q}$, on a $\mathbf{v} \succcurlyeq \mathbf{u}^{(1)}$, et donc $\mathbf{v} \in S(\mathbf{u}^{(1)})$. On peut poursuivre par induction jusqu'à obtenir $\mathbf{v} \succcurlyeq \mathbf{w}$. □

5.3.4 Conclusion de la preuve du théorème.

Nous pouvons achever la preuve du théorème 5.13. D'après le lemme 5.18, le graphe $\mathcal{Q}_d(\mathbf{q})$ est fini et il existe au moins un état final. Comme $\mathrm{Fin}(\mathbf{q}) \subset S(\mathbf{q})$, l'ensemble $S(\mathbf{q})$ n'est pas vide. On choisit un élément $\widehat{\mathbf{q}} \in S(\mathbf{q})$, tel que $\underline{\mathcal{E}}(\widehat{\mathbf{q}}) = \min_{\mathbf{u} \in S(\mathbf{q})} \underline{\mathcal{E}}(\mathbf{u})$. Soit $\mathbf{w} \in \mathrm{Fin}(\mathbf{q})$. Comme $\widehat{\mathbf{q}}$ a une énergie minimale, on a $\underline{\mathcal{E}}(\widehat{\mathbf{q}}) \leq \underline{\mathcal{E}}(\mathbf{w})$. On déduit du corollaire 5.20 que $\widehat{\mathbf{q}} \succcurlyeq \mathbf{w}$. Le lemme 5.16 implique que $\mathbf{w} = \widehat{\mathbf{q}}$ et donc que la configuration finale est unique. On déduit aussi la relation

$$\mathrm{Fin}(\mathbf{q}) = \{\widehat{\mathbf{q}}\} \quad \text{avec} \quad \widehat{\mathbf{q}} = \operatorname*{argmin}_{\mathbf{u} \in S(\mathbf{q})} \underline{\mathcal{E}}(\mathbf{u}).$$

D'après ce qui précède le modèle atteint le même état final (ou point fixe) après un nombre fini d'étapes, et ce, quelle que soit la stratégie utilisée. En revenant au *cfg*, on en déduit les deux assertions du théorème. En particulier, pour le *cfg*, c'est une application directe de l'équation (5.6) qui prouve qu'à chaque étape l'énergie $\mathcal{E}(\mathbf{c})$ diminue de $2h$, jusqu'à atteindre l'état final $\widehat{\mathbf{c}}$.

Remarque. On voit donc que les deux analyses –détermination du nombre d'itérations, et détermination de la configuration finale et de son énergie– sont très liées. On va procéder à ces analyses dans deux cas particuliers importants de deux types de *cfg* de base : les *cfg* de base à une seule pile et les *cfg* de base "positifs".

5.4 Réduction d'un *cfg* avec une seule pile

On s'intéresse d'abord aux configurations initiales de *cfg* qui contiennent une seule pile, ce qui'on appellera aussi les cfg "uni-tas". Ce type de configurations sera rencontré dans le chapitre suivant, en particulier dans la modélisation des réseaux cryptographiques de type sac-à-dos ou de type NTRU. Ces résultats ont d'abord été obtenus par Goles et Kiwi [44] quand la taille d de *cfg* n'intervient pas dans le processus de réduction. Nous nous les avons adaptés à un cadre général d'un *cfg* à une seule pile, de masse Υ et de dimension d, pour tout couple (Υ, d).

On travaille d'abord dans le cas de base, et on commence par décrire précisément la configuration de sortie $\widehat{\mathbf{c}}$ du *cfg* et donc l'énergie finale $\mathcal{E}(\widehat{\mathbf{c}})$. On en déduit ensuite le nombre d'itérations, toujours dans le cas de base. On revient alors au cas générique, et on obtient en particulier l'asymptotique quand $d \to \infty$ et pour les différents régimes de Υ en fonction de d.

Les figures 5.4 et 5.3 donnent des exemples de *cfg* «unitas».

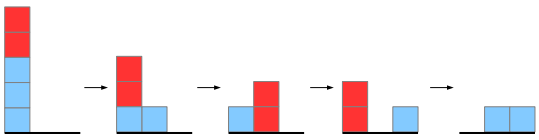


FIGURE 5.3: Un *cfg* avec un seul tas, avec $m = 1$, $d = 4$, $\Upsilon = 5$, $k = 3$ et $r = 2$.

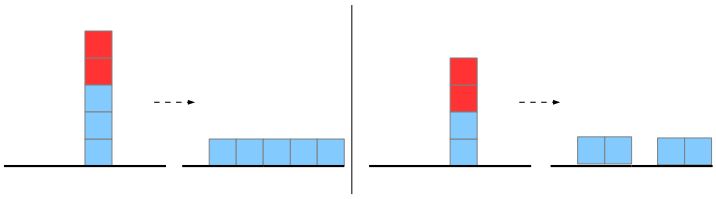


FIGURE 5.4: Un *cfg* avec un seul tas à la position $m = 4$, avec $d = 7$, $k = 3$, $r = 1$, $\Upsilon = 5$ (à gauche), et $\Upsilon = 4$ (à droite).

5.4.1 Configuration finale dans le cas de base.

Proposition 5.21 (Réduction d'un *cfg* avec une seule pile : configuration finale)**.** *On considère un cfg* $\mathcal{C}_d(\mathbf{c}, 1, 1)$ *avec une seule pile en position* m *de hauteur* Υ*. La configuration finale* $\widehat{\mathbf{c}}$ *est décrite ci-dessous, selon quatre cas :*

Cas (i). *Si* $\frac{\Upsilon(\Upsilon+1)}{2} \leq m\Upsilon \leq d\Upsilon - \frac{\Upsilon(\Upsilon+1)}{2}$*, on écrit* $m\Upsilon = a\Upsilon + \frac{\Upsilon(\Upsilon+1)}{2} + r$ *avec* $a \geq 0$ *et* $0 \leq r < \Upsilon$*.*

- *Si* Υ *est pair alors nécessairement* $r = \frac{\Upsilon}{2}$ *et* $a = m - \Upsilon - 1$ *et donc*

$$\widehat{c}_i = \begin{cases} 1 & si\ m - \frac{\Upsilon}{2} \leq i \leq m + \frac{\Upsilon}{2}\ et\ i \neq m \\ 0 & sinon. \end{cases}$$

- *Si* Υ *est impair,* $r = 0$ *et* $a = m - \frac{\Upsilon-1}{2} - 1$ *et donc*

$$\widehat{c}_i = \begin{cases} 1 & si\ m - \frac{\Upsilon-1}{2} \leq i \leq m + \frac{\Upsilon-1}{2} \\ 0 & sinon. \end{cases}$$

Cas (ii). *Si* $d\Upsilon - m\Upsilon \leq \frac{d(d-1)}{2}$ *et* $d\Upsilon - m\Upsilon \leq \frac{\Upsilon(\Upsilon+1)}{2} - 1$*, on pose* $(d-m)\Upsilon = \frac{k(k+1)}{2} + r$ *avec* $0 \leq r \leq k$*. Alors on a*

$$\widehat{c}_i = \begin{cases} 1 & si\ i \geq d - k - 1\ et\ i \neq d - k - 1 + r \\ 0 & sinon. \end{cases}$$

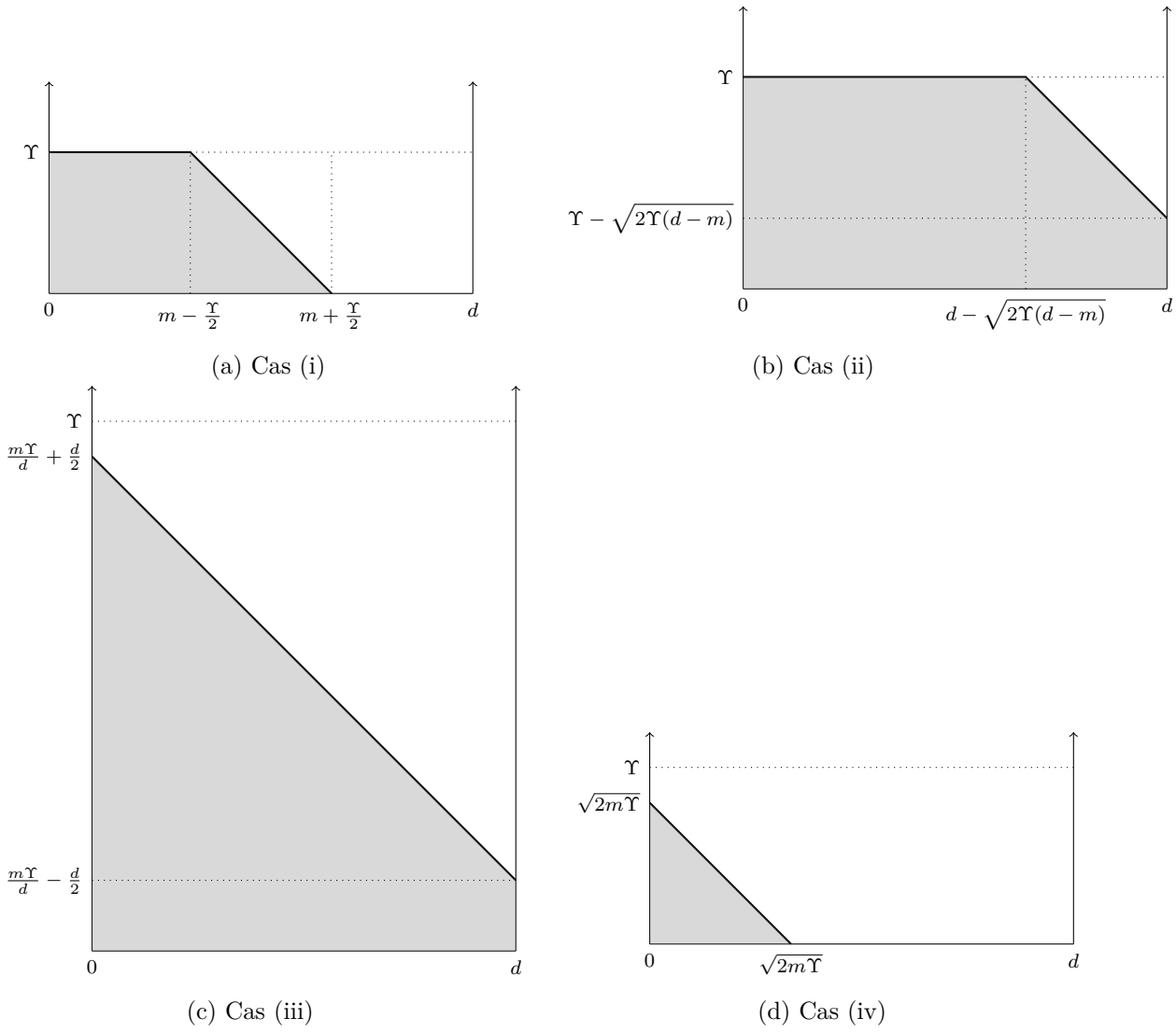
Cas (iii). *Si* $\frac{d(d-1)}{2} + 1 \leq m\Upsilon \leq d\Upsilon - \frac{d(d-1)}{2} - 1$*, on écrit* $m\Upsilon = gd + \frac{d(d-1)}{2} + r$ *avec* $g \geq 0$ *et* $0 \leq r < d$*. Alors on a*

$$\widehat{c}_i = \begin{cases} 1 & si\ 1 \leq i \leq d\ et\ i \neq d - r \\ 0 & sinon. \end{cases}$$

Cas (iv) : *Si* $m\Upsilon \leq \frac{\Upsilon(\Upsilon+1)}{2} - 1$ *et* $m\Upsilon \leq \frac{d(d-1)}{2}$*, on pose* $m\Upsilon = \frac{k(k+1)}{2} + r$ *avec* $0 \leq r \leq k$*. Alors on a*

$$\widehat{c}_i = \begin{cases} 1 & si\ 1 \leq i \leq k+1\ et\ i \neq k - r + 1 \\ 0 & sinon. \end{cases}$$

Les configurations finales pour les tas de sable correspondants sont schématisées sur la figure 5.5.



(a) Cas (i)

(b) Cas (ii)

(c) Cas (iii)

(d) Cas (iv)

FIGURE 5.5: Les quatre possibilités pour la configuration finale d'un tas de sable après réduction (pour un cfg initial avec une seule pile).

Démonstration. La preuve est toujours basée sur le même principe. Nous allons l'illustrer ici en regardant le cas où la pile est située à l'extrême gauche ($m = 1$).

Soit $\mathcal{Q}_d(\mathbf{q}, 1, 1)$ le tas de sable correspondant au *cfg* $\mathcal{C}_d(\mathbf{c}, 1, 1)$, où $q_1 = \Upsilon$ et $q_i = 0$ pour $\forall i \in [2\,..\,d]$. On désigne par $\widehat{\mathbf{q}}$ la configuration finale (i.e., $\mathrm{Fin}(\mathbf{q}) = \{\widehat{\mathbf{q}}\}$).

Soit $\mathbf{u} \in S(\mathbf{q})$ et $\underline{\mathcal{M}}(\mathbf{u}) = \Upsilon$. Par définition $S(\mathbf{q})$, contient tout les éléments de $\mathbf{w} \in \mathbb{Z}^d$ tel que $\forall i \leq d$, $\sum_{j\geq i} w_j \geq \sum_{j\geq i} q_j$. Dans le cas de la configuration initiale $\mathbf{q} = (\Upsilon, 0, \ldots, 0)$, $S(\mathbf{q})$ contient tous les éléments $\mathbf{w} \in \mathbb{Z}^d$ tel que $\underline{\mathcal{M}}(\mathbf{w}) \geq \Upsilon$. On définit $I(\mathbf{u}) := \{i : u_i = u_{i+1} > 0\}$. On suppose qu'on a au moins deux éléments dans $I(\mathbf{u})$ et on note avec $i_{\min} = \min_{i\in I} i$ et $i_{\max} = \max_{i\in I} +1$. Alors si on déplace un grain du tas de $i_{\max}$ sur $i_{\min}$, la nouvelle configuration $\mathbf{u}'$ vérifiera aussi $\mathbf{u}' \in S_d(\mathbf{q})$. De plus on a $\underline{\mathcal{E}}(\mathbf{u}') = \underline{\mathcal{E}}(\mathbf{u}) - i_{\max} + i_{\min} < \underline{\mathcal{E}}(\mathbf{u})$ et $I(\mathbf{u}') = I(\mathbf{u}) - 2$. En répétant cette procédure, on atteint un point fixe $\widehat{\mathbf{u}}$ tel que $\mathrm{Card}\, I(\widehat{\mathbf{u}})) < 2$ et $\underline{\mathcal{E}}(\widehat{\mathbf{u}}) < \underline{\mathcal{E}}(\mathbf{u})$. Comme $\underline{\mathcal{M}}(\widehat{\mathbf{u}}) = \Upsilon$, on a $\widehat{\mathbf{u}} \in S(\mathbf{q})$. Par les corollaires 5.20 et 5.16, on en déduit que $\mathrm{Fin}(\mathbf{q}) = \widehat{\mathbf{u}}$.

On distingue plusieurs cas selon la valeur de Υ.

Cas $\Upsilon < d(d-1)/2$. La seule configuration finale $\widehat{\mathbf{q}}$ qui satisfait les deux conditions

$$\mathrm{Card}(I(\widehat{\mathbf{q}})) < 2 \quad \text{et} \quad \underline{\mathcal{M}}(\widehat{\mathbf{q}}) = \Upsilon = \frac{1}{2}k(k-1) + r, \text{ avec } r < k,$$

est $\widehat{\mathbf{q}} = (k-1, k-2, \ldots, r+1, r, r, r-1, \ldots, 2, 1, 0, \ldots, 0)$, qui correspond à la configuration $\widehat{\mathbf{c}}$ finale du *cfg*

$$\widehat{c}_i = \begin{cases} 0 & \text{si } i = d - r, \\ 1 & \text{sinon.} \end{cases}$$

Cas $\Upsilon \geq d(d-1)/2$. La seule configuration finale $\widehat{\mathbf{q}}$ qui satisfait les deux conditions

$$\mathrm{Card}(I(\widehat{\mathbf{q}})) < 2 \quad \text{et} \quad \underline{\mathcal{M}}(\widehat{\mathbf{q}}) = \Upsilon = g \times d + \frac{1}{2}d(d-1) + r, \text{ avec } r < n$$

est $\widehat{\mathbf{q}} = (g+d-1, g+d-2, \ldots, g+r+1, g+r, g+r, g+r-1, \ldots, g+2, g+1, g)$, qui correspond à la configuration finale $\widehat{\mathbf{c}}$ du *cfg* correspondant

$$\widehat{c}_i = \begin{cases} 0 & \text{si } i = d - r, \\ 1 & \text{sinon.} \end{cases}$$

On peut généraliser et adapter cette preuve. Il y a quatre cas qui correspondent aux configurations finales (de tas de sable) schématisées sur la figure 5.5. □

5.4.2 Nombre d'itérations dans le cas de base.

Une fois la configuration finale caractérisée, on calcule le nombre K d'itérations avec l'équation (5.6), $K = (\mathcal{E}(\mathbf{c}) - \mathcal{E}(\widehat{\mathbf{c}}))/2$. La proposition suivante rassemble les résultats en ne donnant que le premier ordre du développement asymptotique. Le nombre d'itérations peut être calculé exactement mais pour la suite on aura besoin seulement de l'asymptotique.

Proposition 5.22 (réduction d'un *cfg* avec une seule pile : nombre d'itérations et énergie)**.** *Soit un cfg* $\mathcal{C}_d(\mathbf{c}, 1, 1)$ *avec une seule pile en position* m *de hauteur* Υ *de configuration finale* $\widehat{\mathbf{c}}$*. Par définition l'énergie initiale est*

$$\mathcal{E}(\mathbf{c}) = m(d-m)\Upsilon.$$

Le nombre K d'itérations pour la réduction et l'énergie finale $\mathcal{E}(\widehat{\mathbf{c}})$ est selon les cas[2]*.*

Cas (i). *Si $\frac{\Upsilon(\Upsilon+1)}{2} \leq m\Upsilon \leq d\Upsilon - \frac{\Upsilon(\Upsilon+1)}{2}$, on a pour $\Upsilon \to \infty$*

$$K = \frac{\Upsilon^3}{24} + O(\Upsilon^2), \qquad \mathcal{E}(\widehat{\mathbf{c}}) = m(d-m)\Upsilon - \frac{\Upsilon^3}{12} + O(\Upsilon^2).$$

Cas (ii). *Si $d\Upsilon - m\Upsilon \leq \frac{d(d-1)}{2}$ et $d\Upsilon - m\Upsilon \leq \frac{\Upsilon(\Upsilon+1)}{2} - 1$, on a pour $(d-m)\Upsilon \to \infty$*

$$K = \frac{\sqrt{2}}{3}\left(\Upsilon(d-m)\right)^{\frac{3}{2}} - \frac{1}{2}\Upsilon\,(d-m)^2 + O((d-m)\Upsilon),$$
$$\mathcal{E}(\widehat{\mathbf{c}}) = (d-m)d\Upsilon - \frac{2\sqrt{2}}{3}\left(\Upsilon(d-m)\right)^{\frac{3}{2}} + O((d-m)\Upsilon).$$

Cas (iii). *Si $\frac{d(d-1)}{2} + 1 \leq m\Upsilon \leq d\Upsilon - \frac{d(d-1)}{2} - 1$, on a pour $d \to \infty$*

$$K = \frac{1}{2}m(d-m)\Upsilon - \frac{d^3}{12} + O(d^2), \qquad \mathcal{E}(\widehat{\mathbf{c}}) = \frac{d^3}{6} + O(d^2).$$

Cas (iv). *Si $m\Upsilon \leq \frac{\Upsilon(\Upsilon+1)}{2} - 1$ et $m\Upsilon \leq \frac{d(d-1)}{2}$, on a pour $m\Upsilon \to \infty$*

$$K = \frac{\sqrt{2}}{3}\left(m\Upsilon\right)^{\frac{3}{2}} - \frac{1}{2}m^2\Upsilon + O(m\Upsilon), \qquad \mathcal{E}(\widehat{\mathbf{c}}) = dm\Upsilon - \frac{2\sqrt{2}}{3}\left(m\Upsilon\right)^{\frac{3}{2}} + O(m\Upsilon).$$

5.4.3 Comportement asymptotique quand $d \to \infty$ (cas de base)

Dans les modèles reliant *cfg* et réseaux de type cryptographique décrits au chapitre suivant, on utilise cette proposition 5.22 dans plusieurs cas particuliers lorsque $d \to \infty$. Nous énonçons ici deux propositions qui seront utilisées au chapitre 6. Le résultat se présente différemment selon que la pile est "sur le bord" ou non.

La pile n'est pas "sur les bords". Il y a un comportement différent selon le rapport entre la masse Υ et la dimension d :

Proposition 5.23. *Soit un cfg $\mathcal{C}_d(\mathbf{c}, 1, 1)$ avec une seule pile en position $m = \beta d$ $(0 < \beta < 1)$ de hauteur $\Upsilon = \Theta(d^a)$.*

- *Si $0 < a < 1$, alors on a*

$$\mathcal{E}(\widehat{\mathbf{c}}) \sim \mathcal{E}(\mathbf{c}) = \Theta(d^{2+a}), \quad K = \Theta(d^{3a}) \ll \mathcal{E}(\mathbf{c}).$$

- *Si $a = 1$, alors $\mathcal{E}(\widehat{\mathbf{c}})$, $\mathcal{E}(\mathbf{c})$ et K ne sont pas équivalents quand $d \to \infty$ mais on a*

$$\mathcal{E}(\widehat{\mathbf{c}}) = \Theta(d^3), \quad \mathcal{E}(\mathbf{c}) = \Theta(d^3), \quad K = \Theta(d^3).$$

- *Si $a > 1$, alors on a*

$$\mathcal{E}(\widehat{\mathbf{c}}) = \Theta(d^3) \ll \mathcal{E}(\mathbf{c}), \quad K \sim \frac{1}{2}\mathcal{E}(\mathbf{c}) = \Theta(d^{2+a}).$$

2. Comme dans la proposition précédente, les configurations finales pour les tas de sable correspondants sont schématisées sur la figure 5.5.

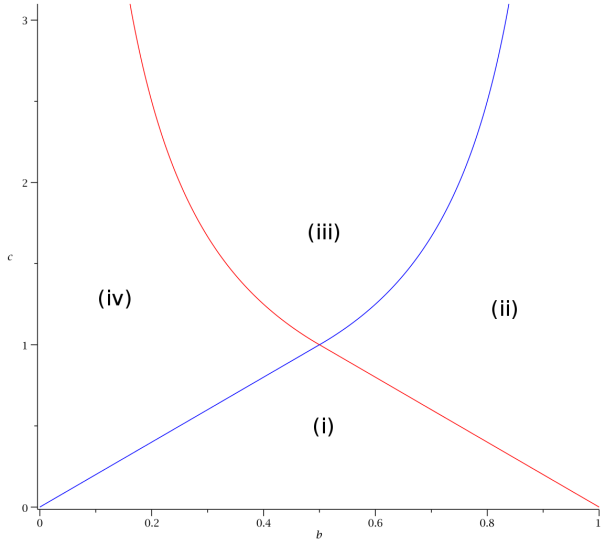


FIGURE 5.6: Les différents cas lorsque $\Upsilon = cd$, $m = \beta d$ $(0 < \beta < 1)$ et d tend vers l'infini (en fonction de β en abscisse et c en ordonnée). Les cas correspondent aux configurations finales décrites sur la figure 5.5.

Démonstration. On vérifie facilement que le cas $0 < a < 1$ correspond au cas *(i)* pour la proposition 5.22. De la même manière, on voit que le cas $a > 1$ correspond au cas *(iii)* de cette même proposition. Le cas $a = 1$ est un peu plus complexe à examiner puisque les quatre cas *(i)*-*(iv)* de la proposition 5.22 sont possibles (en fonction des valeurs de β et $c := \Upsilon/d$). Le tracé des régions correspondant à ces cas est représenté sur la figure 5.6 (pour $d \to \infty$). On vérifie ensuite en appliquant les formules de la proposition 5.22 que sur chacun des domaines, le terme dominant reste de l'ordre de d^3 (il n'y pas de «simplification»). □

La pile est "sur les bords". Il y a un comportement différent selon le rapport entre la masse Υ et le carré d^2 de la dimension :

Proposition 5.24. *Soit un cfg $\mathcal{C}_d(\mathbf{c}, 1, 1)$ avec une seule pile en position $m \in \{1, d-1\}$ de hauteur $\Upsilon = \Theta(d^a)$ $(a > 0)$.*

- *Si $a < 2$, alors on a*

$$\mathcal{E}(\widehat{\mathbf{c}}) \sim \mathcal{E}(\mathbf{c}) = \Theta(d^{a+1}), \quad K = \Theta(d^{\frac{3a}{2}}) \ll \mathcal{E}(\mathbf{c}).$$

- *Si $a = 2$, alors $\mathcal{E}(\widehat{\mathbf{c}})$, $\mathcal{E}(\mathbf{c})$ et K ne sont pas équivalents quand $d \to \infty$*

$$\mathcal{E}(\widehat{\mathbf{c}}) = \Theta(d^3), \quad \mathcal{E}(\mathbf{c}) = \Theta(d^3), \quad K = \Theta(d^3).$$

- *Si $a > 2$, alors on a*

$$\mathcal{E}(\widehat{\mathbf{c}}) = \Theta(d^3) \ll \mathcal{E}(\mathbf{c}) = \Theta(d^{a+1}), \quad K \sim \frac{1}{2}\mathcal{E}(\mathbf{c}) = \Theta(d^{a+1}).$$

Démonstration. Une fois encore, la preuve repose sur l'application de la proposition 5.22. Si $a < 2$, on vérifie qu'on est dans les cas *(iv)* ou *(ii)* de la proposition 5.22 respectivement si $m = 1$ ou $m = d - 1$. Si $a > 2$ on est dans le cas *(iii)* pour $m \in \{1, d-1\}$. Si $a = 2$, tout dépend de la position de $c := \Upsilon/d^2$ par rapport à $1/2$. Si $c < \frac{1}{2}$, on est dans les cas *(iv)* ou *(ii)* respectivement si $m = 1$ ou $m = d - 1$. Si $a = 2$ et $c \geq \frac{1}{2}$ alors on est dans le cas *(iii)* pour $m \in \{1, d-1\}$. □

5.4.4 Réduction d'un *cfg* "unitas" générique.

On revient maintenant au cas générique. On rappelle qu'un *cfg* générique "unitas" en position m n'est pas... stricto sensu "unitas". Sa configuration initiale vérifie

$$c_m \geq H - h, \qquad c_i \in]H - 2h, H - h] \quad \text{pour } i \neq m.$$

On peut décrire la configuration finale assez précisément en remplaçant dans le théorème le résultat $c_i = 1$ par $c_i \in]H - h, H]$ et $c_i = 0$ par $c_i \in]H - 2h, H - h]$. Mais ces encadrements ne sont pas assez fins pour obtenir des résultats sur le nombre d'itérations aussi précis que dans le cas de base, notamment quand l'énergie initiale $\mathcal{E}(\mathbf{c})$ est d'ordre $\Theta(d^e)$ avec $e \leq 3$.

Proposition 5.25. *Soit un cfg $\mathcal{C}_d(\mathbf{c}, h, H)$ avec une seule pile en position m de hauteur $\Upsilon = \Theta(d^a)$.*
Si la pile n'est pas sur les bords, i.e., $m = \beta d$ avec $\beta \in]0, 1[$
- *Si $0 < a \leq 1$, alors $K = O(d^3)$*
- *Si $a > 1$, alors*

$$\mathcal{E}(\widehat{\mathbf{c}}) = O(d^3) \ll \mathcal{E}(\mathbf{c}) = \Theta(d^{2+a}), \quad K \sim \frac{1}{2h}\mathcal{E}(\mathbf{c}) = \Theta(d^{2+a}).$$

Si la pile est sur les bords, en position $m \in \{1, d-1\}$:
- *Si $a \leq 2$, alors on a $K = O(d^3)$*
- *Si $a > 2$, alors on a*

$$\mathcal{E}(\widehat{\mathbf{c}}) = O(d^3) \ll \mathcal{E}(\mathbf{c}) = \Theta(d^{a+1}), \quad K \sim \frac{1}{2h}\mathcal{E}(\mathbf{c}) = \Theta(d^{a+1}).$$

5.5 Réduction d'un *cfg* positif

Les résultats de Goles et Kiwi [44] pour les tas de sable avec un seul tas ont été généralisés dans le cas des *cfg* positifs par Madritsch et Vallée. On décrit ici leur résultats.

On adopte le même cheminement que dans la section précédente : on travaille d'abord dans le cas de base, et on commence par décrire précisément la configuration de sortie $\widehat{\mathbf{c}}$ du *cfg* et donc l'énergie finale $\mathcal{E}(\widehat{\mathbf{c}})$. On en déduit ensuite le nombre d'itérations, toujours dans le cas de base. On revient alors au cas générique.

5.5.1 Configuration finale d'un *cfg* positif de base.

Pour les *cfg* de base strictement positifs, la configuration finale peut se décrire très précisément.

Théorème 5.26. *Soit un cfg de base avec une configuration initiale $\mathbf{c}$. On désigne par $\widehat{\mathbf{c}}$ la configuration finale du cfg. On a*

(i) Si la configuration initiale est positive, alors

$$\widehat{c}_i \in \{0, 1\} \quad \textit{pour} \quad \forall i \in [1 \,..\, d-1].$$

(ii) Si la configuration initiale est strictement positive, on désigne par

$$r := \left(\underline{\mathcal{E}}^-(\mathbf{c}) - \frac{d(d-1)}{2}\right) \mod d = \left(\sum_{i=1}^{d-1} i(c_i - 1)\right) \mod d$$

et la configuration finale $\widehat{\mathbf{c}}$ satisfait

$$\widehat{c}_i = 0 \quad \textit{si } i = d - r, \qquad \widehat{c}_i = 1 \quad \textit{sinon.}$$

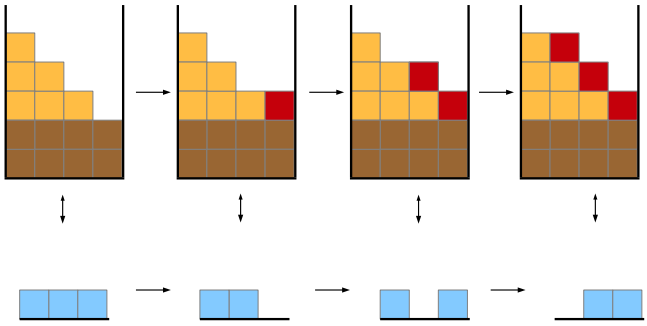


FIGURE 5.7: Les états finaux possibles pour un *cfg* (tas de sable) de taille $d = 4$.

Démonstration. Si $c_i \geq 0$, les règles de transition du *cfg* empêchent $\widehat{c}_i$ de devenir strictement négatif. Comme $\widehat{\mathbf{c}}$ est un état final, alors $\widehat{c}_i \leq 1$, d'où on déduit (i). Si $c_i > 0$, à nouveau la valeur de la pile d'indice i ne peut pas diminuer pour un *cfg* de base à cause des règles des transitions.

Il reste de démontrer le cas (ii). On considère un tas de sable $\mathbf{q}$ strictement décroissant associé au *cfg*.

D'après [71], on procède par induction sur d. Pour $d = 2$, on pose

$$\underline{\mathcal{M}}(\mathbf{q}) = q_1 + q_2 = 1 + 2g + r, \quad \text{avec } r \in \{0, 1\} \text{ et } g \geq 0.$$

On vérifie que la configuration finale est

$$\widehat{\mathbf{q}} = \left(\left\lceil \frac{\underline{\mathcal{M}}(\mathbf{q})}{2} \right\rceil, \left\lfloor \frac{\underline{\mathcal{M}}(\mathbf{q})}{2} \right\rfloor \right).$$

– Si $r = 1$, alors $\widehat{\mathbf{q}} = (g + 1,\, g + 1)$, et $e_0 = 0, e_1 = 1$.
– Si $r = 0$, alors $\widehat{\mathbf{q}} = (g + 1,\, g)$, et $e_0 = 0, e_1 = 0$.

Supposons maintenant le résultat correct pour $d > 2$. En utilisant le théorème 5.13, on peut d'abord réduire les derniers d éléments $\mathbf{q}' = (q_2, \ldots, q_{d+1})$. Soit

$$\underline{\mathcal{M}}(\mathbf{q}') = g' \times d + \frac{d(d-1)}{2} + r' \quad \text{avec } r' \in [0\,..\,d-1] \text{ et } g' \geq 0.$$

Par hypothèse, la configuration finale de $\mathbf{q}' = (q_2, \ldots, q_{d+1})$ est de la forme, pour $i \in [2\,..\,d+1]$,

$$\widehat{q'_i} = g' + (d - i + 1) + e'_i, \quad \text{avec } e'_{i+1} = \begin{cases} 0 & \text{si } i \leq d - r' \\ 1 & \text{si } i > d - r' \end{cases}.$$

Notons qu'on a $e'_2 = 0$, le tas de sable est strictement décroissant, donc $q_1 > q_2 \geq \widehat{q}_2 = g' + (d-1)$.

On écrit q_1 sous la forme $q_1 = g' + g'' \times (d+1) + r''$ avec $0 \leq r'' < d+1$. Plusieurs situations peuvent se poser en fonction de la valeur de g'' et la valeur de $r' + r'' + 1$ par rapport à d.

– Si $g'' = 0$, alors $q_1 = g' + r'' > g' + (d-1)$, donc $r'' = d$, et la configuration finale est de la forme

$$\widehat{q}_i = g' + (d + i - 1) + e'_i, \quad \text{avec } e'_{i+1} = \begin{cases} 0 & \text{si } i \leq d - r' \\ 1 & \text{si } i > d - r' \end{cases}.$$

– Si $g'' \geq 1$ nous répétons g'' fois le processus suivant : on applique d fois f_1, $d-1$ fois f_2, ..., deux fois f_{d-1} et enfin une fois f_d. Cela fait grandir la base (la partie inférieure rectangulaire) du tas de sable de g' à $g'+g''$ et conduit à la configuration

$$\check{q}_1 = g' + g'' + r'', \quad \check{q}_i = g' + g'' + (d-i+1) + e'_i \quad \text{pour } i \in [2\,..\,d+1].$$

Il y a maintenant deux cas :
– *Cas (a) :* $r' + r'' + 1 \leq d$. On pose $r := r' + r'' + 1$, et $g := g' + g'' - 1$. Nous propageons successivement $r'' - 1$ des grains de la première position vers la droite.
– *Cas (b) :* $r' + r'' + 1 > d$. On pose $g := g' + g''$ et $r := r' + r'' - d$. Alors, nous propageons successivement r'' des grains de la première position vers la droite.
Dans les deux cas, r satisfait $0 \leq r \leq d$ et on obtient une configuration finale de la forme :

$$\widehat{q}_i := g + (i-1) + e_i, \qquad \text{avec } e_i = \begin{cases} 0 & \text{si } i \leq d+1-r, \\ 1 & \text{si } i > d+1-r. \end{cases}$$

De plus la masse totale $\underline{\mathcal{M}}(\mathbf{q})$ satisfait

$$\underline{\mathcal{M}}(\mathbf{q}) = \sum_{i=1}^{d} q_i = q_1 + \underline{\mathcal{M}}(\mathbf{q}') = g \times (d+1) + \frac{d(d+1)}{2} + r.$$

□

5.5.2 Nombre d'itérations pour un *cfg* positif de base.

On traduit maintenant les résultats précédents sur le nombre d'itérations nécessaires à la réduction d'un *cfg*.

Théorème 5.27. *Considérons un cfg $\mathcal{C}_d(\mathbf{c}, 1, 1)$ avec configuration initiale $\mathbf{c}$ et configuration finale $\widehat{\mathbf{c}}$. Posons de plus $A(d) := d(d^2-1)/12$. Alors le nombre d'itérations $K(\mathbf{c})$ du système pour atteindre l'état final vérifie ce qui suit :*

(i) Si le cfg est positif, on a

$$\frac{1}{2}\mathcal{E}(\mathbf{c}) - A(d) \leq K(\mathbf{c}) \leq \frac{1}{2}\mathcal{E}(\mathbf{c}).$$

(ii) Si le cfg est strictement positif, alors on a

$$K(\mathbf{c}) = \frac{1}{2}\mathcal{E}(\mathbf{c}) - A(d) + \frac{1}{2}(d-r)r,$$

où

$$r := \left(\underline{\mathcal{E}}(\mathbf{c}) - \frac{d(d-1)}{2}\right) \mod d = \left(\sum_{i=1}^{d-1} i(c_i - 1)\right) \mod d$$

Démonstration. On applique le théorème 5.13 avec la configuration finale décrite dans le théorème 5.26. □

5.5.3 Réduction d'un *cfg* positif générique

Comme on a la correspondance entre les *cfg* génériques et les *cfg* de base (voir la définition 5.3), il suffit de considérer le graphe de base $\mathcal{C}_d(\phi(\mathbf{c}), 1, 1)$ associé au graphe générique $\mathcal{C}_d(\mathbf{c}, H, h)$ via l'application ϕ, puis de calculer sa configuration finale et enfin de revenir au graphe générique. Cette procédure permet à généraliser les résultats du théorème 5.27 dans le cas des *cfg* génériques.

