Géométrie des réseaux euclidiens, gaussiennes et applications

Ecole Normale Supérieure - Département de Mathématiques et Applications

Marius Berthoumieux et Thomas Serafini

sous la direction de M. François Charles

6 août 2022

Table des matières

Introduction

Le problème SVP, algorithmes connus et applications

Ce mémoire est consacré à l'étude des réseaux euclidiens et de l'algorithme le plus rapide connu à ce jour pour résoudre le problème SVP. Un réseau est un sous-groupe discret Λ de $\mathbb{R}^m$, et est toujours de la forme $\Lambda = \mathbf{B}\mathbb{Z}^n$ pour $\mathbf{B} \in \mathcal{M}_{m,n}(\mathbb{R})$. Les vecteurs colonnes de $\mathbf{B}$ forment la base du réseau. La formulation informelle du problème SVP est alors la suivante :
« Etant donné la base $\mathbf{B}$ d'un réseau Λ, trouver $\mathbf{x} \in \Lambda$ non-nul de norme minimale. ».

Ce problème est très difficile à résoudre - les meilleurs algorithmes connus avant la publication de [ADRS15] étaient un algorithme en temps $2^{2,465n+o(n)}$ dû à [PS09] et un algorithme déterministe en temps $\tilde{O}(4^n)$ dû à [MV13] (où n est le rang du réseau). L'algorithme que nous présentons représente une amélioration considérable de ce résultat, puisqu'il a une complexité de $2^{n+o(n)}$ en temps, mais n'est pas déterministe : il a une probabilité exponentielle en $-n$ de ne pas retourner un petit vecteur.

Leur intérêt se trouve notamment en cryptographie à clé publique - on citera par exemple le système d'encryption GGH [GGH] où un message est encodé sous la forme d'un $n-$uplet d'entiers, la clé publique est une base $\mathbf{B}'$ d'un réseau, et la clé privée est une "bonne" base $\mathbf{B}$ (composée par exemple de petits vecteurs) du même réseau. Casser ce système cryptographique nécessite d'être capable, à partir d'une base $\mathbf{B}'$ d'un réseau, d'en calculer une bonne base $\mathbf{B}$, ce qui fait assez rapidement apparaître SVP ou des problèmes analogues comme CVP, où l'on cherche le vecteur d'un réseau Λ le plus proche d'un $\mathbf{u} \in \mathbb{R}^n$ donné.

L'intérêt des problèmes de réseaux ne se limite pas à la cryptographie : beaucoup de problèmes de combinatoire peuvent en effet se réduire à la recherche de petits vecteurs dans des réseaux de grande dimension. Citons par exemple la preuve que la conjecture de Mertens est fausse par Andrew Odlyzko et Herman te Riele [OR85] utilisant l'algorithme LLL de réduction de base.

L'algorithme d'échantillonnage gaussien

Nous présentons dans ce mémoire un algorithme dû à Divesh Aggarwal, Daniel Dadush, Oded Regev et Noah Stephen-Davidovitz dans [ADRS15] et [ADRS18], qui, étant donnée la base $\mathbf{B}$ d'un réseau Λ de rang n fournie en entrée, retourne un de ses petits vecteurs avec probabilité $1 - \exp\big(-\Omega(n)\big)$, en temps $2^{n+o(n)}$.

La première partie de ce mémoire sera dédiée à des définitions, ainsi que des résultats et algorithmes classiques sur les réseaux euclidiens comme l'algorithme LLL (1.13), la formule sommatoire de Poisson (1.20) ou l'inégalité de transférence (1.27).

Nous nous intéresserons ensuite à la résolution de SVP par échantillonnage gaussien : on aura, lors de notre étude préliminaire des réseaux euclidiens, constaté l'importance de la gaussienne discrète ρ_s : $\mathbf{x} \mapsto e^{-\pi||\mathbf{x}||^2/s^2}$. La mesure de probabilité associée μ_s^Λ possède par ailleurs une propriété qui la rend intéressante pour SVP : les vecteurs les plus probables, après $\mathbf{0}$, sont les $\mathbf{x} \in \Lambda$ vérifiant $||\mathbf{x}|| = \lambda_1(\Lambda)$. On introduit donc dans (2.3) le problème DGS qui formalise l'idée de "tirer au sort selon μ_s^Λ. On aimerait donc pouvoir tirer au sort selon μ_s^Λ pour un paramètre s bien choisi - en effet, si s est trop faible, la probabilité d'obtenir $\mathbf{x} \neq \mathbf{0}$ risque d'être excessivement faible, alors que si s est trop grand, on court le risque de tirer trop de vecteurs différents de $\mathbf{x}$.

On observera dans (2.2) et (2.3) que s'il est théoriquement facile, modulo quelques bornes sur ρ_s, de réduire SVP à DGS, résoudre DGS est en fait a priori très difficile, au moins à bas paramètre. En effet, lorsque s est grand devant $\lambda_1(\Lambda)$, on dispose d'un moyen efficace de tirer selon μ_s^Λ, donné par (2.12).

C'est ici qu'intervient l'idée principale de l'article [ADRS15] : faire des moyennes. Si $\mathbf{X}, \mathbf{Y}$ indépendants suivent une loi normale centrée de paramètre σ^2, alors il est connu que leur moyenne suit une normale centrée de paramètre $\sigma^2/2$. On peut se demander dans quelle mesure cette propriété reste vraie quand on passe à la gaussienne discrète - et la réponse détaillée se trouve dans le lemme (2.15).

Cet effort ne sera cependant pas suffisant pour pouvoir tirer selon la gaussienne discrète : si l'on tire selon μ_s^Λ, on aura approximativement $\rho_s(c)$ vecteurs dans chaque classe c modulo Λ, alors que le moyennage nécessite de disposer de $\rho_s(c)^2$ vecteurs dans chaque classe c. Ce problème peut être résolu en utilisant un algorithme d'échantillonnage selon la loi carré (3.1) pour rejeter certains vecteurs.

On introduit alors l'outil principal de [ADRS18], les mélanges de Gaussiennes (3.4), qui permettent de montrer que si un ensemble de vecteurs est trouvable par un algorithme avec rejet, alors il est trouvable par le même algorithme sans l'étape de rejet (3.11). Le résultat est un algorithme étonnamment simple (3.12) qui résout SVP en temps $2^{n+o(n)}$.

Quelques mots sur les problèmes algorithmiques

Les problèmes considérés ici seront des problèmes algorithmiques. Un problème est donc caractérisé par la forme de ses entrées et par la sortie attendue. On appelle une instance d'un problème la donnée d'une entrée valide.

On distingue les problèmes décisionnels, où il s'agit juste d'accepter ou de rejeter une entrée, des problèmes d'évaluation ou il s'agit de donner une sortie d'un certain type qui est la solution de l'instance considérée.

Pour le sudoku, on peut par exemple prendre le problème d'évaluation donnée par, en entrée une grille de sudoku résoluble et en sortie attendue la grille d'entrée résolue. Un problème de décision associé est donné par, en entrée une grille partiellement remplie et une grille complétée et qui accepte l'entrée si et seulement si la grille complétée est une solution de la première grille. On peut aussi penser au problème qui prend en entrée une grille et l'accepte si et seulement si elle est résoluble. Le problème SVP tel que nous l'introduisons est donc un problème d'évaluation.

Un algorithme résout un problème si, pour n'importe quelle instance du problème, il renvoie une solution valide en ayant accès uniquement à l'entrée du problème. Pour les algorithmes probabilistes, on s'autorise une certaine probabilité d'erreur qui dépend uniquement de l'entrée du problème et qui est précisée avec l'algorithme

On dira qu'il existe une réduction d'un problème P' à un problème P si on dispose un algorithme qui résout le problème P' faisant appel à un algorithme qui résout le problème P. Cela correspond à la notion de Turing-reduction en informatique.

Si l'on prend par exemple P le problème qui demande, étant donné un entier n, d'en trouver un facteur premier, et P' qui demande, étant donné un entier n, d'en trouver sa décomposition en facteurs premiers, P' se réduit à P. En effet, si $\mathcal{A}$ est un algorithme résolvant P, alors il suffit d'appeler récursivement $\mathcal{A}$ sur n/p où p est un facteur premier de n trouvé par $\mathcal{A}$.

Introduisons également la notion de complexité en temps d'un algorithme comme le nombre maximal d'opérations élémentaires (principalement des additions binaires) qu'il effectuera avant de donner un retour pour une entrée de taille donnée(en bits).

On peut mesurer la complexité dans le pire cas ou la complexité moyenne. Précisons que « complexité moyenne »désigne la moyenne sur les entrées des nombres d'opérations, pas la moyenne sur une seule et même entrée des temps possibles pour un algorithme probabiliste. Nous mesurerons la complexité dans le pire cas de notre algorithme, et toute mention de « temps moyen »ou de « complexité moyenne »fera référence à la moyenne pour une entrée donnée des différents temps possibles du fait du caractère aléatoire

de l'algorithme.

Il est utile de noter que, lors d'une réduction, la complexité en temps de l'algorithme qui opère la réduction vient s'ajouter à la complexité de l'algorithme initial.

Pour mesurer la complexité en temps d'un algorithme, on introduit les notations suivantes pour $f, g : \mathbb{N} \to \mathbb{R}$:

- $g = O(f)$ s'il existe $C > 0$ tel que pour tout n assez grand, $g(n) \leqslant Cf(n)$. On dira alors éventuellement que g est bornée ou dominée par f.
- $g = \Omega(f)$ s'il existe $C > 0$ tel que pour tout n assez grand, $g(n) \geqslant Cf(n)$ (ce qui revient à dire $f = O(g)$). On dira que g domine f.
- $g = \Theta(f)$ si f domine g et g domine f.
- $g = o(f)$ si g/f tend vers 0 quand n tend vers l'infini (pour peu que f ne soit pas nulle une infinité de fois). On dira alors que g est négligeable devant f.
- $g = \tilde{O}(f)$ s'il existe k tel que $g = O\big(f \cdot \log^k(f)\big)$

1 Généralités sur les réseaux

Cette première partie s'inspire notamment de [MG] pour présenter des définions, algorithmes et théorèmes classiques sur les réseaux euclidiens.

1.1 Premières définitions

Définition 1.1. *Un **réseau euclidien** est la donnée d'un couple $(\Lambda, \langle ., . \rangle)$ où Λ est un groupe abélien libre de rang fini et $\langle ., . \rangle : \Lambda \times \Lambda \to \mathbb{R}$ est une forme $\mathbb{Z}$-bilinéaire symétrique définie positive sur Λ.*

Le produit scalaire sur Λ lui fournit une structure d'espace métrique discret, et permet d'associer une norme à ses éléments. Nous introduisons également l'objet qui nous occupera principalement pour ce mémoire.