Théorème 5.28. *Soit un cfg avec une configuration initiale* $\mathbf{c}$*. On désigne par* $\widehat{\mathbf{c}}$ *la configuration finale.*

(i) Si la configuration initiale est positive, alors

$$H - 2h < \widehat{c}_i \leq H \quad \textit{pour} \quad \forall i \in [1\,..\,d-1].$$

(ii) Si la configuration initiale est strictement positive, alors il existe $j \in [1\,..\,d-1]$ *tel que :*

$$(\forall i \neq j) \quad H - h < \widehat{c}_i \leq H \qquad \textit{et} \quad H - 2h < \widehat{c}_j \leq H - h.$$

Démonstration. On désigne par $\widehat{\mathbf{c}}'$ la configuration du *cfg* de base correspondant à $\widehat{\mathbf{c}}$ ($\widehat{\mathbf{c}}' = \phi(\widehat{\mathbf{c}})$) et on remarque que la condition $c'_i = 0$ est équivalente à $H - 2h < \widehat{c}_i \leq H - h$ et $c'_i = 1$ est équivalent à $H - h < \widehat{c}_i \leq H$. □

Théorème 5.29. *Considérons un cfg* $\mathcal{C}_d(\mathbf{c}, H, h)$ *de configuration initiale* $\mathbf{c}$*. Soit* $K(\mathbf{c})$ *le nombre d'itérations de la réduction. Posons de plus* $A(d) := d(d^2-1)/6$*.*

(i) Pour un cfg positif, on a

$$\frac{1}{2h}\mathcal{E}(\mathbf{c}) - \frac{A(d)}{2h}H \leq K(\mathbf{c}) \leq \frac{1}{2h}\mathcal{E}(\mathbf{c}) + \frac{A(d)}{2h}(2h - H)$$

(ii) Si le cfg est strictement positif, alors l'énergie de la configuration finale vérifie

$$A(d)[H-h] - \frac{d^2}{4} \leq \mathcal{E}(\widehat{\mathbf{c}}) \leq A(d)H$$

$$\frac{1}{2h}\mathcal{E}(\mathbf{c}) - \frac{A(d)}{2h}H \leq K(\mathbf{c}) \leq \frac{d^2}{8h} + \frac{1}{2h}\mathcal{E}(\mathbf{c}) + \frac{A(d)}{2h}\,(h - H)$$

On remarque donc que si l'énergie initiale est strictement supérieure à $A(d)H$, alors le nombre d'itérations est $\Theta(d^3)$. Si l'énergie initiale est plus petite, inférieure à $A(d)H$, le nombre d'itérations est en $O(d^3)$. Et si l'énergie initiale est $\Theta(d^e)$, avec $e > 3$, alors le théorème fournit l'équivalent asymptotique exact du nombre d'itérations, $K(\mathbf{c})) \sim (1/2h)\mathcal{E}(\mathbf{c})$.

5.6 Cas des *cfg* à "trous". Condition d'indépendance pour les blocs

Dans cette section, nous discutons le cas de *cfg* avec des configurations contenant des valeurs négatives. En effet, une pile de valeur négative (ou une suite contigüe de piles de valeurs négatives) constitue un «trou» qui sépare deux blocs formés de piles positives. En particulier, si un trou ne se remplit pas trop lors de la réduction de ses deux blocs voisins positifs, on peut alors envisager le système comme s'il était constitué de *cfg* indépendants (voir la figure 5.8). Par conséquent, la réduction de chacun des *cfg* "positifs" peut être exécutée en parallèle et l'état final du système résulte de la juxtaposition des deux configurations finales séparées par un trou.

On dira que deux blocs formés de configurations positives, séparés par un trou sont indépendants si la réduction du système global aboutit au même résultat que la réduction de chacun des blocs (sous-*cfg*) entre les «trous».

Définition 5.30. *Soit un cfg de paramètres (H, h), qui est la concaténation de trois cfg de la forme suivante*

- *un cfg $\mathcal{C}_p(\mathbf{c}^-, H, h)$ avec une configuration initiale $\mathbf{c}^-$ formée de piles strictement positives,*
- *un cfg $\mathcal{C}_r(\mathbf{c}^\dagger, H, h)$ avec une configuration initiale $\mathbf{c}^\dagger$ formée de piles négatives ou nulles,*
- *un cfg $\mathcal{C}_n(\mathbf{c}^+, H, h)$ avec une configuration initiale $\mathbf{c}^+$ formée de piles strictement positives,*

On dit que les blocs des cfg $\mathcal{C}_p(\mathbf{c}^-, H, h)$ et $\mathcal{C}_n(\mathbf{c}^+, H, h)$ sont indépendants s'il existe une configuration $\widehat{\mathbf{c}}^\dagger$ réduite telle que la configuration finale du cfg total est $\widehat{\mathbf{c}}^- \cdot \widehat{\mathbf{c}}^\dagger \cdot \widehat{\mathbf{c}}^+$ où $\widehat{\mathbf{c}}^-$ et $\widehat{\mathbf{c}}^+$ sont respectivement les configurations finales des cfg $\mathcal{C}_p(\mathbf{c}^-, H, h)$ et $\mathcal{C}_n(\mathbf{c}^+, H, h)$.

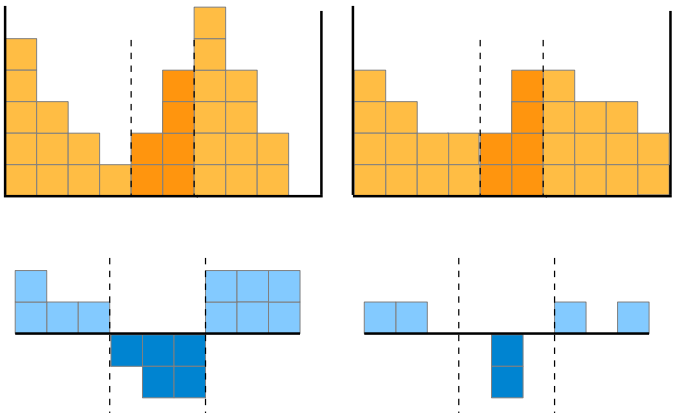


FIGURE 5.8: L'exemple d'un tas de sable (et du *cfg* correspondant en bas) qui contient deux blocs indépendants séparés pas un bloc négatif (configuration initiale à gauche et configuration finale à droite).

Notre objectif est de trouver une condition suffisante qui assure l'indépendance de deux blocs. Clairement, il suffit que le trou soit suffisamment profond pour qu'il puisse "absorber" sans déborder le sable que les deux cfg positifs, celui de gauche et celui de droite, vont déverser dans ce trou. La proposition suivante est donc essentielle dans ce cadre.

Proposition 5.31. *Soit un cfg $\mathcal{C}_d(\mathbf{c}, H, h)$ strictement positif. La quantité de sable qui sort par la gauche, désignée par k_- et la quantité de sable qui sort par la droite, désignée par k_+, s'expriment en fonction des énergies gauche et droite, initiale et finale. Plus précisément*

$$k_- = \frac{1}{d}\left(\mathcal{E}^-(\mathbf{c}) - \mathcal{E}^-(\widehat{\mathbf{c}})\right) \qquad k_+ = \frac{1}{d}\left(\mathcal{E}^+(\mathbf{c}) - \mathcal{E}^+(\widehat{\mathbf{c}})\right).$$

Démonstration. On fait la preuve dans le cas de base. Elle se généralise facilement au cas générique.

Si k_i désigne le nombre d'itérations où on effectue la transition g_i alors k_1 est aussi égal à la quantité k_- de sable qui sort du *cfg* par la gauche et k_{d-1} est aussi égal à la quantité de sable k_+ qui sort du *cfg* par la droite. On veut déterminer k_- et k_+.

On adopte une approche via les séries génératrices, et on introduit les trois polynômes

$$R(z) := \sum_{i=1}^{d-1} k_i\, z^i, \qquad C(z) := \sum_{i=1}^{d-1} c_i\, z^i, \qquad \widehat{C}(z) := \sum_{i=1}^{d-1} \widehat{c}_i\, z^i.$$

Les relations

$$k_0 = k_d = 0, \qquad 2k_i - k_{i-1} - k_{i+1} = c_i - \widehat{c}_i \quad \text{pour } i \in [1\,..\,d-1],$$

expriment le fonctionnement du *cfg*. On en déduit donc l'égalité entre polynômes

$$R(z)(1-2z+z^2) = z\left(C(z) - \widehat{C}(z)\right) - z(k_1 + k_{d-1}z^d).$$

Le polynôme de droite a donc une racine double en $z=1$, et donc sa valeur en $z=1$ et la valeur de sa dérivée en $z=1$ sont toutes deux nulles. Cela donne deux relations

$$k_1 + k_{d-1} = \sum_{i=1}^{d-1}(c_i - \widehat{c}_i), \qquad d\,k_{d-1} = \sum_{i=1}^{d-1} i\,(c_i - \widehat{c}_i).$$

La première relation exprime juste que le sable qui sort ne peut sortir que par la droite ou par la gauche. Et la seconde relation conduit aux égalités cherchées,

$$k_{d-1} = \frac{1}{d}\sum_{i=1}^{d-1} i\,(c_i - \widehat{c}_i) \qquad k_1 = \frac{1}{d}\sum_{i=1}^{d-1}(d-i)\,(c_i - \widehat{c}_i).$$

□

Théorème 5.32. *Soit un cfg de paramètres (H,h), qui est la concaténation de trois cfg de la forme suivante*

- *un cfg $\mathcal{C}_p(\mathbf{c}^-, H, h)$ avec une configuration initiale $\mathbf{c}^-$ formée de piles strictement positives,*
- *un cfg $\mathcal{C}_r(\mathbf{c}^\dagger, H, h)$ avec une configuration initiale $\mathbf{c}^\dagger$ formée de piles négatives ou nulles,*
- *un cfg $\mathcal{C}_n(\mathbf{c}^+, H, h)$ avec une configuration initiale $\mathbf{c}^+$ formée de piles strictement positives,*

Alors, les blocs $\mathcal{C}_p(\mathbf{c}^-, H, h)$ et $\mathcal{C}_n(\mathbf{c}^+, H, h)$ sont indépendants si et seulement si

$$(r-1) - \mathcal{M}(\mathbf{c}^\dagger) + \frac{1}{p}\left(\underline{\mathcal{E}}^+(\mathbf{c}^-) - \underline{\mathcal{E}}^+(\widehat{\mathbf{c}}^-)\right) + \frac{1}{n}\left(\underline{\mathcal{E}}^-(\mathbf{c}^+) - \underline{\mathcal{E}}^+(\widehat{\mathbf{c}}^-)\right) \le 0$$

Des conditions suffisantes pour l'indépendance et la dépendance sont,

(i) pour l'indépendance :

$$\frac{1}{p}\underline{\mathcal{E}}^+(\mathbf{c}^-) + \frac{1}{n}\underline{\mathcal{E}}^-(\mathbf{c}^+) - \mathcal{M}(\mathbf{c}^\dagger) \le \frac{1}{2}(p+n-2r)H - 2h$$

(ii) pour la dépendance :

$$\frac{1}{p}\underline{\mathcal{E}}^+(\mathbf{c}^-) + \frac{1}{n}\underline{\mathcal{E}}^-(\mathbf{c}^+) - \mathcal{M}(\mathbf{c}^\dagger) \ge \frac{1}{2}(p+n-2r)H$$

Démonstration. La première assertion est claire, et exprime que la quantité de sable qui se déverse dans le trou est inférieure à la masse que le trou peut absorber. La proposition précédente permet d'exprimer la quantité de sable qui se déverse dans le trou, venant du *cfg* de gauche, ou du *cfg* de droite. Par ailleurs, la quantité de sable (positive) que peut absorver le trou est égale à $(r-1) - \mathcal{M}(\mathbf{c}^\dagger)$. Le théorème 5.26 décrit précisément les configurations de sortie $\widehat{\mathbf{c}}^-$ et $\widehat{\mathbf{c}}^+$, et, en adaptant au cas générique, on obtient

$$\frac{p-1}{2}H - h \le \frac{1}{p}\mathcal{E}^+(\widehat{\mathbf{c}}^-) \le \frac{p-1}{2}H, \qquad \frac{n-1}{2}H - h \le \frac{1}{n}\mathcal{E}^-(\widehat{\mathbf{c}}^+) \le \frac{n-1}{2}H$$

ce qui fournit les deux conditions suffisantes.

□

Remarquons que dans le cas où les trois configurations initiales sont "plates" de hauteurs respectives $c^-, -c^\dagger, c^+$, les énergies gauche et droite se relient facilement à la masse,

$$\frac{1}{p}\,\underline{\mathcal{E}}^+(\mathbf{c}^-) = \frac{1}{2}\mathcal{M}(\mathbf{c}^-) = \frac{p-1}{2}c^-, \qquad \frac{1}{n}\,\underline{\mathcal{E}}^-(\mathbf{c}^+) = \frac{1}{2}\mathcal{M}(\mathbf{c}^+) = \frac{n-1}{2}c^+,$$

et donc la condition suffisante d'indépendance s'écrit

$$\mathcal{M}(\mathbf{c}^-) + \mathcal{M}(\mathbf{c}^+) - 2\mathcal{M}(\mathbf{c}^\dagger) \leq (2r+p+n)H - 2h,$$

et la condition suffisante de dépendance s'écrit

$$\mathcal{M}(\mathbf{c}^-) + \mathcal{M}(\mathbf{c}^+) - 2\mathcal{M}(\mathbf{c}^\dagger) \geq (2r+p+n)H.$$

5.7 Réduction des réseaux dans le modèle M1.

On revient maintenant sur la modélisation M1 de l'algorithme LLL, décrite dans le chapitre 4 pour appliquer les résultats de ce chapitre.

5.7.1 Relation entre paramètres du réseau et paramètres du *cfg*.

Dans le modèle $\mathtt{M1}(\alpha)$, la réduction d'un réseau donné par une base B est modélisée comme un *tas de sable* $\mathcal{Q}_d(\mathbf{q}, 1, \alpha)$ (ou un *cfg* $\mathcal{C}_d(\mathbf{c}, 1, \alpha)$, $\mathbf{c} = \nabla(\mathbf{q})$) avec :

$$\mathbf{q} = (\log_s \ell_1, \ldots, \log_s \ell_d), \qquad \mathbf{c} = (\log_s r_1, \ldots, \log_s r_{d-1}),$$

où $\ell_i = \|\mathbf{b}_i\|$ pour $i \in [1\,..\,d]$ est la longueur du i^{e} vecteur de la base $B^\star$, orthogonalisée de la base d'entrée B, et $r_i = \ell_{i+1}/\ell_i$ pour $i \in [1\,..\,d-1]$ est le rapport de Siegel associé.

Les tas de sable ou *cfg* ainsi associés à une base B, ont alors leurs principaux paramètres, comme la masse et l'énergie, décrits dans la section 5.2.4 qui sont complètement définis par des paramètres importants de la base d'entrée B correspondante, notamment son déterminant $\det(B)$, ses deux types de potentiel $D(B)$ ou $\Delta(B)$. On a en effet

$$\begin{aligned} \log_s \Delta(B) &= \mathcal{E}(\mathbf{c}) \\ \log_s \det(B) &= \underline{\mathcal{M}}(\mathbf{q}) \\ \log_s \prod_{i=1}^{d-1} r_i &= \mathcal{M}(\mathbf{c}) \end{aligned}$$

Le dernier paramètre, la masse du *cfg*, peut s'interpréter comme un coefficient global de non réduction pour le réseau associé. En tous cas, pour un réseau totalement non réduit, la masse du *cfg* associé quantifie la difficulté que le réseau à se laisser réduire.

5.7.2 Configuration de sortie dans le modèle M1.

Les rapports de Siegel $\widehat{r}_i$ de la configuration de sortie sont, par définition, tous minorés par $1/s$. Mais, on se pose la question de leur distribution dans le segment $[1/s, \infty]$ et on cherche à estimer le défaut de Hermite de la base finale $\widehat{B}$, par définition (voir 2.2)

$$\gamma(\widehat{B}) = \left(\frac{\|\widehat{\mathbf{b}}_1\|}{(\det B)^{(1/d)}}\right)^2.$$

Lemme 5.33. *Soit deux paramètres s et $\alpha > 0$. On considère une base B d'entrée dont tous les rapports de Siegel vérifient $r_i \leq (1/s)^{1-2\alpha}$. Alors, après la réduction de la base B dans le modèle* $\texttt{M1}(\alpha)$*, les rapports de Siegel de la base finale $\widehat{B}$ vérifient*

$$\widehat{r_i} \in \left[\frac{1}{s}, \frac{s^\alpha}{s}\right],$$

et le défaut de Hermite de la base finale vérifie

$$\frac{1}{d-1}\log_s \gamma(\widehat{B}) \in [1-\alpha, 1].$$

Démonstration. C'est une application immédiate du théorème 5.28. □

Ce résultat est compatible avec les expérimentations faites par Nguyen et Stehlé [91]. Les auteurs observent expérimentalement (voir la figure 5.9) que la plupart des bases de sortie $\widehat{B}$ ont un défaut d'Hermite $\gamma(\widehat{B})$ proche de β^{d-1}. Ils obtiennent une valmeur de $\beta \approx 1.04$.

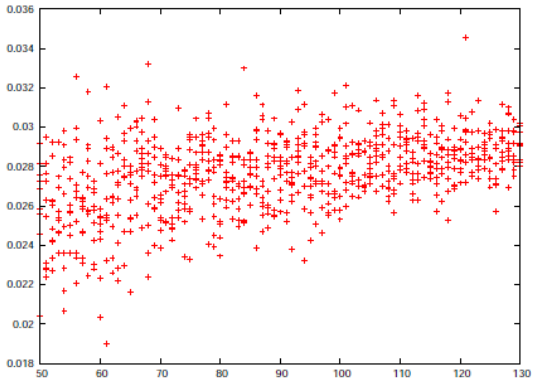


FIGURE 5.9: La distribution de paramètre $\frac{1}{d-1}\log_s \gamma(\widehat{B})$ en fonction de la dimension d.

5.7.3 Nombre d'itérations dans le modèle M1.

Le nombre d'itérations dans le modèle $\texttt{M1}(\alpha)$ vérifie (voir l'équation (4.14))

$$K(B) = \frac{1}{2\alpha}\log_s \frac{\Delta(B)}{\Delta(\widehat{B})} = \frac{1}{2\alpha}\left(\mathcal{E}(\mathbf{c}) - \mathcal{E}(\widehat{\mathbf{c}})\right).$$

Quand la base initiale B est totalement non réduite, avec une énergie initiale $\Delta(B)$ dont le logarithme est d'ordre $\Theta(d^e)$ avec $e > 3$, le *cfg* associé est lui-même totalement non réduit et on peut alors appliquer le théorème qui montre l'équivalent asymptotique suivant quand $d \to \infty$

$$K(B) \sim \frac{1}{2\alpha}\log_s \Delta(B).$$

Dans le chapitre suivant, on décrira des modèles probabilistes d'entrée où l'on pourra préciser davantage la configuration de sortie et le nombre d'itérations.

Chapitre 6

Modélisations des entrées. Exemples des réseaux cryptographiques

Sommaire

Dans les modélisations de l'exécution de l'algorithme LLL, les approches de cette thèse font jouer un rôle central aux longueurs des orthogonalisés ℓ_i et aux rapports de Siegel $r_i := \ell_{i+1}/\ell_i$. Le point de vue additif adopté nous conduit alors à étudier l'exécution de l'algorithme via les variables $q_i := -\log_s \ell_i$ et $c_i := \log_s r_i$, et c'est ainsi que la modélisation de l'exécution via les tas de sable (pour les variables q_i) ou le chip firing game –ou *cfg*– (via les variables c_i) s'impose naturellement. Dans ce chapitre, nous nous concentrons sur le modèle `M1`, le plus simple, où nous cherchons à faire une analyse probabiliste, qui soit à la fois suffisamment générique, mais qui soit aussi applicable à la réalité cryptographique. Puisque nous nous intéressons à l'analyse probabiliste de la réduction, il est essentiel que les entrées soient aussi modélisées dans le même cadre, celui des tas de sable, ou, mieux encore celui des *cfg*. Comme nous cherchons une modélisation générique des entrées, qui puisse être aussi appliquée à la "réalité cryptographique",

nous nous posons ici deux questions principales. Quels sont les principaux types de *cfg* choisis en entrée qui vont permettre de modéliser différents types d'exécution ? Ces types se rencontrent-ils naturellement en cryptographie ?

Nous introduisons ici trois types de *cfg* : les *cfg* "grands tas", formés de piles, toutes de grande hauteur positive – les *cfg* "uni-tas" constitués d'une seule grande pile positive, les autres piles étant de hauteur nulle – enfin les *cfg* "tas à trous" constitués d'alternance régulière de piles de hauteur positive et de piles de hauteur négative, qu'on appelle les trous. Il est clair que ces trois classes de *cfg* constituent des briques de base naturelles qui permettent de construire des *cfg* généraux par concaténation. Ces types de *cfg* sont déjà étudiés dans le chapitre précédent, car ils se présentent naturellement dans la problématique interne des tas de sable et des *cfg*. Mais, ici, nous montrons aussi que ces types de *cfg* se présentent naturellement dans les applications cryptographiques, que nous avons décrites au chapitre 3. Les réseaux introduits par Ajtai donnent lieu naturellement à des entrées "grands tas" ; les *cfg* "uni-tas" qui semblent pourtant des *cfg* très particuliers sont liés à des réseaux typiques de la cryptanalyse (sac-à-dos, factorisation, NTRU) ; les *cfg* "tas à trous" modélisent des cas particuliers de la méthode de Coppersmith pour trouver de petites racines modulaires, et posent la question de l'indépendance des blocs, question déjà abordée au chapitre précédent, sur laquelle nous revenons ici.

Ce chapitre s'organise en trois parties qui correspondent aux trois grandes familles de *cfg*. Pour chacune d'entre elles, nous décrivons la modélisation probabiliste générale, nous mentionnons la "réalité cryptographique" qui se trouve ainsi modélisée. Nous utilisons les résultats du chapitre précédent pour obtenir l'analyse probabiliste de la réduction, quand l'exécution est modélisée par le modèle `M1`, et quand les entrées "vivent" dans chacun des trois modèles.

Généralités. Dans ce chapitre, tous les réseaux sont de dimension pleine ($n = d$) et la base d'entrée $B := (\mathbf{b}_1, \ldots, \mathbf{b}_d)$ s'exprime dans la base canonique $(\mathbf{e}_1, \ldots, \mathbf{e}_d)$ par une matrice triangulaire inférieure $\mathcal{T}$, où on peut toujours supposer sans perte de généralité que les coefficients diagonaux $t_{i,i}$ sont positifs. On a donc

$$\mathbf{b}_i = t_{i,i} \cdot \mathbf{e}_i + \sum_{j=1}^{i-1} t_{i,j} \cdot \mathbf{e}_j.$$

Les vecteurs orthogonaux de Gram-Schmidt sont alors colinéaires aux vecteurs de la base canonique, de la forme $\mathbf{b}_i^\star = t_{i,i}\mathbf{e}_i$, de sorte que la longueur de Siegel ℓ_i vérifie $\ell_i = |t_{i,i}| = t_{i,i}$. Posant alors $t_{i,j} := \ell_j \cdot m_{i,j}$, on obtient l'écriture

$$\mathbf{b}_i = \mathbf{b}_i^\star + \sum_{j=1}^{i-1} m_{i,j} \cdot \mathbf{b}_j^\star,$$

qui montre que la grandeur $m_{i,j}$ précédemment introduite est exactement "notre" coefficient $m_{i,j}$, coefficient général de la matrice $\mathcal{P}$ du chapitre 2 qui joue un rôle central dans le processus de réduction.

Comme on peut toujours commencer par rendre la base d'entrée propre, sans modifier les longueurs ℓ_i, on suppose que les coefficients $m_{i,j}$ initiaux vérifient $|m_{i,j}| \leq 1/2$. Et finalement, dans ce cadre "triangulaire-propre", l'entrée de l'algorithme de réduction, appelée « entrée calculée » dans le chapitre 4, formée de la suite (ℓ_i) et de la matrice $\mathcal{P}$, est immédiate à obtenir à partir de la véritable entrée, donnée par la matrice $\mathcal{T}$

$$\ell_i := t_{i,i}, \qquad m_{i,j} := \frac{t_{i,j}}{t_{j,j}} \in [-\frac{1}{2}, +\frac{1}{2}].$$

De plus, du point de vue général adopté dans cette thèse, ce sont les longueurs ℓ_i qui jouent le rôle principal, et les coefficients $m_{i,j}$ qui jouent un rôle secondaire. En effet, le rôle de ces coefficients $m_{i,j}$ n'est pas pris en compte explicitement dans le modèle simplifié `M1`. Dans les modèles semi-simplifiés, ces coefficients sont considérés comme fixes tout au long de l'exécution de l'algorithme dans le modèle `M2`, ou variant aléatoirement dans l'intervalle $[-1/2, +1/2]$ tout au long de l'exécution de l'algorithme dans le modèle `M3`. Compte-tenu de ces simplifications opérées dans la modélisation de l'exécution de l'algorithme, il nous paraît légitime d'opérer le même type de simplification dès la modélisation de l'entrée, et de toujours supposer que les coefficients $m_{i,j}$ sont choisis aléatoirement dans l'intervalle $[-1/2, +1/2]$.

Finalement, les modèles d'entrée sont des modèles probabilistes, essentiellement définis par la distribution des coefficients de la diagonale $\ell_i = t_{i,i}$, qui peut varier considérablement selon les modèles, tandis que les autres coefficients $t_{i,j}$ pour $j < i$, seront toujours choisis indépendamment, et de la même manière,

$$\frac{t_{i,j}}{t_{j,j}} \quad \text{uniforme dans l'intervalle } \left[-\frac{1}{2}, +\frac{1}{2}\right].$$

Ainsi la matrice $\mathcal{P}$ aura tous ses coefficients diagonaux égaux à 1, et ses autres coefficients $m_{i,j}$ pour $j < i$ indépendants, et aléatoires dans $[-1/2, +1/2]$.

Mais, la difficulté de la réduction dépend non pas des longueurs ℓ_i, mais de leur rapports $r_i := \ell_{i+1}/\ell_i$ pour $i \in [1\,..\,d-1]$. Et c'est cette distribution qui nous intéresse, et mieux encore la distribution des hauteurs $c_i := -\log_s r_i = \log_s \ell_i - \log_s \ell_{i+1}$ de piles associées. Dans toutes les modélisations qui suivent dans ce chapitre, la distribution des entrées sera complètement – et uniquement – définie par la distribution des c_i.

Nous présentons les trois principaux modèles, et décrivons la réalité cryptographique qui se trouve ainsi modélisée. Nous effectuons dans chacun des cas l'analyse probabiliste de la réduction dans le modèle `M1`.

6.1 Le modèle dit d'Ajtai : Les entrées "grands tas"

Cette section propose un modèle pour les bases de réseaux qui donnent lieu à des configurations d'entrée "difficiles" à réduire. Ces bases ont d'ailleurs été introduites par Ajtai [2] pour cette raison, dans le cadre d'une réduction pire cas/cas moyen. Ajtai a en effet montré que, sous certaines conditions, la réduction de ces réseaux est en moyenne au moins aussi difficile que la résolution dans le pire des cas d'une variante du problème $\mathcal{SVP}$. Le *cfg* associé est constitué par de hautes piles – "des grands tas" –.

6.1.1 Description du modèle

Modèle général $\mathcal{A}(\Upsilon, d, g)$. Rappelons d'abord qu'un *cfg* de base est totalement non réduit si $c_i > 1$. Cela correspond à un réseau pour lequel toutes les boîtes sont non réduites. Le modèle $\mathcal{A}(\Upsilon, d, g)$ regroupe un ensemble de *cfg* défini par trois paramètres : la masse totale moyenne Υ, la dimension d (qui donne lieu à un *cfg* de longueur $d-1$) et une densité g strictement positive définie sur $[0, 1]$ et de classe $\mathcal{C}^1$ qui vérifie donc $\int_0^1 g(t)dt = 1$. Remarquons que, puisque le *cfg* est totalement non réduit, la masse moyenne Υ vérifie $\Upsilon > (d-1)$.

La somme de Riemann

$$\frac{1}{d-1} \sum_{i=1}^{d-1} g\left(\frac{i}{d}\right)$$

tend vers 1 quand $d \to \infty$. Le modèle $\mathcal{A}(\Upsilon, d, g)$ regroupe, par définition, des *cfg* dont la valeur moyenne $\mathbb{E}[c_i]$ de la hauteur de la i^{e} pile vérifie

$$\mathbb{E}[c_i] = \frac{1}{d-1}\,\Upsilon\, g\left(\frac{i}{d}\right).$$

Et la masse totale moyenne, définie par

$$\mathbb{E}[\mathcal{M}] = \sum_{i=1}^{d-1} \mathbb{E}[c_i] = \frac{1}{d-1}\,\Upsilon \sum_{i=1}^{d-1} g\left(\frac{i}{d}\right)$$

est bien proche de Υ. L'énergie moyenne vérifie alors

$$\mathbb{E}[\mathcal{E}] = \sum_{i=1}^{d-1} i(d-i)\mathbb{E}[c_i] = d^2\Upsilon\left(\frac{1}{d-1}\sum_{i=1}^{d-1}\frac{i}{d}\left(1-\frac{i}{d}\right) g\left(\frac{i}{d}\right)\right),$$

et donc, quand $d \to \infty$ l'énergie moyenne vérifie

$$\mathbb{E}[\mathcal{E}] \sim d^2\,\mathbb{E}[\mathcal{M}]\, I(g) \quad \text{avec} \quad I(g) := \int_0^1 x(1-x)g(x)dx\,.$$

Dans la suite, puisque la masse totale Υ est d'ordre au moins linéaire en la dimension, l'énergie moyenne dans ce modèle est donc une fonction de la dimension d'ordre au moins cubique.

Modèle original d'Ajtai. Dans la définition initiale de la distribution d'Ajtai, les rapports r_i sont fixés et choisis très petits de la forme

$$r_i = 2^{-(a+1)(2n-i)^a},$$

de sorte que

$$c_i = \log_s(2)\cdot(a+1)(2d-i)^a = \log_s(2)\cdot(a+1)d^a\left(2-\frac{i}{d}\right)^a \quad \text{avec } a>0.$$

La densité g est donc proportionnelle à $(2-x)^a$, et plus précisément

$$g_a(x) = \frac{a+1}{2^{a+1}-1}(2-x)^a,$$

de sorte que

$$\mathbb{E}[\mathcal{M}] \sim \log_s(2)\cdot(d-1)\left[2^{a+1}-1\right]d^a, \qquad \mathbb{E}[\mathcal{M}] = \Theta(d^{a+1}),$$
$$\mathbb{E}[\mathcal{E}] \sim d^2\,\mathbb{E}[\mathcal{M}]\, I(g_a), \qquad \mathbb{E}[\mathcal{E}] = \Theta(d^{a+3}).$$

Modèle à loi puissance. Il y a une autre sous-classe dans ce modèle, introduit par Madritsch et Vallée, dans [71], dans laquelle les rapports de Siegel r_i suivent des lois puissance de la forme

$$\mathbb{P}[r_i \le x] = x^{\frac{1}{\theta_i}} \qquad \text{avec} \qquad x \in [0,1].$$

Les auteurs justifient ce choix des lois puissance parce qu'il fournit une modélisation des distributions sphériques (voir les travaux [8, 26], déjà décrits dans le chapitre 4) et une généralisation des densités "à valuation" introduites en dimension 2 (voir les travaux de Vallée et Vera [112] décrits dans le chapitre 4).

Dans ce modèle, la hauteur $c_i = -\log_s r_i$ de la pile d'indice i suit une loi exponentielle de la forme,

$$\mathbb{P}[c_i \geq y] = s^{-\frac{y}{\theta_i}} \qquad \text{avec} \qquad y \in [0, +\infty].$$

En utilisant des calculs classiques sur les lois exponentielles, on montre que la moyenne et l'écart type sont égaux et satisfont tous deux

$$\mathbb{E}[c_i] = \sigma(c_i) = \frac{\theta_i}{\log s}.$$

Il n'y a pas de phénomène de concentration et la hauteur de chaque pile s'écarte avec une probabilité non négligeable de la moyenne. Cette remarque est cohérente avec la figure 6.5 qui montre que, même si la plupart des piles sont non réduites, il existe des piles qui sont déjà réduites. La difficulté moyenne de la réduction est fonction de l'exposant des lois puissance et devient plus importante quand les paramètres θ_i deviennent plus grands.

FIGURE 6.1: *cfg* de entrée de type Ajtai

Modèle à loi puissance modifiée. On peut modifier le modèle pour imposer que le *cfg* soit totalement non réduit, autrement dit que tous les c_i vérifient $c_i > 1$. Nous considérons un modèle où la distribution des c_i vérifie

$$\mathbb{P}[c_i \geq y + 1] = s^{-\frac{y}{\theta_i}} \qquad \text{avec} \qquad y \in [0, +\infty] \qquad \text{et donc} \qquad \mathbb{E}[c_i] = 1 + \frac{\theta_i}{\log s}.$$

6.1.2 Etude en moyenne des paramètres de réduction dans le modèle double $[\mathtt{M1}, \mathcal{A}]$.

Nous adaptons maintenant les résultats du chapitre précédent, en les "transformant" en analyse en moyenne.

Configuration de sortie. Puisque le *cfg* est totalement non réduit, et vérifie $c_i > 1$, on peut appliquer les résultats de la section 5.5 du chapitre 5. On rappelle que le modèle d'exécution $\mathtt{M1}(\alpha)$ est associé à un décrément fixé $\alpha > 0$, lui-même relié à un facteur de décroissance ρ fixe de la forme $\rho = 1/s^\alpha$. Dans le modèle d'entrée $\mathcal{A}(\Upsilon, d, g)$, la configuration de sortie $\widehat{\mathbf{c}}$ vérifie

$$\forall i \in [1\,..\,d-1], \quad \widehat{c}_i \geq 1 - \alpha, \qquad \text{ce qui implique} \quad \widehat{r}_i \in \left[\frac{1}{s}, \frac{1}{\rho s}\right],$$

et est donc indépendante des paramètres du modèle d'entrée. Remarquons aussi que ce résultat n'est pas (seulement) un résultat en moyenne, mais que c'est un résultat plus fort, vrai "point

à point". On en déduit que l'énergie finale $\widehat{\mathcal{E}} = \mathcal{E}(\widehat{c})$ est une fonction cubique de la dimension et vérifie

$$\frac{\widehat{\mathcal{E}}}{A(d)} \in [1-\alpha, 1] \quad \text{avec} \quad A(d) = d \cdot \frac{d^2-1}{6}.$$

Nombre d'itérations. Dans le modèle $\mathcal{A}(\Upsilon, d, g)$, la masse moyenne Υ est une fonction au moins linéaire de la dimension, et l'énergie initiale $\mathcal{E} = \mathcal{E}(\mathbf{c})$ moyenne vérifie

$$\mathbb{E}[\mathcal{E}] \sim d^2\, \Upsilon\, I(g), \qquad \text{avec} \quad I(g) = \int_0^1 x(1-x)g(x)dx.$$

Il y a deux cas principaux :

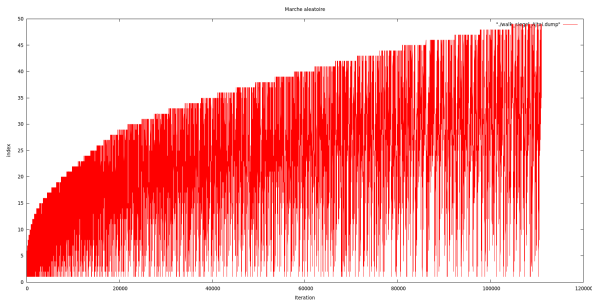
- *Cas où la masse moyenne* Υ *vérifie* $\Upsilon = \Theta(d^a)$ *avec* $a > 1$. Cela correspond par exemple au modèle historique d'Ajtai. Dans ce cas, l'énergie moyenne initiale $\mathbb{E}[\mathcal{E}]$ est d'ordre $\Theta(d^{a+2})$, tandis que l'énergie moyenne finale est d'ordre cubique. Il s'ensuit que le nombre moyen d'itérations dans le modèle $[\mathtt{M1}(\alpha), \mathcal{A}(\Upsilon, d, g)]$ est également d'ordre $\Theta(d^{a+2})$ et vérifie asymptotiquement

  $$\mathbb{E}[K] \sim \frac{1}{2\alpha}\mathbb{E}[\mathcal{E}] \sim \frac{1}{2\alpha} d^2\, \Upsilon\, I(g), \qquad \text{avec} \quad I(g) = \int_0^1 x(1-x)g(x)dx.$$

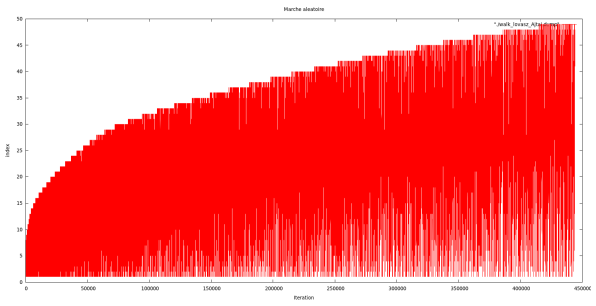
  Ce résultat correspond bien à l'estimation expérimentale du nombre d'itérations obtenu par Nguyen et Stehlé pour l'exécution de l'algorithme LLL [88] sur le modèle initial d'Ajtai.
- *Cas où la masse moyenne* Υ *est d'ordre linéaire en* d *et asymptotique à* $C \cdot d$, *avec* $C > 1$. Alors l'énergie initiale moyenne vérifie quand $d \to \infty$ $\mathbb{E}[\mathcal{E}] \sim C \cdot I(g) \cdot d^3$. Si la relation $6CI(g) > 1$ est vérifiée, alors l'énergie moyenne initiale est strictement supérieure à l'énergie moyenne finale. Le nombre moyen d'itérations est donc cubique et vérifie asymptotiquement quand $d \to \infty$

  $$\frac{d^3}{12\alpha}[6CI(g) - 1] \leq \mathbb{E}[K] \leq \frac{d^3}{12\alpha}\left(6CI(g) - 1 + \alpha\right).$$

  La condition $6CI(g) > 1$ est satisfaite dès que C est suffisamment grand ou g assez proche d'une densité uniforme.



M4 : Algorithme de Siegel



M5 : Algorithme de Lovász

FIGURE 6.2: La marche aléatoire sur une base d'Ajtai $\mathcal{A}(\Upsilon, d, g)$ pour $\Upsilon = 50, d = 50$ et la densité uniforme g

La figure 6.2 montre la marche aléatoire dans les couples de modèles $[\mathtt{M4}, \mathcal{A}]$ ou dans $[\mathtt{M5}, \mathcal{A}]$. L'enveloppe supérieure de la marche décrit donc la courbe $d = f^{-1}(K)$ où f est la fonction qui exprime le nombre d'itérations K en fonction de la dimension d.

6.2 Les modèles “uni-tas”

6.2.1 *cfg* “uni-tas”.

Les *cfg* “uni-tas” sont des *cfg* qui ne possèdent qu’une seule grande pile située à un indice $m = 1 + \lceil \beta(d-2) \rceil$ pour $\beta \in [0,1]$. La hauteur des autres piles est soit beaucoup plus petite, soit nulle. En modifiant le modèle $\mathcal{A}(\Upsilon, d, g)$, et en remplaçant la densité g de classe $\mathcal{C}^1$ par une fonction g partout nulle sauf en $x = \beta$ où elle est égale à 1, on obtient le modèle $\mathcal{U}(\Upsilon, d, \beta)$ qui rassemble des *cfg* de masse moyenne Υ de dimension d dont la seule pile de hauteur non nulle est située à la position $m \approx \beta d$, pour $d \to \infty$, et donc assez proche du modèle précis décrit ici, où $m = 1 + \lceil \beta(d-2) \rceil$. La hauteur moyenne de la pile d’indice m vérifie

$$\mathbb{E}[c_i] = \Upsilon \quad \text{si } i = m = 1 + \lceil \beta(d-2) \rceil \ , \qquad \mathbb{E}[c_i] = 0, \quad \text{sinon}$$

de sorte que la masse totale moyenne est bien Υ. L’énergie moyenne vérifie alors, pour $d \to \infty$,

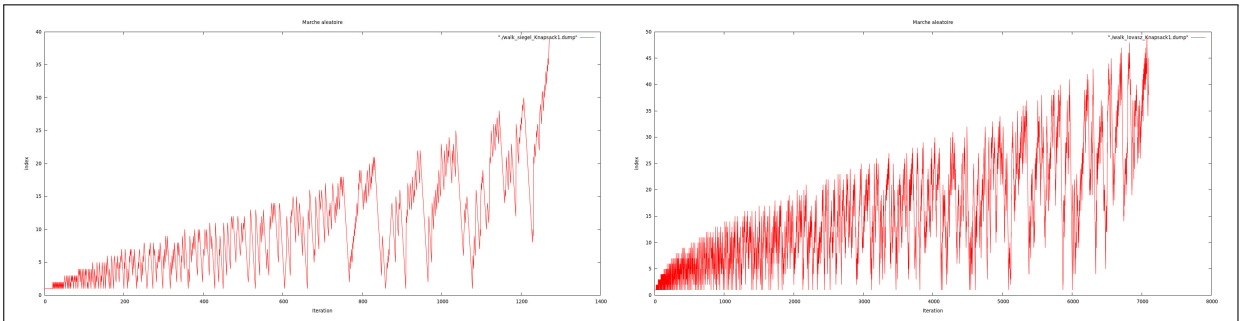
$$\mathbb{E}[\mathcal{E}] = \sum_{i=1}^{d-1} i(d-i)\mathbb{E}[c_i] \sim \begin{cases} d^2\beta(1-\beta)\,\Upsilon & \text{si } \beta \in]0,1[\\ d\,\Upsilon & \text{si } \beta \in \{0,1\} \end{cases}.$$

6.2.2 Nombre d’itérations de la réduction des *cfg* “uni-tas”.

Nous reformulons le théorème que nous avons obtenu dans le chapitre précédent. On suppose que $\Upsilon = \Theta(d^a)$ avec $a > 0$ et que la dimension d tend vers ∞.

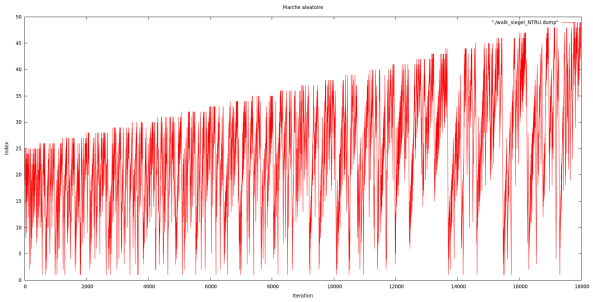
L’unique pile n’est pas sur les bords. Dans ce cas, l’énergie moyenne initiale $\mathbb{E}[\mathcal{E}]$ est d’ordre $\Theta(d^{a+2})$. Pour $a \leq 1$, le nombre d’itérations est $O(d^3)$. Si $a > 1$, l’énergie initiale moyenne est d’ordre strictement supérieur à l’énergie finale moyenne, et le nombre d’itérations est d’ordre $K(\mathbf{c}) = 1/(2\alpha)\mathcal{E}(\mathbf{c}) = \Theta(d^{a+2})$. Dans le cas d’un *cfg* de base, et si $a \leq 1$, on sait que le nombre d’itérations est $\Theta(d^{3a})$ et aussi $\Theta(\Upsilon^3)$.

L’unique pile est sur les bords. Dans ce cas, l’énergie moyenne initiale $\mathbb{E}[\mathcal{E}]$ est d’ordre $\Theta(d^{a+1})$. Pour $a \leq 2$, le nombre d’itérations est $O(d^3)$. Si $a > 2$, l’énergie initiale moyenne est d’ordre strictement supérieur à l’énergie finale moyenne, et le nombre d’itérations est d’ordre $K(\mathbf{c}) = 1/(2\alpha)\mathcal{E}(\mathbf{c}) = \Theta(d^{a+1})$. Dans le cas d’un *cfg* de base, et si $a \leq 2$, on sait que le nombre d’itérations est $\Theta(d^{\frac{3a}{2}})$ et aussi $\Theta(\Upsilon^{3/2})$.

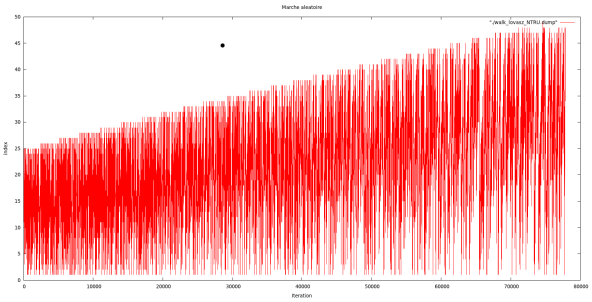


Algorithme de Siegel. Algorithme de Lovász.

FIGURE 6.3: Marche aléatoire de l’algorithme de Siegel et l’algorithme LLL sur un réseau de $\mathcal{U}(\Upsilon, d, \beta)$ pour $\Upsilon = 100, d = 50, \beta = 0$



Algorithme de Siegel



Algorithme LLL

FIGURE 6.4: Marche aléatoire de l'algorithme de Siegel et l'algorithme LLL sur un réseau de $\mathcal{U}(\Upsilon, d, \beta)$ pour $\Upsilon = 50, d = 50, \beta = 1/2$

Les figures 6.3, 6.4 montrent la forme de la marche aléatoire dans les modèles $[\mathtt{M4}, \mathcal{U}]$ et $[\mathtt{M5}, \mathcal{U}]$. Comme précédemment, l'enveloppe supérieure est une courbe qui exprime la dimension d en fonction du nombre d'itérations K.

6.2.3 Structure générale des réseaux uni-tas cryptographiques

Dans la "réalité cryptographique", les réseaux "uni-tas" arrivent dans le cadre suivant : on considère sept éléments :

(*i*) un entier d et un réel $\beta \in [0, 1]$; l'entier $i \in [1\,..\,d-1]$ est défini par la relation $i := 1 + \lfloor \beta(d-2) \rfloor$,

(*ii*) deux vecteurs $\mathbf{x} = (x_1, \ldots, x_i)$ et $\mathbf{y} = (y_{i+1}, \ldots, y_d)$ à composantes strictement positives,

(*iii*) deux entiers X et Y strictement positifs, avec $X \gg Y$,

(*iv*) une matrice $\mathcal{T} = (t_{j,k})_{j \in [1..d-i],\ k \in [1..i]}$.

Le réseau $\mathcal{L}(d, \beta, \mathbf{x}, \mathbf{y}, X, Y, \mathcal{T})$ est le réseau engendré par les lignes de la matrice triangulaire inférieure $\mathcal{B}$ construite par blocs à l'aide de la matrice $\mathcal{T}$ et des matrices diagonales $\mathcal{D}(X\mathbf{x})$, $\mathcal{D}(Y\mathbf{y})$ dont les diagonales sont respectivement égales à $X\mathbf{x}$ et à $Y\mathbf{y}$

$$\mathcal{B} := \begin{pmatrix} \mathcal{D}(X\mathbf{x}) & 0 \\ X \cdot \mathcal{T} & \mathcal{D}(Y\mathbf{y}) \end{pmatrix} = \left(\begin{array}{ccc|ccc} X\,x_1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & X\,x_i & 0 & \cdots & 0 \\ \hline X\,t_{1,1} & \cdots & X\,t_{1,i} & Y\,y_{i+1} & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ X\,t_{d-i,1} & \cdots & X\,t_{d-i,i} & 0 & \cdots & Y\,y_d \end{array}\right).$$

Les longueurs de Siegel se lisent sur la diagonale,

$$\ell_j = Xx_j \quad \text{pour } j \in [1\,..\,i], \qquad \ell_j = Yy_j, \quad \text{pour } j \in [i+1\,..\,d].$$

et les rapports de Siegel vérifient

$$r_j = \frac{x_{j+1}}{x_j} \quad \text{pour } j \in [1\,..\,i-1], \quad r_j = \frac{y_{j+1}}{y_j} \quad \text{pour } j \in [i+1\,..\,d-1], \quad r_i = \frac{Y}{X}\frac{y_{i+1}}{x_i}.$$

Les piles c_j du *cfg* associé vérifient donc :

$$c_j = \log_s x_j - \log_s x_{j+1} \quad \text{pour } j \in [1\,..\,i-1], \quad c_j = \log_s y_j - \log_s y_{j+1} \quad \text{pour } j \in [i+1\,..\,d-1]$$

$$c_i = \log_s X - \log_s Y + \log_s x_i - \log_s y_{i+1}.$$

Dans la réalité cryptographique, les variables x_i et y_j ont souvent la même distribution, et donc

$$\mathbb{E}[c_j] = 0 \quad \text{si } j \neq i, \qquad \mathbb{E}[c_i] \sim \log_s X,$$

et, dans ce cas, le réseau $\mathcal{L}(d, \beta, \mathbf{x}, \mathbf{y}, X, Y, \mathcal{T})$ donne lieu à un *cfg* de l'ensemble $\mathcal{U}(\log_s X, d, \beta)$, dont l'énergie moyenne vérifie

$$\mathbb{E}[\mathcal{E}] = \sum_{i=1}^{d-1} i(d-i)\mathbb{E}[c_i] \sim \begin{cases} d^2\beta(1-\beta)\log_s X & \text{si } \beta \in]0,1[\\ d \log_s X & \text{si } \beta \in \{0,1\} \end{cases}.$$

6.2.4 Réseaux cryptographiques donnant lieu à des *cfg* uni-tas

En cryptographie, les réseaux utilisés dans l'attaque efficace des protocoles (sac-à-dos, NTRU) ou bien dans l'aide à la résolution de problèmes algorithmiques difficiles (comme la factorisation) donnent des exemples naturels de réseaux uni-tas.

Réseaux Sac-à-dos. Le problème du sac-à-dos est présenté en détail dans le chapitre 3 ainsi que le protocole de Merkle et Hellman fondé sur ce problème.
Problème : étant donnés un entier S et une famille d'entiers $(a_1, \ldots, a_{d-1})$ positifs pour lesquels il existe une suite $(x_i) \in \{0,1\}^{d-1}$ vérifiant $S = \sum_{i=1}^{d-1} x_i a_i$, trouver la suite (x_i).

Comme le chapitre 3 le rappelle, la solution de ce problème est reliée à un vecteur court du réseau engendré par les lignes de la matrice suivante

$$\mathcal{B} := \left(\begin{array}{c|ccc} C \cdot S & 0 & \cdots & 0 \\ \hline C \cdot a_1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C \cdot a_{d-1} & 0 & \cdots & 1 \end{array}\right).$$

C'est un réseau du modèle $\mathcal{U}(\Upsilon, d, 0)$ avec $\Upsilon := \log_s S + \log_s C$. Dans la pratique cryptographique, on s'intéresse à des sacs à dos de densité δ, où la densité δ, liée à l'information du cryptosystème, est définie par

$$\delta = \frac{d}{\max_i \log a_i} \sim \frac{d}{\log S}.$$

Donc, dans les applications, la masse du *cfg* $\log_s S$ est une fonction linéaire de la dimension. Mais le succès de la méthode impose souvent de choisir C avec une longueur binaire quadratique en la dimension d, et dans ce cas Υ est d'ordre $\Theta(d^2)$.

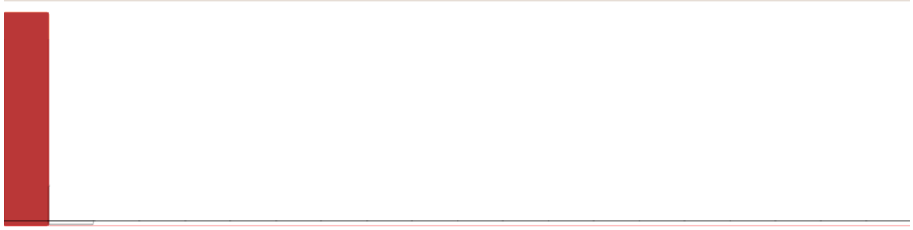


FIGURE 6.5: *cfg* de l'entrée de type Knapsack

Réseaux de Schnorr. Comme il est rappelé dans le chapitre 2, Schnorr a décrit dans [100] une méthode de factorisation de l'entier N qui s'appuie sur la recherche de couples (u, v) formés d'entiers u et v pour lesquels u et $u - vN$ sont tous les deux B-lisses (pour un grand entier B). Schnorr a montré un lien fort entre les couples (u, v) de cette forme et les vecteurs courts du réseau $\mathcal{L}(\mathcal{B})$ de dimension d engendré par les vecteurs de la matrice $\mathcal{B}$ suivante

$$\mathcal{B} = \left(\begin{array}{c|cccc} N^c \log N & 0 & 0 & \cdots & 0 \\ \hline N^c \log p_1 & \log p_1 & 0 & \cdots & 0 \\ N^c \log p_2 & 0 & \log p_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ N^c \log p_{d-1} & 0 & 0 & \cdots & \log p_{d-1} \end{array}\right).$$

Dans cette matrice, $p_1, \ldots, p_{d-1}$ sont les $d-1$ premiers nombres premiers, de sorte que la borne B est égale à p_{d-1}. Pour que le réseau soit adapté à cette recherche de couples (u, v), Schnorr propose le choix $p_{d-1} = (\log N)^a$, avec une constante a proche de 2. Comme le d^{e} nombre premier est asymptotique à $d \log d$, la dimension du réseau d est d'ordre approximatif $(\log N)^a$, tandis que la masse de la première pile du *cfg* associé est d'ordre approximatif $c \log N$, soit d'ordre $d^{1/a}$ ou encore d'ordre proche de $\sqrt{d}$.
Remarquons que les autres piles du *cfg* ont une hauteur très faiblement négative, puisque la pile d'indice i a une hauteur égale à

$$c_i := \log_s \left(\frac{p_i}{p_{i+1}}\right) \sim \frac{1}{i},$$

de sorte que la masse totale négative est de l'ordre de $\log d$, complètement négligeable devant la hauteur de la première pile.

Réseaux de type NTRU. Le protocole NTRU a été introduit au chapitre 3. Les réseaux sous-jacents sont engendrés par des matrices de la forme

$$\mathcal{B} = \left(\begin{array}{c|c} X \cdot I_i & 0 \\ \hline \mathcal{T} & I_{d-i} \end{array}\right),$$

avec X un entier qui est souvent une puissance de 2 (par exemple 128) et $\mathcal{T}$ une matrice dont les coefficients sont des entiers de l'intervalle $[-X/2, +X/2]$. Une version du cryptosystème NTRU (voir [49]) admet comme paramètres $d = 2m$, $i = m$ et $\mathcal{T} = \mathcal{T}(\mathbf{h})$ une matrice circulante relative à un vecteur $\mathbf{h} = (h_1, h_2, \ldots, h_m)$.

C'est donc un réseau de la famille $\mathcal{L}(d, 1/2, \mathbf{1}, \mathbf{1}, X, 1, \mathcal{T})$. Si X est un grand entier, seule la pile $c_{d/2}$ a une hauteur importante, et ce réseau donne lieu à un *cfg* de la famille $\mathcal{U}(\log_s X, d, 1/2)$. La figure 6.6 donne un exemple d'un tel *cfg*.

Dans les attaques contre ce cryptosystème, le paramètre q est une fonction polynomiale de la dimension d, si bien que la masse totale moyenne Υ est logarithmique en la dimension d.

Réseaux de type SIS (dits q-aires). Les réseaux qui interviennent dans le problème SIS sont engendrés également par des matrices de la forme $\mathcal{B}$. Ils font partie de la famille $\mathcal{L}(d, \beta, \mathbf{1}, \mathbf{1}, X, 1, \mathcal{T})$. L'entier X est un entier polynomial en d et la position de pile est $i = 1 + \lfloor \beta(d-2) \rfloor$ avec $\beta \sim 1 - 1/\log d$. Donc ce réseau donne lieu à un *cfg* de la famille $\mathcal{U}(\log_s X, d, \beta)$, avec $\beta = 1 - (1/\log d)$. Comme la masse Υ est logarithmique en la dimension d, on a

$$\mathbb{E}(\mathcal{E}) \sim d^2 \beta(1-\beta)\,\Upsilon = \Theta\left(d^2\right).$$

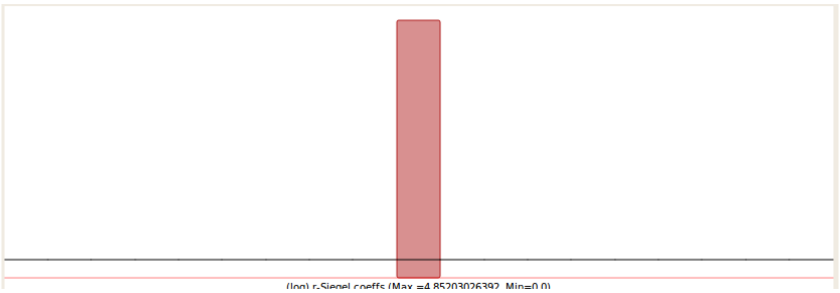


FIGURE 6.6: *cfg* de l’entrée de type NTRU

6.3 Les modèles de Coppersmith : “tas à trous”.

Jusqu’à présent, tous les modèles rassemblent des *cfg* dont toutes les piles ont des hauteurs positives ou nulles. Mais, la méthode de Coppersmith décrite au chapitre 3 donne lieu à des *cfg* dont les piles peuvent avoir des hauteurs de signe variable. Nous réservons le nom de pile à une pile de hauteur positive ou nulle, tandis qu’une pile de hauteur strictement négative sera appelée un “trou”. Lors de la réduction d’un *cfg*, le sable contenu dans une pile peut se déverser dans un trou adjacent, et si le trou est assez profond, il pourra absorber le sable apporté sans déborder, et rester donc réduit.

6.3.1 Réseaux de Coppermith à une variable

Au chapitre 3, nous avons introduit la méthode de Coppersmith avec une variable. Quand n est le degré du polynôme, et N le module, on considère un entier $X \sim N^{1/n}$ et un entier m, et on travaille avec un réseau $\mathcal{L}(U)$, donné par une matrice triangulaire inférieure U. Cette matrice est composée de $(m+1)$ blocs triangulaires de longueur fixe n, et s’écrit sous la forme (ici en prenant $m = 2$ et $n = 3$) ,

$$U = \left(\begin{array}{ccc|ccc|cccc} N^2 & & & & & & & & \\ - & N^2X & & & & & & & \\ - & - & N^2X^2 & & & & & & \\ \hline - & - & - & NX^3 & & & & & \\ - & - & - & - & NX^4 & & & & \\ - & - & - & - & - & NX^5 & & & \\ \hline - & - & - & - & - & - & X^6 & & \\ - & - & - & - & - & - & - & X^7 & \\ - & - & - & - & - & - & - & - & X^8 \end{array}\right).$$

La diagonale $\mathbf{l}^{(k)}$ du bloc d’indice k est

$$\mathbf{l}^{(k)} = N^{m-k}X^{kn}\left(1, X, \ldots, X^i, \ldots, X^{n-1}\right) = N^m\left(\frac{X^n}{N}\right)^k\left(1, X, \ldots, X^i, \ldots, X^{n-1}\right)$$

Les rapports de Siegel r_i de chaque bloc sont constants et égaux à X. Chaque bloc donne lieu à un *cfg* de longueur $n-1$ dont toutes les piles ont la même hauteur égale à $-\log_s X$ qui est négative. C’est donc un “grand trou” de longueur $n-1$ (en vert sur la figure 6.7). Il existe une pile positive (en rouge sur la figure) à la frontière de chaque bloc, entre le dernier élément du bloc d’indice k et le premier élément du bloc d’indice $k+1$. Ces piles sont toutes de même hauteur positive, égale à $\log_s(N/X) = \log_s N - \log_s X$.

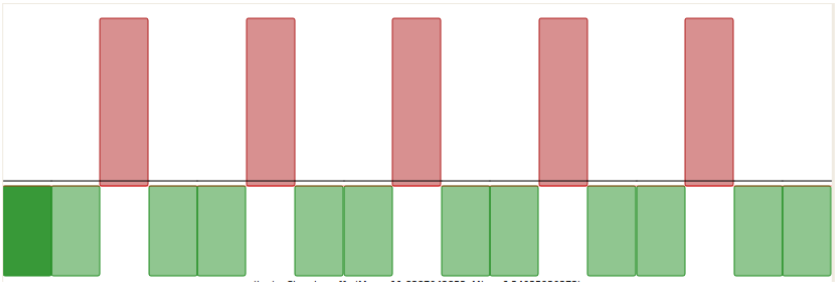


FIGURE 6.7: *cfg* : type Coppersmith avec une variable pour $n = 3$.

6.3.2 Réseaux de Coppermith à deux variables

La méthode de Coppersmith introduite dans le Chapitre 3 et utilisée pour le cassage de RSA utilise un réseau donné par une base $\mathcal{B}$, définie à partir de deux matrices triangulaires $\mathcal{C}_x$ et $\mathcal{C}_y$, et une autre matrice triangulaire $\mathcal{T}$ sous la forme

$$\mathcal{B} := \begin{pmatrix} \mathcal{C}_x & 0 \\ \mathcal{T} & \mathcal{C}_y \end{pmatrix}$$

La matrice $\mathcal{T}$ ne joue donc plus de rôle par la suite. Chacune de ces matrices $\mathcal{C}_x, \mathcal{C}_y$ est formée de $m+1$ blocs, chacun d'eux étant lui-même triangulaire, et dépend de trois entiers E, X, Y, qui vérifient $Y = E^{1/2}, X = E^{\delta}$ avec $\delta \in [1/4, 1/2]$.

Matrice $\mathcal{C}_x$. Elle contient $m+1$ blocs, indexés de $k = 0$ à m. Le bloc d'indice k est de longueur $k+1$ et sa diagonale $\mathbf{l}_x^{(k)}$ vérifie

$$\mathbf{l}_x^{(k)} = E^m X^k \left(1, Z, \ldots, Z^i, \ldots, Z^k\right), \qquad \text{avec} \quad Z := \frac{Y}{E} = E^{-1/2}, \quad X = E^{\delta}.$$

La matrice $\mathcal{C}_x$ est alors de la forme

$$C_x = \left(\begin{array}{ccccccccc}
E^m & & & & & & & & \\
\hline
- & E^m X & & & & & & & \\
- & - & E^{m-1}XY & & & & & & \\
\hline
- & - & - & \ddots & & & & & \\
\hline
- & - & - & - & E^m X^m & & & & \\
- & - & - & - & - & E^{m-1}X^m Y & & & \\
- & - & - & - & - & - & E^{m-2}X^m Y^2 & & \\
- & - & - & - & - & - & - & \ddots & \\
- & - & - & - & - & - & - & - & X^m Y^m
\end{array}\right).$$

Le bloc d'indice k conduit à un *cfg* de longueur k dont toutes les piles sont de même hauteur positive $L/2$ avec $L := \log_s E$. À la frontière des deux blocs d'indice k et $k+1$, il existe une unique pile de hauteur négative (donc un trou), dont la profondeur est

$$-L\left(\delta + \frac{k}{2}\right).$$

Matrice $\mathcal{C}_y$. Elle est aussi formée de $m+1$ blocs, chacun d'eux étant de longueur constante t et diagonal. La diagonale du bloc d'indice k, notée $\mathbf{l}_y^{(k)}$ est

$$\mathbf{l}_y^{(k)} = E^{m-k}X^kY^{k+1}\left(1, Y, \ldots, Y^i, \ldots, Y^{t-1}\right) = E^mY\left(\frac{XY}{E}\right)^k\left(1, Y, \ldots, Y^i, \ldots, Y^{t-1}\right).$$

La matrice $\mathcal{C}_y$ est donc de la forme

$$C_y = \left(\begin{array}{ccccccccccc}
E^mY & & & & & & & & & & \\
- & E^mY^2 & & & & & & & & & - \\
- & - & \ddots & E^mY^t & & & & & & & \\
\hline
- & - & - & - & E^{m-1}XY^2 & & & & & & \\
- & - & - & - & - & E^{m-1}XY^3 & & & & & \\
- & - & - & - & - & - & \ddots & E^{m-1}XY^{t+1} & & & \\
\hline
 & & & & & & & \ddots & & & \\
\hline
- & - & - & - & - & - & - & - & X^mY^{m+1} & & \\
- & - & - & - & - & - & - & - & \ddots & X^mY^{m+t} &
\end{array}\right).$$

Chaque bloc donne lieu à un *cfg* de longueur $t-1$ dont toutes les piles sont de même hauteur négative, égale à $-L/2$ (toujours avec $L = \log_s E$). Chaque bloc donne donc lieu à un trou. À la frontière entre le bloc d'indice k et celui d'indice $k+1$, il existe une unique pile de hauteur positive

$$c_{k+1,k} = L\left(\frac{t+1}{2} - \delta\right).$$

Le *cfg* complet associé à un réseau de type Coppersmith est donné par la concaténation des x-shifts et y-shifts (voir figure 6.10).

FIGURE 6.8: *cfg* associé aux x-shifts

6.3.3 Modèle général de Coppersmith.

Il y a donc trois cas : le *cfg* associé à la méthode de Coppersmith à 1 variable, désigné par `Cop1`, et les deux *cfg* dans le cas de 2 variables, le *cfg* des x-shifts désigné par `Cop2x`, et celui des y-shifts désigné par `Cop2y`. Dans les trois cas, le bloc d'indice k a pour diagonale

$$\mathbf{l}^{(k)} = CA^k\left(1, B, \ldots, B^i, \ldots, B^{n(k)-1}\right).$$

Mais il y a deux cas bien distincts :

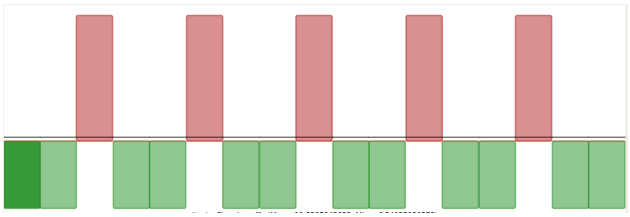


FIGURE 6.9: *cfg* associés aux y-shifts

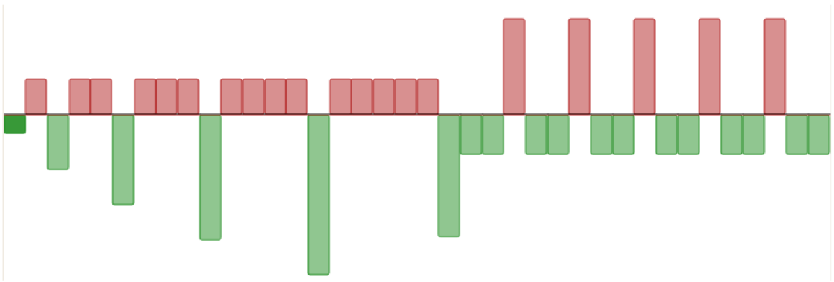


FIGURE 6.10: *cfg* : type Coppersmith deux variables

– Pour `Cop2x`, on a : $B < 1$ et la longueur du bloc d'indice k est $n(k) = k$.
– Pour `Cop1` et `Cop2y`, on a $B > 1$ et la longueur $n(k)$ du bloc d'indice k est fixe, égale à n.

Dans tous les cas, si on pose $b := -\log_s B, a := -\log_s A$, le *cfg* correspondant au k^{e} bloc est de longueur $n(k) - 1$, et vérifie

$$\mathbf{c}^{(k)} = (b, b, \ldots, b)\,.$$

Il est de masse $(n(k) - 1)\, b$, qui est positive pour `Cop2x`, et négative dans les autres cas.

Dans tous les cas, la pile de transition entre les deux blocs est unique, de hauteur $a-(n(k) - 1)\, b$, mais il y a deux cas distincts :

– pour `Cop2x`, elle est négative
– pour `Cop1` et `Cop2y`, elle est positive.

Pour `Cop1`, on a $A < 1$ mais proche de 1, de telle sorte que a est positif, et donc la pile de transition a une masse positive. Mais la somme des deux masses est égale à a et est donc positive, de sorte que le trou est trop petit pour absorber le sable qui tombe de la pile positive.

Pour `Cop2y`, c'est la même chose, car XY vérifie $XY < E$.

Un essai de modélisation des réseaux de Coppersmith. Il y a de fait deux sortes de *cfg* associés aux réseaux de Coppersmith :

cfg de Coppersmith de type positif. C'est un *cfg* qui généralise le *cfg* de type `Cop2x`. C'est un *cfg* où les blocs sont positifs, de longueur variable (le k^{e} bloc est de longueur k), et la pile entre les blocs est négative (c'est un trou). La configuration de base d'un tel réseau est formée de trois blocs

– à gauche, un bloc positif de longueur k, formé de piles toutes de même hauteur c_g^+,
– à droite, un bloc positif de longueur $k + 1$, formé de piles toutes de même hauteur c_d^+,
– au milieu, un trou de longueur 1, formé d'une pile de profondeur c^-.

cfg de Coppersmith de type négatif. C'est un *cfg* qui généralise les *cfg* de type `Cop1` et `Cop2y`. C'est un *cfg* où les blocs sont négatifs (de longs trous, toujours de même longueur $n-1$) et la pile entre les blocs est positive. La configuration de base est formée de trois blocs
- à gauche, une pile positive de hauteur c_g^+,
- à droite, une pile positive de hauteur c_d^+,
- au milieu, un trou de longueur $n-1$ formé de piles de même hauteur négative c^-.

cfg de Coppersmith quelconque. La configuration de base est formée de trois blocs
- à gauche, un bloc positif de longueur n_g^+, formé de piles, toutes de même hauteur c_g^+,
- à droite, un bloc positif de longueur n_d^+, formé de piles toutes de même hauteur c_d^+,
- au milieu, un trou de longueur n^- formé de piles toutes de même hauteur négative c^-.

6.3.4 Indépendance des blocs des *cfg* de Coppersmith

Ainsi, les *cfg* associés à la méthode de Coppermith sont à la fois composés de trous et de piles. Sur ces entrées, l'algorithme de réduction des *cfg* vide les piles dans les trous. Si les trous sont suffisamment profonds, ils absorberont le sable fourni par les piles positives, et la configuration de base pourra se réduire indépendamment de ses voisines. Autrement dit, ces configurations seront indépendantes et pourront être réduites en parallèle.

Au chapitre 5, nous avons introduit la condition d'indépendance de blocs (Théorème 5.32). Nous allons ici appliquer cette condition pour les blocs x-shifts et pour les blocs y-shifts dans le modèle `M1`(α). C'est plus facile ici, car les blocs (positifs ou négatifs) sont “plats”, formés de piles de même hauteur. Les conditions du chapitre précédent s'écrivent comme suit :
Condition suffisante d'indépendance :

$$n_g^+ c_g^+ + n_d^+ c_d^+ - 2n^- c^- \leq (2n^- + n_g^+ + n_d^+) - 2\alpha,$$

Condition suffisante de dépendance :

$$n_g^+ c_g^+ + n_d^+ c_d^+ - 2n^- c^- \geq (2n^- + n_g^+ + n_d^+).$$

Indépendance des configurations de Coppersmith ; cas des réseaux de type positif. On a, dans le cas des réseaux `Cop2x` ;

$$n_g^+ c_g^+ + n_d^+ c_d^+ = k\frac{L}{2} + (k+1)\frac{L}{2}, \quad n^- c^- = L\left(\delta + \frac{k}{2}\right).$$

La condition s'écrit donc

$$L\left(\frac{1}{2} - 2\delta\right) \leq (2k+3) - 2\alpha$$

qui est toujours vérifiée dès que $\delta > 1/4$ pour $L \to \infty$ et k fixe, ce qui est le cas dans le cadre de la cryptanalyse. Pour les réseaux utilisés dans le cryptanalyse de RSA la valeur de $\delta \in [1/4, 1/2]$ (voir le chapitre 3). Dans ce cas, il est possible de paralléliser la réduction des blocs.

Indépendance des configurations ; cas des réseaux de type négatif. On étudie successivement les deux cas des réseaux `Cop1` puis `Cop2y`.

Dans le cas des réseaux considérés de type `Cop1`, on a

$$n_g^+(c_g^+ - 1) + n_d^+(c_d^+ - 1) = 2L(1 - \frac{1}{t}), \qquad n^-(c^- + 1) = (t-1)\left(\frac{L}{t} + 1\right).$$

La condition s'écrit $(t-1) \leq 0$, et n'est jamais vérifiée.

Dans le cas des réseaux considérés de type `Cop2y`, on a

$$n_g^+(c_g^+ - 1) + n_d^+(c_d^+ - 1) = 2L\left(\frac{t+1}{2} - \delta\right), \qquad n^-(c^- + 1) = (t-1)\left(\frac{L}{2} + 1\right).$$

La condition s'écrit $L(1-\delta) \leq (t-1)$, et n'est jamais vérifiée puisque $L \to \infty$ et t est fixe.

6.3.5 Nombre d'itérations dans le cas des réseaux de Coppersmith vérifiant la condition d'indépendance

Nous considérons le nombre d'itérations effectuées par la réduction sur la partie x-shift. En appliquant le théorème 5.29, le nombre d'itérations dans le modèle `M1` (α) pour réduire le bloc d'indice k est :

$$K_k = \frac{k^3}{12\alpha}\left(\frac{L}{2} - 1\right). \tag{6.1}$$

Le théorème d'indépendance, dans la version 5.32 du chapitre 5, ou dans la version de ce chapitre, montre qu'on peut adopter une stratégie parallèle pour réduire le *cfg* correspondant à la partie des x-shifts. Le résultat suivant compare les deux stratégies, séquentielle ou parallèlle :

Théorème 6.1. *Considérons le réseau $\mathcal{L}$ engendré par la base $\mathcal{C}_x$, correspondant à la partie des x-shifts. Alors, dans le modèle du cfg `M1`(α), les blocs du réseau sont toujours indépendants, et la réduction peut être effectuée en parallèle sur chaque bloc. Le nombre d'itérations $K_{\langle p\rangle}$ effectuées dans une stratégie parallèle est :*

$$K_{\langle p\rangle} = \frac{m^3}{12\alpha}\left(\frac{L}{2} - 1\right),$$

tandis que le nombre d'itérations $K_{\langle s\rangle}$ effectuées dans une stratégie séquentielle est :

$$K_{\langle s\rangle} = \sum_{i=1}^{m} K_i \approx \frac{m^4}{48\alpha}\left(\frac{L}{2} - 1\right).$$

6.3.6 Expérimentations dans le modèle $[\mathtt{M5}, \mathcal{C}]$.

Bien sûr, l'exécution de l'algorithme LLL ne correspond pas exactement au modèle du *cfg*. On observe les faits expérimentaux suivants dans le modèle `M5` :

Sur le bloc des x-shifts. On peut comparer le résultat du théorème 6.1 avec l'exécution de LLL (voir Figure 6.11 gauche). L'algorithme LLL effectue un grand nombre d'itérations dans chaque bloc de la partie des x–shifts, car les blocs sont complètement non réduits. Même si les blocs ne sont pas complètement indépendants dans le modèle `M5`, on voit qu'ils sont presque indépendants. On remarque en effet que la concaténation des blocs réduits donne lieu à une base presque réduite, car le nombre supplémentaire d'étapes nécessaire pour obtenir une base réduite reste très petit (voir Figure 6.11 droite).

Sur l'ensemble du cfg. Sur la partie qui correspond aux x-shifts, on observe clairement sur la Figure 6.12 que la réduction successive des blocs s'opère avec un faible travail supplémentaire pour corriger "les débordements". Par contre, la marche aléatoire sur la deuxième partie de l'exécution n'est plus du tout de même nature, et présente de nombreux retours en arrière. C'est

la dépendance des blocs sur la partie des y-shifts qui explique un tel comportement de cette marche aléatoire.

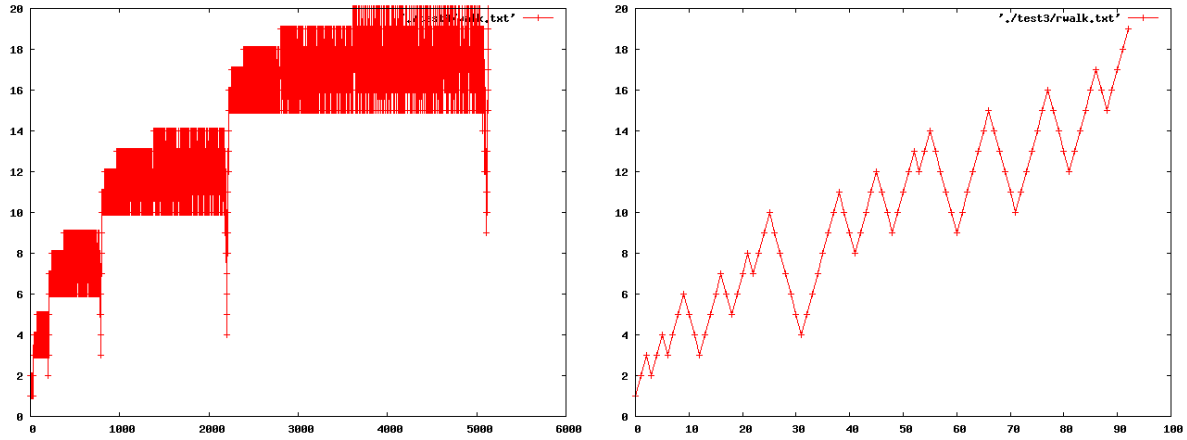


FIGURE 6.11: La marche aléatoire de l'algorithme LLL pour un réseau Coppersmith (x-shift) du dimention $d = 21$ ($m = 5$) à gauche. La marche aléatoire de LLL sur une base obtenue après réduction de chaque bloc en avance.