Définition 1.2. *Un réseau de $\mathbb{R}^m$ est caractérisé par la donnée d'une matrice $\mathbf{B} \in \mathcal{M}_{m,n}(\mathbb{R})$ de rang n. On peut alors définir $\Lambda := \mathbf{B}\mathbb{Z}^n \subset \mathbb{R}^m$.*

Un réseau de $\mathbb{R}^m$ est muni d'une structure de réseau par le produit scalaire euclidien ambient, ce qui justifie l'appellation.

On peut d'ailleurs étendre cette définition à un espace euclidien E de dimension m, et appeler "réseau de E" un sous groupe discret de E. Le choix d'une base orthonormée de E fournit une isométrie avec $\mathbb{R}^m$ qui envoie les réseaux de E sur ceux de $\mathbb{R}^m$.

Par ailleurs, étant donné un groupe abélien libre de rang fini Λ et un produit scalaire euclidien $\langle ., . \rangle$ à $\mathbf{B}\mathbb{Z}^n \subset \mathbb{R}^n$, on peut considérer le produit tensoriel de $\mathbb{Z}-$modules $\Lambda \otimes_{\mathbb{Z}} \mathbb{R}$, que l'on munit d'une structure de $\mathbb{R}$-espace vectoriel en décrétant que $\alpha \cdot (\mathbf{v} \otimes x) := \mathbf{v} \otimes (\alpha x)$. On note $\Lambda_{\mathbb{R}}$ cet espace vectoriel, que l'on munit de la norme induite par le produit scalaire sur Λ étendu par $\mathbb{R}-$linéarité, à savoir $\langle \mathbf{v}_1 \otimes x_1, \mathbf{v}_2 \otimes x_2 \rangle := x_1 x_2 \langle \mathbf{v}_1, \mathbf{v}_2 \rangle$.

Signalons finalement un cas intéressant de réseau, le cas où $n = m$: alors on a $\mathbf{B} \in \mathrm{GL}(n, \mathbb{R})$. Comme il est toujours possible de se ramener à ce cas, par construction de $\Lambda_{\mathbb{R}}$ ou définition de $E := \mathrm{vect}(\Lambda)$, puis choix d'une isométrie avec $\mathbb{R}^n$, nous nous placerons dans ce cadre dès que l'on veut prouver des formules.

Dans le cas d'un algorithme, la donnée est en général la base $\mathbf{B}$ du réseau.

Définition 1.3 (Minima successifs)**.** *Soit Λ un réseau euclidien de dimension n.*
Les minimas du réseau sont les $\lambda_k(\Lambda), 1 \leqslant k \leqslant n$ tels que :

$$\lambda_k(\Lambda) := \inf\left\{\max_{1\leq i\leq k} ||\mathbf{x}_i|| \ : \mathbf{rg}(\mathbf{x}_1,...,\mathbf{x}_k) = k, \mathbf{x}_1,...,\mathbf{x}_k \in \Lambda\right\}$$

ou, de manière équivalente :

$$\lambda_k(\Lambda) := \inf\left\{r > 0 \ : \dim\big(\mathrm{Vect}(\Lambda \cap \mathcal{B}(0,r))\big) \geqslant k\right\}.$$

On notera souvent $\lambda_k(\Lambda) = \lambda_k$ quand cela n'est pas ambigu.

Il convient de faire attention au fait que $\lambda_k(\Lambda)$ n'est pas nécessairement le $k-$ème plus petit vecteur du réseau : on peut par exemple penser à $\Lambda = \mathbb{Z} \oplus 4\mathbb{Z} \subset \mathbb{R}^2$: alors, $\lambda_1(\Lambda) = 1$ et $\lambda_2(\Lambda) = 4$.

On s'autorisera à les noter simplement λ_k quand aucune confusion ne sera possible

Nous sommes maintenant en mesure de formuler le problème qui occupera le reste de ce mémoire :

Définition 1.4 (Shortest Vector Problem)**.**
Entrée : Un réseau Λ représenté par les coordonnées d'une base dans $\mathbb{R}^m$
Sortie : Les coordonées d'un vecteur $\mathbf{x}$ de Λ tel que $||\mathbf{x}|| = \lambda_1(\Lambda)$

Il existe aussi des variantes où l'on cherche seulement à trouver un vecteur qui est petit (inférieur en norme à λ_1 à une constante multiplicative près ne dépendant que de la dimension)

Définition 1.5 (Pavé élémentaire)**.** *Soit Λ un réseau de rang n. Un pavé élémentaire de Λ est un sous-ensemble de $\Lambda_{\mathbb{R}}$ de la forme :*

$$\mathcal{P}_{\mathbf{B}} := \left\{\sum_{i=1}^{n} t_i\mathbf{b}_i : t_1,...,t_n \in [0,1)^n\right\}$$

où $\mathbf{B} = (\mathbf{b}_1,...,\mathbf{b}_n)$ est une base de Λ.

Ces pavés ont tous le même volume. Pour s'en convaincre, on peut supposer $\Lambda \subset \mathbb{R}^n$ en choisissant une isométrie, qui ne changera donc pas le volume.

Si on prend alors $\mathbf{B}, \mathbf{B}' \in \mathrm{GL}(n,\mathbb{R})$ représentant deux bases de Λ, on a que $\mathbf{B}^{-1}\mathbf{B}'\mathbb{Z}^n = \mathbf{B}^{-1}\mathbf{B}\mathbb{Z}^n = \mathbb{Z}^n$. Mais pour $\mathbf{M} \in \mathrm{GL}(n,\mathbb{R})$, on a $\mathbf{M}\mathbb{Z}^n = \mathbb{Z}^n \iff \mathbf{M} \in \mathrm{GL}(n,\mathbb{Z})$.

Le volume de $\mathcal{P}_{\mathbf{B}}$ étant alors le déterminant de $\mathbf{B}$, il est invariant par multiplication à droite de $\mathbf{B}$ par une matrice de $\mathrm{GL}(n,\mathbb{Z})$, qui sera de déterminant ± 1.

Cette remarque permet de définir la notion de covolume d'un réseau.

Définition 1.6 (Covolume d'un réseau)**.** *Soit Λ un réseau. Le covolume de Λ est la mesure d'un pavé élémentaire.*

Plus concrètement, si $\mathbf{B}$ est la matrice représentant cette base par rapport à une base orthonormale de $\Lambda_{\mathbb{R}}$, on a :

$$\mathbf{covol}(\Lambda) = |\det(\mathbf{B})|$$

On définit également le degré de Λ comme :

$$\mathbf{deg}(\Lambda) = -\log(\mathbf{covol}(\Lambda))$$

On peut donner une définition (équivalente) du covolume en remarquant que la mesure de Lebesgue (définie par la norme) sur $\Lambda_{\mathbb{R}}$ passe au quotient $\Lambda_{\mathbb{R}}/\Lambda$, et on peut alors poser $\mathbf{covol}(\Lambda) = \mu(\Lambda_{\mathbb{R}}/\Lambda)$

Remarque. *Une méthode pour calculer le covolume de Λ réseau de dimension n de $\mathbb{R}^m$ quand m et n ne sont pas nécessairement égaux étant donnée une base $\mathbf{B}$ (vus comme vecteurs de $\mathbb{R}^m$) est de calculer son orthogonalisée de Gram-Schmidt $\tilde{\mathbf{B}}$. On a alors $\mathbf{covol}(\Lambda) = \prod_{i=1}^{n} ||\tilde{\mathbf{b}}_i||$. Ainsi, on obtient que $\mathbf{covol}(\Lambda) = \sqrt{\det\left({}^t\mathbf{BB}\right)}$ et si Λ est à coordonnées entières dans $\mathbb{R}^m$ alors $\mathbf{covol}(\Lambda)^2$ est un entier.*

Cette remarque permet également de définir $\mathbf{covol}(\Lambda)$ sans construction supplémentaire, en définissant simplement la matrice de Gram de Λ comme $\mathbf{G}(\Lambda) = \left(\langle \mathbf{b}_i, \mathbf{b}_j \rangle\right)_{1 \leqslant i,j \leqslant n}$ et en posant $\mathbf{covol}(\Lambda) := \sqrt{\det\left(\mathbf{G}(\Lambda)\right)}$.

Si Λ de dimension n est à coordonées entières dans $\mathbb{R}^m$, ça n'implique pas qu'il est à coordonnées entières dans le sous-espace de dimension n qu'il engendre.

Définition 1.7 (Réseau dual)**.** *Le réseau dual de Λ est défini comme :*

$$\Lambda^* := \mathrm{Hom}(\Lambda, \mathbb{Z}) \subset (\Lambda_{\mathbb{R}})^* .$$

On le munit de la norme opérateur associée à la norme euclidienne sur $\Lambda_{\mathbb{R}}$.

Il faut vérifier que Λ^* défini ainsi est effectivement un réseau. Pour s'en convaincre, il convient de rappeler qu'il existe une isométrie entre $\Lambda_{\mathbb{R}}$ et son dual, donnée par $\mathbf{u} \mapsto \langle \mathbf{u}, . \rangle$.

En se ramenant au cas de $\mathbb{R}^n$ et de son dual, le produit scalaire peut être vu comme un produit matriciel, et alors pour $\mathbf{u} \in \mathbb{R}^n$, on a $\mathbf{u} \cdot \mathbf{x} \in \mathbb{Z}$ pour tout $\mathbf{x} \in \Lambda$ si et seulement si $\mathbf{u} \cdot \mathbf{Bk} \in \mathbb{Z}$ pour tout $\mathbf{k} \in \mathbb{Z}^n$, ce qui revient à dire ${}^t\mathbf{Bu} \in \mathbb{Z}^n$ - ainsi, la matrice correspondant à la base de Λ^* est ${}^t\mathbf{B}^{-1}$. Ce fait implique en particulier que $\mathbf{covol}(\Lambda^*) = \mathbf{covol}(\Lambda)^{-1}$

Cette identification isométrique entre Λ^* et ${}^t\mathbf{B}^{-1}\mathbb{Z}^n$ sera très utile quand on voudra montrer des formules et des inégalités, comme dans (1.19).

1.2 Théorèmes de Minkowski

Nous poursuivons cette discussion des propriétés des réseaux en présentant un premier théorème de borne sur les minima des réseaux.

Théorème 1.8 (deuxième théorème de Minkowski)**.** *Soit Λ un réseau de dimension n, on a alors :*

$$\left(\prod_{i=1}^{n} \lambda_i\right)^{1/n} \leqslant \sqrt{n}\, \mathbf{covol}(\Lambda)^{1/n}$$

avec une inégalité stricte si $n \geqslant 2$.