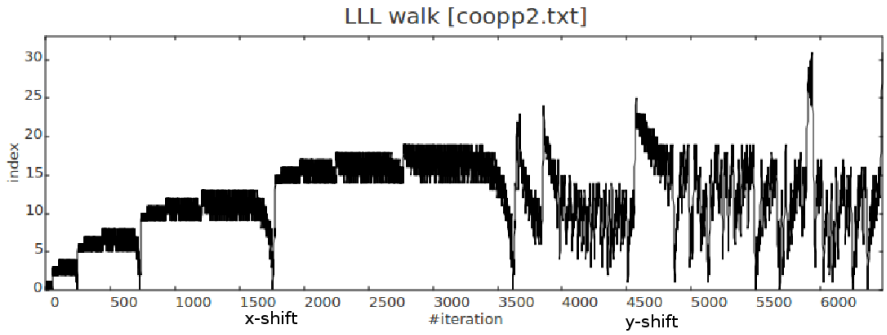


FIGURE 6.12: Marche aléatoire de l'indice sur le réseau de Coppersmith.

Chapitre 7

Étude du modèle M2.

Sommaire

Le chapitre 4 introduit plusieurs modèles simplifiés de l'exécution de l'algorithme LLL. Aux chapitres 5 et 6, l'étude du modèle M1 basé sur les tas de sable a été réalisée en fonction des modèles d'entrées. Dans ce chapitre, nous étudions le modèle d'exécution M2 (voir chapitre 4, section 4.7.2). Ce modèle est plus réaliste que le modèle "chip firing game" puisqu'on suppose ici que la hauteur de sable retirée d'une pile à chaque étape n'est plus constante mais dépend de la hauteur de la pile. Dans l'algorithme LLL, la hauteur de sable retirée dépend aussi du carré de la partie fractionnaire centrée du coefficient sous diagonal courant de la matrice de passage (désignée par μ dans le chapitre 4), mais le modèle M2 considère toujours cette quantité comme constante ; c'est l'hypothèse principale de ce modèle.

Le système dynamique obtenu n'est plus un tas de sable. C'est un système dynamique général, à trou, qui permet de modéliser l'algorithme LLL(t), même pour $t = 1$, alors que l'analyse de l'algorithme LLL(t) n'est pas bien connue pour $t = 1$. Nous analysons ce modèle très finement

en petite dimension pour tout couple (t, μ) avec $t \geq 1$ et $\mu \leq 1/4$. Évidemment, pour $d = 2$, nous ne trouvons rien de nouveau, puisque tout est connu (ou presque) sur l'algorithme de Gauss. Mais, justement, ce modèle $\texttt{M2}(\mu, t)$ "colle" très bien à la réalité, car nous prouvons les mêmes phénomènes qualitatifs que ceux de l'algorithme de Gauss – notamment une distribution géométrique pour le nombre d'itérations. Nous considérons alors le cas $d = 3$, le premier cas où l'algorithme LLL(t) n'est pas bien connu, et nous analysons le modèle M2 pour tout couple (t, μ) avec $t \geq 1$ et $\mu \leq 1/4$, et pour toute densité "à valuation" : nous montrons que le nombre d'itérations est asymptotiquement géométrique, sauf cas exceptionnels où la distribution est "trop symétrique". De plus, la raison de la loi géométrique ne dépend que de μ et non de t. Nous montrons aussi que le nombre d'itérations est asymptotiquement logarithmique en une grandeur directement reliée au potentiel du réseau correspondant. Nous considérons aussi le cas d'une dimension générale, mais, là, nous échouons dans notre étude du cas $t = 1$. Nous montrons cependant que pour $t > 1$ et une entrée $\mathbf{x}$, le nombre d'itérations de l'algorithme LLL(t) dans le modèle $\texttt{M2}(\mu, t)$ reste asymptotiquement logarithmique en une quantité $P(\mathbf{x})$ et ne dépend que de μ et non de t. Du fait d'un calcul intégral un peu complexe (avec beaucoup de paramètres), nous ne sommes pas parvenus [1] à démontrer que le nombre d'itérations suit une loi asymptotiquement géométrique dont la raison ne dépend que de μ et non de t même s'il est très probable qu'un résultat identique existe.

Pour $t = 1$, nous énonçons une conjecture "naturelle" qui permettrait de relier l'analyse du modèle M2 pour $t = 1$ à celle du modèle M2 pour $t > 1$.

Pour "passer" du cas $t > 1$ au cas $t = 1$, nous utilisons une approche similaire à celle usuellement utilisée pour l'algorithme de Gauss, en évaluant le nombre d'itérations qui permet de passer d'une base t-réduite à une base 1-réduite. C'est aussi l'approche que Lenstra [67] et Akhavi [7] ont utilisée dans l'analyse de LLL(1).

La section 7.1 décrit le système dynamique général du modèle M2 avec un accent mis sur les dimensions 1 et 2 (respectivement LLL en dimensions 2 et 3). Nous commençons par l'analyse du système dynamique de la dimension 1 dans la section 7.2. Nous résumons dans la section 7.3 les principales notions classiques des systèmes dynamiques dans $\mathbb{R}^2$. Puis, la section 7.4 est consacrée à l'analyse du système dynamique en dimension 2 (correspondant à la dimension 3 des réseaux). Nous abordons dans la section 7.5 le cas d'une dimension quelconque.

7.1 Système dynamique associé au modèle M2

Dans le modèle simplifié M2, les longueurs des orthogonalisés $\ell_i = \|b_i^*\|$ ont un rôle fondamental alors que la partie fractionnaire centrée du coefficient sous diagonal courant est supposé constant. Nous commençons par le cas le plus simple, celui de l'algorithme de Gauss (LLL en dimension 2) qui se ramène à un système dynamique en dimension 1.

7.1.1 Système dynamique lié à l'algorithme de Gauss

L'algorithme de Gauss commence par remplacer le coefficient $m_{2,1}$ par sa partie fractionnaire centrée, de sorte qu'il vérifie $\|m_{2,1}\| \leq 1/2$. Puis si $\|b_2\| < \|b_1\|$, les deux vecteurs sont échangés et l'échange entraîne le calcul d'une nouvelle base orthogonale de Gram-Schmidt. La nouvelle base vérifie les conditions suivantes :

$$\check{\ell_1}^2 := \ell_2^2 + m_{2,1}^2 \ell_1^2,$$

1. Maple n'a pas réussi non plus.

Entrées : $x \in \mathbb{R}$
Résultat : $\widehat{x} \geq 1-\mu$

tant que $x < 1-\mu$ **faire**
  $x := \dfrac{x}{(x+\mu)^2}$
fintq

FIGURE 7.1: Modèle M2(μ) pour l'algorithme de Gauss.

et l'invariance du déterminant $\check{\ell}_1\check{\ell}_2 = \ell_1\ell_2$ donne la relation

$$\check{\ell}_2^2 = \ell_2^2 \cdot \frac{\ell_1^2}{\ell_2^2 + m_{2,1}^2\ell_1^2}.$$

En posant $x := \ell_2^2/\ell_1^2$, $\check{x} := \check{\ell}_2^2\check{\ell}_1^2$ et $\mu := m_{2,1}^2$ la transformation effectuée par l'algorithme de Gauss s'écrit sous la forme $\check{x} = f_\mu(x)$ avec

$$f_\mu(x) = \frac{x}{(x+\mu)^2} \quad \text{si} \quad x < 1-\mu, \quad f_\mu(x) = x \quad \text{si} \quad x \geq 1-\mu. \tag{7.1}$$

Le modèle simplifié M2= M2(μ) de l'algorithme de Gauss suppose que le coefficient sous-diagonal utilisé reste constant et égal à μ. L'algorithme ainsi simplifié ne dépend que du paramètre μ et est décrit par la figure 7.1.

La fonction f_μ est positive sur $\mathbb{R}^+$, et atteint son maximum en $x = \mu$ où elle est égale à $f_\mu(\mu) = 1/(4\mu)$. Si on désigne par I_μ l'intervalle $\left[0, \frac{1}{4\mu}\right]$, le couple (I_μ, f_μ) est un système dynamique de l'intervalle. C'est un système à trou puisque pour $x > 1-\mu$, $f_\mu(x) = x$.

7.1.2 Système dynamique en dimension supérieure

En dimension d, l'algorithme LLL(t) met en jeu $d-1$ rapports de Siegel, et contrairement aux autres chapitres, nous travaillons ici avec leurs carrés $x_i := \ell_{i+1}^2/\ell_i^2$ pour $i \in [1\,..\,d-1]$, et le vecteur $\mathbf{x} = (x_1, x_2, \ldots, x_i, \ldots, x_{d-1}) \in \mathbb{R}^{d-1}$ formé de ces rapports. Considérons une boite non réduite au sens de t-Lovász et effectuons l'échange entre les deux vecteurs (b_i, b_{i+1}). Si on désigne par $\mu := \lfloor m_{i+1,i} \rfloor^2$, cet échange se traduit par l'application d'une fonction $T_{i,\mu} : \mathbb{R}^{d-1} \to \mathbb{R}^{d-1}$, et le nouveau vecteur noté $\check{\mathbf{x}}$ s'écrit $\check{\mathbf{x}} = T_{i,\mu}(\mathbf{x})$ avec

$$\check{x}_{i-1} = x_{i-1}(x_i+\mu), \quad \check{x}_i = \frac{x_i}{(x_i+\mu)^2}, \quad \check{x}_{i+1} = x_{i+1}(x_i+\mu), \quad \check{x}_j = x_j \ \text{ pour } j \neq i-1, i, i+1.$$

La t-réduction au sens de Lovász s'exprime alors par le fait que chacune des composantes x_i de $\mathbf{x} = (x_1, x_2, \ldots, x_{d-1})$ vérifie $x_i + \mu \geq (1/t)^2$. Autrement dit, le trou de l'algorithme est

$$\mathcal{O}_{\mu,t}^{d-1} \quad \text{avec} \quad \mathcal{O}_{\mu,t} = \left[\frac{1}{t^2} - \mu, +\infty\right[.$$

L'algorithme LLL(t) se modélise comme un système dynamique de $\mathcal{I}_\mu^{d-1}$ avec $I_\mu = [0, 1/(4\mu)]$ où chaque étape correspond à l'application d'une des transformations $T_{i,\mu}$. On sort dès que $\mathbf{x} \in \mathcal{O}_{\mu,t}^{d-1}$. Le choix de la transformation dépend de la stratégie de l'algorithme. Dans sa stratégie

Entrées : Le vecteur $\mathbf{x} = (x_1, \ldots, x_{d-1}) \in \mathcal{I}_\mu^{d-1}$
Résultat : Le vecteur final $\widehat{\mathbf{x}} = (\widehat{x}_1, \ldots, \widehat{x}_{d-1}) \in \mathcal{O}_{\mu,t}^{d-1}$

tant que $\mathbf{x} \notin \mathcal{O}_{\mu,t}^{d-1}$ **faire**
  Choisir i tel que $x_i \notin \mathcal{O}_{\mu,t}$;
  Calculer $\mathbf{x} := T_{i,\mu}(\mathbf{x})$;
  $x_{i-1} := x_{i-1}(x_i + \mu), \qquad x_{i+1} := x_{i+1}(x_i + \mu), \qquad x_i = \dfrac{x_i}{(x_i+\mu)^2}$;
fintq

FIGURE 7.2: Système dynamique du modèle $\mathtt{M2}(t, \mu)$ associé au paramètre $\mu \in]0, 1/4[$ pour modéliser l'algorithme LLL(t). Les ensembles $\mathcal{I}_\mu$ et $\mathcal{O}_{\mu,t}$ sont deux intervalles, avec $\mathcal{I}_\mu = [0, 1/(4\mu)]$ et $\mathcal{O}_{\mu,t} = [(1/t^2) - \mu, +\infty[$.

usuelle, l'algorithme utilise la transformation $T_{i,\mu}$ en choisissant pour i le premier indice pour lequel $x_i \notin \mathcal{O}_{\mu,t}$. La section 7.1.3 présente d'autres stratégies. La figure 7.2 décrit la structure générale du système dynamique $\mathtt{M2}(t, \mu)$ dans $\mathbb{R}^{d-1}$ (ou plus précisément I_μ^{d-1}) qui est associée à LLL et qui dépend des deux paramètres t et μ.

Répétons que le modèle $\mathtt{M2}(t, \mu)$ suppose que la partie fractionnaire centrée du coefficient sous-diagonal courant $m_{i+1,i}$ est constant et égal à μ. Dorénavant, nous omettons la référence à μ et écrirons T_i au lieu de $T_{i,\mu}$.

7.1.3 Stratégies pour l'exécution de l'algorithme LLL

A tout instant j de l'exécution, il y a (en général) plusieurs indices i pour lequel la base locale ne vérifie pas la condition de Lovász. C'est l'ensemble $\mathcal{NR}(j)$ du chapitre 4. L'algorithme LLL(t) choisit l'une de ces bases et effectue la réduction. La stratégie qui gouverne ce choix a une influence sur le temps d'exécution de l'algorithme. Nous revenons sur quelques unes de ces stratégies, déjà décrites dans le chapitre 4.

Stratégie classique de LLL. On choisit le plus petit indice i pour lequel la base locale $(\mathbf{b}_i, \mathbf{b}_{i+1})$ ne satisfait pas la condition de Lovász. En reprenant les notations de la section précédente, l'indice i est défini par

$$i = \min\{j \in [1\,..\,d-1] \;\mid\; x_j \notin \mathcal{O}_{\mu,t}\}.$$

Dans le cas de l'algorithme LLL en dimension 3, le système dynamique est constitué de deux transformations (T_1, T_2). La stratégie s'écrit alors :

1. si T_1 peut être appliquée, on l'applique
2. sinon on applique T_2.

Stratégie inverse. On choisit le plus grand indice i pour lequel que la base locale $(\mathbf{b}_i, \mathbf{b}_{i+1})$ ne satisfait pas la condition de Lovász. Avec les notations de la section précédente, l'indice i est donné par

$$i = \max\{j \in [1\,..\,d-1] \mid x_j \notin \mathcal{O}_{\mu,t}\}.$$

Dans le cas du système dynamique de dimension 2 avec les deux transformations (T_1, T_2), la stratégie s'écrit

1. si T_2 peut être appliquée, on l'applique
2. sinon on applique T_1.

Entrées : le vecteur $(x, y) \in \mathcal{I}_\mu^2$
Résultat : le vecteur final $(\widehat{x}, \widehat{y}) \in \mathcal{O}_{\mu,t}^2$

tant que $(x, y) \notin \mathcal{O}_{\mu,t}^2$ **faire**
Selon la stratégie choisie, choisir T_1 ou T_2,

$$(x, y) := T_1(x, y) = \left(\frac{x}{(x+\mu)^2}, y(x+\mu)\right) \quad \text{ou} \quad (x, y) := T_2(x, y) = \left(x(y+\mu), \frac{y}{(y+\mu)^2}\right)$$

fintq

FIGURE 7.3: Système dynamique du modèle $\texttt{M2}(t, \mu)$ associé au paramètre $\mu \in]0, 1/4[$ pour modéliser l'algorithme LLL(t) en dimension 3. Les ensembles $\mathcal{I}_\mu$ et $\mathcal{O}_{\mu,t}$ sont deux intervalles, avec $\mathcal{I}_\mu = [0, 1/(4\mu)]$ et $\mathcal{O}_{\mu,t} = [(1/t^2) - \mu, +\infty[$.

Stratégie aléatoire. On tire l'indice i aléatoirement et uniformément parmi les indices des bases locales qui ne satisfont pas la condition de Lovász. Dans le cas du système dynamique de dimension 2 avec les deux transformations (T_1, T_2), la procédure est la suivante :

1. Si les deux transformations peuvent être appliquées, on en choisit une au hasard.
2. sinon, on applique la seule transformation qu'on peut appliquer.

Stratégie gloutonne. Cette stratégie choisit l'indice i pour lequel $x_i + \mu$ est le plus petit possible. Dans le cas du système dynamique constitué de deux transformations (T_1, T_2), la stratégie s'écrit :

1. si $x_1 \leq x_2$, alors on applique T_1
2. sinon T_2 est appliquée.

Les deux premières stratégies ont des dynamiques complètement symétriques. Il suffit d'échanger les rôles de x_i et x_{d-i}. Cependant ces stratégies donnent des rôles différents aux fonctions T_i. Par exemple, la stratégie classique en dimension 2 donne la priorité à T_1 par rapport à T_2. A l'inverse, la stratégie gloutonne donne un rôle identique à toutes les fonctions ce qui la rend plus facile à étudier.

7.1.4 Modèle simplifié de l'algorithme LLL en dimension 3

La section 7.4 est essentiellement consacrée à l'analyse du système dynamique de dimension 2 ($d = 3$) avec la stratégie gloutonne. Dans ce cas, le système dynamique est un système à deux variables,

$$x := x_1 = \frac{\ell_2^2}{\ell_1^2}, \qquad y := x_2 = \frac{\ell_3^2}{\ell_2^2}$$

avec deux transformations

$$T_1(x, y) = \left(\frac{x}{(x+\mu)^2}, y(x+\mu)\right) \qquad \text{si } x \notin \mathcal{O}_{\mu,t},$$

$$T_2(x, y) = \left(x(y+\mu), \frac{y}{(y+\mu)^2}\right) \qquad \text{si } y \notin \mathcal{O}_{\mu,t}.$$

Comme le résume la figure 7.3, le système dynamique applique T_1 ou T_2 selon la stratégie tant que $(x, y) \notin \mathcal{O}_{\mu,t}^2$.

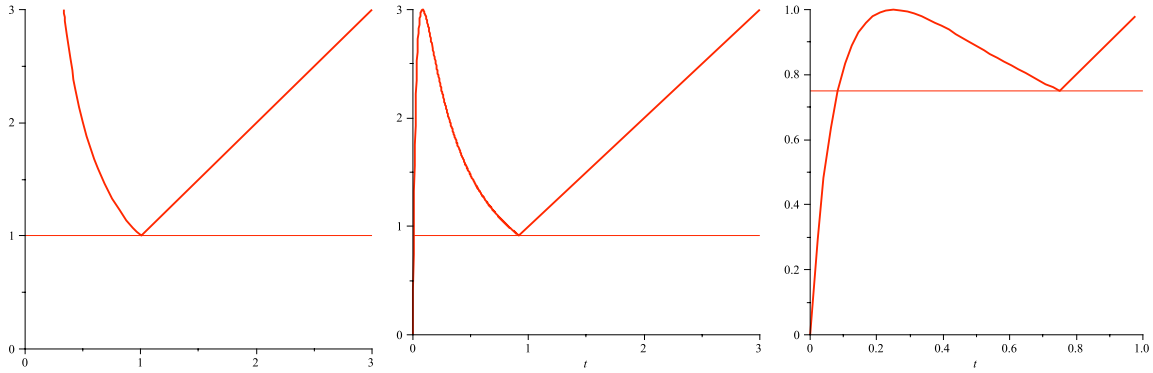


FIGURE 7.4: Les graphes des fonctions f_μ pour trois valeurs de μ égales à $0, 1/12, 1/4$, avec la droite $y = 1 - \mu$. La première figure est tronquée et limitée au carré $[0, 3] \times [0, 3]$.

7.2 Étude du système dynamique de dimension 1 : modèle simplifié de Gauss

Dans cette section, nous étudions le nombre d'itérations effectuées par le système dynamique $\mathtt{M2}(\mu)$ associé à l'algorithme de Gauss ($d = 2$ et $t = 1$). Le véritable algorithme est bien analysé, comme on l'a rappelé dans le chapitre 2, pour l'analyse dans le pire des cas, ou dans le chapitre 4, pour l'analyse en moyenne. Le nombre d'itérations dans le pire des cas ainsi que l'entrée correspondante ont respectivement été explicités par Lagarias [59] puis Vallée [108]. En 1990, Flajolet et Vallée [110] puis Daudé, Flajolet et Vallée en 1994 [27] ont effectué une analyse en distribution du nombre d'itérations dans le modèle uniforme. En parallèle, Laville et Vallée [64] ont étudié les principaux paramètres de sortie (défaut d'Hermite, premier minimum) toujours dans le modèle uniforme. En 2007, Vallée et Vera ont généralisé tous ces résultats dans un modèle aléatoire "avec valuation", introduit dans [112] (voir le chapitre 4, section 4.4.2) et qui permet de faire varier le poids des entrées difficiles à réduire. Toutes les analyses montrent que le nombre d'itérations suit une loi géométrique dont la raison dépend du modèle. Nous allons retrouver ces résultats (de manière qualitative) dans le modèle simplifié M2, et ainsi confronter ce modèle M2 au "véritable" algorithme de Gauss.

La transformation effectuée par le système dynamique de dimension 1 s'écrit sous la forme $f_\mu : I_\mu \mapsto I_\mu$ avec

$$f_\mu(x) = \frac{x}{(x+\mu)^2} \quad \text{si} \quad x < 1 - \mu, \quad f_\mu(x) = x \quad \text{si} \quad x \geq 1 - \mu.$$

Plusieurs représentations en fonction de μ sont données à la Figure 7.4.

7.2.1 Description du domaine $[K = k]$

On travaille avec la variable $K(x)$ qui associe à $x \in I_\mu$ le nombre d'itérations effectué par le système (I_μ, f_μ) sur l'entrée x. La proposition suivante décrit pour tout k, le domaine $[K = k]$ où le système effectue k itérations.

Proposition 7.1. *Considérons le système dynamique (f_μ, I_μ). Pour $k \geq 1$, on désigne par V_k l'ensemble des entrées $x \in I_\mu$ sur lesquelles le système dynamique effectue exactement k étapes. Alors V_1 est de la forme $[a_1, 1 - \mu]$ avec $a_1 = \mu^2/(1 - \mu)$ et pour tout $k \geq 2$, V_k est de la forme $[a_k, a_{k-1}[$ avec $a_{k-1} = f_\mu(a_k)$ et $a_k = \Theta(\mu^{2k})$.*

Preuve. Nous commençons par déterminer le voisinage V_1 de $1-\mu$ dans lequel le système n'effectue qu'une itération. Pour cela, il suffit de résoudre l'équation $f_\mu(x) \geq 1-\mu$ qui se ramène à une équation polynomiale du second degré et qui après résolution est équivalente à la condition $x \in [a_1, 1-\mu]$ avec $a_1 = \mu^2/(1-\mu)$.

On considère l'ensemble V_k des entrées x sur lesquelles le système dynamique effectue k itérations. La fonction f_μ est croissante sur l'intervalle $[0,\mu]$ et $a_1 < \mu$. Ainsi pour $k \geq 2$, V_k est un intervalle de la forme $[a_k, a_{k-1}[$ avec $a_{k-1} = f_\mu(a_k)$. Il reste à montrer que a_k est d'ordre $\Theta(\mu^{2k})$. Nous avons

$$\frac{f_\mu(x)}{x} = \frac{1}{(x+\mu)^2} \leq \frac{1}{\mu^2} \quad \text{et par induction} \quad \frac{f_\mu^k(x)}{x} \leq \frac{1}{\mu^{2k}} \tag{7.2}$$

pour tout k tel que $f^{k-1}(x)$ n'est pas dans le trou. En prenant $x = a_{k+1}$, on montre que $a_{k+1} \geq a_1\mu^{2k}$. En induisant l'égalité dans l'équation (7.2), nous obtenons

$$\frac{x}{f_\mu^k(x)} = \prod_{j=0}^{k-1} (f_\mu^j(x)+\mu)^2,$$

et en prenant $x = a_{k+1}$, l'égalité devient

$$\frac{a_{k+1}}{a_1} = \prod_{j=2}^{k+1} (a_j+\mu)^2 \leq \beta\mu^{2k} \quad \text{avec} \quad \beta = \prod_{j\geq 2} \left(1+\frac{a_j}{\mu}\right)^2.$$

La constante β est bien définie car pour $x \leq a_1$, $x \leq (a_1+\mu)^2 f_\mu(x)$ et par induction, a_j est exponentiellement petit d'ordre $O((a_1+\mu)^{2j})$. En conclusion, nous avons $a_1\mu^{2k} \leq a_{k+1} \leq \beta\mu^{2k}$ ce qui complète la preuve.

□

7.2.2 Loi géométrique pour le nombre d'itérations.

Lorsque les entrées sont choisies selon une loi puissance de paramètre r (on dit encore un modèle de valuation r), le nombre d'itérations de l'algorithme de Gauss suit asymptotiquement une loi géométrique de paramètre $\lambda(2+r)$, comme on l'a décrit dans le chapitre 4. Ici, $\lambda(s)$ est la valeur propre dominante de l'opérateur de transfert associé à l'algorithme d'Euclide. La fonction "valeur propre dominante" vérifie $\lambda(1) = 1$. Plusieurs auteurs ont proposé une méthode efficace de calcul de cette fonction, notamment dans [27], et on a par exemple $\lambda(2) = 0.1994\cdots$

La proposition suivante est une conséquence immédiate de la proposition 7.1. Elle montre que le nombre d'itérations du modèle `M2` suit une loi asymptotiquement géométrique, et ce, dans tout modèle "à valuation". C'est l'équivalent de la loi géométrique obtenue pour l'algorithme de Gauss mais dans le modèle simplifié `M2`(μ).

Proposition 7.2. *Considérons, pour $r > -1$, l'intervalle I_μ muni de la densité $g_r(x) = c_r \cdot x^r$ avec $c_r > 0$, et la variable aléatoire K définie sur I_μ mesurant le nombre d'itérations : pour $x \in I_\mu$, $K(x)$ désigne le nombre d'itérations nécessaires au système dynamique (I_μ, f_μ) pour atteindre le trou à partir de x. Alors, la variable K suit une loi asymptotiquement géométrique de raison $\mu^{2(r+1)}$. Précisément, on a, pour tout $r > -1$,*

$$\mathbb{P}[K > k] = \frac{c_r}{r+1} a_k^{r+1} = \Theta\left(\mu^{2k(r+1)}\right), \quad (k \to \infty).$$

Remarque. La proposition 7.2 reste vraie pour le système dynamique $\mathtt{M2}(t,\mu)$ associé à l'algorithme de t-Gauss dont le trou est donné par $\mathcal{O}_{t,\mu} = \{x \in I_\mu \ : \ x > \frac{1}{t^2} - \mu\}$. En effet pour $x \in \mathcal{O}_{t,\mu}$, la proposition 7.1 montre qu'il suffit d'un nombre constant d'itérations pour que x atteigne le trou $\mathcal{O}_{1,\mu}$. Autrement dit, le nombre d'itérations de $\mathtt{M2}(t,\mu)$ est le nombre d'itérations de $\mathtt{M2}(1,\mu)$ à une constante près.

On ne peut espérer cependant que ce modèle M2 permette de retrouver le comportement probabiliste exact de l'algorithme de Gauss, pour lequel la raison de la progression géométrique pour une valuation r est égale à $\lambda(2+r)$. On peut cependant comparer qualitativement les deux modèles. Quand $r \to -1$, on a :

$$\lambda(2+r) \underset{r\to -1}{\sim} 1 - \frac{\pi^2}{6\ln 2}(r+1), \qquad \mu^{2(r+1)} \underset{r\to -1}{\sim} 1 + 2(r+1)\ln \mu.$$

On pourrait vouloir choisir μ pour que l'égalité suivante soit vraie

$$\log \mu = -\frac{\pi^2}{12\ln 2} \quad \text{soit} \quad \mu = 0.3052\cdots,$$

mais c'est une valeur impossible pour μ puisque μ doit appartenir à l'intervalle $[0, 1/4]$... Si on veut faire "coller" les deux modèles pour une densité initiale uniforme, on voudrait choisir $\mu = \lambda(2)^{1/2} = 0.446\cdots$ mais par hypothèse $\mu \in [0, 1/4]$... Cela montre à la fois les vertus et les limites de notre modèle M2.

7.2.3 Nombre d'itérations pour les entrées difficiles.

Nous terminons la section par un autre résultat qui découle directement de la proposition 7.1. Il décrit le nombre d'itérations lorsque l'entrée initiale x n'est pas du tout réduite, c'est-à-dire lorsque x tend vers 0.

Proposition 7.3. *Le nombre $K(x)$ d'itérations du système dynamique (f_μ, I_μ) sur l'entrée x vérifie*

$$K(x) = \frac{1}{2}\log_\mu x + O(1) \quad \textit{lorsque } x \to 0.$$

Remarque. Comme précédemment, la proposition 7.3 reste vraie pour le système dynamique $\mathtt{M2}(t,\mu)$ associé à l'algorithme de t-Gauss.

7.2.4 Quelques choix naturels pour le paramètre μ.

Lorsque LLL est utilisé avec la condition de Siegel, les coefficients sous-diagonaux de la matrice d'orthogonalisation de Gram-Schmidt semblent se répartir "assez uniformément" dans l'intervalle $[-1/2, 1/2]$. Intuitivement, on peut donc choisir $\mu = \mu_0$ comme étant la valeur moyenne de μ^2 quand μ est uniforme dans l'intervalle $[-1/2, 1/2]$, c'est-à-dire

$$\mu_0 = 2\int_0^{1/2} x^2 dx = \frac{1}{12} = 0.0833\cdots.$$

On peut aussi choisir $\mu = \mu_1$ de manière à assurer que la probabilité d'être dans l'intervalle $[0, 1-\mu]$ est la même que celle d'être dans l'intervalle $[1-\mu, 1/(4\mu)]$ soit

$$1 - \mu = \frac{1}{4\mu} - (1-\mu), \qquad \text{et} \quad \mu \approx 0.15.$$

7.3 Rappels sur les systèmes dynamiques de $\mathbb{R}^2$

Le modèle simplifié de LLL(t) en dimension 3 conduit à un système dynamique de $\mathbb{R}^2$. Dans cette section, nous présentons très succinctement l'étude des systèmes de $\mathbb{R}^2$. Nous renvoyons à [55] pour une présentation beaucoup plus complète.

Dans toute cette section, X est un compact de $\mathbb{R}^2$ et G une application de X dans X. Le couple (G, X) forme un système dynamique. Nous noterons $z_f = (x_f, y_f)$ un point fixe de G, c'est-à-dire, $G(x_f, y_f) = (x_f, y_f)$. Bien entendu, tous les systèmes dynamiques n'ont pas nécessairement de point fixe mais l'étude du modèle simplifié implique l'existence d'un point fixe en $(x_f, y_f) = (1-\mu, 1-\mu)$. L'application G peut se voir comme un couple de fonctions (f, g), autrement dit $G(x, y) = (f(x, y), g(x, y))$, avec f et g définies sur X. Pour un point $z = (x, y)$ de $\mathbb{R}^2$, $\|z\|$ désigne la norme euclidienne de z. Si $r \geq 0$, $V_z(r)$ est la boule ouverte de centre z et de rayon r (pour la norme euclidienne),

$$V_z(r) = \{z' \in \mathbb{R}^2 \mid \|z - z'\| < r\}.$$

7.3.1 Linéarisation et matrice jacobienne

Fixons $z = (x, y)$ un point de X. La matrice jacobienne de G au point z est définie par

$$\mathrm{Jac}_G(z) := \begin{pmatrix} \dfrac{\partial f}{\partial x}(z) & \dfrac{\partial f}{\partial y}(z) \\ \dfrac{\partial g}{\partial x}(z) & \dfrac{\partial g}{\partial y}(z) \end{pmatrix}.$$

La matrice jacobienne joue le même rôle que la dérivée pour les fonctions à une variable. En particulier, on retrouve deux résultats importants que sont le théorème des accroissements finis et le développement de Taylor.

Théorème 7.4 (Développement de Taylor à l'ordre 1)**.** *Si l'application G est de classe C^2 dans un voisinage ouvert V de $z \in X$, alors pour tout $z' \in V$,*

$$G(z') - G(z) = \mathrm{Jac}_G(z)(z' - z) + O(\|z' - z\|^2),$$

où la constante dans le O ne dépend que de G et V.

Autrement dit, l'application $G(z')$ autour du point z est proche de l'application affine $G(z) + \mathrm{Jac}_G(z)(z' - z)$.

Théorème 7.5 (Inégalité des accroissements finis)**.** *Soit U un ouvert de X. Si G est une application de classe C^1 sur U, alors pour tout segment $[z, z']$ inclus dans U, on a*

$$\|G(z') - G(z)\| \leq \left(\sup_{t \in [z, z']} \|\mathrm{Jac}_G(t)\| \right) \|z' - z\|.$$

Pour une matrice M, $\|M\|$ est la norme matricielle subordonnée à la norme $\|\cdot\|$. Notons que l'inégalité des accroissements finis reste vraie quelle que soit la norme utilisée.

Considérons le point fixe z_f. Si les valeurs propres de la matrice $\mathrm{Jac}_G(z_f)$ sont en module strictement plus petites que 1, il existe une norme sur $\mathbb{R}^2$, notée abusivement $\|\cdot\|$, telle que $\|\mathrm{Jac}_G(t)\| < 1$ [ce résultat bien connu se démontre en utilisant les formes de Jordan, voir section 7.3.2]. La fonction G est de classe C^1 et par perturbation [54], il existe un voisinage U de

z' tel que pour tout $z \in U$, $\|\mathrm{Jac}_G(z)\| < 1 - \epsilon$ pour un $\epsilon > 0$. En appliquant récursivement l'inégalité des accroissements finis, on montre que pour tout $z \in U$,

$$\|G^k(z) - z_f\| \leq (1-\epsilon)^k \|z - z_f\|.$$

Autrement dit, le point fixe est un point attractif (ou stable). Le système dynamique converge vers le point fixe. Nous avons le résultat un peu plus fort suivant.

Proposition 7.6. *Le point fixe z_f de l'application G est asymptotiquement stable (ou attractif) si et seulement si le module de toutes les valeurs propres de $\mathrm{Jac}_G(z_f)$ est strictement inférieur à 1.*

7.3.2 Formes de Jordan réelles dans $\mathbb{R}^2$

La proposition 7.6 montre le rôle important joué par les valeurs propres de la matrice jacobienne autour d'un point fixe. Considérons une matrice A carrée et réelle. Alors les valeurs propres de A sont les solutions de l'équation caractéristique

$$\lambda^2 - \mathrm{Tr}(A)\lambda + \det(A) = 0.$$

Si λ_1 et λ_2 sont les racines (complexes) de l'équation, la trace et le déterminant satisfont

$$\mathrm{Tr}(A) = \lambda_1 + \lambda_2 \qquad \text{et} \qquad \det(A) = \lambda_1 \lambda_2.$$

Le discriminant de l'équation est donné par

$$\Delta = \mathrm{Tr}(A)^2 - 4\det(A).$$

Théorème 7.7 (Réduction de Jordan)**.** *Soit A une matrice réelle carrée de dimension 2. Alors A vérifie une et une seule des propriétés suivantes :*

(a) A est de la forme $\lambda \cdot I_2$ avec I_2 la matrice identité et λ l'unique valeur propre double de A (en particulier, $\Delta = 0$).

(b) $\Delta = 0$ et A n'est pas de la forme $\lambda \cdot I_2$. Alors il existe une matrice réelle inversible P telle que

$$P^{-1}AP = \begin{pmatrix} \lambda & 1 \\ 0 & \lambda \end{pmatrix}$$

avec λ l'unique valeur propre de A.

(c) $\Delta > 0$: A admet deux valeurs propres distinctes λ_1 et λ_2 de vecteurs propres associés u_1 et u_2. Alors en posant $P = (u_1|u_2)$, P est inversible et

$$P^{-1}AP = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}.$$

(d) $\Delta < 0$: A admet deux valeurs propres complexes conjuguées $\lambda_1 = a + ib$ et $\lambda_2 = a - ib$. Les vecteurs propres associés u_1 et u_2 sont également conjugués de la forme $u_1 = v + iw$ et $u_2 = v - iw$ avec v et w des vecteurs réels. En posant $P = (v|-w)$, P est inversible et

$$P^{-1}AP = \begin{pmatrix} a & b \\ -b & a \end{pmatrix}.$$

La forme particulière de J rend le calcul de J^n très facile. Comme $A^n = PJ^nP^{-1}$, le comportement asymptotique de A^n est très lié à celui de J^n. Autour d'un point fixe, un système dynamique est très proche de sa matrice jacobienne. Intuitivement, itérer le système revient à itérer la matrice. La décomposition de Jordan éclaire alors le comportement du système dynamique autour du point fixe.

Dans le cas (d), la forme de Jordan peut s'écrire sous une forme plus géométrique. Les deux valeurs propres s'écrivent sous la forme $\lambda_1 = \rho e^{i\alpha}$ et $\lambda_2 = \rho e^{-i\alpha}$. On a alors $a = \rho\cos\alpha$ et $b = \rho\sin\alpha$ et la matrice J devient

$$J = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} = \begin{pmatrix} \rho\cos\alpha & -\rho\sin\alpha \\ \rho\sin\alpha & \rho\cos\alpha \end{pmatrix} = \rho\mathrm{Rot}(\alpha).$$

La matrice $\mathrm{Rot}(\alpha)$ correspond à une rotation d'angle α. On déduit facilement que $J^n = \rho^n\mathrm{Rot}(n\alpha)$ est une similitude avec un rapport ρ^n et un angle de rotation $n\alpha$.

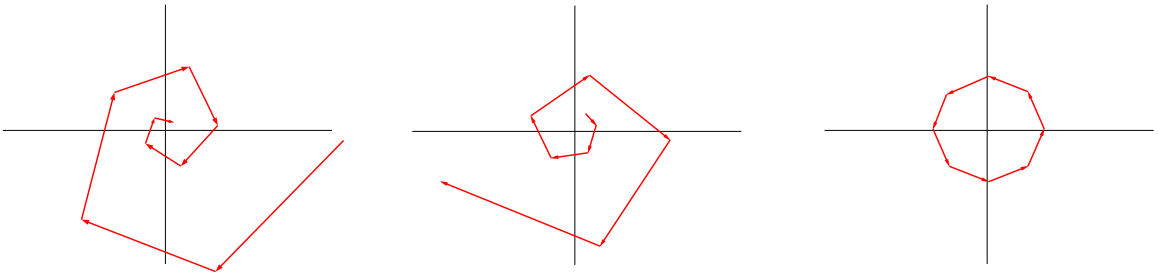
7.3.3 Typologie des trajectoires

Les trajectoires d'une matrice M sont à un changement de base près les trajectoires de la forme de Jordan associée. Nous décrivons ici les trajectoires lorsque l'on itère une matrice J mise sous forme de Jordan. On considère z_0 un élément de $\mathbb{R}^2$ et nous notons $z_n := J^n z_0$.

Cas (a) et (d) : si J est de la forme

$$J = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} = \begin{pmatrix} \rho\cos\alpha & -\rho\sin\alpha \\ \rho\sin\alpha & \rho\cos\alpha \end{pmatrix} = \rho\mathrm{Rot}(\alpha),$$

alors la trajectoire $(x_n)_{n\geq 0}$ tourne autour de 0, à chaque fois d'un angle α, soit en s'éloignant de 0 si $\rho > 1$, soit en s'approchant de 0 si $\rho < 1$, soit en restant toujours à la même distance si $\rho = 1$. Le cas (a) correspond à $\rho = |\lambda|$, $\alpha = 0$ si $\lambda > 0$ et $\alpha = \pi$ si $\lambda < 0$.



$\rho < 1$ $\rho > 1$ $\rho = 1$

FIGURE 7.5: Cas (a) : trajectoires de la forme de Jordan associée.

Cas (b) : Le n-ième itéré de J vérifie alors

$$J^n = \lambda^n \begin{pmatrix} 1 & n/\lambda \\ 0 & 1 \end{pmatrix}.$$

Si $|\lambda| < 1$, alors la trajectoire $(z_n)_n$ s'approche de 0 de manière tangente à l'axe des abscisses. Si $|\lambda| \geq 1$, la trajectoire s'éloigne à la fois de l'axe des abscisses et des ordonnées mais le vecteur $\overrightarrow{Oz_n}$ est de plus en plus tangent à l'axe des abscisses. Dans les deux cas, la trajectoire alterne autour de l'axe des ordonnées si $\lambda < 0$.

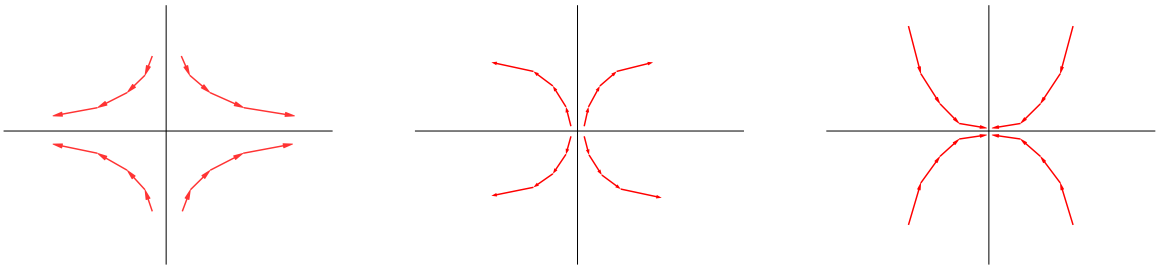
$|\lambda| \geq 1$ $|\lambda| < 1$

FIGURE 7.6: Cas (b) : trajectoires de la forme de Jordan associée.

Cas (c) **:** Le n-ième itéré de J vérifie alors

$$J^n = \begin{pmatrix} \lambda_1^n & 0 \\ 0 & \lambda_2^n \end{pmatrix}.$$

Quitte à échanger λ_1 et λ_2, on peut supposer $|\lambda_2| < |\lambda_1|$. Si $|\lambda_1| < 1$, alors la trajectoire $(z_n)_n$ s'approche de 0 de manière tangente à l'axe des abscisses. Si $|\lambda_2| > 1$, alors la trajectoire s'éloigne de l'axe des abscisses et des ordonnées mais le vecteur $\overrightarrow{Oz_n}$ est de plus en plus tangent à l'axe des abscisses. Si $|\lambda_2| < 1 < |\lambda_1|$, alors la trajectoire s'éloigne de 0 mais s'approche de l'axe des abscisses. Dans tous les cas, la trajectoire alterne autour de l'axe des abscisses (resp. ordonnées) si $\lambda_1 < 0$ (resp. $\lambda_2 < 0$).



$|\lambda_2| < 1 < |\lambda_1|$ $1 < |\lambda_2| < |\lambda_1|$ $|\lambda_2| < |\lambda_1| < 1$

FIGURE 7.7: Cas (c) : trajectoires de la forme de Jordan associée.

7.4 Étude du système dynamique associé à LLL en dimension 3

Le modèle simplifié $\texttt{M2}(t, \mu)$ de l'algorithme LLL(t) en dimension $d = 3$ est un système dynamique de $\mathbb{R}^2$ qui applique selon la stratégie choisie l'une des deux applications suivantes :

$$T_1(x, y) = \left(\frac{x}{(x+\mu)^2}, y(x+\mu)\right) \qquad \text{si } x < \tfrac{1}{t^2} - \mu$$

$$T_2(x, y) = \left(x(y+\mu), \frac{y}{(y+\mu)^2}\right) \qquad \text{si } y < \tfrac{1}{t^2} - \mu$$

Le processus s'arrête lorsque T_1 et T_2 ne peuvent plus s'appliquer c'est à dire lorsque le point (x, y) est dans le trou

$$\mathcal{O}^2_{\mu,t} = \{(x, y) : \quad x \geq \frac{1}{t^2} - \mu; y \geq \frac{1}{t^2} - \mu\}.$$

Stratégie gloutonne. On rappelle la stratégie gloutonne : si $x < y$, on applique T_1 sinon, on applique T_2. A chaque étape, cette stratégie choisit le tas qui fera diminuer au maximum la "quantité de sable". La stratégie gloutonne fait jouer un rôle parfaitement symétrique aux transformations T_1 et T_2. Elle est dans un sens beaucoup plus naturelle. C'est sans doute pour cela que nous avons réussi à faire l'étude du modèle $\texttt{M2}(1, \mu)$ seulement pour cette stratégie. Nous nous placerons toujours dans le cadre de la stratégie gloutonne.

En dimension 2, la quantité qui augmente tout le long de l'algorithme $\texttt{M2}(t, \mu)$ est xy. Cette quantité est reliée à la fois à la masse et à l'énergie du *cfg* correspondant, puisque

$$xy = \exp\left(-2\mathcal{M}(\mathbf{c})\right) = \exp\left(-\mathcal{E}(\mathbf{c})\right).$$

Ici, en désignant par (x', y') le transformé de (x, y) par T avec $T = T_1$ ou $T = T_2$, choisi selon la stratégie gloutonne, nous avons

$$x'y' = \frac{xy}{\min(x, y) + \mu} > xy.$$

Cette formule est fondamentale dans nos analyses et est fortement liée à la stratégie gloutonne. En particulier, les domaines $\mathcal{D}_{t,\mu}$ donnés par

$$\mathcal{D}_{t,\mu} = \{(x, y) \in I_\mu \times I_\mu \mid xy \geq (\frac{1}{t^2} - \mu)^2\} \tag{7.3}$$

sont des domaines stables par les transformations T_1 ou T_2 : on a l'inclusion $T(\mathcal{D}_{t,\mu}) \subset \mathcal{D}_{t,\mu}$ pour $T \in \{T_1, T_2\}$ dès que $t \geq 1$.

Le système dynamique $\texttt{M2}(1, \mu)$ peut être vu comme l'exécution d'un système dynamique $\texttt{M2}(t, \mu)$ avec $t > 1$ suivi d'itérations du système $\texttt{M2}(1, \mu)$. Ceci n'est plus vrai pour l'algorithme LLL(1) mais cette approche est similaire à celles utilisées par Lenstra [67] puis Akhavi [7]. Nous commençons donc par l'analyse de $\texttt{M2}(t, \mu)$ pour $t > 1$ et nous terminons avec celle de $Mdeux(1, \mu)$.

7.4.1 Étude du modèle $\texttt{M2}(t, \mu)$ avec $t > 1$.

Un premier lemme grossier. Le lemme suivant donne une description grossière d'une trajectoire qui sera utilisée pour obtenir une description plus fine.

Lemme 7.8. *Pour $(x_0, y_0) \in I^2_\mu$, on désigne par $(x_0, y_0), (x_1, y_1), \ldots, (x_i, y_i)$ une trajectoire de (x_0, y_0) sous l'action du système dynamique. On désigne par k le premier indice pour lequel $(x_k, y_k) \in \mathcal{O}^2_{t,\mu}$ Alors, pour tout $i \in [0\,..\,k]$, les coordonnées du point (x_i, y_i) vérifient*

$$\min(x_i, y_i) \leq \frac{1}{4\mu}\frac{1}{t^{k-i}}.$$

Démonstration. Pour $i \in [0\,..\,k-1]$, $(x_i, y_i) \notin \mathcal{O}^2_{t,\mu}$ et $\min(x_i, y_i) < \frac{1}{t^2} - \mu$ soit

$$x_{i+1}y_{i+1} = \frac{x_i y_i}{\min(x_i, y_i) + \mu} > t^2 x_i y_i.$$

En itérant cette inégalité, on montre par induction que pour $i \in [0\,..\,k]$,

$$x_i y_i \leq \frac{1}{(4\mu)^2} \frac{1}{t^{2(k-i)}},$$

car $x_k y_k \leq 1/(4\mu)^2$. Le lemme se déduit de l'inégalité $\min(x_i, y_i) \leq \sqrt{x_i y_i}$.

□

Détermination du domaine $[K_{t,\mu} = k]$**.** Le résultat suivant est l'équivalent de la proposition 7.1.

Proposition 7.9. *Il existe deux constantes positives $\alpha = \alpha(\mu, t)$ et $\beta = \beta(\mu, t)$ telles que si le système* $\texttt{M2}(t, \mu)$ *effectue k itérations sur l'entrée (x, y), nous avons*

$$\alpha \mu^k \leq xy \leq \beta \mu^k.$$

Démonstration. Nous reprenons les notations de la preuve du lemme 7.8. En appliquant le lemme 7.8 avec la relation $x_i y_i = (\min(x_i, y_i) + \mu) x_{i+1} y_{i+1}$, nous obtenons l'encadrement

$$\mu \leq \frac{x_i y_i}{x_{i+1} y_{i+1}} \leq \mu \left(1 + \frac{1}{4\mu^2} \frac{1}{t^{k-i}}\right).$$

Le produit terme à terme donne

$$\mu^k \leq \frac{x_0 y_0}{x_k y_k} \leq \mu^k \prod_{i=1}^{k} \left(1 + \frac{1}{4\mu^2} \frac{1}{t^i}\right),$$

et puisque (x_k, y_k) est dans le trou, nous obtenons le lemme en posant

$$\alpha = \left(\frac{1}{t^2} - \mu\right)^2 \qquad \text{et} \qquad \beta = \frac{1}{(4\mu)^2} \prod_{i=1}^{+\infty} \left(1 + \frac{1}{4\mu^2} \frac{1}{t^i}\right).$$

□

Remarques. A partir de la proposition 7.9, nous pouvons étudier la loi du nombre d'itérations du système $\texttt{M2}(t, \mu)$ pour tout $t > 1$. Remarquons cependant que la constante β admet la majoration suivante

$$\beta \leq \frac{1}{(4\mu)^2} \exp\left[\frac{1}{4\mu^2} \frac{1}{t-1}\right],$$

et on ne peut pas faire tendre t vers 1. Mais, nous allons montrer maintenant que l'on peut passer du cas $t > 1$ à $t = 1$ en raffinant l'étude précédente. Nous étudions maintenant le modèle pour $t = 1$.

7.4.2 Étude du modèle $\texttt{M2}(1, \mu)$.

Nous allons travailler avec les ensembles $\mathcal{D}_{t,\mu}$ défini en (7.3) et déterminer un nombre réel $t > 1$ tel que le système $\texttt{M2}(1, \mu)$ effectue au plus un nombre borné d'itérations en partant de $\mathcal{D}_{t,\mu}$.

Étude du domaine $\mathcal{D}_{1,\mu}$. Nous commençons par l'étude du cas $t = 1$.

Lemme 7.10. *Le système dynamique* $\texttt{M2}(1, \mu)$ *effectue au plus une itération sur le domaine* $\mathcal{D}_{1,\mu}$ *défini en (7.3).*

Preuve. Si $(x, y) \in \mathcal{D}_{1,\mu} \backslash \mathcal{O}^2_{1,\mu}$, alors T_1 ou T_2 s'applique. Prenons par exemple T_1, autrement dit $x < y \leq 1/(4\mu)$ et posons $(x', y') = T_1(x, y)$. Nous avons

$$y' = y(x + \mu) = yx + y\mu \geq (1 - \mu)^2 + y\mu.$$

Mais $xy \geq (1 - \mu)^2$ et $x < y$ impliquent $y \geq (1 - \mu)$ soit

$$y' \geq (1 - \mu)^2 + (1 - \mu)\mu = (1 - \mu).$$

De même, $xy \geq (1-\mu)^2$, $y \leq 1/(4\mu)$ et $\mu < 1/4$ impliquent $x \geq 4\mu(1-\mu)^2 > \mu$. Mais $x' = f_\mu(x)$ avec f_μ qui est décroissante pour $x > \mu$. Ainsi, x' est minimum lorsque x est maximum c'est-à-dire $x' \geq f_\mu(1-\mu) = 1-\mu$. L'analyse pour T_2 étant symétrique, cette dernière inégalité complète la preuve. □

Deux lemmes pour "passer" de $t > 1$ à $t = 1$. Nous voulons déterminer un domaine $\mathcal{D}_{t,\mu}$ avec $t > 1$ mais proche de 1 sur lequel le système dynamique $\texttt{M2}(1, \mu)$ effectue au plus un nombre borné d'étapes. Comme le trou $\mathcal{O}^2_{t,\mu}$ est inclus dans $\mathcal{D}_{t,\mu}$, nous en déduisons qu'après avoir appliqué le système $\texttt{M2}(t, \mu)$, le système $\texttt{M2}(1, \mu)$ effectue au plus un nombre borné d'étapes. L'asymptotique sera alors donnée par l'asymptotique de la première phase, décrite par la proposition 7.9. Le lemme suivant est spécifique à la dimension $d = 3$ et nous ne sommes pas parvenus à le généraliser au cas $d > 3$.

Lemme 7.11. *Considérons un paramètre* $t \in]1, 2/\sqrt{3}[$. *Alors, partant d'un point* $(x, y) \in \mathcal{D}_{t,\mu}$, *le système applique alternativement les fonctions* T_1 *et* T_2, *jusqu'à atteindre le trou* $\mathcal{O}^2_{1,\mu}$.

Démonstration. Si $t \in]1, 2/\sqrt{3}[$ alors $1 - (1/t^2) < 1/4$, et on pose $1/t^2 = 1 - \epsilon$. Soit $(x, y) \in \mathcal{D}_{t,\mu}$ tel que l'application T_1 s'applique (le cas où T_2 s'applique se traite de manière symétrique). Alors les relations $xy \geq (1 - \mu - \epsilon)^2$, $y \leq 1/(4\mu)$ et $\mu < 1/4$ impliquent $x \geq 4\mu(1 - \mu - \epsilon)^2 > \mu$ dès que $\epsilon < \frac{1}{4}$. Si $(x', y') = T_1(x, y)$, alors $x' = f_\mu(x)$ qui est minorée sur l'intervalle $[1 - \mu - \epsilon, 1 - \mu]$ par $1 - \mu$. Deux situations sont possibles : soit (x', y') est dans le trou et l'exécution est terminée, soit $y' < 1 - \mu \leq x'$ et T_2 peut s'appliquer. Après une itération de T_1, il y a donc une itération de T_2 (si le trou n'est pas atteint). Par symétrie, il en est de même pour T_2. Ceci complète le lemme. □

Le lemme suivant montre que pour t suffisamment proche de 1, le domaine $\mathcal{D}_{t,\mu}$ convient.

Lemme 7.12. *Pour tout* $\mu \in]0, 1/4]$, *il existe* $t_\mu \leq 2/\sqrt{3}$ *pour lequel, pour tout* $t \leq t_\mu$, *le système* $\texttt{M2}(1, \mu)$ *effectue au plus 3 itérations sur* $\mathcal{D}_{t,\mu}$.

Démonstration. On pose $1/t^2 = 1-\epsilon$. Soit $(x, y) \in \mathcal{D}_{t,\mu}$ tel que T_1 puis T_2 s'appliquent et posons $(x', y') = T_1(x, y)$ et $(x'', y'') = T_2(x', y')$. Par hypothèse, nous avons les deux relations

$$xy \geq (1 - \mu - \epsilon)^2 \quad \text{et} \quad y' = y(x + \mu) \leq 1 - \mu,$$

qui combinées entre elles donnent

$$x \geq \frac{\mu(1 - \mu - \epsilon)^2}{(1 - \mu) - (1 - \mu - \epsilon)^2} = 1 - \mu - \frac{2}{\mu}\epsilon + O(\epsilon^2). \tag{7.4}$$

Nous montrons maintenant qu'il existe $\epsilon_2 > 0$ tel que si $x > 1-\mu-\epsilon_2$, alors $T_2 \circ T_1(x,y) \in \mathcal{D}_{1,\mu}$. Nous avons $T_2 \circ T_1(x,y) \in \mathcal{D}_{1,\mu}$ si et seulement si $x''y'' \geq (1-\mu)^2$ ce qui après calcul de $T_2 \circ T_1$ revient à

$$\frac{xy}{(x+\mu)(y(x+\mu)+\mu)} \geq (1-\mu)^2.$$

La fonction h définie par

$$h(x,y) = xy - (1-\mu)^2(x+\mu)(y(x+\mu)+\mu).$$

est une fonction affine par rapport à la variable y, et la dérivée partielle $\partial h/\partial y(x,y)$ ne dépend que de x et vérifie

$$\frac{\partial h}{\partial y}(1-\mu, 1-\mu) = \mu(1-\mu) > 0.$$

Par continuité, il existe $\epsilon_1 > 0$ tel que pour tout $x \in [1-\mu-\epsilon_1, 1-\mu]$, la dérivée partielle est strictement positive. La fonction $y \to h(x,y)$ atteint alors son minimum en $y = x$ (car par hypothèse, $y \geq x$). Nous sommes ramenés à étudier la fonction $g(x) = h(x,x)$. Pour $x = 1-\mu$, on a

$$g'(1-\mu) = (1-\mu)\left[2 - (1-\mu)(3-\mu)\right] = -(1-\mu)(\mu-2)^2$$

qui est toujours strictement négatif pour $\mu \in [0, 1/4]$. Par continuité, il existe $\epsilon_2 < \epsilon_1$ tel que g est décroissante sur $[1-\mu-\epsilon_2, 1-\mu]$ avec $g(1-\mu) = 0$. Pour x dans cet intervalle, $g(x) \geq 0$ et $x''y'' \geq (1-\mu)^2$.

Pour conclure, l'équation (7.4) montre que pour ϵ suffisamment petit, la condition $x > 1-\mu-\epsilon_2$ est satisfaite ; par suite, le point $T_2 \circ T_1(x,y)$ appartient au domaine $\mathcal{D}_{1,\mu}$. Une étape étant au plus nécessaire pour quitter $\mathcal{D}_{1,\mu}$, 3 étapes sont au plus nécessaires à (x,y) pour atteindre le trou.

Le cas $T_1 \circ T_2$ est complètement symétrique ce qui achève de prouver le lemme.

□

Description du domaine $[K_{1,\mu} = k]$. La proposition qui suit décrit pour chaque entier k, l'ensemble $[K_{1,\mu} = k]$ des entrées sur lesquelles l'algorithme $\texttt{M2}(1,\mu)$ effectue exactement k itérations.

Proposition 7.13. *Il existe deux constantes positives $\alpha = \alpha(\mu)$ et $\beta = \beta(\mu)$ telles que le domaine $[K_{1,\mu} = k]$ où l'algorithme $\texttt{M2}(1,\mu)$ effectue k itérations vérifie*

$$[K_{1,\mu} = k] \subset \{(x,y); \quad \alpha \cdot \mu^k \leq xy \leq \beta \cdot \mu^k\}$$

Démonstration. Considérons t_μ défini dans le lemme 7.12. Alors $\mathcal{O}^2_{t_\mu,\mu} \subset \mathcal{D}_{t_\mu,\mu}$ et selon la proposition 7.9, il existe deux constantes $\alpha'(\mu) := \alpha(t_\mu,\mu) > 0$ et $\beta'(\mu) := (t_\mu,\mu) > 0$ pour lesquelles le domaine $[K_{t_\mu,\mu} = k]$ vérifie

$$[K_{t_\mu,\mu} = k] \subset \{xy; \quad \alpha'(\mu)\mu^k \leq xy \leq \beta'(\mu)\mu^k\}.$$

Le lemme 7.12 prouve l'inclusion $[K_{1,\mu} = k] \subset [K_{t_\mu,\mu} = k+3]$ et donc l'inclusion

$$[K_{1,\mu} = k] \subset \{xy; \quad \alpha(\mu)\mu^k \leq xy \leq \beta(\mu)\mu^k\},$$

avec $\alpha(\mu) := \mu^3\alpha'(\mu)$ et $\beta(\mu) = (1/\mu^3)\beta'(\mu)$. Ceci complète la preuve. □

Nombre d'itérations de $\mathtt{M2}(t,\mu)$ sur des entrées difficiles. La proposition suivante est une conséquence immédiate de la proposition 7.13 et est l'équivalent en dimension 2 du corollaire 7.3

Proposition 7.14. *Pour $t \geq 1$ et $\mu \leq 1/4$, le nombre d'itérations $K_{t,\mu}$ du système dynamique $\mathtt{M2}(t,\mu)$ satisfait*

$$K_{t,\mu}(x,y) = \log_\mu xy + O(1) \quad \textit{lorsque } xy \to 0$$

où la constante du O est uniforme pour $t \to 1$.

Loi du nombre d'itérations. Pour $r > -1$ et $s > -1$, le domaine $[0, 1/(4\mu)]^2$ est muni de la loi puissance bivariée $g_{r,s}(x,y)$ définie par

$$g_{r,s}(x,y) = c_{r,s} x^r y^s \qquad \text{avec} \qquad c_{r,s} = (4\mu)^{r+s+2}(r+1)(s+1),$$

Un simple calcul intégral donne le résultat suivant pour R proche de 0,

$$\mathbb{P}\left[xy \leq R\right] = \begin{cases} (4\mu)^{2s+2} \dfrac{r+1}{r-s} R^{s+1} \left(1 + O\left(R^{r-s}\right)\right) & \text{si } r > s \\ (4\mu)^{2r+2} \dfrac{s+1}{s-r} R^{r+1} \left(1 + O\left(R^{s-r}\right)\right) & \text{si } r < s \\ (4\mu)^{2r+2} (r+1) R^{r+1} \left|\ln R\right| \left(1 + O\left(\left|\ln R\right|^{-1}\right)\right) & \text{si } r = s \end{cases}$$

La proposition 7.13 montre aussi qu'il existe deux réels α et β pour lesquels le domaine $[K_{t,\mu} > k]$ vérifie

$$\{(x,y) | xy \leq \alpha\mu^k\} \subset [K_{t,\mu} \geq k] \subset \{(x,y) xy \leq \beta\mu^k\}.$$

Nous en déduisons la proposition suivante.

Proposition 7.15. *Supposons que pour $r > -1$ et $s > -1$, le domaine $[0, 1/(4\mu)]^2$ est muni de la distribution puissance dont la densité est donnée par*

$$g_{r,s}(x,y) = c_{r,s} x^r y^s, \qquad \textit{avec} \quad c_{r,s} = (4\mu)^{r+s+2}(r+1)(s+1).$$

Alors, pour $t \geq 1$ et pour $\mu \leq 1/4$, le nombre d'itérations $K_{t,\mu}$ du modèle $\mathtt{M2}(t,\mu)$ vérifie pour $k \to \infty$

$$\mathbb{P}\left[K_{t,\mu} > k\right] = \begin{cases} \Theta\left(\mu^{k(s+1)}\right) & \textit{si } r > s \\ \Theta\left(\mu^{k(r+1)}\right) & \textit{si } r < s \\ \Theta\left(k\mu^{k(r+1)}\right) & \textit{si } r = s \end{cases}$$

Le nombre d'itérations $K_{t,\mu}$ suit une loi asymptotiquement géométrique dès que les deux valuations r et s sont différentes.

Cette proposition exhibe une différence de comportement selon que les valuations sont choisies égales ou non. On obtient une loi géométrique lorsque les valuations r et s sont différentes, et la raison de cette loi ne dépend que de la valuation minimale –mais la loi n'est plus strictement géométrique quand les valuations r et s sont égales, et l'algorithme effectue plus d'itérations dans ce cas. Nous ne comprenons pas bien ce phénomène.

7.4.3 Étude des trajectoires autour du point fixe

Nous avons déjà vu qu'il existe un voisinage du point fixe dans lequel le système effectue un nombre borné d'itérations en alternant à chaque fois les applications T_1 et T_2. Une autre manière d'obtenir ce résultat est d'utiliser la forme de Jordan de $T_2 \circ T_1$ (ou $T_1 \circ T_2$). La figure 7.8 donne l'exemple d'une trajectoire. On constate l'alternance de T_1 et T_2 juste avant d'atteindre le trou. En regardant un point sur 2, on constate que $T_2 \circ T_1$ a tendance à s'approcher du point fixe tout en tournant autour. Cela suggère que la forme de Jordan de la matrice jacobienne de $T_2 \circ T_1$ est probablement une homothétie-rotation (forme (d) du théorème 7.7).

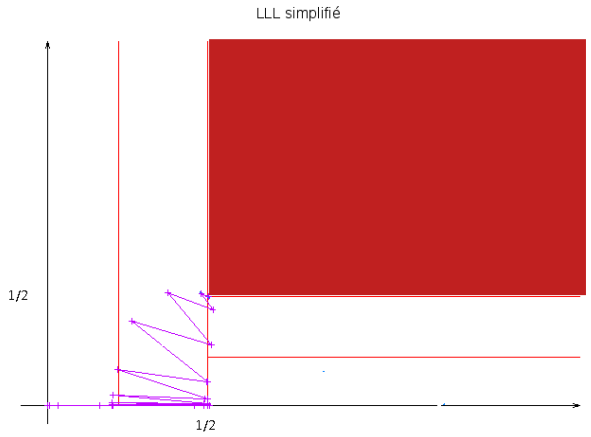


FIGURE 7.8: Trajectoire d'un point pour la stratégie gloutonne

Nous avons

$$\mathrm{Jac}_{T_1}(1-\mu,1-\mu) = \begin{pmatrix} 2\mu-1 & 0 \\ 1-\mu & 1 \end{pmatrix} \quad \text{et} \quad \mathrm{Jac}_{T_2}(1-\mu,1-\mu) = \begin{pmatrix} 1 & 1-\mu \\ 0 & 2\mu-1 \end{pmatrix}$$

soit

$$\begin{aligned} A \underset{def}{:=} \mathrm{Jac}_{T_2\circ T_1}(1-\mu,1-\mu) &= \mathrm{Jac}_{T_2}(T_1(1-\mu,1-\mu))\mathrm{Jac}_{T_1}(1-\mu,1-\mu) \\ &= \begin{pmatrix} 1 & 1-\mu \\ 0 & 2\mu-1 \end{pmatrix} \begin{pmatrix} 2\mu-1 & 0 \\ 1-\mu & 1 \end{pmatrix} \\ &= \begin{pmatrix} \mu^2 & 1-\mu \\ (1-\mu)(2\mu-1) & 2\mu-1. \end{pmatrix} \end{aligned}$$

La trace, le déterminant et le discriminant satisfont respectivement

$$\mathrm{tr}A = \mu^2 + 2\mu - 1, \quad \det A = (2\mu-1)^2 \quad \Delta(A) = (\mu-1)^2((\mu+3)^2 - 12) < 0.$$

La matrice A vérifie les propriétés (d) du théorème 7.7. Le rapport de l'homothétie est donnée par $\rho = \sqrt{\det A} = 1 - 2\mu$ et l'angle de la rotation vérifie

$$\cos\alpha = \frac{\mathrm{tr}A}{2\rho} = \frac{\mu^2 + 2\mu - 1}{2(1-2\mu)}.$$

C'est un angle négatif qui montre que la trajectoire tend vers le point fixe en tournant dans le sens horaire. Pour montrer que le système dynamique s'arrête, il est possible de calculer les angles de la rotation et du trou (après changement de base) et de montrer que l'angle du second est plus grand que l'angle du premier. Cette méthode est très calculatoire et n'apporte rien de plus que ce qui est déjà connu. Nous ne la présentons donc pas ici.

7.5 Étude du système dynamique associé à LLL en dimension d

Nombre d'itérations. Les méthodes utilisées dans les analyses du nombre d'itérations pour $d = 2$ et $d = 3$ sont très similaires et nous pensons que l'approche peut être adaptée à une dimension d quelconque. Cette section ne contient pas de preuve mais nous expliquons comment il est peut-être possible de généraliser le résultat.

A chaque étape, le système dynamique applique une des transformations T_i ($i \leq d-1$) et le processus s'arrête lorsque le trou $\mathcal{O}_{t,\mu}^{d-1}$ est atteint.

La quantité qui augmente. Dans les cas $d = 2$ et $d = 3$, chaque transformation avait pour effet d'augmenter une certaine quantité qui était x_1 pour $d = 2$ et x_1x_2 pour $d = 3$. Cette quantité correspond à l'énergie $\exp[-2\mathcal{E}(\mathbf{c})]$ du *cfg* correspondant (voir définition 5.3) dans le modèle `M1` et est aussi liée à la notion de potentiel $D(B)$ (voir formule 2.18) pour l'algorithme LLL. Une extension naturelle de la "quantité qui augmente" en dimension supérieure est reliée à l'énergie du *cfg* associé, et aussi au coefficient de réduction du réseau

$$P(\mathbf{x}) := \prod_{i=1}^{d-1} x_i^{i(d-i)} = \exp[-2\mathcal{E}(\mathbf{c}) \log s] = \prod_{i=1}^{d-1} r_i^2,$$

où r_i est le i^{e} rapport de Siegel du réseau associé. Notons que pour $d = 3$, la quantité $P(\mathbf{x}) = x_1^2x_2^2$ est le carré de la grandeur $xy = x_1x_2$ qui décroît de la section 7.4. Ceci explique l'apparition de facteurs $1/2$ ou de carrés dans certains résultats qui généralisent ceux de la section 7.4.

Si la transformation T_i est appliquée, on a

$$P(T_i(\mathbf{x})) = \frac{P(\mathbf{x})}{(x_i + \mu)^2} > P(\mathbf{x}),$$

et lorsque T est choisie suivant la stratégie gloutonne, on obtient

$$\frac{P(\mathbf{x})}{P(T(\mathbf{x}))} = \min(x_i + \mu)^2.$$

Le domaine où il reste peu d'itérations. La définition du domaine $\mathcal{D}_{t,\mu}$ est immédiate et dépend naturellement du potentiel. Précisément, nous posons

$$\mathcal{D}_{t,\mu} = \left\{ \mathbf{x} = (x_1, \ldots, x_{d-1}) \in I_\mu^{d-1} : P(\mathbf{x}) \geq (\frac{1}{t^2} - \mu)^{d(d^2-1)/6} \right\}.$$

Le domaine $\mathcal{D}_{t,\mu}$ est stable par n'importe quelle transformation et il contient strictement le point fixe du système dynamique $\mathtt{M2}(1, \mu)$. En utilisant le même type d'arguments que pour les dimensions $d = 2$ et $d = 3$, nous obtenons le résultat suivant.

Proposition 7.16. *Désignons par $K_{t,\mu}(\mathbf{x})$ le nombre d'itérations du système dynamique $\mathtt{M2}(t, \mu)$ pour atteindre le trou $\mathcal{O}_{t,\mu}^{d-1}$ avec la stratégie gloutonne. Alors, il existe $\alpha = \alpha(d, t, \mu) > 0$ et $\beta = \beta(d, t, \mu) > 0$ tels que le domaine $[K_{t,\mu}(\mathbf{x}) = k]$ vérifie pour tout $k \geq 0$,*

$$[K_{t,\mu}(\mathbf{x}) = k] \subset \{\mathbf{x} \in I_\mu^{d-1}; \quad \alpha\mu^{2k} \leq P(\mathbf{x}) \leq \beta\mu^{2k}\}.$$

Démonstration. Soit $\mathbf{x}_0 \notin \mathcal{O}_{t,\mu}^{d-1}$. On pose $\mathbf{x}_{i+1} = T(\mathbf{x}_i)$ quand T est la transformation du système $\mathtt{M2}(t,\mu)$ choisie dans la stratégie gloutonne. Alors on a l'égalité

$$\frac{P(\mathbf{x}_i)}{P(\mathbf{x}_{i+1})} = \min\{(x_{i,j}+\mu)^2; \quad \mathbf{x}_i = (x_{i,1},\dots,x_{i,d-1})\} \tag{7.5}$$

qui permet de montrer

$$\frac{P(\mathbf{x}_i)}{P(\mathbf{x}_{i+1})} \le \left(\frac{1}{t^2}\right)^2, \quad \text{et par induction} \quad P(\mathbf{x}_i) \le t^{-4(k-i)} P(\mathbf{x}_k) \le t^{-4(k-i)} \left(\frac{1}{4\mu}\right)^{(1/6)d(d-1)^2}.$$

De plus, $P(\mathbf{x}_i) \ge \min\{x_{i,j} : \mathbf{x}_i = (x_{i,1},\dots,x_{i,d-1})\}^{d(d-1)^2/6}$ soit pour tout $i \in [1\,..\,k]$,

$$\min\{x_{i,j} : \mathbf{x}_i = (x_{i,1},\dots,x_{i,d-1})\} \le \frac{1}{4\mu} t^{-24\frac{k-i}{d(d-1)^2}}.$$

Utilisant une fois de plus l'égalité (7.5) on en déduit l'encadrement

$$\mu^2 \le \frac{P(\mathbf{x}_i)}{P(\mathbf{x}_{i+1})} \le \mu^2 \left(1 + \frac{1}{4\mu^2} t^{-24\frac{k-i}{d(d-1)^2}}\right)^2$$

et le produit terme à terme donne

$$\mu^{2k} \le \frac{P(\mathbf{x}_0)}{P(\mathbf{x}_k)} \le \mu^{2k} \prod_{i=1}^{k} \left[1 + \frac{1}{4\mu^2}\left(t^{-24/(d(d-1)^2)}\right)^i\right]^2$$

Comme $\mathbf{x}_k \in \mathcal{O}_{t,\mu}^{d-1}$, nous avons

$$\left(\frac{1}{t^2} - \mu\right)^{d(d-1)^2/6} \le P(\mathbf{x}_k) \le \left(\frac{1}{4\mu}\right)^{d(d-1)^2/6}$$

et nous obtenons le lemme avec

$$\alpha = \left(\frac{1}{t^2} - \mu\right)^{d(d-1)^2/6} \qquad \text{et} \qquad \beta = \left[\left(\frac{1}{4\mu}\right)^{d(d-1)^2/6}\right]\left[\prod_{i=1}^{+\infty}\left(1 + \frac{1}{4\mu^2} t^{-24\frac{i}{d(d-1)^2}}\right)\right]^2.$$

□

Avec cette proposition, nous pouvons évaluer la probabilité que l'algorithme $\mathtt{M2}(t,\mu)$ effectue au moins k itérations par

$$\mathbb{P}\left[P(\mathbf{x}) \le \alpha\mu^{2k}\right] \le \mathbb{P}\left[K_{t,\mu}(\mathbf{x}) \ge k\right] \le \mathbb{P}\left[P(\mathbf{x}) \le \beta\mu^{2k}\right].$$

Avec une densité à valuation, nous sommes ramenés au calcul d'une intégrale multiple avec $d-1$ paramètres. Malheureusement, nous ne sommes pas arrivés à bout de ce calcul même si très probablement nous aurions une fois de plus obtenu des lois géométriques.

Mais ici, et contrairement à ce qui se passait pour les dimensions $d = 2$ et $d = 3$, nous ne sommes pas parvenus à démontrer qu'il existe $t > 1$ pour lequel le nombre d'itérations pour atteindre le trou $\mathcal{O}_{1,\mu}^{d-1}$ à partir du trou $\mathcal{O}_{t,\mu}^{d-1}$ est borné. En dimension $d = 2$, il n'y avait qu'une transformation. Pour $d = 3$, il était nécessaire de considérer $T_1 \circ T_2$. Pour une dimension d quelconque, il faut peut-être étudier "toutes" les combinaisons de ces $d-1$ transformations, ce qui augmente la combinatoire... Toutefois, nous pensons qu'un résultat similaire existe et nous proposons la conjecture suivante : [2]

2. cette conjecture peut paraître "osée" mais elle est due à l'échelle générale du problème, qui est cubique.

Conjecture 7.17. *Il existe η, tel que, pour tout $d \geq 3$ et tout $\mu \in]0, 1/4]$, il existe $t_\mu > 1$ pour lequel, pour tout $t < t_\mu$, le système $\mathtt{M2}(1, \mu)$ en dimension d effectue au plus $O(d^\eta)$ itérations sur le trou $\mathcal{O}_{t,\mu}^{d-1}$.*

On peut montrer que si $t' > t > 1$, alors le nombre d'itérations pour passer de $\mathcal{D}_{t',\mu}$ à $\mathcal{D}_{t,\mu}$ est une fonction cubique de d. Cela nous laisse penser que le nombre d'itérations nécessaire pour passer de $\mathcal{O}_{t,\mu}^{d-1}$ à $\mathcal{O}_{1,\mu}^{d-1}$ est au moins cubique en d.

La conjecture précédente et le lemme 7.16 conduisent ensemble à la généralisation de la proposition 7.13 au cas d'une dimension d quelconque.

Proposition 7.18. *Sous la conjecture 7.17, pour toute dimension d et pour tout $\mu \in]0, 1/4]$, il existe deux constantes positives $\alpha = \alpha(d, \mu)$ et $\beta = \beta(d, \mu)$ telles que, pour tout $t \geq 1$, le domaine $[K_{t,\mu}^{(d)} = k]$ formé des entrées pour lesquelles le modèle $\mathtt{M2}(t, \mu)$ effectue k itérations en dimension d satisfait*

$$[K_{t,\mu}^{(d)} = k] \subset \left\{ \mathbf{x} \in I_\mu^{d-1} \;\mid\; \alpha \cdot \mu^{2k+O(d^\eta)} \leq P(\mathbf{x}) \leq \beta \cdot \mu^{2k+O(d^\eta)} \right\}.$$

Le corollaire suivant est une conséquence immédiate de la proposition 7.18 et décrit précisément le nombre d'itérations en dimension d quelconque.