On obtient comme conséquence directe :

Corollaire 1.9 (premier théorème de Minkowski)**.** *Soit Λ un réseau de dimension n, on a alors :*

$$\lambda_1(\Lambda) \leqslant \sqrt{n}\, \mathbf{covol}(\Lambda)^{1/n}.$$

Pour prouver le résultat on va avoir besoin du lemme :

Lemme 1.10 (Théorème des corps convexes de Minkowski)**.** *Soit Λ un réseau de $\mathbb{R}^n$ de dimension n, et $S \subset \mathbb{R}^n$ convexe et symétrique par rapport à l'origine.*
Si $\mu(S) > 2^n \mathbf{covol}(\Lambda)$, alors A contient un point non nul de Λ.

Démonstration. Choisissons une base $\mathbf{B}$ de Λ. Définissons $(\mathcal{P}_\mathbf{x})_{\mathbf{x}\in\Lambda}$ par $\mathcal{P}_0 := \mathcal{P}_\mathbf{B}$ et $\mathcal{P}_\mathbf{x} = \mathbf{x} + \mathcal{P}_0$: ces ensembles forment une partition de $\mathbb{R}^n$.
Ainsi les $S_\mathbf{x} := \frac{1}{2}S \cap \mathcal{P}_\mathbf{x}$ forment une partition de $\frac{1}{2}S$. Supposons alors qu'il n'existe aucun élément non-nul de Λ dans S : alors, on montre que pour $\mathbf{x} \neq \mathbf{y}$, $(S_\mathbf{x} - \mathbf{x}) \cap (S_\mathbf{y} - \mathbf{y})$ est vide.

En effet, si $\mathbf{w}$ appartient à l'intersection, on peut l'écrire $\mathbf{w} = \mathbf{u}/2 - \mathbf{x} = \mathbf{v}/2 - \mathbf{y}$ avec $\mathbf{u}, \mathbf{v} \in S$. On en déduit que $\frac{\mathbf{u}-\mathbf{v}}{2} = \mathbf{x} - \mathbf{y} \in \Lambda$. Mais comme S est symétrique par rapport à l'origine, $-\mathbf{v} \in S$. La convexité de S implique alors $\frac{\mathbf{u}-\mathbf{v}}{2} = \mathbf{x} - \mathbf{y} \in S$. On observe finalement que $S_\mathbf{x} - \mathbf{x} \subset \mathcal{P}_0$.

On a alors

$$\mu(\mathcal{P}_0) \geqslant \mu\left(\bigcup_{\mathbf{x}\in\Lambda} (S_\mathbf{x} - \mathbf{x})\right) = \sum_{\mathbf{x}\in\Lambda} \mu(S_\mathbf{x})$$

car Λ est dénombrable.
On en déduit $\mathbf{covol}(\Lambda) = \mu(\mathcal{P}_0) \geqslant \mu(\frac{1}{2}S) > \mathbf{covol}(\Lambda)$, ce qui est absurde. Il existe donc un élément non-nul de Λ dans S.

□

Démonstration du théorème 1.8. Le cas n=1 est trivial, on suppose donc $n \geqslant 2$. Le théorème porte uniquement sur les longueurs des vecteurs et le covolume de Λ, on peut donc supposer que Λ est plongé dans $\mathbb{R}^n$.
Soit $(\mathbf{x}_i)_{1\leqslant i\leqslant n}$ une famille de vecteurs de Λ qui qui atteignent les minima ($||\mathbf{x}_i|| = \lambda_i$) (Une telle famille existe parce que Λ est discret)- alors, ils forment une base de $\mathbb{R}^n$.
Notons également $(\tilde{\mathbf{x}}_i)_{1\leqslant i\leqslant n}$ la base obtenue par orthogonalisation de Gram-Schmidt. Supposons par l'absurde que

$$\left(\prod_{i=1}^{n} \lambda_i\right)^{1/n} \geqslant \sqrt{n}\, \mathbf{covol}(\Lambda)^{1/n}$$

On définit alors $\mathbf{L}$ linéaire par son action sur la base $(\tilde{\mathbf{x}}_i)$ en posant $\mathbf{L}(\tilde{\mathbf{x}}_i) := \tilde{\mathbf{x}}_i \lambda_i \tilde{\mathbf{x}}_i$, et on pose $\mathcal{B}$ la boule unité de $\mathbb{R}^n$.

$$\begin{aligned}
\mu(\mathbf{L}(\mathcal{B})) &= \prod_{i=1}^{n} \lambda_i \mu(\mathcal{B}) && \text{(Par Fubini-Tonelli)} \\
&\geqslant (\sqrt{n})^n \mu(\mathcal{B}) \mathbf{covol}(\Lambda) \\
&= \mu(\sqrt{n}\mathcal{B}) \mathbf{covol}(\Lambda).
\end{aligned}$$

Mais $\mu(\sqrt{n}\mathcal{B}) > 2^n$ puisque la boule contient l'hypercube $[-1, 1]^n$ de côté 2. Le lemme donne alors l'existence de $\mathbf{x} = c_1\tilde{\mathbf{x}}_1 + \cdots + c_n\tilde{\mathbf{x}}_n \in U$ tq $\mathbf{y} := \mathbf{L}(\mathbf{x}) \in \Lambda$.
En notant k entier maximal tel que $c_k \neq 0$ et $k' < k$ entier minimal tel que $\lambda_{k'} = \lambda_k$ on a, comme $c_k \neq 0$ que $\mathbf{y}$ est linéairement indépendant de $(\mathbf{x}_1, \ldots, \mathbf{x}_{k'-1})$ et comme $||\mathbf{x}|| < 1$, on trouve que $||\mathbf{y}|| < \lambda_{k'}$ - cela donne une famille de k' vecteurs indépendants dont le maximum des normes est strictement inférieur à $\lambda_{k'}$, ce qui est absurde. □

1.3 L'algorithme LLL

Une première idée pour chercher des petits vecteurs dans un réseau est de chercher des vecteurs "presque orthogonaux". En effet dans le cas des résaux de dimension 2 par exemple, si l'on dispose d'un vecteur, on peut en trouver un autre petit non colinéaire en cherchent le plus orthogonal possible (qui forme toujours une base avec le premier vecteur).

C'est sur cette idée que repose le premier algorithme que nous allons considérer,qui est présenté dans [MG] et dû à Lenstra, Lenstra et Lovász, et repose sur la réduction de base.

On considèrera des réseaux dans $\mathbb{R}^n$ et on se donnera une base $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ ainsi que son orthogonalisée de Gram-Schmidt $\tilde{\mathbf{B}} = (\tilde{\mathbf{b}}_1, \dots, \tilde{\mathbf{b}}_n)$ donnée récursivement par $\tilde{\mathbf{b}}_1 = \mathbf{b}_1$, et

$$\tilde{\mathbf{b}}_{i+1} = \mathbf{b}_{i+1} - \sum_{k=1}^{i} \frac{\left\langle \mathbf{b}_{i+1}, \tilde{\mathbf{b}}_k \right\rangle}{||\tilde{\mathbf{b}}_k||^2} \tilde{\mathbf{b}}_k.$$

On supposera de plus que les vecteurs du réseau sont à coordonnées entières, on peut se ramener, par une homotétie, à ce cas si on a un réseau à coordonnées rationnelles. Cela se justifie dans le contexte d'algorithme puisqu'une machine ne stocke de toute façon pas de manière fidèle les réels.

On a alors :

Définition 1.11 (Base LLL réduite)**.** *Soit $\delta > 0$, Λ un réseau de $\mathbb{R}^m$ dimension n.*
Une base $\mathbf{B}$ de Λ est dite δ-LLL réduite si :

- $\forall 1 \leqslant i < j \leqslant n, \mu_{j,i} := \frac{\left\langle \mathbf{b}_j, \tilde{\mathbf{b}}_i \right\rangle}{||\tilde{\mathbf{b}}_i||^2} \leqslant \frac{1}{2}$
- $\forall 1 \leqslant i \leqslant n-1, \delta||\pi_i(\mathbf{b}_i)||^2 = \delta||\tilde{\mathbf{b}}_i||^2 \leqslant ||\pi_i(\mathbf{b}_{i+1})||^2$
  où π_i est la projection orthogonale sur $\mathrm{Vect}(\tilde{\mathbf{b}}_j)_{i \leqslant j \leqslant n}$.

Cette condition va imposer que le premier vecteur de la base ne soit pas trop grand. Plus précisément :

Proposition 1.12. *Soit Λ un réseau de dimension n, $\delta > \frac{1}{4}$.*
Si $\mathbf{B}$ est une base δ-LLL réduite de Λ, alors $||\mathbf{b}_1|| \leqslant \lambda_1(\delta - \frac{1}{4})^{\frac{1-n}{2}}$.

Démonstration. Premièrement, on remarque que pour $1 \leqslant i \leqslant n-1$,

$$\begin{aligned} \delta||\tilde{\mathbf{b}}_i||^2 &\leqslant ||\pi_i(\mathbf{b}_{i+1})||^2 \\ &= ||\tilde{\mathbf{b}}_i \mu_{i+1,i} + \tilde{\mathbf{b}}_{i+1}||^2 \\ &\leqslant \frac{1}{4}||\tilde{\mathbf{b}}_i||^2 + ||\tilde{\mathbf{b}}_{i+1}||^2 \end{aligned}$$

donc $||\mathbf{b}_{i+1}|| \geqslant \left(\delta - \frac{1}{4}\right)||\tilde{\mathbf{b}}_i||$. Puis, comme $\delta \in]0,1[$, on a pour $1 \leqslant i \leqslant n$, l'inégalité$||\mathbf{b}_{i+1}|| \geqslant \left(\delta - \frac{1}{4}\right)^{\frac{n-1}{2}} ||\mathbf{b}_1||$.

Finalement, il suffit de constater, par Cauchy-Schwarz, que $\forall k \in \mathbb{Z}^n \backslash \{0\}, ||k\mathbf{B}|| \geqslant ||\tilde{\mathbf{b}}_\mathbf{i}||$ pour un certain $i, 1 \leqslant i \leqslant n$ donc $\lambda_1 \geqslant (\delta - \frac{1}{4})^{\frac{n-1}{2}} ||\mathbf{b}_1||$ □

Introduisons donc l'algorithme de réduction de base présenté par Lenstra, Lenstra et Lovász en 1982.

Algorithme 1.13 (LLL)**.**

Entrée : Une base $\mathbf{B} = (\mathbf{b}_i)_{1 \leqslant i \leqslant n}$ d'un réseau Λ de dimension $n \in \mathbb{N}$ et un paramètre δ
Sortie : Une base $\mathbf{B}'$ de Λ qui est $\delta-$LLL-réduite.

Tant que $\mathbf{B}$ n'est pas réduite :
  pour $1 \leqslant i \leqslant n$ dans l'ordre croissant et les $1 \leqslant j < i$ dans l'ordre décroissant :
  (i) remplacer $\mathbf{b}_i$ par $\mathbf{b}_i - k\mathbf{b}_j$ avec k entier, de manière a assurer que $\mu_{i,j} \leqslant \frac{1}{2}$
  (ii) Si il existe $1 \leqslant i \leqslant n-1$ tq $\delta||\tilde{\mathbf{b}}_i||^2 > ||\pi_i(\mathbf{b}_{i+1})||^2$:
    échanger $\mathbf{b}_i$ et $\mathbf{b}_{i+1}$
renvoyer $\mathbf{B}$

Il est clair que si l'algorithme se termine, la base obtenue est δ-LLL réduite, il reste à voir la terminaison.