Corollaire 7.19. *Si $K(\mathbf{x})$ désigne le nombre d'itérations du système dynamique sur l'entrée $\mathbf{x}$, sous l'hypothèse de la conjecture 7.17, nous avons*

$$K(\mathbf{x}) = \frac{1}{2} \log_\mu P(\mathbf{x}) + O(d^\eta) \quad \textit{lorsque } P(\mathbf{x}) \to 0.$$

Notez la présence du $1/2$ qui n'existe pas dans la proposition 7.14 et qui est due à la définition de $P(\mathbf{x})$ pour $d = 3$ ($P(x, y) = x^2 y^2$). Nous appliquons maintenant le corollaire 7.19 aux modèles d'entrée décrits dans le chapitre 6. Le nombre d'itérations pour chacun des modèles est donné par le tableau 7.1 suivant :

Modèles	K **pour** $\mathtt{M2}(1, \mu)$
$\mathcal{A}(\Upsilon, d, g)$	$K = \left\lvert \dfrac{\log s}{\log \mu} \right\rvert d^2 \Upsilon I(g) + O(d^\eta)$
$\mathcal{K}(\Upsilon, d)$	$K = \left\lvert \dfrac{\log s}{\log \mu} \right\rvert (d-1)\Upsilon + O(d^\eta)$
$\mathcal{U}(\Upsilon, d, \beta)$	$K = \left\lvert \dfrac{\log s}{\log \mu} \right\rvert d^2 (1-\beta)\beta\Upsilon + O(d^\eta)$

TABLE 7.1: Asymptotique du nombre d'itérations K dans le cas de modèle $\mathtt{M2}$. On suppose $\Upsilon/d^2 \to \infty$ pour le modèle $\mathcal{K}$ et $\Upsilon/d \to \infty$ dans les deux autres modèles.

Si on admet que $\eta = 3$, alors l'asymptotique n'est obtenue pour $\mathcal{K}(\Upsilon, d)$ que lorsque $\Upsilon/d^2 \to \infty$. Elle n'est obtenue dans les autres cas que si $\Upsilon/d \to \infty$. La complexité semble dépendre de s, mais c'est une fausse impression car nous rappelons que nous avons posé

$$\Upsilon = \log_s \prod_{i=1}^{d-1} r_i$$

où r_i sont les rapports de Siegel. Ici, il est plus clair de travailler avec la version standard

$$\widetilde{\Upsilon} := \log \prod_{i=1}^{d-1} r_i$$

et dans ce cas, on a autre tableau (table 7.2).

Modèles	K **pour** $\mathtt{M2}(1,\mu)$
$\mathcal{A}(\Upsilon,d,g)$	$K = \dfrac{1}{\lvert\log\mu\rvert} d^2\,\widetilde{\Upsilon}\,I(g) + O(d^\eta)$
$\mathcal{K}(\Upsilon,d)$	$K = \dfrac{1}{\lvert\log\mu\rvert} (d-1)\,\widetilde{\Upsilon} + O(d^\eta)$
$\mathcal{U}(\Upsilon,d,\beta)$	$K = \dfrac{1}{\lvert\log\mu\rvert} d^2\,(1-\beta)\beta\,\widetilde{\Upsilon} + O(d^\eta)$

TABLE 7.2: Asymptotique du nombre d'itérations K dans le cas de modèle M2. On suppose $\Upsilon/d^2 \to \infty$ pour le modèle $\mathcal{K}$ et $\Upsilon/d \to \infty$ dans les deux autres modèles.

Configuration de sortie. On sait que les rapports de Siegel $r_i = \sqrt{x_i}$ vérifient à la fin de l'algorithme $\mathtt{M2}(1,\mu)$, $r_i \geq \sqrt{1-\mu}$. Le défaut d'Hermite est alors donné par la proposition suivante.

Proposition 7.20. *Considérons le modèle d'exécution* $\mathtt{M2}(1,\mu)$. *Soit* $\mathcal{L} = \mathcal{L}(B)$ *un réseau engendré par une base* B. *Après l'exécution du modèle* $\mathtt{M2}(1,\mu)$, *la configuration de sortie vérifie*

$$\gamma(B) \leq \left(\frac{1}{1-\mu}\right)^{\frac{d-1}{2}}.$$

Nous n'avons pas étudié la distribution de la configuration de sortie, en particulier quand on considère une distribution d'entrées à valuation. Ce serait une étude intéressante à faire, et à comparer avec les travaux de Vallée et Vera.

7.6 Conclusion

Après avoir étudié le modèle "Tas de sable" au chapitre 5, nous nous sommes intéressés au modèle plus complexe M2, qui est aussi plus proche de l'algorithme LLL. Nous sommes parvenus à démontrer des résultats équivalents à ceux connus sur l'algorithme de Gauss dans le modèle $\mathtt{M2}(t,\mu)$ pour $t \geq 1$. Concernant l'algorithme LLL(t) avec la stratégie gloutonne, nous avons exhibé la loi du nombre d'itérations du modèle $\mathtt{M2}(t,\mu)$ associé pour toutes les dimensions et pour $t > 1$, ce résultat n'étant pas connu pour LLL. Pour $t = 1$, une analyse complète dans le cas $d = 3$ a été réalisée. Sauf cas particulier pour la densité initiale, le nombre d'étapes suit toujours une loi géométrique.

Notons que, sous réserve d'une conjecture, la méthode employée peut s'adapter à toutes les dimensions.

Chapitre 8

Modèles simplifiés et réalité algorithmique

Sommaire

C'est un chapitre expérimental qui vise à confronter nos modélisations simplifiées avec la réalité algorithmique. Dans le cadre du modèle `M1`, nous avons prouvé que la géométrie de la configuration de sortie est largement indépendante du modèle d'entrée, pourvu qu'il donne lieu à un cfg d'énergie initiale suffisamment grande. Par contre, l'exécution de l'algorithme (et en particulier son nombre d'itérations) dépend largement du modèle d'entrée. Les expériences de Gama, Nguyen et Stehlé [34, 88] montrent que ces phénomènes apparaissent aussi dans la réalité algorithmique, qui travaille avec le modèle de l'algorithme LLL dit modèle `M5`. Ces travaux fournissent une preuve expérimentale indirecte de la légitimité de nos modèles simplifiés. Mais nous nous posons ici ces questions de "légitimité" directement. Les hypothèses de simplification que nous avons faites dépendent du comportement du facteur de décroissance ρ, de sa contrepartie additive, qui est le décrément $\alpha := -\log_s \rho$. Ce chapitre est principalement une étude expérimentale de ce décrément α.

C'est aussi un chapitre expérimental qui veut comparer les modèles :

– les modèles d'exécution, au nombre de cinq (voir le chapitre 4) : les modèles `M1`, `M2`, `M3`, `M4`, `M5`. Il faut choisir aussi, pour les deux premiers modèles, des paramètres : pour le modèle `M1`,

décrit au chapitre 5, il faut choisir le paramètre α, qui joue le rôle de décrément ; pour le modèle `M2`, décrit au chapitre 7, il faut choisir le paramètre μ qui joue le rôle du carré du coefficient sous-diagonal courant.

– les modèles d'entrée (voir chapitre 6, au nombre de trois [1] : le modèle d'Ajtai, désigné par $\mathcal{A}$, le modèle de NTRU, désigné par $\mathcal{N}$ et le modèle du sac-à dos, désigné par $\mathcal{K}$ (Knapsack). Chacun des modèles d'entrée $\mathcal{N}$ ou $\mathcal{K}$ est un *cfg* unitas, déterminé par sa masse Υ, sa dimension d, et la position de son unique pile $m = 1$ pour le modèle $\mathcal{K}$ et $m = d/2$ pour le modèle $\mathcal{N}$. Le modèle $\mathcal{A}$ est un modèle qui est aussi défini par sa masse Υ et sa dimension d, et nous considérons le cas où la hauteur des piles suit une loi puissance, de telle manière que la masse moyenne soit égale à Υ. Dans les expérimentations, la masse est donnée dans la base 2 et notée Υ_2 ($\Upsilon = \log_s 2 \cdot \Upsilon_2 \approx 4.4\Upsilon_2$).

En fait, nous ferons la plupart des expérimentations dans les modèles d'exécution "réalistes"– à savoir les véritables algorithmes `M4` et `M5`–, et nous ferons varier les modèles d'entrée dans $\{\mathcal{A}, \mathcal{N}, \mathcal{K}\}$. Nous aurons donc le plus souvent six couples (modèle d'entrée, modèle d'exécution) pour nos expérimentations.

A la fin de ce chapitre, nous tenterons une comparaison de tous les modèles d'exécution entre eux, vis-à-vis du nombre d'itérations.

8.1 Les principales questions.

8.1.1 Description d'une exécution

Une exécution d'un algorithme de réduction sur une entrée B se décrit avec les principaux paramètres suivants :

- la dimension d,
- le nombre d'itérations K,
- la marche aléatoire, qui est le graphe de la fonction $P : [1\,..\,K] \to [1\,..\,d-1]$ qui associe au temps $j \in [1\,..\,K]$ la position $i = P(j)$, c'est-à -dire l'indice de la boîte (ou l'indice de la pile du cfg) qu'on réduit (voir la figure 8.1),
- le graphe de l'application $\alpha : [1\,..\,K] \to \mathbb{R}^+$ qui, au temps j associe la valeur $\alpha(j)$ du décrément au temps j (voir la figure 8.2),
- le graphe de l'application $\nu : [1\,..\,K] \to [-1/2, +1/2]$ qui au temps j associe la valeur $\nu(j)$ de la partie fractionnaire du coefficient sous-diagonal $m_{i+1,i}$ quand la position $P(j)$ est égale à i.

8.1.2 Les principales questions.

A un instant j de l'exécution, où la position $P(j)$ est égale à i, le décrément α se calcule en fonction de la hauteur de pile c_i et du coefficient sous-diagonal $m_{i+1,i}$ comme suit,

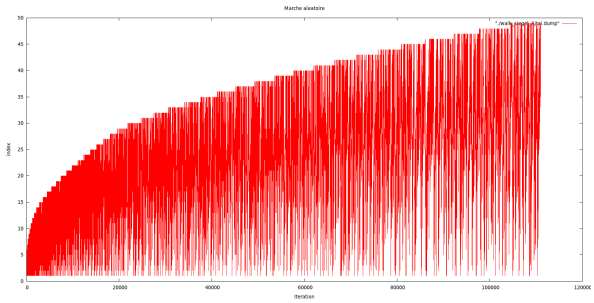
$$\alpha := -\frac{1}{2}\log_s \left[\left(\frac{1}{s^2}\right)^{c_i} + \lfloor m_{i+1,i}^2 \rceil\right],$$

et la valeur s du paramètre usuellement utilisé est

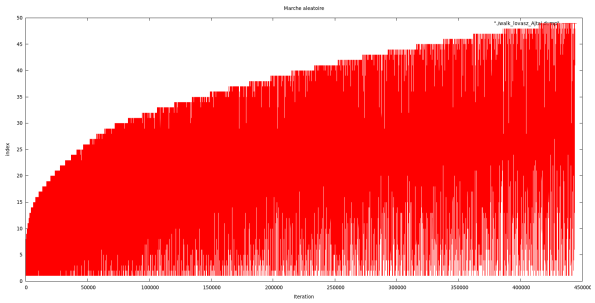
$$s = \sqrt{\frac{1}{\delta - \eta^2}}, \qquad \text{avec} \quad (\delta, \eta) = (0.99, 0.51), \quad s \approx 1.17.$$

On considèrera aussi la valeur limite $s_0 = 2/\sqrt{3}$.

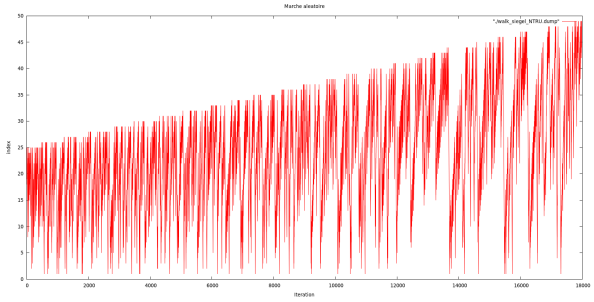
1. Nous ne faisons pas d'expérimentations supplémentaires dans le modèle de Coppersmith.



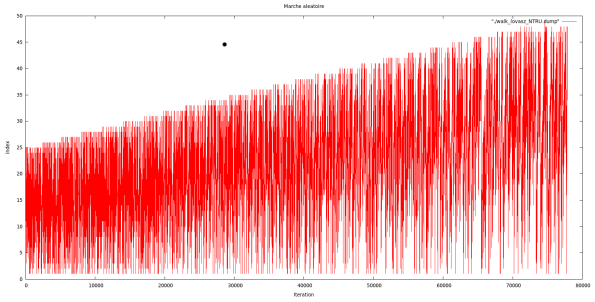
$[\mathcal{A}, \Upsilon_2 = 50, \texttt{M4}]$



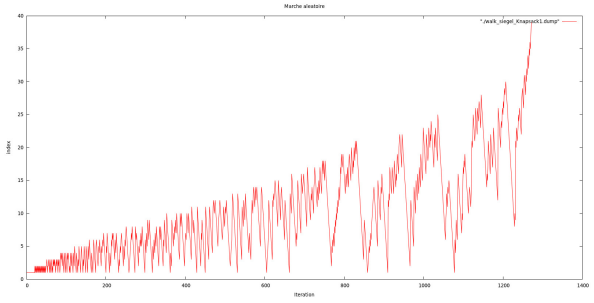
$[\mathcal{A}, \Upsilon_2 = 50, \texttt{M5}]$



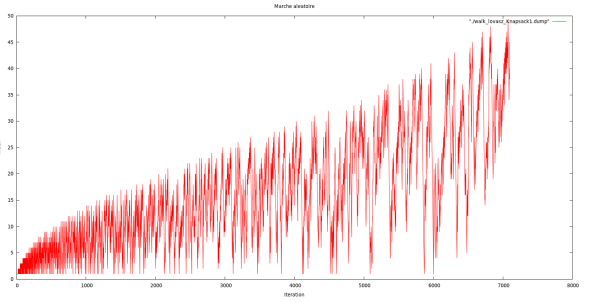
$[\mathcal{N}, \Upsilon_2 = 50, \texttt{M4}]$



$[\mathcal{N}, \Upsilon_2 = 50, \texttt{M5}]$

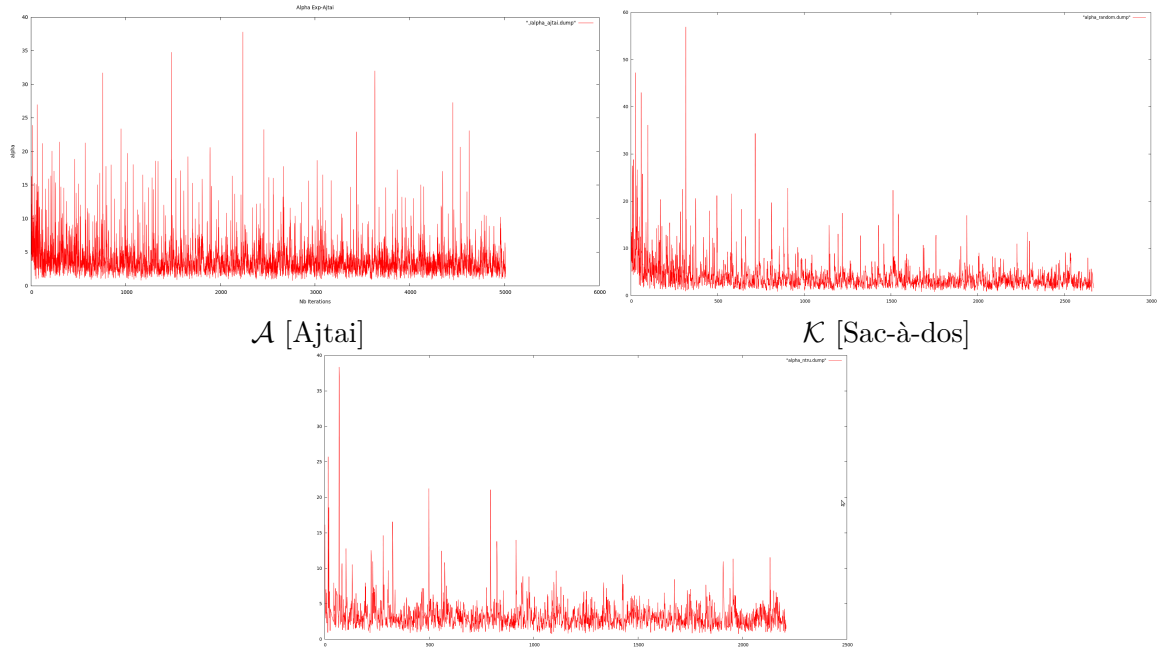


$[\mathcal{K}, \Upsilon_2 = 100, \texttt{M4}]$.



$[\mathcal{K}, \Upsilon_2 = 100, \texttt{M5}]$

FIGURE 8.1: La marche aléatoire (L'indice de réduction en fonction du temps de l'exécution) en dimension $d = 50$, pour chacun des tros modèles d'entrée, et chacun des deux modèles d'exécution `M4` et `M5`.



$\mathcal{A}$ [Ajtai] $\mathcal{K}$ [Sac-à-dos]

$\mathcal{N}$ [NTRU]

FIGURE 8.2: L'évolution du décrément α lors d'une exécution de l'algorithme de Siegel (modèle `M4`) dans chacun des trois modèles d'entrée. La dimension d est égale à 20 et la masse Υ_2 est égale à 50.

Distribution du paramètre ν. Le paramètre ν est une fonction $j \mapsto \nu(j)$ qui associe au temps j la valeur $\nu(j)$ de la partie fractionnaire du coefficient sous-diagonal $m_{i+1,i}$ quand la position $P(j)$ est égale à i. Ce paramètre est sans doute très difficile à modéliser, mais nous avons fait l'hypothèse qu'il variait de manière aléatoire et indépendante de la hauteur c_i de la pile courante.

Cette hypothèse est-elle légitime ? Que peut-on dire de la distribution du paramètre ν ? Cette distribution de ν varie-t-elle au cours du temps ? dépend-elle de la position i où ν est calculé ?

Distribution du paramètre α. On se pose des questions similaires pour le décrément α.

Que peut-on dire de la distribution du paramètre α ? Cette distribution dépend-elle de la dimension ? de la masse Υ de l'entrée ? de la position i où il est calculé ? du moment j de l'exécution où il est calculé ?

Évolution du décrément α en fonction du temps. C'est certainement la question la plus naturelle à propos de ce décrément. Nous avons déjà abordé cette question au Chapitre 4, et nous résumons ici le comportement que α doit avoir au cours de l'exécution.

Au début de l'exécution. Quand le réseau n'est pas de tout réduit, la hauteur c de la pile courante est très grande et c'est ν qui domine. Si on suppose que ν est distribué uniformément dans l'intervalle $[-1/2, +1/2]$, alors la valeur moyenne du décrément au début de l'algorithme vaut

$$-\frac{2}{\log s_0}\int_0^{1/2} \log x dx = \frac{1}{\log s_0}(\log 2 + 1) \approx 11.7.$$

À la fin de l'exécution. La valeur de c est proche de 1, et si on suppose toujours que ν est distribué uniformément dans l'intervalle $[-1/2, +1/2]$, alors la valeur moyenne à la fin de l'algorithme est

$$\frac{-1}{\log s_0}\int_0^{1/2}\log\left(\frac{3}{4}+x^2\right)dx = \frac{1}{\log s_0}\left[-1+\sqrt{3}\arctan\left(\frac{1}{\sqrt{3}}\right)\right] = \frac{1}{\log s_0}\left[-1+\sqrt{3}\frac{\pi}{6}\right] \approx 0.65.$$

Au cours de l'exécution. Plus géneralement, la valeur moyenne du décrément α appliqué à une pile de hauteur c vaut

$$\begin{aligned}&\frac{-1}{\log s_0}\int_0^{1/2}\log\left[\left(\frac{3}{4}\right)^c+x^2\right]dx\\ &\quad=\frac{1}{\log s_0}\left[\log 2+1-\frac{1}{2}\log\left(1+4\left(\frac{3}{4}\right)^c\right)-2\left(\frac{3}{4}\right)^{c/2}\arctan\left(\frac{1}{2}\left(\frac{4}{3}\right)^{c/2}\right)\right].\end{aligned}$$

La valeur moyenne le long d'une exécution. C'est un paramètre fondamental car c'est lui qui intervient dans le calcul du nombre d'itérations (voir l'égalité (4.14)).

Comment α varie-t-il au cours d'une exécution ? Que peut-on dire de sa valeur moyenne $\underline{\alpha}$ le long d'une exécution ?

8.1.3 Principes d'étude.

Discrétisations. Nous commençons par transformer les ensembles variables en des ensembles discrets et fixes. Nous séparons l'intervalle de départ $[1\,..\,K]$ (variable) en quarante intervalles de même longueur, et nous remplaçons l'intervalle $[1\,..\,K]$ par l'intervalle fixe $[1\,..\,40]$. Nous séparons l'intervalle "d'arrivée" $[1\,..\,d-1]$ de la fonction P en vingt sous-intervalles de même longueur, et nous remplaçons l'intervalle $[1\,..\,d-1]$ par l'intervalle fixe $[1\,..\,20]$.

Nous excluons les valeurs de α supérieures à 20, et discretisons le domaine "d'arrivée" $[1, 20]$ de la fonction α en le transformant en l'intervalle discret $[1\,..\,20]$. Nous séparons l'intervalle "d'arrivée" $[-1/2, +1/2]$ de la fonction ν en dix sous -intervalles de longueur $1/10$, et nous remplaçons l'intervalle $[-1/2, +1/2]$ par l'intervalle discret $[1\,..\,10]$.

Densités associées au paramètre ν. Elles sont là pour répondre à la question "le paramètre ν est-il dépendant d'autres paramètres, notamment du temps d'exécution ? de la position ?". Nous travaillons avec la densité q du paramètre ν et certaines de ses densités conditionnelles $q_{(t)}$ et $q^{(x)}$ définies comme suit.

(a) Nous considérons les temps d'exécutions où la valeur discrétisée de ν est proche de $a \in [1\,..\,10]$,

$$\mathcal{J}(a) := \left\{j : \quad \nu(j) \in \left[-\frac{1}{2}+\frac{a-1}{10}, -\frac{1}{2}+\frac{a}{10}\right]\right\}, \quad J(a) := |\mathcal{J}(a)|,$$

et la densité q de ν est définie par

$$q(a) := \frac{J(a)}{K}, \qquad a \in [1\,..\,10].$$

(b) Nous considérons les temps d'exécutions proches d'un temps discret $t \in [1\,..\,40]$,

$$\mathcal{K}_{(t)} := \left\{j : \quad j \in \left[(t-1)\frac{K}{40}, t\frac{K}{40}\right]\right\}, \qquad \text{avec} \quad K_{(t)} = \frac{K}{40},$$

et aussi les temps d'exécution proches d'un temps discret $t \in [1\,..\,40]$, où la valeur discrétisée de ν est proche de a,

$$\mathcal{J}_{(t)}(a) := \left\{ j : j \in \left[(t-1)\frac{K}{40}, t\frac{K}{40} \right], \quad \nu(j) \in \left[-\frac{1}{2} + \frac{a-1}{10}, -\frac{1}{2} + \frac{a}{10} \right] \right\},$$

avec $J_{(t)}(a) := |\mathcal{J}_{(t)}(a)|$, et la densité q de la variable ν "près du temps discret t" est définie par

$$q_{(t)}(a) := \frac{J_{(t)}(a)}{K_{(t)}}.$$

(c) Nous considérons les temps d'exécution j où $P(j)$ proche d'une position discrète x

$$\mathcal{J}^{(x)} := \left\{ j : \quad P(j) \in \left[(x-1)\frac{d-1}{10}, x\frac{d-1}{10} \right] \right\} \qquad J^{(x)} := |\mathcal{J}^{(x)}|$$

et les temps d'exécution j où $P(j)$ proche d'une position discrète x et la valeur discrétisée de ν est proche de a,

$$\mathcal{J}^{(x)}(a) := \left\{ j : \quad P(j) \in \left[(x-1)\frac{d-1}{10}, x\frac{d-1}{10} \right], \quad \nu(j) \in \left[-\frac{1}{2} + \frac{a-1}{10}, -\frac{1}{2} + \frac{a}{10} \right] \right\},$$

avec $J^{(x)}(a) := |\mathcal{J}^{(x)}(a)|$, et la densité q de la variable ν "près de la position discrète x" est définie par

$$q^{(x)}(a) := \frac{J^{(x)}(a)}{J^{(x)}}.$$

Densités associées au paramètre α. Elles sont là pour répondre à la question "le paramètre α est-il dépendant d'autres paramètres, notamment du temps d'exécution ? de la position ? Nous travaillons avec la densité p du paramètre α et certaines de ses densités conditionnelles $p_{(t)}$ et $p^{(x)}$ définies comme suit.

(a') Nous considérons les temps d'exécutions où la valeur de α est proche de $a \in [1\,..\,20]$,

$$\mathcal{K}(a) := \{ j : \quad \alpha(j) \in [a-1, a] \}, \quad \text{avec} \quad K(a) := |\mathcal{K}(a)|,$$

et la densité p de α est définie par

$$p(a) := \frac{K(a)}{K}, \qquad a \in [1\,..\,20].$$

(b') Nous considérons les temps d'exécutions proches d'un temps discret $t \in [1\,..\,40]$,

$$\mathcal{K}_{(t)} := \left\{ j : j \in \left[(t-1)\frac{K}{40}, t\frac{K}{40} \right] \right\}, \qquad \text{avec} \quad K_{(t)} = \frac{K}{40},$$

et aussi les temps d' exécution proches d'un temps discret $t \in [1\,..\,40]$, où la valeur de α est proche de $a \in [1\,..\,20]$,

$$\mathcal{K}_{(t)}(a) := \left\{ j : \quad j \in \left[(t-1)\frac{K}{40}, t\frac{K}{40} \right] \quad \text{et} \quad \alpha(j) \in [a-1, a] \right\}, \qquad \text{avec} \quad K_{(t)}(a) := |\mathcal{K}_t(a)|,$$

et la densité $p_{(t)}$ de α "près du temps discret t" est alors définie par

$$p_{(t)}(a) = \frac{K_{(t)}(a)}{K_{(t)}}.$$

(c') Nous considérons enfin les temps d'exécution pour lesquelles la position est proche de la position discrète x. Là encore, nous transformons l'intervalle $[1\,..\,d-1]$ en un intervalle fixe $[1\,..\,20]$ et pour $x \in [1\,..\,20]$, nous considérons les temps d'exécution pour lesquels la position est proche de la position x

$$\mathcal{K}^{(x)} := \left\{ j : \quad P(j) \in \left[(x-1)\frac{d-1}{20}, x\frac{d-1}{20} \right] \right\}, \qquad \text{avec} \quad K^{(x)} := |\mathcal{K}^{(x)}|$$

et aussi les temps d'exécution pour lesquels la position est proche de la position x et la valeur de α est proche de a,

$$\mathcal{K}^{(x)}(a) := \left\{ j; \quad P(j) \in \left[(x-1)\frac{d-1}{20}, x\frac{d-1}{20} \right], \quad \alpha(j) \in [a-1, a] \right\},$$

avec $K^{(x)}(a) := |\mathcal{K}^{(x)}(a)|$, et la densité $p^{(x)}$ de α "près de la position discrète x" est définie par

$$p^{(x)}(a) = \frac{K^{(x)}(a)}{K^{(x)}}.$$

Évolution de α le long de l'exécution. Nous transformons l'intervalle $[1\,..\,K]$ (variable) en un intervalle fixe $[1\,..\,40]$ et pour $t \in [1\,..\,40]$, nous considérons l'ensemble

$$\mathcal{K}_t := \left\{ j : j \in \left[(t-1)\frac{K}{40}\,..\,t\frac{K}{40} \right] \right\},$$

et nous transformons le paramètre α en un paramètre $\underline{\alpha}$ défini sur $[1\,..\,40]$. Pour $t \in [1\,..\,40]$, on définit $\underline{\alpha}$ comme la moyenne de $\alpha(j)$ quand j varie dans $\mathcal{K}_t$,

$$\underline{\alpha}(t) = \frac{1}{K_t} \sum_{j \in \mathcal{K}_t} \alpha(j), \qquad \text{avec } K_t = \frac{K}{40}.$$

Valeurs moyennes le long d'une exécution. On veut comparer la valeur moyenne $\underline{\alpha}$ du décrément α, calculée sur la totalité de l'exécution, et la comparer à la valeur moyenne $\underline{\alpha}^*$ du décrément α calculée dans le dernier quart de l'exécution. On définit donc

$$\underline{\alpha} := \frac{1}{K} \sum_{j=1}^{K} \alpha(j), \qquad \underline{\alpha}^* := \frac{4}{K} \sum_{j=(3/4)K}^{K} \alpha(j).$$

Densités "moyennisées". Tous les paramètres définis ci-dessus sont des variables aléatoires, et dépendent donc de la base d'entrée B. Par exemple, $K(B)$ est le nombre d'itérations de l'algorithme sur la base B ; pour $j \in [1\,..\,K(B)]$, la notation $\alpha(j)(B)$ désigne la valeur du paramètre α à la j-e itération sur la base B, etc.

Nous effectuons un nombre d'expériences égal à N. Très souvent, nous avons choisi $N = 64$, et nous avons donc choisi N bases d'entrée que nous désignons par $B_1, B_2, \ldots, B_N$. L'indice $n \in [1\,..\,N]$ désigne l'indice de l'expérience, et au lieu de considérer des variables qui dépendent de la base B_n, nous conserverons uniquement l'indice n entre crochet. Par exemple, $K[n]$ désigne le nombre d'itérations sur l'entrée B_n et $\alpha(j)[n]$ est la valeur du paramètre α à la j-e itération sur la base B_n.

Les versions moyennisées des six densités précédentes sont obtenues en remplaçant à chaque fois le numérateur et le dénominateur par leur valeurs cumulatives. On rajoute à chaque fois un chapeau sur ces variables, pour bien faire la différence. On a ainsi :

$$\widehat{p}(a) := \frac{\sum_{n=1}^{N} K(a)[n]}{\sum_{n=1}^{N} K[n]}, \quad \widehat{p}_{(t)}(a) = \frac{\sum_{n=1}^{N} K_{(t)}(a)[n]}{\sum_{n=1}^{N} K_{(t)}[n]}, \quad \widehat{p}^{(x)}(a) = \frac{\sum_{n=1}^{N} K^{(x)}(a)[n]}{\sum_{n=1}^{N} K^{(x)}[n]}$$

$$q(a) := \frac{\sum_{n=1}^{N} J(a)[n]}{\sum_{n=1}^{N} K[n]}, \quad \widehat{q}_{(t)}(a) := \frac{\sum_{n=1}^{N} J_{(t)}(a)[n]}{\sum_{n=1}^{N} K_{(t)}[n]}, \quad \widehat{q}^{(x)}(a) := \frac{\sum_{n=1}^{N} J^{(x)}(a)[n]}{\sum_{n=1}^{N} J^{(x)}[n]}.$$

De même, on remplace la function $t \mapsto \underline{\alpha}$ par sa version moyennisée

$$\widehat{\alpha}(t) = \frac{\sum_{n=1}^{N} \sum_{j \in \mathcal{K}_t[n]} \alpha(j)}{\sum_{n=1}^{N} K_t[n]}.$$

Et on considère aussi les versions "moyennisées" $\widehat{\alpha}$ et $\widehat{\alpha}^*$ des valeurs moyennes $\underline{\alpha}$, calculée sur la totalité de l'exécution, et $\underline{\alpha}^*$, calculée sur le dernier quart,

$$\widehat{\alpha} := \frac{\sum_{n=1}^{N} \sum_{j=1}^{K[n]} \alpha(j)[n]}{\sum_{n=1}^{N} K[n]} \qquad \alpha^* := 4\,\frac{\sum_{n=1}^{N} \sum_{j=(3/4)K[n]}^{K[n]} \alpha(j)[n]}{\sum_{n=1}^{N} K[n]}.$$

8.2 Distribution du coefficient sous-diagonal ν.

Influence du temps sur la distribution de ν. La figure 8.3 décrit la distribution $a \mapsto \widehat{q}_{(t)}(a)$ de la variable ν, au début ($t = 1$), au milieu ($t = 20$) et à la fin ($t = 40$).

Influence de la position sur la distribution de ν. La Figure 8.4 montre la densité $a \mapsto \widehat{q}^{(x)}(a)$ pour quatre valeurs de $x \in [1\,..\,20]$.

Discussion. Ce qui frappe ici, c'est la différence de comportement entre le comprtement de ν dans le modèle `M5` (algorithme de Lovász) et le modèle `M4` (algorithme de Siegel). Pour le modèle `M4`, la densité de ν apparaît à peu près uniforme, excepté pour le modèle $\mathcal{N}$ sur lequel nous reviendrons. Par contre, ce n'est plus le cas dans le modèle `M5`, où il y a une forte concentration des valeurs de ν aux bords de l'intervalle $[-1/2, +1/2]$, et à la fin de l'algorithme `M5`, il n'est plus raisonnable de supposer que ν a une densité uniforme. La distribution de ν suit la forme du domaine fondamental, et il y a une accumulation "dans les coins" du domaine fondamental.

Dans chacun des deux modèles d'exécution, la distribution de ν ne semble pas dépendre de la position.

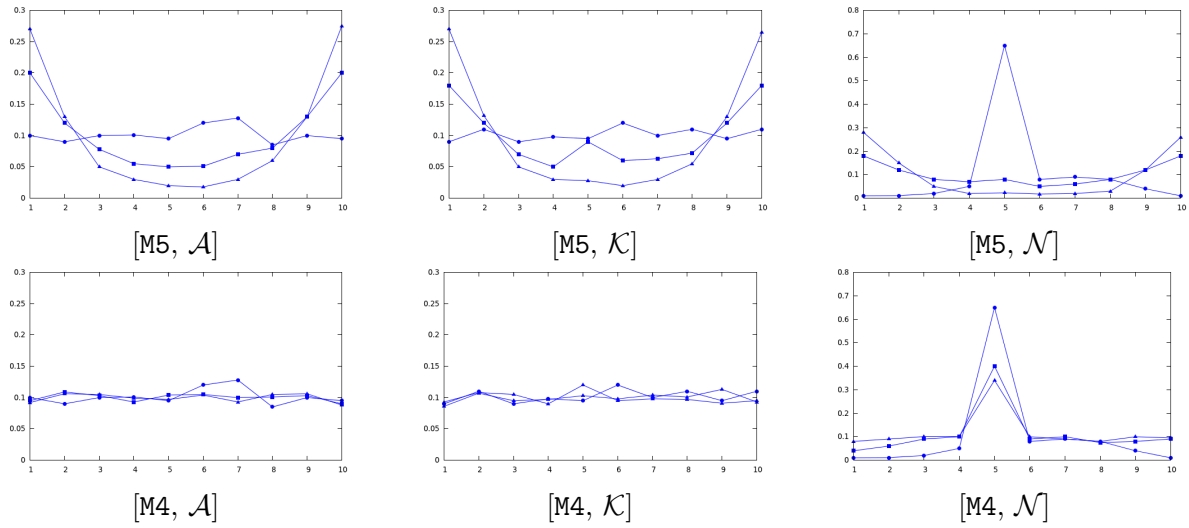


FIGURE 8.3: La distribution $a \mapsto \widehat{q}_{(t)}(a)$ de ν lors de l'exécution : au début $(t = 1, \bullet)$, au milieu $(t = 20, \blacksquare)$ et à la fin $(t = 40, \blacktriangle)$. La dimension d est fixée à $d = 50$ et la masse Υ_2 est fixée à $\Upsilon_2 = 50$.

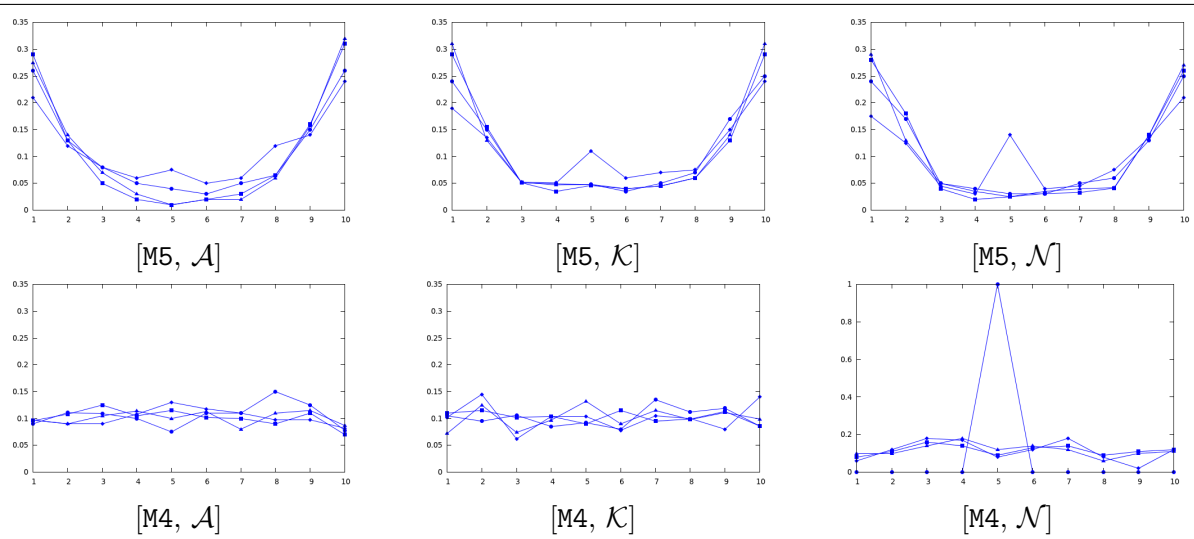


FIGURE 8.4: La distribution $a \mapsto \widehat{q}^{(x)}(a)$ de ν à la position $x = 1$ $(\bullet)$, $x = 8$ $(\blacksquare)$, $x = 14$ $(\blacktriangle)$ et $x = 20$ $(\blacklozenge)$. La dimension d est fixée à $d = 50$ et la masse Υ_2 est fixée à $\Upsilon_2 = 50$.

Les expérimentations de [88] montrent aussi les mêmes phénomènes : une concentration du coefficient ν autour les valeurs $\pm\frac{1}{2}$. Les auteurs observent que, à la sortie, les boîtes $\widehat{B}_i = (\mathbf{b}_i^\star, m_{i+1,i}\mathbf{b}_i^\star + \mathbf{b}_{i+1}^\star)$, ont des propriétés géométriques qui sont largement indépendantes de l'indice i. Dans la Figure 8.5, chaque point (de coordonnées $m_{i+1,i}$ et $r_i = \ell_{i+1}/\ell_i$) correspond à une base locale. La valeur moyenne de $|m_{i+1,i}|$ est proche de 0.38 et $r_i \approx 1/\beta$ avec la même valeur $\beta = 1.04$, qui est la valeur dont nous avons déjà parlé dans le chapitre 5, Figure 5.9).