On va pour voir cela utiliser le covolume des réseaux Λ_i pour $1 \leqslant i \leqslant n$ qui sont les réseaux générés par les i premiers vecteurs de la base a une étape donnée de l'algorithme. On note avec un exposant (k) les valeurs des variables après la k-ième boucle de l'algorithme.

On pose

$$d_k = \prod_{i=1}^{n} \mathbf{covol}\left(\Lambda_i^{(k)}\right)^2$$

qui est un entier d'après les remarques précédentes.

On remarque d'après l'expression du covolume en fonction des vecteurs de $\tilde{\mathbf{B}}$ que $\mathbf{b}_k$ est inchangé par l'étape (i) de l'algorithme et si on prend i sélectionné à l'étape (ii) on a $\forall j \neq i, \Lambda_j^{(k)} = \Lambda_j^{(k+1)}$ et ainsi

$$\begin{aligned}\frac{||\mathbf{b}_{k+1}||}{||\mathbf{b}_k||} &= \frac{\mathbf{covol}\left(\Lambda_i^{(k+1)}\right)^2}{\mathbf{covol}\left(\Lambda_i^{(k)}\right)^2} \\ &= \frac{||\pi_i^{(k)}(\mathbf{b}_{i+1}^{(k)})||^2}{||\tilde{\mathbf{b}}_i||^2} \\ &< \delta.\end{aligned}$$

donc $d_k \leqslant \delta^k d_0 < 1$ pour $k = -\frac{\log(d_0)}{\log(\delta)}$. Pour ce k l'algorithme a donc déjà renvoyé une base et le calcul de d_0 est en temps polynomial en l'entrée donc son logarithme est polynomial en l'entrée. Et chaque boucle s'exécute en temps polynomial car les nombres considérés sont tous bornés par des polynômes en l'entrée.

1.4 A la recherche de petites bases

Pour travailler sur un réseau, on aura souvent besoin d'en connaître une "petite" base (en tout cas une base dont on contrôle la taille). On utilisera cette notion.

Définition 1.14 (Base γ-réduite)**.** *Soit $\gamma \geq 1$, Λ un réseau de $\mathbb{R}^m$ de dimension n. Une base $\mathbf{B}$ de Λ est dite $\gamma-$réduite si :*

- $||\mathbf{b}_1|| \leqslant \gamma\lambda_1(\Lambda)$
- $\left(\pi_1(\mathbf{b}_k)\right)_{2 \leqslant k \leqslant n}$ *est une base γ-réduite de $\pi_1(\Lambda)$.*

Nous aurons besoin, pour appliquer 2.7, du théorème suivant :

Théorème 1.15 (Petites bases)**.** *Il existe un algorithme qui, étant donné un paramètre $a \geq 2$ et un réseau Λ de dimension n, donne une base $a^{n/a}$-réduite de Λ en temps* $\exp(O(a)) \cdot \mathrm{poly}(n)$

Ce théorème est difficile à prouver. Sa version la plus facile, mais avec des bornes en temps un peu moins efficaces, est trouvable dans [Sch87], et une version plus efficace est trouvable dans [AKS01] (Corollaire 15).

1.5 Formule sommatoire de Poisson

Nous commençons par une revue rapide de l'espace $\mathcal{S}(\mathbb{R}^n)$. Pour un multi-indice $\alpha = (\alpha_1, ..., \alpha_n) \in \mathbb{N}^n$, un vecteur $\mathbf{u} = (u_1, ..., u_n) \in \mathbb{R}^n$ et $\varphi : \mathbb{R}^n \to \mathbb{R}$ de classe $\mathcal{C}^\infty$, on note $|\alpha| := \sum_{j=1}^n \alpha_j$, $\mathbf{u}^\alpha := u_1^{\alpha_1} \cdot ... \cdot u_n^{\alpha_n}$, $\partial^\alpha \varphi = \frac{\partial^{|\alpha|}}{\partial u_1^{\alpha_1} ... \partial u_n^{\alpha_n}} \varphi$ et $\hat{\varphi}$ la transformée de Fourier de φ, pour laquelle on prendra la convention de la transformée de Fourier en fréquence.

Définition 1.16 (Transformée de Fourier)**.** *Soit $\varphi \in L^1(\mathbb{R}^n)$. On définit la transformée de Fourier de φ par*

$$\hat{\varphi}(\mathbf{v}) := \int_{\mathbb{R}^n} \varphi(\mathbf{u}) e^{-2i\pi \mathbf{u}\cdot\mathbf{v}} \mathrm{d}\mathbf{u}.$$

Cette définition n'est pas standard, mais ne diffère de la TF classique $\int_{\mathbb{R}^n} \varphi(\mathbf{u}) e^{-2i\pi \mathbf{u}\cdot\boldsymbol{\xi}} \mathrm{d}\mathbf{u}$ que par une dilatation de 2π (et éventuellement une renormalisation) : ses propriétés d'intégrabilité, ainsi que celles de ses dérivées, sont par conséquent inchangées.

On donne alors la définition suivante :

Définition 1.17 (Espace de Schwartz)**.** *L'espace de Schwartz $\mathcal{S}(\mathbb{R}^n)$ est défini comme l'ensemble des fonctions $\varphi : \mathbb{R}^n \to \mathbb{R}$ de classe $\mathcal{C}^\infty$ telles que pour tout multi-indices α, β, la fonction*

$$\mathbf{u} \mapsto \mathbf{u}^\beta \big(\partial^\alpha \varphi\big)(\mathbf{u})$$

est bornée sur $\mathbb{R}^n$.

Notons qu'il est équivalent de demander que pour tout $k \in \mathbb{N}$, $|\mathbf{u}|^k (\partial^\alpha \varphi)$ soit bornée.
La proposition suivante est utile dans la preuve de la formule sommatoire de Poisson, au moins si l'on veut se passer de trop d'hypothèses sur φ. Pour une preuve, on peut se référer par exemple aux notes de cours niveau M1/agrégation de Jean-Marc Bouclet sur la transformation de Fourier (https ://www.math.univ-toulouse.fr/ bouclet/Notes-de-cours-exo-exam/page-enseignement.html).

Proposition 1.18 (Propriétés des fonctions à décroissance rapide)**.** *Soit $\varphi \in \mathcal{S}(\mathbb{R}^n)$. Alors :*

- *pour tous multi-indices α, β, on a $\mathbf{u}^\beta \partial^\alpha \varphi \in \mathcal{S}(\mathbb{R}^n)$.*
- *pour $p \geqslant 1$ réel, $\varphi \in L^p(\mathbb{R}^n)$.*
- *$\hat{\varphi} \in \mathcal{S}(\mathbb{R}^n)$.*

On peut alors montrer la formule sommatoire de Poisson, qui sera au coeur des formules et inégalités sur la gaussienne discrète.

Proposition 1.19 (Formule sommatoire de Poisson)**.** *Soit $\varphi \in \mathcal{S}(\mathbb{R}^n)$. Alors :*

$$\sum_{\mathbf{k}\in\mathbb{Z}^n} \varphi(\mathbf{k}+\mathbf{u}) = \sum_{\mathbf{k}\in\mathbb{Z}^n} \hat{\varphi}(\mathbf{k}) e^{2i\pi \mathbf{k}\cdot\mathbf{u}}.$$

Démonstration. Définissons $F(\mathbf{u}) := \sum_{\mathbf{k}\in\mathbb{Z}^n} \varphi(\mathbf{k}+\mathbf{u})$. Comme F est $\mathbb{Z}^n$-périodique, on peut se restreindre à $\mathbf{u} \in [0,1]^n$. La décroissance rapide de φ implique alors la convergence normale de la somme par

comparaison série-intégrale, ce qui entraîne la continuité de F. On travaille alors dans $L^2(\mathbb{T}^n)$, identifié à $L^2([0,1]^n)$. Une base hilbertienne en est donnée par $(e_{\mathbf{k}})_{\mathbf{k}\in\mathbb{Z}^n}$, avec $e_{\mathbf{k}}(\mathbf{u}) = e^{2i\pi\mathbf{k}\cdot\mathbf{u}}$. Notons $S_N(F)$ la somme $\sum_{|\mathbf{k}|<N}\langle F, e_{\mathbf{k}}\rangle e_{\mathbf{k}}$. Le théorème de Riesz-Fischer assure la convergence au sens L^2 de $S_N(F)$ vers F : $\int_{[0,1]^n}|F(\mathbf{u}) - S_N(F)(\mathbf{u})|^2 \mathrm{d}\mathbf{u} \underset{N\to\infty}{\longrightarrow} 0$.

On vérifie alors $\langle F, e_{\mathbf{k}}\rangle = \hat{\varphi}(\mathbf{k})$:

$$\begin{aligned}
\langle F, e_{\mathbf{k}}\rangle &= \int_{[0,1]^n} \sum_{\mathbf{j}\in\mathbb{Z}^n} \varphi(\mathbf{j}+\mathbf{u}) e^{-2i\pi\mathbf{k}\cdot\mathbf{u}} \mathrm{d}\mathbf{u} \\
&= \sum_{\mathbf{j}\in\mathbb{Z}^n} \int_{[0,1]^n} \varphi(\mathbf{j}+\mathbf{u}) e^{-2i\pi\mathbf{k}\cdot\mathbf{u}} \mathrm{d}\mathbf{u} \\
&= \sum_{\mathbf{j}\in\mathbb{Z}^n} \int_{\mathbb{R}^n} \varphi(\mathbf{j}+\mathbf{u}) e^{-2i\pi\mathbf{k}\cdot\mathbf{u}} \mathbf{1}_{[0,1)^n}(\mathbf{u}) \mathrm{d}\mathbf{u} \\
&= \sum_{\mathbf{j}\in\mathbb{Z}^n} \int_{\mathbb{R}^n} \varphi(\mathbf{u}) e^{-2i\pi\mathbf{k}\cdot\mathbf{u}} \mathbf{1}_{[0,1)^n+\mathbf{j}}(\mathbf{u}) \mathrm{d}\mathbf{u} \\
&= \int_{\mathbb{R}^n} \varphi(\mathbf{u}) e^{-2i\pi\mathbf{k}\cdot\mathbf{u}} \sum_{\mathbf{j}\in\mathbb{Z}^n} \mathbf{1}_{[0,1)^n+\mathbf{j}}(\mathbf{u}) \mathrm{d}\mathbf{u}.
\end{aligned}$$

Les interversions somme-intégrale sont justifiées dans un premier temps par le fait que l'intégration concerne des fonctions continues et facilement dominées (par $\sum|\varphi(\mathbf{j})|$) sur un compact où la convergence est uniforme, et dans un second temps par le fait qu'on peut dominer uniformément la somme $\left|\sum_{|\mathbf{j}|\leq M} \varphi(\mathbf{u}) e^{2i\pi\mathbf{k}\cdot\mathbf{u}} \mathbf{1}_{[0,1)^n+\mathbf{j}}\right|$ par $|\varphi(\mathbf{u})|$ qui est intégrable sur $\mathbb{R}^n$.