Ces phénomènes ne semblent pas dépendre du modèle d'entrée. Le pic autour de 0 est en fait un artefact, lié à la forme des matrices d'entrée : la première partie d'une matrice NTRU est une matrice diagonale avec des ν initiaux qui sont donc nuls. Le pic n'existe que pour des positions qui sont "au début", et disparaît au cours de l'exécution.

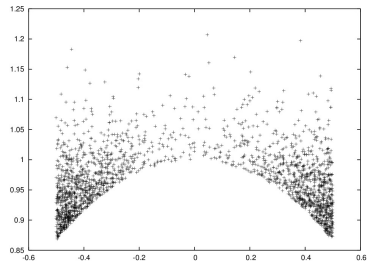


FIGURE 8.5: La distribution de sortie des bases locales

8.3 Distribution du décrément α

Influence de la masse Υ, sur la distribution du décrément α. Dans ces expériences, la dimension d est fixée à $d = 20$ et Υ_2 varie dans l'ensemble $\{5, 20, 100, 200, 500\}$. Pour chaque choix de Υ, pour chaque modèle d'entrée $\mathcal{A}, \mathcal{K}, \mathcal{N}$, et pour chaque modèle d'exécution `M4` (Siegel) ou `M5` (Lovasz), nous avons calculé la densité $\widehat{p}$ de α. Voir Figure 8.6.

Influence de la dimension, sur la distribution du décrément α. Pour ces expériences, on fixe Υ_2 à $\Upsilon_2 = 50$, et on fait varier la dimension d dans l'ensemble $\{10, 20, 30, 40, 50\}$. Pour chaque choix de la dimension d, pour chaque modèle d'entrée $\mathcal{A}, \mathcal{K}, \mathcal{N}$, et pour chaque modèle d'exécution `M4` (Siegel) ou `M5` (Lovasz), nous avons calculé la densité $\widehat{p}$ de α. Voir Figure 8.7.

Influence du temps sur la distribution du décrément α. Nous avons calculé la densité $a \mapsto \widehat{p}_{(t)}(a)$ pour différentes valeurs de t : $t = 1$, $t = 20$ et $t = 40$ .La dimension d est fixée à $d = 50$ et la masse Υ_2 est égale à 50 pour les modèles $\mathcal{A}, \mathcal{N}$ et à 100 pour le modèle $\mathcal{K}$. Voir Figure 8.8.

Influence de la position sur la distribution de α. Nous avons calculé la densité $a \mapsto \widehat{p}^{(x)}(a)$ pour différentes valeurs de la position x $x = 1$, $x = 10$ et $x = 20$ (▲). Nous avons fait $N = 64$ expériences à chaque fois. La dimension d est fixée à $d = 50$ et la masse Υ_2 est égale à 50 pour les modèles $\mathcal{A}, \mathcal{N}$ et à 100 pour le modèle $\mathcal{K}$. Voir Figure 8.9.

Observations. Pour chaque couple de modèles fixé, [modèle d'entrée, modèle d'exécution], on peut faire trois remarques essentielles :

– la densité dépend très peu de la masse Υ, et de la dimension d : pour chacun des six cas considérés, les cinq courbes de densité sont presque indiscernables.

– La densité varie aussi assez peu en fonction du modèle d'entrée, car les courbes sur chaque horizontale ont vraiment la même forme, mais cette forme est vraiment différente, selon qu'il s'agit du modèle d'exécution `M5` (Lovász, en haut) ou du modèle d'exécution `M4` (Siegel, en bas).
– Dans le cas du modèle `M4`, il y a un phénomène de concentration autour d'une valeur α proche de 3, alors que le même phénomène de concentration existe dans le cas du modèle `M5`, mais autour d'une valeur α très petite et proche de 0.

Explications. Nous tentons quelques explications :

– Dans le modèle `M5`, l'algorithme effectue une longue phase dans laquelle les boîtes B_i sont presque réduites, et dans cette phase, α prend des petites valeurs, proches de 1/2. Dans le modèle `M4`, cette phase est bien plus courte.

– La distribution de α ne varie pas beaucoup en fonction de la masse Υ. Au début, les piles ont une grande hauteur c ; mais, dans ce cas, le décrément α peut être assimilé à $\mathbb{E}[\alpha_c]$ où α_c est défini et étudié dans le chapitre 4. Par exemple, quand la pile courante c est très haute, proche de $c = \infty$, le décrément est proche de 10, et la hauteur de pile diminue rapidement, tandis que le décrément reste à peu près fixe car la courbe du chapitre 4 montre que $c \mapsto \mathbb{E}[\alpha_c]$ atteint vite son asymptote. Mais il faut aussi modérer cet argument, car lorsque c est grand, l'écart type $\sigma(\alpha_c)$ est lui-même grand, et le décrément α présente aussi une grande variabilité. Puis, dans la deuxième phase qui est plus longue, la hauteur des piles ne dépend plus beaucoup de la masse initiale Υ. Cette interprétation s'appuie aussi sur la Figure 8.10 où on trace l'évolution de α pendant l'exécution.

– La figure 8.8 montre que, pour les entrées du modèle $\mathcal{N}$ (NTRU), la distribution de α varie peu au cours du temps, et que les valeurs du décrément α sont concentrées dans un petit intervalle. Par contre, pour les entrées des modèles $\mathcal{K}$ et $\mathcal{A}$, les valeurs de α au début de l'algorithme présentent une grande variabilité, ce qui correspond à la première phase (voir 8.10), qui est plus longue dans les cas des modèles $\mathcal{K}$ et $\mathcal{A}$.

– Le modèle d'entrée $\mathcal{K}$ est constitué d'une pile unique au début. Et, comme nous avons choisi ici les valeurs ($\Upsilon_2 = 100, d = 50$), la masse Υ est beaucoup plus petite que le carré d^2 de la dimension ; dans ce cas, nous savons que dans le modèle `M1` (le *cfg*), la pile ne s'étale pas sur tout l'intervalle $[1\,..\,d-1]$.

On remarque effectivement, que, sur la figure 8.1, pour ces valeurs-là, toutes les positions ne sont pas atteintes dans le modèle `M4`, qui se rapproche ainsi du modèle `M1`. Par contre, pour le modèle `M5`, où α prend des valeurs bien plus petites, la pile s'étale sur tout l'intervalle $[1\,..\,d-1]$.

– Dans chacun des trois modèles, et pour les derniers indices, on remarque que les valeurs de α ne sont pas concentrés, comme elles le sont pour les indices au début et au milieu. Ceci s'explique car la réduction commence par les indices du début. La quasi-totalité de la réduction se passe pour des indices "du début" ou "du milieu" et les grandes valeurs de l'indice sont prises plus rarement. Il y a donc moins de données disponibles, et une plus grande variabilité.

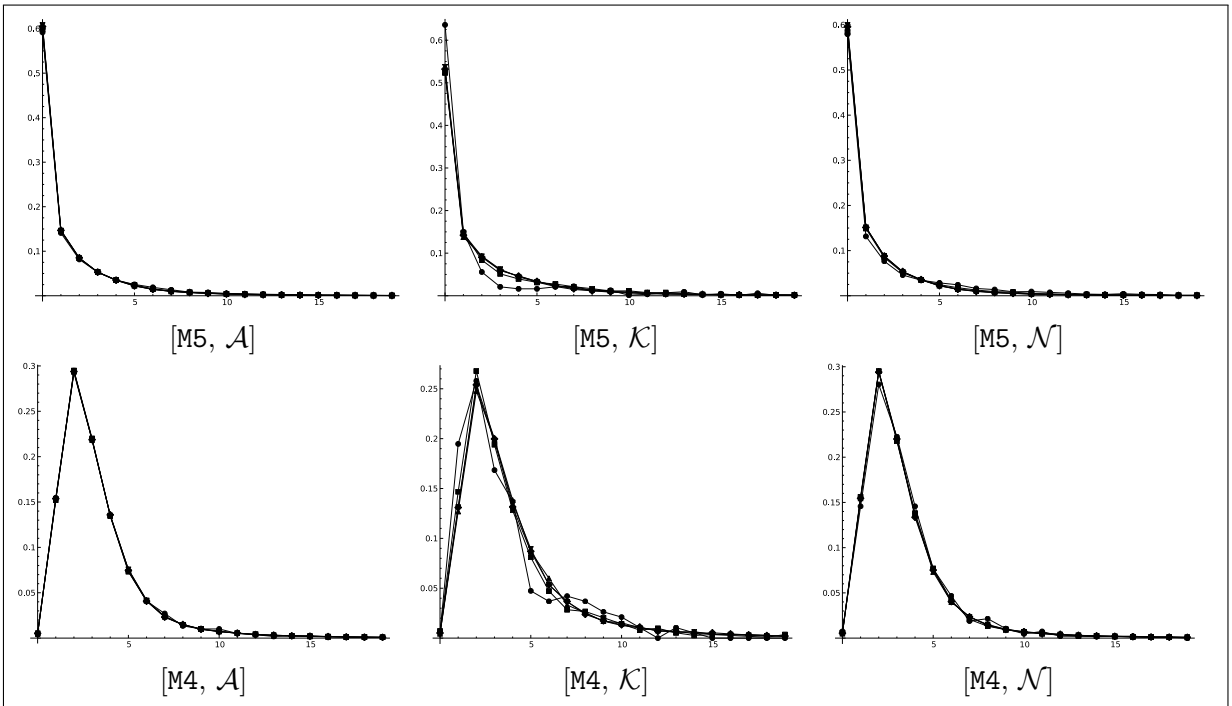


FIGURE 8.6: La densité $a \mapsto \widehat{p}(a)$ pour une dimension $d = 20$ et différentes valeurs de Υ_2 : $\Upsilon_2 = 5$ (•), $\Upsilon_2 = 20$ (■), $\Upsilon_2 = 100$ (▲), $\Upsilon_2 = 200$ (♦) et $\Upsilon_2 = 500$ (▼).

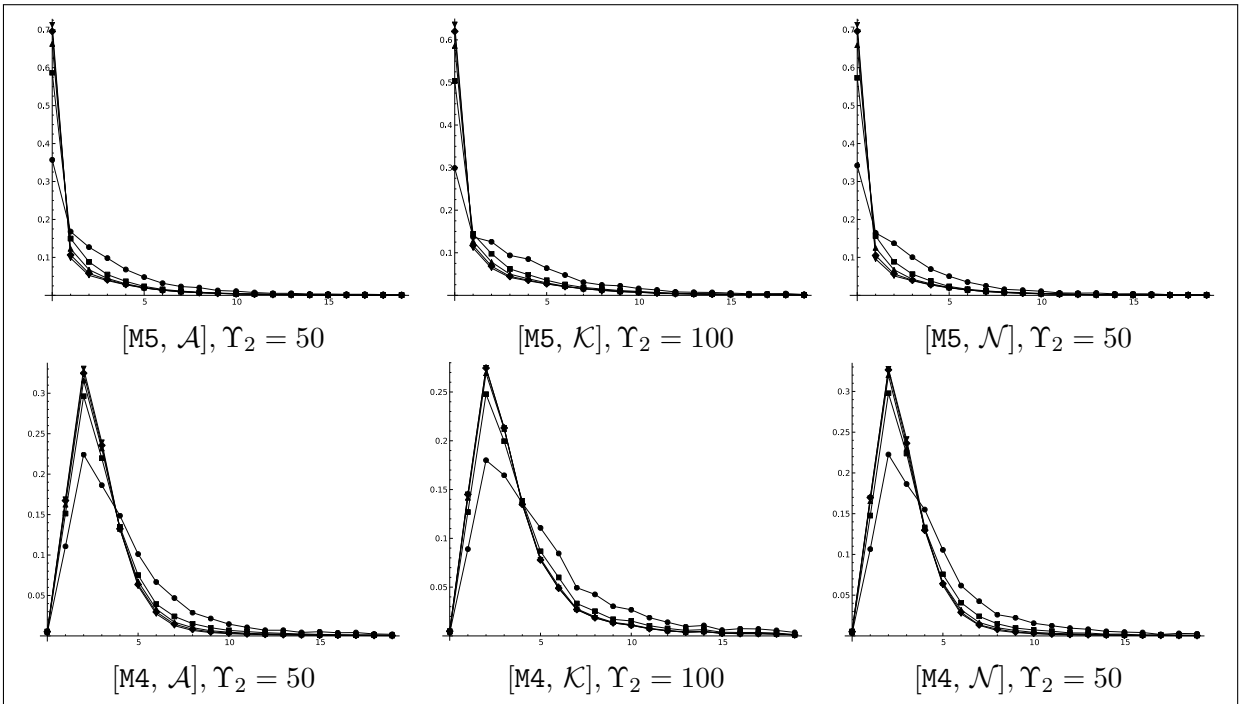


FIGURE 8.7: La densité $a \mapsto \widehat{p}(a)$ pour différentes valeurs de d : $d = 10$ (•), $d = 20$ (■), $d = 30$ (▲), $d = 40$ (♦) et $d = 50$ (▼).

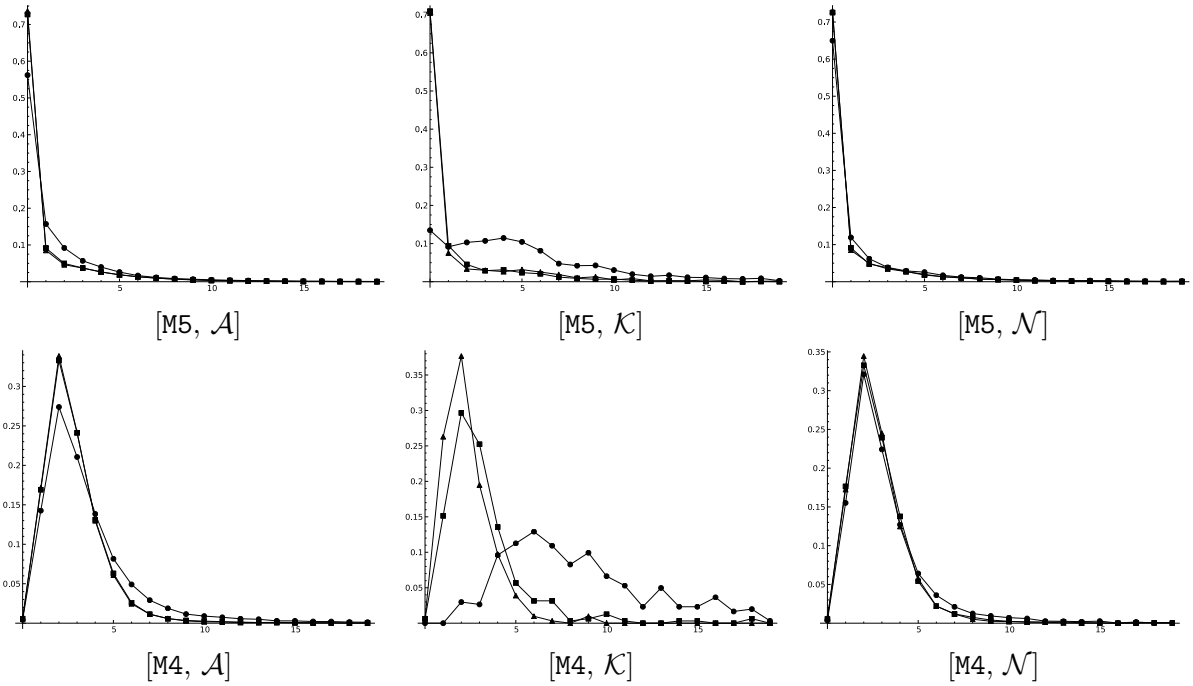


FIGURE 8.8: La densité $a \mapsto \widehat{p}_{(t)}(a)$ pour différentes valeurs de t : $t = 1$ (•), $t = 20$ (■) et $t = 40$ (▲). La dimension d est fixée à $d = 50$ et la masse Υ_2 est égale à 50 pour les modèles $\mathcal{A}, \mathcal{N}$ et à 100 pour le modèle $\mathcal{K}$.

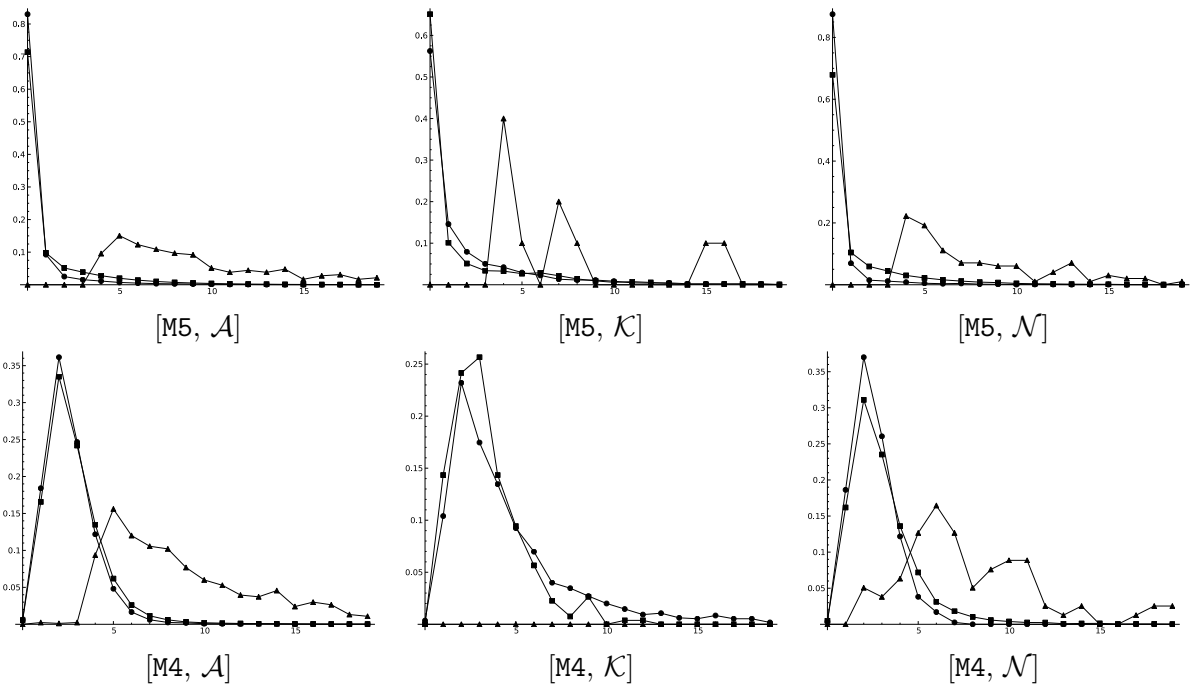


FIGURE 8.9: La densité $a \mapsto \widehat{p}^{(x)}(a)$ pour différentes valeurs de la position x $x = 1$ (•), $x = 10$ (■) and $x = 20$ (▲). La dimension d est fixée à $d = 50$ et la masse Υ_2 est égale à 50 pour les modèles $\mathcal{A}, \mathcal{N}$ et à 100 pour le modèle $\mathcal{K}$.

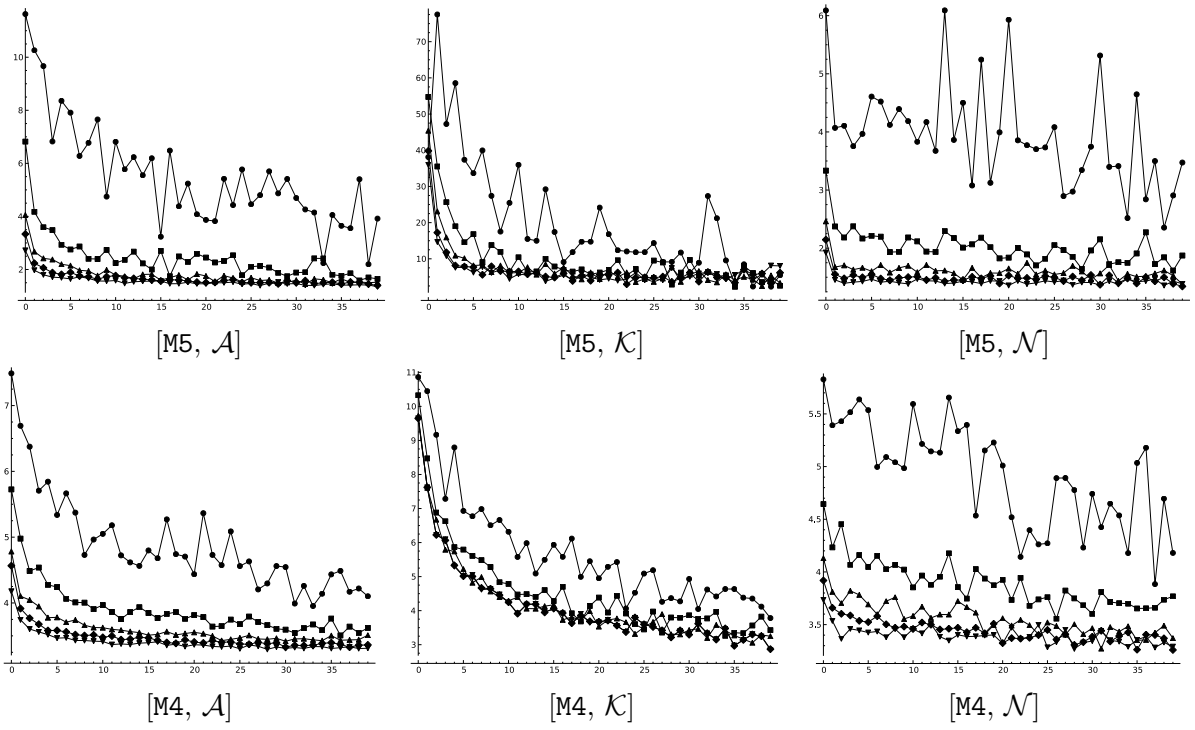


FIGURE 8.10: L'évolution de α pendant la réduction : Graphe de la fonction $t \mapsto \widehat{\alpha}(t)$ pour différentes valeurs de la dimension : $d = 10$ (•), $d = 20$ (■), $d = 30$ (▲), $d = 40$ (♦) et $d = 50$ (▼). La masse Υ_2 est égale à 50 pour les modèles $\mathcal{A}, \mathcal{N}$ et à 100 pour le modèle $\mathcal{K}$.

8.4 L'évolution du décrément α pendant l'exécution.

Évolution du décrément α pendant l'exécution. On étudie pour chaque couple de modèles la fonction $t \mapsto \widehat{\alpha}(t)$ pour différentes valeurs de la dimension : $d = 10$, $d = 20$, $d = 30$, $d = 40$ et $d = 50$ (▼). La masse Υ_2 est égale à 50 pour les modèles $\mathcal{A}, \mathcal{N}$ et à 100 pour le modèle $\mathcal{K}$. Voir Figure 8.10.

Observations. Pour tous les couples (modèle d'entrée, modèle d'exécution), avec un modèle d'exécution `M4` ou `M5`, nous pouvons observer deux phases lors de l'exécution de l'algorithme, où α se comporte assez différemment :

– une première phase, où α est en général grand, mais est aussi très variable,

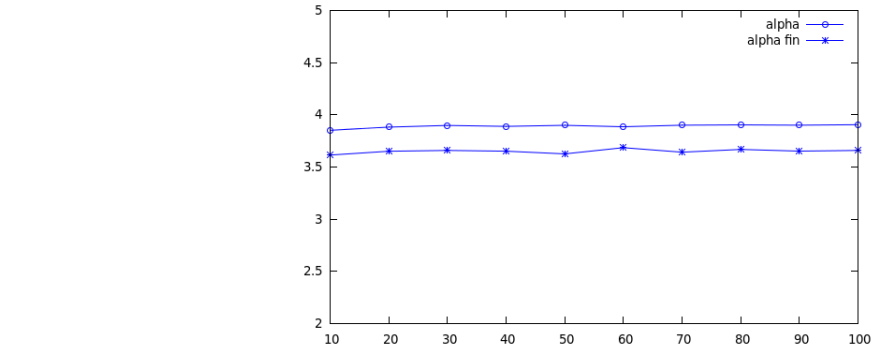
– une deuxième phase, où α est plus petit et aussi plus concentré.

Ces deux phases sont schématisées dans la figure 8.12. Mais la longueur respective des deux phases semble dépendre à la fois du modèle d'entrée et du modèle d'exécution :

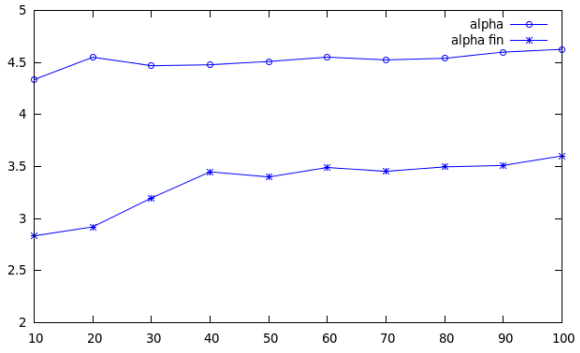
– Selon le modèle d'exécution. La dernière phase du modèle `M5` est beaucoup plus "plate" que celle du modèle `M4`, avec des valeurs de α à la fois petites et concentrées.

– Selon le modèle d'entrée. Cette séparation en deux phases est plus prononcée dans le modèle $\mathcal{K}$ que dans les deux autres modèles – mais cela peut-être dû au fait que le choix de la masse Υ n'est pas le même dans tous les modèles...

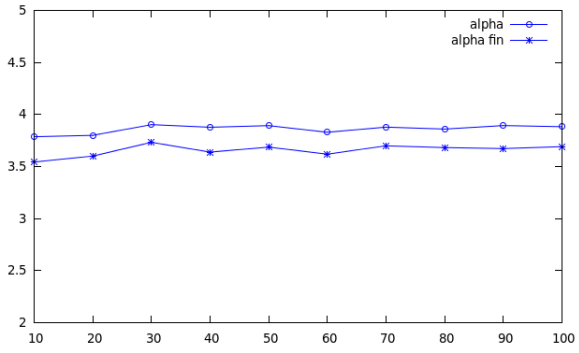
Dans la première phase, qui dépend des entrées (en particulier leur masse et leur type), les piles sont grandes et restent proportionnelles à la taille qu'elles avaient au début. La taille des piles s'égalise assez "rapidement", car la quantité avec laquelle les piles diminuent reste constante



Modèle $\mathcal{A}$



Modèle $\mathcal{K}$



Modèle $\mathcal{N}$

FIGURE 8.11: L'influence de la masse totale Υ_2 sur les deux valeurs moyennes $\widehat{\alpha}$ et $\widehat{\alpha}^*$ pour une dimension $d = 20$, le modèle d'exécution `M4`, et dans chacun des trois modèles d'entrée.

pendant cette phase. La deuxième phase est la plus longue, et elle assez indépendante de l'entrée (masse, forme). Cette interprétation permet d'expliquer pourquoi la géométrie de la sortie de l'algorithme semble être largement indépendante de la distribution des entrées (voir [34, 88]).

Valeur moyenne du décrément dans le dernier quart de l'exécution. La valeur moyenne de α, calculée sur la totalité de l'exécution n'est pas clairement représentative, car α peut prendre de grandes valeurs au début de l'exécution, et se stabilise à la fin de l'exécution. C'est bien sûr vrai dans le modèle `M5`, mais on veut étudier ce qui se passe dans le modèle `M4`, où la valeur de α semble aussi se stabiliser en fin d'exécution. On va donc étudier, pour le modèle d'exécution `M4` (Algorithme de Siegel), la valeur moyenne du décrément α, calculée dans le dernier quart de l'exécution [2], et la comparer à la valeur moyenne du décrément α calculée sur la totalité de l'exécution.

On veut aussi étudier l'influence de la masse Υ et du type des entrées, sur ces deux valeurs moyennes – valeur moyenne totale et valeur moyenne du dernier quart–. On fixe la dimension $d = 20$ et on fait varier la masse Υ_2 entre $\Upsilon_2 = 10$ et $\Upsilon_2 = 100$, aussi bien que le type des entrées $\mathcal{A}, \mathcal{K}, \mathcal{N}$. Voir la figure 8.11.

Observations. Les valeurs $\widehat{\alpha}$ et $\widehat{\alpha}^*$ varient peu en fonction de Υ, ce qui confirme les observations de la figure 8.6. Ces valeurs moyennes se situent essentiellement dans l'intervalle $[3, 4.5]$.

2. ce choix du dernier quart est sans doute un peu arbitraire.

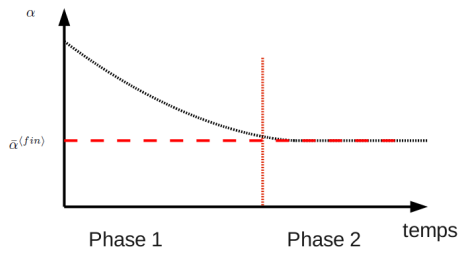


FIGURE 8.12: Les deux phases dans l'exécution dans le modèle `M4` et `M5`

Pour cette dimension $d = 20$, ces deux valeurs moyennes semblent être largement indépendantes de la masse Υ et du type des entrées.

Conclusion Pour chaque modèle d'entrées, on peut distinguer deux phases dans l'algorithme (voir Figure 8.12), pour chacun des deux modèles `M4` et `M5` où α se comporte assez différemment :
- une première phase, où α est en général grand, mais est aussi très variable
- une deuxième phase, où α est plus petit et aussi plus concentré.

Pour simuler l'exécution de l'algorithme LLL (modèles `M4` et `M5`), on peut utiliser une modélisation mixte : dans la Phase 1, il faut utiliser un des modèles simplifiés `M2` ou `M3` qui prend en compte la variation de la hauteur de la pile, mais, dans la Phase 2, l'utilisation du modèle simplifié `M1` avec une valeur de décrément $\widehat{\alpha}^*$ semble tout à fait justifiée.

8.5 Nombre moyen d'itérations

Cette partie vise à étudier le paramètre "nombre moyen d'itérations", et à comparer ce nombre d'itérations dans chacun des modèles d'exécution `M1,M2,M3,M4,M5`, et en étudiant l'influence des modèles d'entrées, et des paramètres de ces modèles d'entrée, notamment la masse Υ et la dimension d.

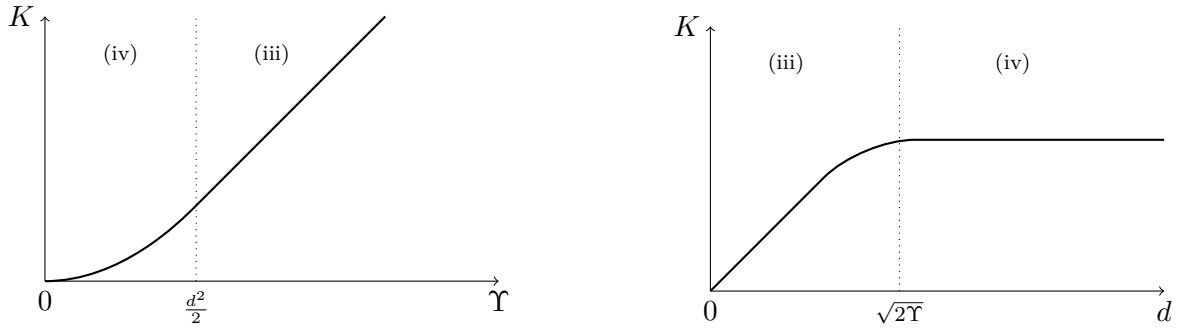
Evidemment, pour comparer ces algorithmes entre eux, il faut définir une valeur pour le paramètre α et pour le paramètre μ. Pour le paramètrre α, nous utilisons nos expériences précédentes, et nous choisissons pour α la valeur moyenne $\widehat{\alpha}^*$ dans le dernier quart de l'exécution. On rappelle ces valeurs, pour chacun des trois modèles

$$\text{pour } \mathcal{A}, \quad \widehat{\alpha}^* = 3.9, \qquad \text{pour } \mathcal{N}, \quad \widehat{\alpha}^* = 3.8, \qquad \text{pour } \mathcal{K}, \quad \widehat{\alpha}^* = 4.6.$$

Pour le paramètre μ nous choisissons pour μ la valeur moyenne, et donc $\mu = 1/12$.

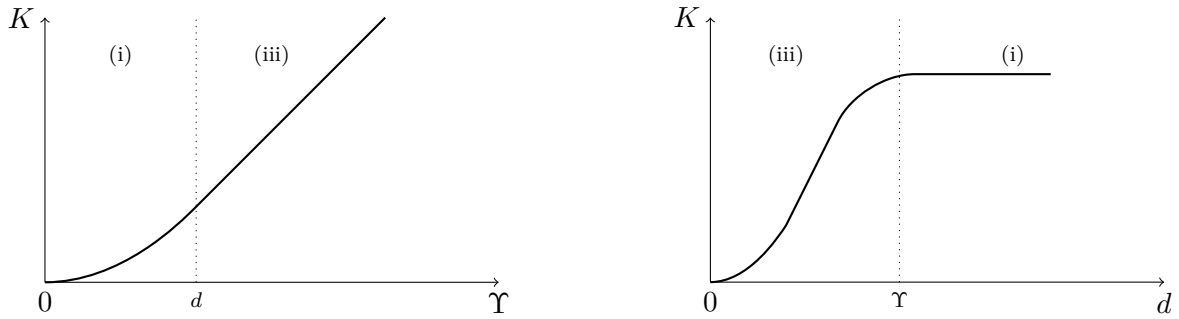
Rappel du nombre d'itérations dans le modèle `M1`. Dans le chapitre 5, nous avons étudié le nombre d'itérations dans le modèle d'exécution `M1`, dans chacun des trois modèles $\mathcal{A}, \mathcal{K}$ et $\mathcal{N}$ en fonction de la dimension d et de la masse Υ. Nous rappelons ici les principaux résultats pour un *cfg* de base : Pour le modèle $\mathcal{K}$, on a, dans le modèle `M1`(1),

$$K \sim \frac{\sqrt{2}}{3}\Upsilon^{3/2} \quad \text{pour} \quad \frac{\Upsilon}{d^2} \to 0, \qquad K \sim \frac{1}{2}d\Upsilon \quad \text{pour} \quad \frac{\Upsilon}{d^2} \to \infty.$$



(a) d fixé (b) Υ fixé

FIGURE 8.13: Graphe pour le nombre K d'itérations dans le modèle $\mathcal{K}$ (Knapsack).



(a) d fixé (b) Υ fixé

FIGURE 8.14: Graphe pour le nombre K d'itérations dans le modèle $\mathcal{N}$.

Pour le modèle $\mathcal{N}$, on a, dans le modèle `M1`(1),

$$K \sim \frac{1}{24}\Upsilon^3 \quad \text{pour} \quad \frac{\Upsilon}{d} \to 0, \qquad K \sim \frac{1}{8}d^2\Upsilon \quad \text{pour} \quad \frac{\Upsilon}{d} \to \infty.$$

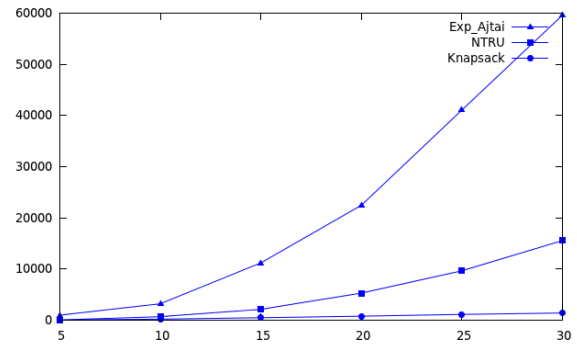
Les courbes sont rappelées ci-dessous dans les figures 8.13 (pour le modèle $\mathcal{K}$) et 8.14 (pour les modèles $\mathcal{A}, \mathcal{N}$.

8.5.1 Influence du modèle d'entrée et de ses paramètres sur le nombre moyen d'itérations.

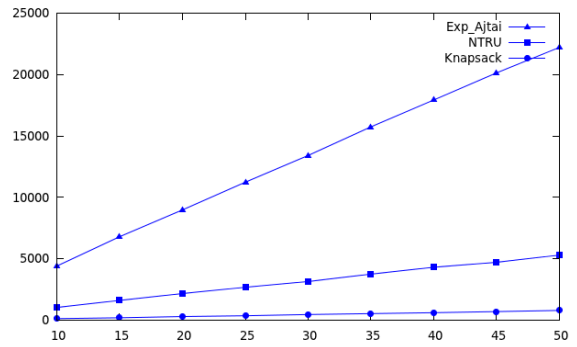
Nous voulons maintenant faire la même étude dans le modèle `M5`. *Comment le nombre d'itérations varie-t-il en fonction de la masse Υ, la dimension d et le type des entrées ?*

On étudie séparément l'influence de la masse Υ et l'influence de la dimension d. Voir Figure 8.15. On fixe d'abord la dimension $d = 20$ et on étudie l'influence du modèle d'entrée et de la masse Υ sur le nombre d'itérations. On fait varier la masse Υ_2 de 10 à 50.
Puis on fixe la masse $\Upsilon_2 = 50$, et on étudie l'influence du modèle d'entrée et de la dimension d sur le nombre d'itérations. On fait varier la dimension d de 5 à 30.

Il faut remarquer que notre choix du couple de paramètres (Υ, d) ne permet pas de conclure franchement à l'adéquation des modèles `M1` et `M5`, car ces couples ne se situent pas clairement par rapport aux seuils, égaux à $\Upsilon = d^2/2$ pour le modèle $\mathcal{K}$ et à $\Upsilon = d$ pour le modèle $\mathcal{N}$. On remarque que le nombre d'itérations dans le modèle `M5`, pour chacun des modèles d'entrée fixé



en fonction de d – à masse fixée $\Upsilon_2 = 50$



en fonction de Υ_2 –à dimension fixée $d = 20$

FIGURE 8.15: Nombre d'itérations K dans le modèle `M5` pour chacun des trois modèles d'entrée $\mathcal{A}, \mathcal{N}, \mathcal{K}$

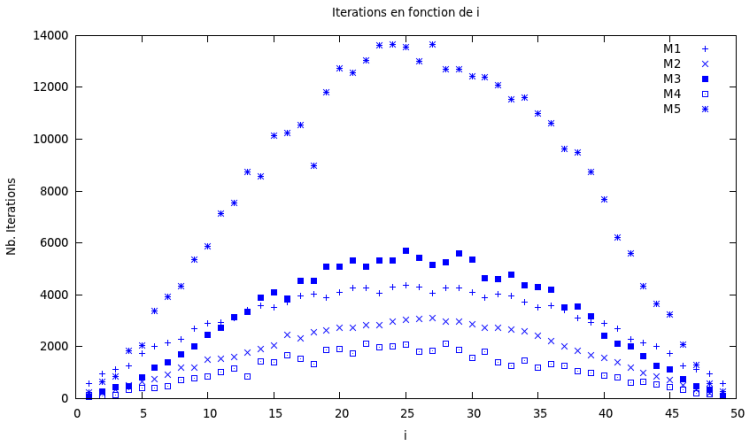


FIGURE 8.16: Influence de la position de l'unique pile sur le nombre moyen d'itérations. La dimension est fixée à $d = 50$ et la masse à $\Upsilon_2 = 25$.

et à dimension fixée, apparaît être une fonction linéaire de la masse Υ. Cela semble généraliser ce qui se passe dans le modèle `M1`.