On observe alors aisément que $\sum_{\mathbf{j}\in\mathbb{Z}^n} \mathbf{1}_{[0,1)^n+\mathbf{j}} = 1$, ce qui montre $\langle F, e_{\mathbf{k}}\rangle = \hat{\varphi}(\mathbf{k})$ - ce qui prouve que $S_N(F)$ converge vers une fonction limite $S(F)$.

Cela implique immédiatement que $S(F) = F$ presque partout : en effet, puisque $S_N(F)$ converge vers F dans L^2, il en existe une sous-suite qui converge presque partout vers F - le fait que $S_N(F)$ converge uniformément (et donc simplement) permet de conclure.

Puisque $\hat{\varphi}$ est également à décroissance rapide, la série $\sum\hat{\varphi}(\mathbf{k})e_{\mathbf{k}}$ converge aussi normalement, donc uniformément : cela assure la continuité de la fonction somme. L'égalité presque partout avec F, elle aussi continue, garantit en fait l'égalité partout.

□

On généralise alors cette formule à tout réseau Λ par changement de variable.

Théorème 1.20 (Formule sommatoire de Poisson, version réseaux)**.** *Soit $\Lambda := \mathbf{B}\mathbb{Z}^n$ un réseau de $\mathbb{R}^m$ et $\varphi \in \mathcal{S}(\mathbb{R}^m)$. Alors, pour $\mathbf{u} \in \mathrm{vect}(\Lambda)$:*

$$\varphi(\Lambda + \mathbf{u}) = \mathbf{covol}(\Lambda^*) \sum_{\mathbf{y}\in\Lambda^*} \hat{\varphi}(\mathbf{y}) e^{2i\pi\mathbf{y}\cdot\mathbf{u}}$$

Démonstration. On se ramène à $\Lambda \subseteq \mathbb{R}^n$ et $\mathbf{B} \in \mathrm{GL}(n,\mathbb{R})$ par choix d'une isométrie entre le sous-espace de $\mathbb{R}^m$ engendré par Λ et $\mathbb{R}^n$. Remarquons tout d'abord que φ est à décroissance rapide, alors $\chi_{\mathbf{B}} := \varphi(\mathbf{B}\,\cdot)$ l'est aussi pour $\mathbf{B} \in \mathrm{GL}(n,\mathbb{R})$.

On peut dans ce cas calculer la transformée de Fourier de $\chi_{\mathbf{B}}$:

$$\begin{aligned}
\hat{\chi}_{\mathbf{B}}(\mathbf{x}) &= \int_{\mathbb{R}^n} \chi_{\mathbf{B}}(\mathbf{u}) e^{2i\pi \mathbf{u}\cdot\mathbf{x}} \mathrm{d}\mathbf{u} \\
&= \int_{\mathbb{R}^n} \varphi(\mathbf{B}\mathbf{u}) e^{2i\pi \mathbf{u}\cdot\mathbf{x}} \mathrm{d}\mathbf{u} \\
&= \int_{\mathbb{R}^n} \varphi(\mathbf{v}) e^{2i\pi \mathbf{v}\cdot {}^t\mathbf{B}^{-1}\mathbf{x}} |\det(\mathbf{B})|^{-1} \mathrm{d}\mathbf{v} \qquad (\mathbf{v} = \mathbf{B}\mathbf{u}) \\
&= |\det(\mathbf{B})|^{-1} \hat{\varphi}\left({}^t\mathbf{B}^{-1}\mathbf{x}\right).
\end{aligned}$$

L'application de (1.19) à $\chi_{\mathbf{B}}$ donne :

$$\begin{aligned}
\sum_{\mathbf{x}\in\Lambda} \varphi(\mathbf{x}+\mathbf{u}) &= \sum_{\mathbf{k}\in\mathbb{Z}^n} \chi_{\mathbf{B}}(\mathbf{k}+\mathbf{B}^{-1}\mathbf{u}) \\
&= \sum_{\mathbf{k}\in\mathbb{Z}^n} \hat{\chi}_{\mathbf{B}}(\mathbf{k}) e^{2i\pi \mathbf{k}\cdot(\mathbf{B}^{-1}\mathbf{u})} \\
&= \det(\mathbf{B})^{-1} \sum_{\mathbf{k}\in\mathbb{Z}^n} \hat{\varphi}\left({}^t\mathbf{B}^{-1}\mathbf{k}\right) e^{2i\pi({}^t\mathbf{B}^{-1}\mathbf{k})\cdot\mathbf{u}} \\
&= \mathbf{covol}(\Lambda^*) \sum_{\mathbf{y}\in\Lambda^*} \hat{\varphi}(\mathbf{y}) e^{2i\pi \mathbf{y}\cdot\mathbf{u}}.
\end{aligned}$$

□

1.6 La gaussienne discrète

Nous introduisons à présent l'outil central de nos discussions.

Définition 1.21 (Gaussienne discrète)**.** *Pour un réseau Λ, on définit la gaussienne discrète de paramètre $s > 0$ sur Λ comme la fonction :*

$$\rho_s : \mathbf{x} \in \Lambda \mapsto \exp\left[-\frac{||\mathbf{x}||^2}{s^2}\right]$$

Si $s = 1$, on la notera simplement ρ. On notera également, pour $S \subset \Lambda$, $\rho_s(S) := \sum_{\mathbf{x}\in S} \rho_s(\mathbf{x})$, mais aussi μ_s^Λ pour désigner la mesure de probabilité sur Λ donnée par $\mu_s^\Lambda(\{\mathbf{x}\}) := \rho_s(\mathbf{x})/\rho_s(\Lambda)$

Une remarque est de mise avant de continuer : la définition de ρ "avec paramètre s" peut paraître redondante : en effet, en considérant le réseau $\Lambda_s := s^{-1}\Lambda$, les résultats valables pour ρ seront valables pour les ρ_s à quelques renormalisations près. Le paramètre s sera néanmoins un intermédiaire de calcul très utile, notamment pour (1.27).

On a alors le corollaire suivant de (1.20).

Corollaire 1.22 (Formule sommatoire de Poisson, gaussiennes)**.** *Soit Λ un réseau de rang n. Alors, pour $\mathbf{u} \in \Lambda_{\mathbb{R}}$:*

$$\rho_s(\Lambda + \mathbf{u}) = \mathbf{covol}(\Lambda^*) s^n \sum_{\mathbf{y}\in\Lambda^*} \rho_{1/s}(\mathbf{y}) e^{2i\pi \mathbf{y}\cdot\mathbf{u}}$$

Démonstration. On se ramène au cas $\Lambda \subseteq \mathbb{R}^n$ en choisissant une isométrie entre $\Lambda_{\mathbb{R}}$ et $\mathbb{R}^n$, qui identifie l'action de Λ^* au produit scalaire par l'élément de ${}^t\mathbf{B}^{-1}\mathbb{Z}^n$ correspondant. On applique ensuite (1.20) à $\varphi(\mathbf{x}) := \exp(-\pi||\mathbf{x}||^2/s^2)$, qui coïncide avec ρ_s sur Λ et est clairement dans $\mathcal{S}(\mathbb{R}^n)$. Pour la convention choisie pour la TF, on a $\hat{\varphi} = s^n \rho_{1/s}$, ce qui donne le résultat voulu. □

En notant $h^0(\Lambda) := \log(\rho(\Lambda))$ et $h^1(\Lambda) := \log(\rho(\Lambda^*))$, cette égalité se réécrit :

$$h^0(\Lambda) - h^1(\Lambda) = \mathbf{deg}(\Lambda)$$

De la formule sommatoire de Poisson, on peut tirer un grand nombre d'inégalités qui nous seront extrêmement utiles dans toute la suite : nous proposons ainsi d'établir un "formulaire" des fonctions ρ_s, qui listera toutes les égalités et inégalités que nous utiliserons dans la suite.

Nous aurons besoin, pour l'une de ces équations, du lemme suivant, tiré de [PS09].

Lemme 1.23 (Petits vecteurs, [PS09]). *Posons $\beta := 2^{0.401}$. Pour un réseau Λ de rang n, le nombre de vecteurs de taille au plus r est inférieur ou égal à $\beta^{n+o(n)} \left(\frac{r}{\lambda_1(\Lambda)}\right)^n$*

Proposition 1.24 (Formulaire). *Soit Λ réseau de rang n de $\mathbb{R}^n$. Les fonctions ρ_s vérifient ce qui suit.*

(i) Pour $s \geq 1$, $\mathbf{u} \in \mathbb{R}^n, r > 0$:

$$\rho_s\big((\Lambda+\mathbf{u})\backslash\mathcal{B}(\mathbf{0},r)\big) \geq e^{\pi\left(1-\frac{1}{s^2}\right)r^2}\rho\big((\Lambda+\mathbf{u})\backslash\mathcal{B}(\mathbf{0},r)\big) \tag{1.1}$$

(ii) Pour $s \geq 1$, $\mathbf{u} \in \mathbb{R}^n$:

$$\rho_s(\Lambda+\mathbf{u}) \leqslant s^n\rho(\Lambda) \tag{1.2}$$

(iii) Pour $\mathbf{u} \in \mathbb{R}^n$:

$$|\rho(\Lambda+\mathbf{u}) - \mathbf{covol}(\Lambda^*)| \leqslant \rho(\Lambda^*\backslash\{\mathbf{0}\})\mathbf{covol}(\Lambda^*) \tag{1.3}$$

(iv) Pour $s > \sqrt{2\pi/n}\lambda_1(\Lambda)$:

$$\rho_s(\Lambda) \leqslant 1 + \left(\frac{\beta^2 s^2 n}{2\pi e\lambda_1^2}\right)^{n/2+1}\beta^{o(n)} \tag{1.4}$$

(v) Pour $\mathbf{x},\mathbf{y} \in \mathbb{R}^n, s > 0$:

$$\rho_s(\mathbf{x})\rho_s(\mathbf{y}) = \rho_{s/\sqrt{2}}\left(\frac{\mathbf{x}+\mathbf{y}}{2}\right)\rho_{s/\sqrt{2}}\left(\frac{\mathbf{x}-\mathbf{y}}{2}\right) \tag{1.5}$$

Démonstration. (i) On remarque que $1 - 1/s^2 \geqslant 0$ car $s \geq 1$, et on peut développer.

$$\begin{aligned}
\rho_s\big((\Lambda+\mathbf{u})\backslash\mathcal{B}(\mathbf{0},r)\big) &= \sum_{\mathbf{x}\in\Lambda+\mathbf{u},||\mathbf{x}||\geqslant r} e^{-\pi\frac{||\mathbf{x}||^2}{s^2}} \\
&= \sum_{\mathbf{x}\in\Lambda+\mathbf{u},||\mathbf{x}||\geqslant r} e^{-\pi||\mathbf{x}||^2}e^{\pi\left(1-\frac{1}{s^2}\right)||\mathbf{x}||^2} \\
&\geqslant \sum_{\mathbf{x}\in\Lambda+\mathbf{u},||\mathbf{x}||\geqslant r} e^{-\pi||\mathbf{x}||^2}e^{\pi\left(1-\frac{1}{s^2}\right)r^2} \\
&\geqslant \rho\big((\Lambda+\mathbf{u})\backslash\mathcal{B}(\mathbf{0},r)\big)e^{\pi\left(1-\frac{1}{s^2}\right)r^2}.
\end{aligned}$$

(ii) $\rho_s(\Lambda+\mathbf{u}) \leq \rho_s(\Lambda)$ découle d'une application de l'inégalité triangulaire à (1.22). En remarquant que $\rho_{\frac{1}{s}} \leqslant \rho$ pour $s \geqslant 1$, on a :

$$\rho_s(\Lambda+\mathbf{u}) \leqslant \rho_s(\Lambda) = \mathbf{covol}(\Lambda^*)s^n\rho_{\frac{1}{s}}(\Lambda^*) \leqslant \mathbf{covol}(\Lambda^*)s^n\rho(\Lambda^*) = s^n\rho(\Lambda).$$

(iii) On applique la formule sommatoire de Poisson pour obtenir :

$$|\rho(\Lambda+\mathbf{u}) - \mathbf{covol}(\Lambda^*)| = \mathbf{covol}(\Lambda^*)\left|\sum_{\mathbf{y}\in\Lambda^*,\mathbf{y}\neq 0}\rho(\mathbf{y})e^{2i\pi\mathbf{y}\cdot\mathbf{u}}\right|.$$

Une application de l'inégalité triangulaire permet de conclure.

(iv) Quitte à remplacer s par s/λ_1, on peut supposer $\lambda_1 = 1$. Pour pouvoir utiliser (1.23), on va découper Λ en boules. On pose $r := 1 + 1/n$, et pour $k \geq 0$, on pose $\mathcal{B}_k := \mathcal{B}(\mathbf{0}, r^{k+1}) \backslash \mathcal{B}(\mathbf{0}, r^k)$. On a donc $\mathbb{R}^n \backslash \{\mathcal{B}(\mathbf{0}, 1)\} = \bigcup_{k \geq 0} \mathcal{B}_k$. Grâce à (1.23), on remarque que $|\Lambda \cap \mathcal{B}_k| \leqslant \beta^{n+o(n)} r^{kn}$, donc $\rho_s(\Lambda \cap \mathcal{B}_k) \leqslant e^{-\pi \frac{r^{2k}}{s^2}} \beta^{n+o(n)} r^{(k+1)n}$. On peut alors se lancer dans les calculs :

$$\begin{aligned}\rho_s(\Lambda) &= 1 + \sum_{k \geqslant 0} \rho_s(\Lambda \cap \mathcal{B}_k) \\ &\leqslant 1 + r^n \beta^{n+o(n)} \sum_{k \geq 0} e^{-\pi \frac{r^{2k}}{s^2}} r^{kn}.\end{aligned}$$

Remarquons que pour k grand devant s, n (dans le sens où $k \geqslant sP(n) \geqslant (1 + \log(2/\pi) + \log s)^2 \cdot P(n)$ où P est un polynôme quadratique ne dépendant que de la raison géométrique choisie), $\exp(-\pi r^{2k}/s^2) r^{kn}$ décroît plus vite qu'une suite géométrique choisie (la constante $\log(2/\pi)$ correspond simplement à une raison de e^{-1}, pratique pour les calculs) - en particulier, au moins pour $s > \sqrt{2\pi/n}$ la série des termes à partir de $k_0 = sP(n)$ est en $e^{-sP(n)} \leqslant e^{-n}$ - et la somme des k premiers termes est inférieure ou égale à :

$$\begin{aligned}\sum_{k=0}^{k_0} r^n e^{-\pi \frac{r^{2k}}{s^2}} r^{nk} &\leqslant seP(n) \max_{t \geq 1} e^{-\pi \frac{t^2}{s^2}} t^n \\ &\leqslant s\beta^{o(n)} e^{-n/2} \left(\frac{s^2 n}{2\pi}\right)^{n/2}.\end{aligned}$$

En effet, le maximum de $t \mapsto -at^2 + b\log(t)$ pour $a, b > 0$ est atteint en $\sqrt{b/2a}$.
On remarque finalement que pour $s > \sqrt{2\pi/n}$, on a :

$$s = \frac{\beta^2 s^2 n}{2\pi e} \cdot \frac{2\pi e}{\beta^2 s n} \leqslant \frac{\beta^2 s^2 n}{2\pi e} \beta^{o(n)}.$$

On a finalement bien l'inégalité asymptotique voulue.

(v) On utilise l'égalité du parallélogramme.

$$\begin{aligned}\rho_{s/\sqrt{2}}\left(\frac{\mathbf{x}+\mathbf{y}}{2}\right) \rho_{s/\sqrt{2}}\left(\frac{\mathbf{x}-\mathbf{y}}{2}\right) &= \exp\left(-\frac{||\mathbf{x}+\mathbf{y}||^2}{2s^2}\right) \exp\left(-\frac{||\mathbf{x}-\mathbf{y}||^2}{2s^2}\right) \\ &= \exp\left(-\frac{||\mathbf{x}+\mathbf{y}||^2 + ||\mathbf{x}-\mathbf{y}||^2}{2s^2}\right) \\ &= \exp\left(-\frac{||\mathbf{x}||^2 + ||\mathbf{y}||^2}{s^2}\right) \\ &= \rho_s(\mathbf{x})\rho_s(\mathbf{y}).\end{aligned}$$

□

1.7 Inégalité de transférence

Après cette introduction, nous allons prouver un premier résultat concret qui utilise la formule de Poisson, et la transformée de Fourier en général. Ce résultat ne sera pas utile dans la suite de nos

considérations : c'est cependant une bonne manière de clore cette partie introductive, puisqu'il permet d'obtenir un résultat a priori non-trivial à l'aide de gaussiennes.

Nous allons prouver l'inégalité de transférence, qui dit que pour un réseau Λ de rang n, on a :

$$1 \leqslant \lambda_1(\Lambda)\lambda_n(\Lambda^*) \leqslant n.$$

Notons que c'est une bien meilleure borne que ce que l'on peut obtenir par des outils plus naïfs, comme le théorème de Minkowski.
L'idée de la preuve repose sur la définition suivante.

Définition 1.25 (rayon de recouvrement)**.** *Soit Λ un réseau de $\mathbb{R}^n$ de rang maximal, $\mathbf{B}$ une base de Λ. On définit le rayon de recouvrement $\mu(\Lambda)$ par :*

$$\mu(\Lambda) := \sup_{\mathbf{x}\in\mathbb{R}^n} d(\mathbf{x},\Lambda) = \sup_{\mathbf{x}\in\overline{\mathcal{P}}_{\mathbf{B}}} d(\mathbf{x},\Lambda).$$

On montre la proposition suivante :

Proposition 1.26. *Soit Λ un réseau de $\mathbb{R}^n$ de rang maximal. Alors $\mu(\Lambda) \geqslant \frac{1}{2}\lambda_n(\Lambda)$.*

Démonstration. Par définition de λ_n, tout point de $\Lambda \cap \mathcal{B}(0,\lambda_n)$ est dans un certain hyperplan H de dimension $n-1$. En particulier, si on choisit $\mathbf{u}$ orthogonal à H, de norme $\frac{\lambda_n}{2}$ exactement, alors $\mathcal{B}\left(\mathbf{u},\frac{\lambda_n}{2}\right)$ ne peut contenir aucun point du réseau - d'une part, elle ne peut pas contenir des points de H, et d'autre part, puisque $\mathcal{B}\left(\mathbf{u},\frac{\lambda_n}{2}\right) \subseteq \mathcal{B}(\mathbf{0},\lambda_n)$, les points de H sont les seuls points de Λ candidats à être dans $\mathcal{B}\left(\mathbf{u},\frac{\lambda_n}{2}\right)$. □

Nous prouvons donc l'inégalité de transférence en nous inspirant largement de la preuve fournie par Oded Regev dans ses cours "Lattices in computer science" à l'université de Tel Aviv, qui prouve la borne suivante, avec n à la place de $n/2$. Nos modifications de la preuve nous amènent malheureusement à perdre les cas $n = 2$ et $n = 3-$ nous prouverons donc seulement l'inégalité de transférence pour $n > 3$.

Théorème 1.27 (Inégalité de transférence)**.** *Soit Λ un réseau de rang n.*

$$\lambda_1(\Lambda)\mu(\Lambda^*) \leqslant \frac{n}{2}$$

Démonstration. On se ramène au cas où $\Lambda \subseteq \mathbb{R}^n$ est un réseau de rang maximal. On procède par l'absurde : supposons $\lambda_1(\Lambda)\mu(\Lambda^*) > n/2$

Quitte à remplacer Λ par $t\Lambda$ pour un $t > 0$ approprié, on peut supposer $\lambda_1(\Lambda), \mu(\Lambda^*) > \sqrt{n/2}$: en effet, $\lambda_1(t\Lambda) = t\lambda_1(\Lambda)$ et $\mu\big((t\Lambda)^*\big) = \mu(t^{-1}\Lambda^*) = t^{-1}\mu(\Lambda^*)$.

On va s'atteler à étudier la décroissance de $\rho(\Lambda + \mathbf{u})$ quand $\mathbf{u}$ s'éloigne de l'origine. Celle-ci est quantifiée par l'inégalité (1.1) avec $r = \sqrt{n/2}$, qui, conjointement avec (1.2), donne :

$$s^n\rho(\Lambda) \geq \rho_s(\Lambda + \mathbf{u}) \geq \rho_s\big((\Lambda + \mathbf{u})\backslash\mathcal{B}(\mathbf{0},\sqrt{n/2})\big) \geq e^{\frac{\pi}{2}\left(1-\frac{1}{s^2}\right)n}\rho\big((\Lambda + \mathbf{u})\backslash\mathcal{B}(\mathbf{0},\sqrt{n/2})\big).$$

On en tire en particulier :

$$\rho\big((\Lambda + \mathbf{u})\backslash\mathcal{B}(\mathbf{0},\sqrt{n/2})\big) \leq \left(se^{-\frac{\pi}{2}\left(1-\frac{1}{s^2}\right)}\right)^n \rho(\Lambda).$$

Notons α le minimum de $s \mapsto se^{-\frac{\pi}{2}\left(1-\frac{1}{s^2}\right)}$ sur $[1,\infty)$ - ce minimum est atteint pour $s \approx 1.77$ et est strictement inférieur à 0.61.