Quand Υ est fixé, et à modèle d'entrée fixé, le nombre d'itérations apparaît être une fonction linéaire de d pour le modèle d'entrée $\mathcal{K}$ et une fonction quadratique de d pour les modèles $\mathcal{A}, \mathcal{N}$. Cela semble généraliser ce qui se passe dans le modèle `M1`.

Influence de la position de l'unique pile sur le nombre moyen d'itérations du modèle `M5`. Avec ces expériences, on veut étudier la dépendance du nombre d'itérations pour les modèles `M1`, `M2`, `M3`, `M4` et `M5` en fonction de la position m de la pile unique. Pour cela, on fixe la dimension à $d = 50$, la masse Υ_2 à $\Upsilon_2 = 25$ et on fait varier la position m de l'unique pile dans l'intervalle $[1 \,..\, d-1]$. Voir la figure 8.16.

8.5.2 Comparaison du nombre moyen d'itérations pour différents modèles d'exécution.

Maintenant, nous voulons comparer les cinq modèles de l'exécution, vis-à-vis du nombre d'itérations.

Pour la première expérience, on fixe la dimension d et on fait varier la masse Υ et dans la deuxième expérience on fixe la masse Υ et on fait varier la dimension d. Voir Figure 8.17

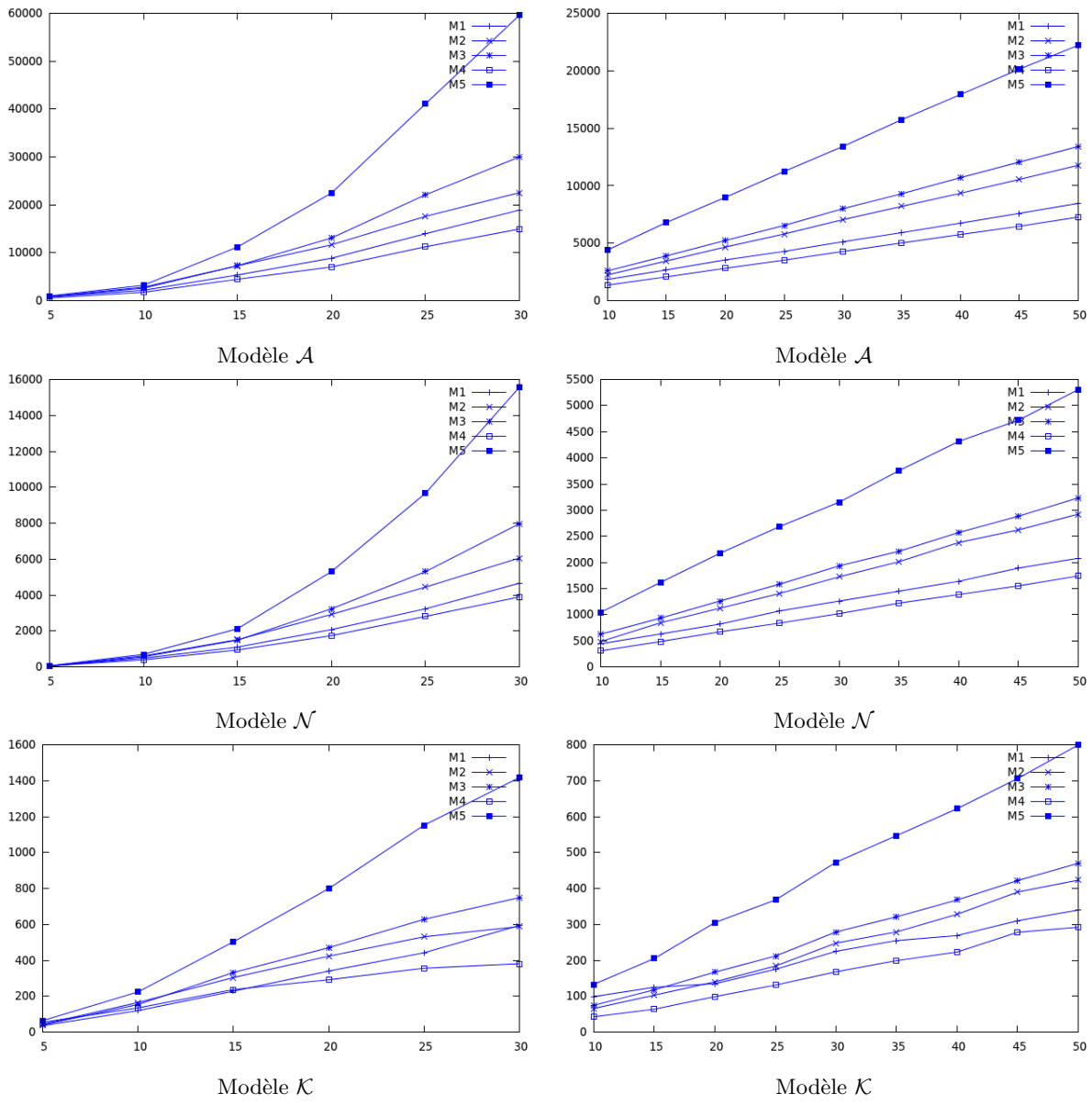


FIGURE 8.17: Nombre d'itérations : à gauche, en fonction de la dimension d, avec une masse $\Upsilon_2 = 50$; à droite, en fonction de la masse Υ_2, avec une dimension $d = 20$.

Pour chacun des cinq modèles d'exécution, le nombre d'itérations apparaît être linéaire en Υ et quadratique en d pour les modèles d'entrée $\mathcal{A}$ et $\mathcal{N}$. Le nombre d'itérations apparaît être linéaire en Υ et linéaire en d pour les modèle d'entrée $\mathcal{K}$. On remarque aussi sur la Figure 8.17 que les graphiques correspondant aux modèles simplifiés se trouvent entre les deux versions de

LLL – M5 avec la condition de Lovász et M4 avec la condition de Siegel–. Le modèle M1 se révèle assez proche de M4. Il est sans doute judicieux de simuler l'algorithme de Siegel avec le modèle M1. Par contre, pour simuler l'algorithme de Lovász (modèle M5), les modèles M2 et M3 semblent préférables.

8.6 Comparaison des modèles et conclusion.

Dans les modèles simplifiés, et notamment le modèle M1, on peut prouver des phénomènes qui semblent "qualitativement" conformes faits expérimentaux observés dans les modèles de la réalité algorithmique M4 et M5.

Les tableaux 8.1 et 8.2, montrent que les équivalents asymptotiques prouvés dans les modèles simplifiées sont "conformes" aux résultats expérimentaux. Même si ces modélisations ne sont réalistes que lorsque certaines hypothèses de régularité sont vérifiées, la campagne d'expérimentations décrite dans ce chapitre a établi, à la fois de manière directe et indirecte, la validité "qualitative" de ces hypothèses.

Modèles	K **pire des cas**	K **pour** M1(α)	K **pour** M2(μ)	K **expérimental**
$\mathcal{A}(\Upsilon, d)$	$\frac{1}{12\log t} d(d+1)\widetilde{\Upsilon}$	$\frac{1}{12\alpha} d(d+1)\widetilde{\Upsilon}$	$\frac{1}{6\lvert\log\mu\rvert} d(d+1)\widetilde{\Upsilon}$	$\Theta(d^2\widetilde{\Upsilon})$
$\mathcal{N}(\Upsilon, d)$	$\frac{1}{8\log t} d^2\widetilde{\Upsilon}$	$\frac{1}{8\alpha}\, d^2\widetilde{\Upsilon}$	$\frac{1}{4\lvert\log\mu\rvert} d^2\widetilde{\Upsilon}$	$\Theta(d^2\widetilde{\Upsilon})$
$\mathcal{K}(\Upsilon, d)$	$\frac{1}{2\log t} d\widetilde{\Upsilon}$	$\frac{1}{2\alpha} d\widetilde{\Upsilon}$	$\frac{1}{\lvert\log\mu\rvert} d\widetilde{\Upsilon}$	$\Theta(d\widetilde{\Upsilon})$

TABLE 8.1: Nombre K d'itérations de l'algorithme LLL dans le pire des cas, dans le modèle M1(α), et dans le modèle M2(μ) avec une stratégie gloutonne. Ici, d est la dimension du réseau et $\widetilde{\Upsilon}$ est le coefficient de non-réduction du réseau $\widetilde{\Upsilon} = -\log(\prod r_i)$, où r_i sont les rapports de Siegel. La réduction est supposée difficile, et $\widetilde{\Upsilon}/d^a \to \infty$ avec $a = 1$ pour les modèles $\mathcal{A}, \mathcal{N}$ et $a = 2$ pour le modèle $\mathcal{K}$. La deuxième et la troisième colonnes fournissent des équivalents asymptotiques quand $d \to \infty$.

Paramètres	**LLL pire des cas**	**Modèle** M1(α)	**Modèle** M2(μ)	**LLL expérimental [88]**
$\widehat{r}_i$	$\geq \frac{1}{s}$	$\left[\frac{1}{s}, \frac{1}{\rho s}\right]$	$\geq (1-\mu)^{1/2}$	$\approx \frac{1}{\beta}$
$\gamma(\widehat{B})$	$\leq s^{d-1}$	$\left[\rho(s\rho)^{d-1}, s^{d-1}\right]$	$\leq \left(\frac{1}{1-\mu}\right)^{(d-1)/2}$	$\approx \beta^{d-1}$

TABLE 8.2: Comparaison entre les majorations prouvées pour les rapports de Siegel $\widehat{r}_i$ à la fin de l'exécution et le défaut d'Hermite $\gamma(\widehat{B})$. Ici $\rho = s^{-\alpha}$ et $\beta \approx 1.04$.

Nos expériences mettent aussi en évidence un déroulement en deux phases : une première phase, où l'exécution dépend de la masse et du type de l'entrée, et où il est préférable d'utiliser

la modélisation `M2` et une seconde phase, plus longue et plus régulière qu'on peut simuler avec le modèle `M1`. Nos modélisations simplifiées donnent une idée intuitive de l'exécution de l'algorithme LLL ; ils permettent aussi de développer des heuristiques, et de donner une explication pédagogique de l'algorithme à des publics non familiers de la réduction des réseaux.

Conclusion

Résumé des principales contributions. Nous avions deux principaux objectifs : nous voulions d'abord analyser l'efficacité des cryptanalyses qui utilisent les réseaux euclidiens et mieux évaluer la sécurité des systèmes cryptographiques fondés sur les réseaux euclidiens ; pour cela, nous devions mieux comprendre le processus même de réduction des réseaux, et analyser les algorithmes de réduction de manière probabiliste.

Devant la grande difficulté d'une analyse exacte de l'algorithme LLL, nous avons proposé toute une classe de modèles simplifiés pour l'exécution de l'algorithme, du plus simple, déjà proposé par Madrisch et Vallée, au plus compliqué, qui correspond à l'algorithme LLL lui-même. Nous sommes revenus sur l'analyse du modèle le plus simple, le modèle `M1` en adoptant le point de vue du chip firing game, directement applicable, plutôt que celui du tas de sable, plus classique mais moins adapté à notre point de vue.

Nous avons aussi cherché à modéliser, dans ce cadre de *cfg*, les principales entrées qui nous intéressaient, correspondant à des réseaux cryptographiques. Nous avons été conduits à trois familles des réseaux cryptographiques : les réseaux dit réseaux d'Ajtai qui donnent lieu à des tas "tous très pleins", les réseaux sac-à-dos ou réseaux NTRU, qui donnent lieu à des tas de sable "avec un seul tas" et enfin les réseaux de Coppersmith, qui donnent lieu à des tas de sable "avec des trous". Nous avons ainsi retrouvé des types de configurations classiques, mais aussi d'autres moins classiques, liés notamment aux réseaux que Coppersmith utilise dans ses cryptanalyses. Nous avons ainsi complèté une étude initiée par Madritsch et Vallée.

Nous avons ensuite étudié un modèle moins simplifié d'exécution, mais sans aucun doute plus proche de la réalité, le modèle dit `M2`, dans laquelle le décrément α dépend de la hauteur de la pile courante. Nous avons effectué une analyse précise de ce modèle d'exécution : analyse totale pour le cas de la dimension 2, qui correspond à la dimension 3 dans le monde des réseaux, où l'analyse du véritable algorithme LLL est déjà mal comprise –analyse partielle en dimension générale.

Enfin, nous avons fait des expérimentations afin de rechercher une validation expérimentale des hypothèses qui mènent aux modèles simplifiés. D'abord une validation directe, où nous avons testé le comportement du décrément α, objet central sur lequel reposent les principes de modélisation. Nous avons opéré aussi une validation indirecte en comparant expérimentalement le comportement probabiliste de tous les modèles d'exécution, en faisant aussi varier le modèle d'entrées à l'intérieur de l'ensemble des modèles cryptographiques réalistes. Malgré la simplification que nous avons opérée, nous prouvons des phénomènes qualitatifs de même nature que ceux qui sont obtenus expérimentalement dans une modélisation exacte. Pour le modèle simplifié `M1`, nous montrons, à la suite de Madritsch et Vallée, que la géométrie de la configuration de sortie est largement indépendante du modèle d'entrée, pourvu qu'il donne lieu à un *cfg* d'énergie initiale suffisamment grande. Par contre, l'exécution de l'algorithme (et en particulier son nombre d'itérations) dépend largement du modèle d'entrée. Nos expériences confirment les expériences de Gama, Nguyen et Stehlé et montrent que ces phénomènes apparaissent aussi dans la réalité

algorithmique, qui travaille avec le modèle `M5` de l'algorithme LLL.

Perspectives. Nous décrivons ici quelques pistes qui peuvent prolonger ce travail de thèse :

(*i*) On peut modéliser d'autres réseaux euclidiens qui interviennent dans la cryptologie ou dans d' autres applications de l'algorithme LLL et traduire leur exécution dans les modèles `M1` et `M2`. Cela peut peut-être conduire à des optimisations, dans l'esprit de ce que nous avons fait pour les réseaux de Coppersmith.

(*ii*) Maintenant que nous comprenons mieux l'algorithme, grâce à nos travaux de modélisation complétés par nos expérimentations, nous pouvons reprendre nos expérimentations, en adaptant mieux les principaux paramètres (la masse Υ et la dimension d) à la réalité algorithmique, et en précisant leurs relations. On peut aussi faire des expériences sur les différentes stratégies (classique, gloutonne ou aléatoire).

(*iii*) Pour le modèle `M2`, il faudrait prouver la conjecture, qui permettrait d'avoir un modèle simplifié pour l'algorithme LLL associé au paramètre $t = 1$.

(*iv*) Il faudrait aussi s'attaquer au modèle `M3`, qui est un modèle plus évolué qui permet de prendre en compte l'évolution du paramètre ν, égal à la partie fractionnaire centée du coefficient sous-diagonal courant. Il faut d'abord mieux comprendre comment ce paramètre évolue. Il faut ensuite analyser ce modèle `M3`. C'est un système dynamique probabiliste, sans doute assez difficile à étudier, dont l'analyse nous semble cependant possible. Et c'est sans doute ce modèle qui rend le mieux compte du comportement probabiliste de l'algorithme LLL...

Bibliographie

[1] Aharonov, D., and Regev, O. Lattice problems in NP ∩ coNP. *J. ACM 52*, 5 (Sept. 2005), 749–765. (Cité page 24.)

[2] Ajtai, M. Generating hard instances of lattice problems (extended abstract). In *STOC* (1996), pp. 99–108. (Cité pages 24, 47, 53, 54, 57, et 125.)

[3] Ajtai, M. The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract). In *STOC* (1998), pp. 10–19. (Cité pages 3 et 24.)

[4] Ajtai, M., and Dwork, C. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC* (1997), pp. 284–293. (Cité pages 3, 53, 54, 57, et 58.)

[5] Ajtai, M., Kumar, R., and Sivakumar, D. Sampling short lattice vectors and the closest lattice vector problem. In *Proceedings of the 17th IEEE Annual Conference on Computational Complexity* (Washington, DC, USA, 2002), CCC '02, IEEE Computer Society, pp. 53–. (Cité pages 23, 24, 30, et 32.)

[6] Akhavi, A. Random lattices, threshold phenomena and efficient reduction algorithms. *Theoretical Computer Science 287*, 2 (2002), 359 – 385. (Cité page 73.)

[7] Akhavi, A. The optimal LLL algorithm is still polynomial in fixed dimension. *Theor. Comput. Sci. 297*, 1-3 (2003), 3–23. (Cité pages 41, 142, et 153.)

[8] Akhavi, A., Marckert, J.-F., and Rouault, A. On the reduction of a random basis. In *ANALCO* (2007), pp. 265–270. (Cité pages 73, 75, et 126.)

[9] Akhavi, A., and Vallée, B. Average bit-complexity of euclidean algorithms. In *ICALP* (2000), pp. 373–387. (Cité page 78.)

[10] Bak, Tang, and Wiesenfeld. Self-organized criticality : An explanation of the 1/f noise. *Phys. Rev. Lett. 59 (4)* (Jul 1987), 381–384. (Cité pages 90 et 98.)

[11] Baladi, V., and Vallée, B. Euclidean algorithms are gaussian. *CoRR cs.DS/0307062* (2003). (Cité page 78.)

[12] Boneh, D., and Durfee, G. Cryptanalysis of RSA with private key d less than $N^{0.292}$. *IEEE Transactions on Information Theory 46*, 4 (2000), 1339–1349. (Cité pages 3, 61, 63, et 66.)

[13] Bourdon, J., Daireaux, B., and Vallée, B. Dynamical analysis of alpha-euclidean algorithms. *J. Algorithms 44*, 1 (2002), 246–285. (Cité page 78.)

[14] Brakerski, Z., Gentry, C., and Vaikuntanathan, V. Fully homomorphic encryption without bootstrapping. *Electronic Colloquium on Computational Complexity (ECCC) 18* (2011), 111. (Cité page 54.)

[15] Brakerski, Z., Langlois, A., Peikert, C., Regev, O., and Stehlé, D. Classical hardness of learning with errors. In *STOC* (2013), pp. 575–584. (Cité page 54.)

[16] Brakerski, Z., and Vaikuntanathan, V. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS* (2011), pp. 97–106. (Cité page 54.)

[17] Brakerski, Z., and Vaikuntanathan, V. Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In *CRYPTO* (2011), pp. 505–524. (Cité page 54.)

[18] Cohen, H. *A Course in Computational Algebraic Number Theory*, vol. 138 of *Graduate Texts in Mathematics*. Springer-Verlag, 1993. (Cité page 10.)

[19] Cohn, H., and Kumar, A. The densest lattice in twenty-four dimensions. *Electron. Res. Announc. Am. Math. Soc. 10*, math.MG/0408174 (2004), 58–67. (Cité page 18.)

[20] Coppersmith, D. Finding a small root of a univariate modular equation. In *EUROCRYPT* (1996), pp. 155–165. (Cité pages 60 et 61.)

[21] Coppersmith, D. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptology 10*, 4 (1997), 233–260. (Cité pages 3, 61, et 63.)

[22] Coppersmith, D., and Shamir, A. Lattice attacks on NTRU. In *EUROCRYPT* (1997), pp. 52–61. (Cité page 60.)

[23] Coster, M. J., Joux, A., LaMacchia, B. A., Odlyzko, A. M., Schnorr, C.-P., and Stern, J. Improved low-density subset sum algorithms. *Computational Complexity 2* (1992), 111–128. (Cité page 56.)

[24] Daemen, J., and Rijmen, V. *The Design of Rijndael : AES - The Advanced Encryption Standard*. Springer, 2002. (Cité page 50.)

[25] Daireaux, B., Maume-Deschamps, V., and Vallée, B. The Lyapunov tortoise and the dyadic hare. In *2005 International Conference on Analysis of Algorithms* (2005), C. Martínez, Ed., vol. AD of *DMTCS Proceedings*, Discrete Mathematics and Theoretical Computer Science, pp. 71–94. (Cité page 78.)

[26] Daireaux, B., and Vallée, B. Dynamical analysis of the parametrized Lehmer-Euclid Algorithm. *Combinatorics, Probability & Computing 13*, 4-5 (2004), 499–536. (Cité pages 78 et 126.)

[27] Daudé, H., Flajolet, P., and Vallée, B. An average-case analysis of the Gaussian Algorithm for lattice reduction. *Combinatorics, Probability & Computing 6*, 4 (1997), 397–433. (Cité pages 79, 80, 146, et 147.)

[28] Daudé, H., and Vallée, B. An upper bound on the average number of iterations of the LLL algorithm. *Theoretical Computer Science 123* (1994), 95–115. (Cité pages 73 et 75.)

[29] Diffie, W., and Hellman, M. E. New directions in cryptography. *IEEE Transactions on Information Theory 22*, 6 (1976), 644–654. (Cité pages 49 et 51.)

[30] Dupré, A. Sur le nombre de divisions à effectuer pour obtenir le plus grand commun diviseur entre deux nombres entiers. *Journal de Mathématiques Pures et Appliquées 11* (1846), 41–74. (Cité page 33.)

[31] Ernst, M., Jochemsz, E., May, A., May, E., and de Weger, B. Partial key exposure attacks on RSA up to full size exponents. In *In CRYPTO 2005, volume 3494 of LNCS* (2005), Springer-Verlag, pp. 371–386. (Cité page 3.)

[32] Fincke, U., and Pohst, M. A procedure for determining algebraic integers of given norm. In *EUROCAL* (1983), pp. 194–202. (Cité pages 23 et 41.)

[33] Gama, N., and Nguyen, P. Q. Finding short lattice vectors within mordell's inequality. In *STOC* (2008), pp. 207–216. (Cité pages 10 et 24.)

[34] Gama, N., and Nguyen, P. Q. Predicting lattice reduction. In *EUROCRYPT* (2008), pp. 31–51. (Cité pages 72, 163, et 177.)

[35] Gamal, T. E. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory 31*, 4 (1985), 469–472. (Cité pages 9, 51, et 52.)

[36] Gauss, C. F. *Disquisitiones Arithmeticae*. Leipzig, 1801. (Cité page 32.)

[37] Gentry, C. Fully homomorphic encryption using ideal lattices. In *STOC* (2009), pp. 169–178. (Cité pages 3, 47, 53, et 54.)

[38] Gentry, C., and Halevi, S. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *FOCS* (2011), pp. 107–109. (Cité page 54.)

[39] Gentry, C., Halevi, S., and Smart, N. P. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT* (2012), pp. 465–482. (Cité page 54.)

[40] Gentry, C., Peikert, C., and Vaikuntanathan, V. Trapdoors for hard lattices and new cryptographic constructions. In *STOC* (2008), pp. 197–206. (Cité page 54.)

[41] Goldreich, O., and Goldwasser, S. On the limits of nonapproximability of lattice problems. *J. Comput. Syst. Sci. 60*, 3 (2000), 540–563. (Cité page 24.)

[42] Goldreich, O., Goldwasser, S., and Halevi, S. Eliminating decryption errors in the Ajtai-Dwork cryptosystem. In *Advances in Cryptology - CRYPTO 97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings* (1997), B. S. K. Jr., Ed., vol. 1294 of *Lecture Notes in Computer Science*, Springer, pp. 105–111. (Cité pages 57 et 58.)

[43] Goldreich, O., Goldwasser, S., and Halevi, S. Public-key cryptosystems from lattice reduction problems. In *CRYPTO* (1997), pp. 112–131. (Cité pages 3, 53, et 54.)

[44] Goles, E. C., and Kiwi, M. A. Games on line graphs and sand piles. *Theor. Comput. Sci. 115*, 2 (1993), 321–349. (Cité pages 105, 108, et 114.)

[45] Hanrot, G., and Stehlé, D. Improved analysis of Kannan's shortest lattice vector algorithm. In *Proceedings of the 27th annual international cryptology conference on Advances in cryptology* (Berlin, Heidelberg, 2007), CRYPTO'07, Springer-Verlag, pp. 170–186. (Cité page 23.)

[46] Håstad, J. Solving simultaneous modular equations of low degree. *SIAM J. Comput. 17*, 2 (1988), 336–341. (Cité page 61.)

[47] Hensley, D. The number of steps in the Euclidean algorithm. *Journal of Number Theory 2*, 49 (1994), 149–182. (Cité page 78.)

[48] Hermite, C., and Picard, E. *Œuvres : Publiées sous les auspices de l'Académie des Sciences*. No. vol. 1. Gauthier-Villars, 1905. (Cité pages 9, 26, et 27.)

[49] Hoffstein, J., Pipher, J., and Silverman, J. H. NTRU : A Ring-based public key cryptosystem. In *ANTS* (1998), pp. 267–288. (Cité pages 3, 53, 54, 59, et 132.)

[50] Howgrave-Graham, N. *Finding Small Roots of Univariate Modular Equations Revisited*. 1997. (Cité page 61.)

[51] Jensen, H. J. Self-organized criticality. *Cambridge Lecture Notes in Physics. Cambridge University Press, Cambridge. Emergent complex behavior in physical and biological systems. 10* (1998). (Cité page 98.)

[52] Kannan, R. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing* (New York, NY, USA, 1983), STOC '83, ACM, pp. 193–206. (Cité pages 23, 30, 32, 41, et 42.)

[53] Karp, R. M. Reducibility among combinatorial problems. In *Complexity of Computer Computations* (1972), pp. 85–103. (Cité page 54.)

[54] Kato, T. *Perturbation Theory for Linear Operators*. Springer-Verlag, 1980. (Cité page 149.)

[55] Katok, A., and Hasselblatt, B. *Introduction to the Modern Theory of Dynamical Systems*. Cambridge University Press, New York, NY, 1995. (Cité page 149.)

[56] Khot, S. Hardness of approximating the shortest vector problem in lattices. In *FOCS* (2004), pp. 126–135. (Cité page 24.)

[57] Koblitz, N. Elliptic curve cryptosystems. *Mathematics of Computation 48*, 177 (Jan. 1987), 203–209. (Cité page 53.)

[58] Korkine, A., and Zolotarev, G. Sur les formes quandratiques. *Mathematishe Annalen 6* (1873), 336–389. (Cité pages 9, 16, et 26.)

[59] Lagarias, J. C. Worst-case complexity bounds for algorithms in the theory of integral quadratic forms. *J. Algorithms 1*, 2 (1980), 142–186. (Cité page 146.)

[60] Lagarias, J. C. The computational complexity of simultaneous diophantine approximation problems. In *FOCS* (1982), pp. 32–39. (Cité pages 43 et 47.)

[61] Lagarias, J. C., Jr., H. W. L., and Schnorr, C.-P. Korkin-Zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica 10*, 4 (1990), 333–348. (Cité page 30.)

[62] Lagarias, J. C., and Odlyzko, A. M. Solving low-density subset sum problems. *Journal of the ACM 32*, 1 (Jan. 1985), 229–246. (Cité page 56.)

[63] Lagrange, J. L. Recherches d'arithmétiques. In *Nouveaux mémoires de l'Académie royale des sciences et des belles-lettres de Berlin*. 1773-1775. (Cité page 9.)

[64] Laville, H., and Vallée, B. Distribution de la constante d'hermite et du plus court vecteur dans les réseaux de dimension deux. *Journal de théorie des nombres de Bordeaux 6*, 1 (1994), 135–159. (Cité pages 79 et 146.)

[65] Lenstra, A., Lenstra, H., and Lovász, L. Factoring polynomials with rational coefficients. *Mathematische Annalen 261*, 4 (1982), 515–534. (Cité pages 1, 9, 24, 26, 32, et 41.)

[66] Lenstra, H. W. Integer Programming in a Fixed Number of Variables. *Math. Oper. Res. 8* (1983), 538–548. (Cité pages 42 et 47.)

[67] Lenstra Jr., H. W. Flags and lattice basis reduction. In *R. M. Miro-Roig, 3rd European congress on mathematics, Barcelona* (2001), C. Casacuberta, J. Verdera, and S. Xambo-Deschamps, Eds., Springer-Verlag, pp. 37–52. (Cité pages 41, 142, et 153.)

[68] Lhote, L., and Vallée, B. Gaussian laws for the main parameters of the Euclid algorithms. *Algorithmica 50*, 4 (2008), 497–554. (Cité page 78.)

[69] Lovász, L. *An algorithmic theory of numbers, graphs and convexity*, vol. 50 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1986. (Cité pages 10 et 23.)

[70] Lyubashevsky, V., Peikert, C., and Regev, O. On Ideal lattices and Learning with Errors over Rings. In *EUROCRYPT* (2010), pp. 1–23. (Cité page 54.)

[71] Madritsch, M., and Vallée, B. Modelling the LLL algorithm by sandpiles. In *Proceedings of the 9th Latin American conference on Theoretical Informatics* (Berlin, Heidelberg, 2010), LATIN'10, Springer-Verlag, pp. 267–281. (Cité pages 2, 84, 90, 98, 105, 115, et 126.)

[72] Mahler, K. A theorem on inhomogeneous diophantine inequalities. *Nederl. Akad. Wetensch., Proc.* (1938). (Cité page 30.)

[73] Martinet, J. *Perfect lattices in Euclidean spaces*. Grundlehren der mathematischen Wissenschaften = A series of comprehensive studies in mathematics. Springer, Berlin, New York, Hong Kong, 2003. Traduit de : Réseaux parfaits des espaces euclidiens. (Cité pages 10 et 18.)

[74] May, A. Using LLL-reduction for solving RSA and factorization problems. In *The LLL Algorithm*. 2010, pp. 315–348. (Cité page 63.)

[75] McEliece, R. A public-key cryptosystem based on algebraic number theory. Tech. rep., Jet Propulsion Laboratory, 1978. DSN Progress Report 42-44. (Cité page 53.)

[76] Menezes, A. J., Vanstone, S. A., and Oorschot, P. C. V. *Handbook of Applied Cryptography*, 1st ed. CRC Press, Inc., Boca Raton, FL, USA, 1996. (Cité page 50.)

[77] Merkle, R., and Hellman, M. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions in Information Theory IT-24* (1978), 525–530. (Cité page 54.)

[78] Micciancio, D. Improving lattice based cryptosystems using the Hermite normal form. In *CaLC* (2001), pp. 126–145. (Cité pages 11 et 21.)

[79] Micciancio, D. The shortest vector in a lattice is hard to approximate to within some constant. *SIAM J. Comput. 30*, 6 (Dec. 2001), 2008–2035. (Cité page 24.)

[80] Micciancio, D. Cryptographic functions from worst-case complexity assumptions. In *The LLL Algorithm*. 2010, pp. 427–452. (Cité page 48.)

[81] Micciancio, D. Lattice-based cryptography. In *Encyclopedia of Cryptography and Security (2nd Ed.)*. 2011, pp. 713–715. (Cité page 48.)

[82] Micciancio, D., and Goldwasser, S. *Complexity of Lattice Problems : a cryptographic perspective*, vol. 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, Mar. 2002. (Cité page 22.)

[83] Micciancio, D., and Regev, O. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput. 37*, 1 (Apr. 2007), 267–302. (Cité pages 3, 53, et 54.)

[84] Micciancio, D., and Regev, O. Lattice-based cryptography. In *Post-quantum Cryprography*, D. Bernstein and J. Buchmann, Eds. 2008. (Cité page 48.)

[85] Miller, V. S. Use of elliptic curves in cryptography. In *Lecture notes in computer sciences ; 218 on Advances in cryptology—CRYPTO 85* (New York, NY, USA, 1986), Springer-Verlag New York, Inc., pp. 417–426. (Cité page 53.)

[86] Milnor, J. W., and Husemoller, D. *Symmetric bilinear forms*. Springer-Verlag Berlin, New York, 1973. (Cité page 18.)

[87] Minkowski, H. *Geometrie der Zahlen.* B. G. Teubner, Leipzig, 1896. (Cité page 9.)

[88] Nguyen, P. Q., and Stehlé, D. LLL on the average. In *ANTS* (2006), pp. 238–256. (Cité pages 72, 128, 163, 172, 177, et 182.)

[89] Nguyen, P. Q., and Stern, J. Cryptanalysis of the Ajtai-Dwork cryptosystem. In *Advances in Cryptology - CRYPTO 98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings* (1998), H. Krawczyk, Ed., vol. 1462 of *Lecture Notes in Computer Science*, Springer, pp. 223–242. (Cité page 58.)

[90] Nguyen, P. Q., and Stern, J. The two faces of lattices in cryptology. In *CaLC* (2001), pp. 146–180. (Cité page 48.)

[91] Nguyen, P. Q., and Stern, J. Adapting density attacks to low-weight knapsacks. In *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings* (2005), B. K. Roy, Ed., vol. 3788 of *Lecture Notes in Computer Science*, Springer, pp. 41–58. (Cité pages 57 et 121.)

[92] Nguyen, P. Q., and Vallée, B. *The LLL Algorithm : Survey and Applications*, 1st ed. Springer Publishing Company, Incorporated, 2009. (Cité page 10.)

[93] Peikert, C. Public-key cryptosystems from the worst-case shortest vector problem : extended abstract. In *Proceedings of the 41st annual ACM symposium on Theory of computing* (New York, NY, USA, 2009), STOC '09, ACM, pp. 333–342. (Cité page 53.)

[94] Peikert, C. An efficient and parallel Gaussian sampler for lattices. In *CRYPTO* (2010), pp. 80–97. (Cité page 54.)

[95] Pohst, M. On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. *SIGSAM Bull. 15*, 1 (Feb. 1981), 37–44. (Cité page 41.)

[96] Regev, O. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing* (New York, NY, USA, 2005), STOC '05, ACM, pp. 84–93. (Cité page 54.)

[97] Regev, O. Lattice-based cryptography. In *CRYPTO* (2006), pp. 131–141. (Cité page 48.)

[98] Rivest, R. L., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM 26*, 1 (Jan. 1983), 96–99. (Cité page 51.)

[99] Schnorr, C. P. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci. 53*, 2-3 (Aug. 1987), 201–224. (Cité pages 10, 24, et 32.)

[100] Schnorr, C. P. Factoring integers and computing discrete logarithms via diophantine approximation. In *Proceedings of the 10th annual international conference on Theory and application of cryptographic techniques* (Berlin, Heidelberg, 1991), EUROCRYPT'91, Springer-Verlag, pp. 281–293. (Cité pages 45, 47, et 132.)

[101] Schnorr, C. P., and Euchner, M. Lattice basis reduction : improved practical algorithms and solving subset sum problems. *Math. Program. 66*, 2 (Sept. 1994), 181–199. (Cité pages 10, 24, et 26.)

[102] Schnorr, C. P., and Hörner, H. H. Attacking the chor-rivest cryptosystem by improved lattice reduction. In *Proceedings of the 14th annual international conference on Theory and application of cryptographic techniques* (Berlin, Heidelberg, 1995), EUROCRYPT'95, Springer-Verlag, pp. 1–12. (Cité page 26.)

[103] Shamir, A. A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. In *FOCS* (1982), pp. 145–152. (Cité page 56.)

[104] Shor, P. W. Algorithms for quantum computation : discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science* (Washington, DC, USA, 1994), SFCS '94, IEEE Computer Society, pp. 124–134. (Cité pages 3, 52, et 53.)

[105] Siegel, C. L. *Lectures on the Geometry of Numbers*. Springer-Verlag, 1989. (Cité page 10.)

[106] Stehlé, D., and Steinfeld, R. Making NTRU as secure as worst-case problems over ideal lattices. In *Proceedings of the 30th Annual international conference on Theory and applications of cryptographic techniques : advances in cryptology* (Berlin, Heidelberg, 2011), EUROCRYPT'11, Springer-Verlag, pp. 27–47. (Cité pages 54 et 60.)

[107] Vallée, B. Opérateurs de Ruelle-Mayer généralisés et analyse en moyenne des algorithmes d'Euclide et de Gauss. *Acta Arithmetica 81*, 2 (1997), 101–144. (Cité page 78.)

[108] Vallée, B. Gauss' algorithm revisited. *Journal of Algorithms 12* (1991), 556–572. (Cité pages 35 et 146.)

[109] Vallée, B. Dynamics of the binary Euclidean algorithm : functional analysis and operators. *Algorithmica 22* (1998), 660–685. (Cité page 78.)

[110] Vallée, B., and Flajolet, P. The lattice reduction algorithm of Gauss : an average case analysis. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science* (Washington, DC, USA, 1990), SFCS '90, IEEE Computer Society, pp. 830–839 vol.2. (Cité pages 79 et 146.)

[111] Vallée, B., Girault, M., and Toffin, P. How to guess l-th roots modulo n by reducing lattices bases. *Comptes-rendus de AAECC-88, Rome, Lectures Notes in Computer Science 357* (1988), 427–442. (Cité pages 45 et 47.)

[112] Vallée, B., and Vera, A. Lattice reduction in two dimensions : analyses under realistic probabilistic models. *DMTCS Proceedings 0*, 1 (2008). (Cité pages 66, 79, 83, 126, et 146.)

[113] Vallée, B., and Vera, A. Probabilistic analyses of lattice reduction algorithms. In *The LLL Algorithm*, P. Q. Nguyen and B. Vallée, Eds., Information Security and Cryptography. Springer Berlin Heidelberg, 2010, pp. 71–143. (Cité pages 66, 79, et 83.)

[114] van Emde Boas, P. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Tech. Rep. 81-04, Math Inst., University of Amsterdam, Amsterdam, 1981. Available at author's home page. (Cité page 23.)

[115] Verheul, E. R., and van Tilborg, H. C. A. Cryptanalysis of 'less short' RSA secret exponents. *Appl. Algebra Eng. Commun. Comput. 8*, 5 (1997), 425–435. (Cité page 63.)

[116] Wiener, M. J. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory 36*, 3 (1990), 553–558. (Cité page 63.)