Dans ce cas, on a $\rho(\Lambda) - 1 = \rho(\Lambda\backslash\{\mathbf{0}\}) = \rho\big(\Lambda\backslash\mathcal{B}(\mathbf{0}, \sqrt{n/2})\big)$. Ainsi, on obtient $\rho(\Lambda) - 1 \leqslant \alpha^n\rho(\Lambda)$, qu'il est possible de réécrire $\rho(\Lambda) \leqslant \frac{1}{1-\alpha^n}$. On en déduit également que $\rho(\Lambda\backslash\{\mathbf{0}\}) \leqslant \frac{\alpha^n}{1-\alpha^n}$.

On remarque alors qu'en vertu de (1.3), $\mathbf{u} \mapsto \rho(\Lambda^* + \mathbf{u})$ est presque constante :

$$|\rho(\Lambda^* + \mathbf{u}) - \mathbf{covol}(\Lambda)| \leq \rho(\Lambda\backslash\{\mathbf{0}\})\mathbf{covol}(\Lambda) \leq \frac{\alpha^n}{1-\alpha^n}\mathbf{covol}(\Lambda).$$

En prenant $\mathbf{v} \in \mathbb{R}^n$ tel que $d(\mathbf{v}, \Lambda^*) > \sqrt{n/2}$, on a

$$\rho(\Lambda^* - \mathbf{v}) = \rho\big((\Lambda^* - \mathbf{v})\backslash\mathcal{B}(\mathbf{0}, \sqrt{n/2})\big) \leqslant \frac{\alpha^n}{1-\alpha^n}.$$

Par ailleurs, la "presque-constance" de $\rho(\Lambda^* + \mathbf{u})$ implique que :

$$\rho(\Lambda^* - \mathbf{v}) \geq \mathbf{covol}(\Lambda) - \frac{\alpha^n}{1-\alpha^n}\mathbf{covol}(\Lambda) = \frac{1-2\alpha^n}{1-\alpha^n}\mathbf{covol}(\Lambda) \geqslant \frac{1-2\alpha^n}{1-\alpha^n}2^{-\frac{n}{2}}.$$

Ces deux inégalités réunies impliquent :

$$\begin{aligned}\frac{\alpha^n}{1-\alpha^n} \geqslant \frac{1-2\alpha^n}{1-\alpha^n}2^{-\frac{n}{2}} &\implies (\sqrt{2}\alpha)^n \geq 1 - 2\alpha^n \\ &\implies (\sqrt{2}^n + 2)\alpha^n \geq 1.\end{aligned}$$

On vérifie, graphiquement ou par l'étude élémentaire de la fonction $t \mapsto \big(\sqrt{2}^t + 2\big)\alpha^t$, que cette inégalité est fausse dès lors que $n > 3$.

□

Cette inégalité est utile, par exemple, dans le cadre du problème GapSVP, un problème de décision associé à SVP.

Définition 1.28 (GapSVP)**.**
Entrée : un réseau Λ et un réel $d > 0$.
Sortie : « oui »dans le cas où $\lambda_1(\Lambda) \leqslant d$ et « non »dans le cas où $\lambda_1(\Lambda) > nd$.

Dans le cas où $\lambda_1(\Lambda) \leqslant d$, un algorithme répondant « oui »pourra le justifier en fournissant un vecteur de Λ de norme inférieure à d. Ce vecteur ne sera pas nécessairement rapide à trouver, mais il sera rapide de vérifier qu'il appartient effectivement à Λ et qu'il est effectivement de norme inférieure ou égale à d.

Au premier abord, il n'y a pas de possibilité de vérification aussi simple dans le cas $\lambda_1(\Lambda) > nd$. Cependant, l'inégalité de transférence nous apprend que dans ce cas, on a nécessairement $\lambda_n(\Lambda^*) < \frac{1}{d}$, et il devient possible de justifier une réponse « non »en fournissant n vecteurs linéairement indépendants de Λ^*, tous de norme inférieure à $\frac{1}{d}$ - et dans ce cas, il est possible de vérifier la réponse en temps polynomial.

2 L'échantillonage gaussien

Cette section sera dédiée à l'introduction et l'étude du problème DGS. Nous tenterons également d'y comprendre comment le problème SVP peut s'y réduire. Par ailleurs, nous ne considérerons à partir de maintenant que des réseaux de $\mathbb{R}^m$, et nous adoptons la convention "réseau de rang n" = "réseau de $\mathbb{R}^m$ de rang n".

2.1 Préliminaires probabilistes

Définition 2.1 (distance statistique)**.** *Soient $\mathbb{P}, \mathbb{Q}$ deux mesures de probabilité sur un espace $(\Omega, \mathcal{A})$. On définit la distance statistique entre $\mathbb{P}$ et $\mathbb{Q}$ par :*

$$\delta(\mathbb{P}, \mathbb{Q}) := \sup_{A \in \mathcal{A}} |\mathbb{P}(A) - \mathbb{Q}(A)|.$$

On vérifie aisément que c'est effectivement une distance. On a par ailleurs le résultat suivant :

Proposition 2.2. *Si Ω est un espace de probabilité dénombrable (muni de la tribu discrète), et en définissant $p(x), q(x) := \mathbb{P}\big(\{x\}\big), \mathbb{Q}\big(\{x\}\big)$:*

$$\delta(\mathbb{P}, \mathbb{Q}) := \frac{1}{2} \sum_{x \in \Omega} |p(x) - q(x)|.$$

Démonstration. Définissons $U := \{x \in \Omega | p(x) > q(x)\}$ et $V := U^c$. Il est assez facile de vérifier que $\mathbb{P}(U) - \mathbb{Q}(U) = \mathbb{Q}(V) - \mathbb{P}(V)$, et que $p \leqslant q$ sur V.

Remarquons alors que $\delta(\mathbb{P}, \mathbb{Q}) = \mathbb{P}(U) - \mathbb{Q}(U)$. En effet, si $A \subset X$, on a :

$$|\mathbb{P}(A) - \mathbb{Q}(A)| = |\mathbb{P}(A \cap U) - \mathbb{Q}(A \cap U) + \mathbb{P}(A \cap V) - \mathbb{Q}(A \cap V)|\,.$$

Mais $\mathbb{P}(A \cap U) - \mathbb{Q}(A \cap U) \geqslant 0$, $\mathbb{P}(A \cap V) - \mathbb{Q}(A \cap V) \leqslant 0$, ce qui implique :

$$|\mathbb{P}(A) - \mathbb{Q}(A)| \leq \max\big(\mathbb{P}(A \cap U) - \mathbb{Q}(A \cap U), \mathbb{Q}(A \cap V) - \mathbb{P}(A \cap V)\big).$$

On remarque finalement $\mathbb{P}(A \cap U) - \mathbb{Q}(A \cap U) \leqslant \mathbb{P}(U) - \mathbb{Q}(U)$ et $\mathbb{Q}(A \cap V) - \mathbb{P}(A \cap V) \leqslant \mathbb{Q}(V) - \mathbb{P}(V)$, et on en déduit que $\mathbb{P}(U) - \mathbb{Q}(U) = \mathbb{Q}(V) - \mathbb{P}(V) = \delta(\mathbb{P}, \mathbb{Q})$.

La conclusion s'obtient en remarquant :

$$\begin{aligned}
\frac{1}{2} \sum_{x \in \Omega} |p(x) - q(x)| &= \frac{1}{2} \left(\sum_{x \in U} p(x) - q(x) + \sum_{x \in V} q(x) - p(x) \right) \\
&= \frac{1}{2} \left(\mathbb{P}(U) - \mathbb{Q}(U) + \mathbb{Q}(V) - \mathbb{P}(V) \right) \\
&= \frac{1}{2} \cdot 2\delta(\mathbb{P}, \mathbb{Q}) = \delta(\mathbb{P}, \mathbb{Q}).
\end{aligned}$$

□

L'un des avantages principaux de la distance statistique est qu'elle se comporte "bien" par rapport aux différents algorithmes que l'on va utiliser.

2.2 Réduction de SVP à DGS

Définition 2.3 (Problème DGS)**.** *Soient Λ un réseau de rang n, $\varepsilon > 0$, $M \in \mathbb{N}$ et σ un paramètre positif dépendant de Λ. le problème ε-DGS_σ^M demande un algorithme prenant en entrée une base $\mathbf{B}$ d'un réseau de rang n, et donnant en sortie un M-uplet aléatoire $(\mathbf{X}_1, ..., \mathbf{X}_M)$ dont la distribution est $\varepsilon-$proche en distance statistique de M gaussiennes discrètes indépendantes de paramètre $s > \sigma(\Lambda)$ sur Λ.*

On veut ici que σ joue le rôle d'une sorte de "paramètre de lissage" au-dessus duquel on peut échantillonner en temps raisonnable. On omettra ϵ ou σ lorsqu'ils seront nuls.

Le but de cette section est de quantifier dans quelles conditions il est hautement probable d'obtenir un petit vecteur de Λ à partir de M échantillons indépendants de DGS, ce qui implique également un contrôle de l'échelle.

Lemme 2.4 (Paramètre optimal)**.** *Soit Λ un réseau de rang n, β la constante du lemme 1.23. Définissons*

$$\hat{s} := \sqrt{\frac{2\pi e}{\beta^2 n}}\lambda_1(\Lambda).$$

Alors si $||\mathbf{x}|| = \lambda_1(\Lambda)$, $\mu_{\hat{s}}^{\Lambda}(\mathbf{x}) \geqslant \left(e^{-\frac{\beta^2}{2e}}\right)^{n+o(n)} \approx 1.38^{-n-o(n)}$

Démonstration. Il s'agit d'une application de (1.4), puisque $e/\beta^2 > 1$. On a alors $\rho_{\hat{s}}(\Lambda) \leqslant \beta^{o(n)}$, donc :

$$\mu_{\hat{s}}^{\Lambda}(\mathbf{x}) = \frac{e^{-\frac{\beta^2 n}{2e}}}{\rho_{\hat{s}}(\Lambda)} \leqslant e^{-\frac{\beta^2 n}{2e}}\beta^{-o(n)}.$$

Comme β et $e^{\frac{\beta^2}{2e}}$ sont deux constantes strictement supérieures à 1, un $\beta^{o(n)}$ sera également un $\left(e^{\frac{\beta^2}{2e}}\right)^{o(n)}$. □

En appliquant l'inégalité (1.2), on se rend compte que pour $\theta > 0$ et $\hat{s} \leqslant s \leqslant e^{\theta}\hat{s}$, on a :

$$\mu_{s}^{\Lambda}(\mathbf{x}) \geqslant \left(e^{-\frac{\beta^2}{2e}+\theta}\right)^{n+o(n)}. \tag{2.1}$$

dès lors que $\mathbf{x}$ est de norme λ_1.

On peut alors se dire qu'il suffit de tirer au moins $2^{n/2}$ vecteurs selon $\mu_{\hat{s}}^{\Lambda}$, et qu'alors un petit vecteur se trouvera dans l'échantillon avec probabilité $1-\exp(-\Omega(n))$. Un problème subsiste : $\hat{s}$ dépend directement de $\lambda_1(\Lambda)$, que l'on ne connaît pas. Ce problème peut cependant être réglé en l'estimant avec LLL (et quelques autres détails techniques). On a alors effectivement la réduction attendue de SVP à DGS :

Théorème 2.5. *Il existe une réduction de* SVP *à* $\exp\big(-\Omega(n)\big)$-DGS$^{2^{n/2}}$ *L'algorithme tourne en temps* $\mathrm{poly}(n)\cdot 2^{n/2} + O(n)\cdot T$ *où T est le temps de l'algorithme résolvant* DGS*.*

On ne prouvera pas directement ce théorème. On trouve une preuve [ADRS15], et les idées principales de cette preuve sont présentes dans la preuve du bon fonctionnement de l'algorithme 3.12. Cependant, nous allons nous placer dans un cadre légèrement différent - en effet, une fois que nous aurons détaillé les algorithmes d'échantillonnage gaussien, il deviendra clair que la manoeuvre optimale consiste à se placer dans un sous-réseau $\Lambda' \subseteq \Lambda$ contenant tous les petits vecteurs et dont on connaît une petite base grâce à (2.13). On peut faire en sorte que la distance statistique entre μ_s^{Λ} et $\mu_s^{\Lambda'}$ soit suffisamment petite pour pouvoir remplir les conditions du théorème, mais nous préférons l'approche qui consiste à directement passer à SVP sur un sous-réseau puisque nous abandonnerons définitivement l'idée de produire un algorithme qui tire exactement selon μ_s^{Λ} d'ici la fin de la section suivante.

2.3 Résolution exacte de DGS

Puisqu'on peut, étant donné un algorithme d'échantillonnage gaussien, de résoudre SVP, on va expliciter ici un algorithme pour tirer selon une gaussienne discrète en dimension quelconque (2.6). Une fois cet algorithme présenté, notre occupation sera le contrôle de la complexité en temps : rappelons l'algorithme que nous présenterons se distingue par sa remarquable rapidité. L'algorithme de tirage DGS s'appuiera sur trois constats :

- Les réseaux de dimension 1 sont des objets extrêmement simples, sur lesquels il est très facile de faire un tirage gaussien (2.6).
- Etant donné un accès à un moyen d'échantillonnage sur un ensemble discret E selon une loi p, et une "loi cible" q, telles qu'il existe une constante $C > 0$ telle que $q \leq Cp$, il est possible de fournir un algorithme qui échantillonne selon q, avec un temps d'exécution dépendant de C et du temps de l'algorithme qui échantillonne selon p (2.8).
- Il est possible, pour peu que l'on travaille en paramètre assez grand (2.10) de tirer selon une loi "proche" (au sens de 2.8) de la gaussienne en tirant les coordonnées sur la base orhtogonalisée de Gram-Schmidt de celle de Λ, puis en inversant l'orthogonalisation (2.7).

Algorithme 2.6 (tirage DGS exact dimension 1)**.** *Soient, $s > 0$ et $0 \leq u < 1$ deux réels. Pour plus de lisibilité, on note $P_+ := \int_u^\infty \rho_s(t)\mathrm{d}t$, $P_- := \int_{-\infty}^{u-1} \rho_s(t)\mathrm{d}t$. Finalement, on pose $P := P_+ + P_- + \rho_s(u-1) + \rho_s(u)$. L'algorithme suivant permet de tirer selon $\mu_s^{\mathbb{Z}+u}$ en temps moyen inférieur à CT, où T est le temps nécessaire pour estimer $\rho_s(t)$ pour $t \in \mathbb{R}$ et à calculer les intégrales impliquées.*

Pour plus de clarté dans l'algorithme, on introduit une variable aléatoire Z à valeurs dans $\{-2, -1, 0, 1\}$, de probabilités respectives P_-/P, $\rho_s(u-1)/P$, $\rho_s(u)/P$ et P_+/P, ainsi que deux variables Y_+, Y_- suivant les restrictions des gaussiennes à $[u, \infty)$ et $(-\infty, u-1]$.

Entrée : Un réel $u \in [0, 1)$
Sortie : Une variable aléatoire distribuée selon $\mu_s^{\mathbb{Z}+u}$

Initialiser x à u et b à 0
Tant que $b = 0$:
  Faire un tirage de Z
  si $Z = -2$:
    $y :=$ tirage de Y_-
    $x \leftarrow$ le plus grand élément de $\mathbb{Z} + u$ qui soit inférieur ou égal à y
    Tirer b' selon une Bernoulli de paramètre $\rho_s(x)/\rho_s(y)$
    $b \leftarrow b'$
  sinon, si $Z = -1$:
    $x \leftarrow u - 1$
    $b \leftarrow 1$
  sinon, si $Z = 0$:
    $x \leftarrow u$
    $b \leftarrow 1$
  sinon :
    $y :=$ tirage de Y_+
    $x \leftarrow$ le plus petit élément de $\mathbb{Z} + u$ qui soit supérieur ou égal à y
    Tirer b' selon une Bernoulli de paramètre $\rho_s(x)/\rho_s(y)$
    $b \leftarrow b'$
Retourner x

Démonstration. On vérifie tout d'abord que $\rho_s(\mathbb{Z} + u) \geqslant P_+ + P_-$:

$$\begin{aligned}
P_+ + P_- &= \int_u^\infty \rho_s(t)\mathrm{d}t + \int_{-\infty}^{u-1} \rho_s(t)\mathrm{d}t \\
&= \sum_{k=0}^{\infty} \int_{k+u}^{k+1+u} \rho_s(t)\mathrm{d}t + \sum_{k=1}^{\infty} \int_{-k-1+u}^{-k+u} \rho_s(t)\mathrm{d}t \\
&\leqslant \sum_{k=0}^{\infty} \rho_s(k+u) + \sum_{k=1}^{\infty} \rho_s(-k+u) \\
&\leqslant \rho_s(\mathbb{Z}+u).
\end{aligned}$$

Alors, une itération de l'algorithme a probabilité $\rho_s(\mathbb{Z}+u)/P \geqslant 1/2$ de retourner quelque chose, ce qui signifie qu'il se termine presque sûrement, en moins de deux étapes en moyenne.

Il reste alors à vérifier que la distribution en sortie est bien celle voulue : appelons X la variable en sortie. Il suffit alors de regarder une seule itération de l'algorithme - en effet, soit X_i la variable aléatoire à valeurs dans $(\mathbb{Z}+u)\cup\{*\}$ donnée par le fait d'itérer l'algorithme une fois, et, dans le cas où $b'=0$, retourner simplement $*$. Alors, X est exactement la variable qui prend la valeur de X_i avec i le plus petit indice tel que $X_i \neq *$. Les $(X_i)_{i\geqslant 1}$ étant indépendants et identiquement distribués, on peut écrire :

$$\begin{aligned}
\mathbb{P}[X=x] &= \sum_{j=1}^{\infty} \mathbb{P}[X_j = x \cap X_1 = * \cap \ldots \cap X_{j-1} = *] \\
&= \sum_{j=1}^{\infty} \mathbb{P}[X_j = x]\mathbb{P}[X_1 = *]\ldots\mathbb{P}[X_{j-1} = *] \\
&= \sum_{j=1}^{\infty} \mathbb{P}[X_1 = x]\mathbb{P}[X_1 = *]^{j-1} \\
&= \frac{\mathbb{P}[X_1 = x]}{1-\mathbb{P}[X_1 = *]}.
\end{aligned}$$

On calcule alors $\mathbb{P}[X_1 = x]$ pour $x \in \mathbb{Z} + \mathbf{u}$.
Si $x = u$ ou $x = u-1$, il est clair que la probabilité de retourner x est $\rho_s(x)/P$.
Si $x = k+u$ avec $k \geqslant 1$, alors la probabilité globale que l'algorithme retourne x à la première itération est :

$$\begin{aligned}
\mathbb{P}[X=x] &= \mathbb{P}[Z=1]\mathbb{P}[X=x|Z=1] \\
&= \frac{P_+}{P} \cdot \frac{1}{P_+} \int_{k-1+u}^{k+u} \mathbb{P}[b'=1|Y_+ = y]\rho_s(y)\mathrm{d}y \\
&= \frac{1}{P} \int_{k-1+u}^{k+u} \frac{\rho_s(x)}{\rho_s(y)}\rho_s(y)\mathrm{d}y = \rho_s(x)/P
\end{aligned}$$

Un calcul similaire sur la partie $(-\infty, u-1]$ donne le même résultat : $\mathbb{P}[X=x]$ est proportionnel à $\mathbb{P}[X_1 = x]$, qui est proportionnel à $\rho_s(x)$. X est donc nécessairement distribué selon la gaussienne discrète. □

Remarque. *Dans ce cas, T est au plus polynomial en le* log *de la précision demandée - nous pouvons donc, en temps polynomial en n, obtenir une précision en $\left(1-e^{-\Omega(n)}\right)$, suffisante pour ne pas impacter le résultat du théorème précédent.*

Notons également que cet algorithme permet de tirer selon la gaussienne sur $\alpha\mathbb{Z}+u$ pour $\alpha>0$ en absorbant la constante α dans le paramètre s.

L'algorithme suivant explique comment passer de la dimension 1 à la dimension n - on va essentiellement tirer dans les coordonnées orthogonales puis repasser à Λ en inversant le procédé de Gram-Schmidt

Algorithme 2.7 (Tirage presque DGS en dimension quelconque)**.** *Soit Λ un réseau de dimension n, de base $\mathbf{B}=(\mathbf{b}_1,...,\mathbf{b}_n)$. On note $\tilde{\mathbf{B}}=(\tilde{\mathbf{b}}_1,...,\tilde{\mathbf{b}}_n)$ son orthogonalisée de Gram-Schmidt, $||\tilde{\mathbf{B}}||:=\max_i||\tilde{\mathbf{b}}_i||$, et finalement $\overline{\mathbf{B}}$ son orthonormalisée, qui sont calculables en temps polynomial en n. On présente un algorithme qui prend en entrée un réseau Λ, donné par sa base $\mathbf{B}$, ainsi qu'un paramètre $s>0$ quelconque, et donne en sortie un vecteur aléatoire de Λ dont la distribution ν_s^Λ est proche de μ_s^Λ dans le sens où il existe une constante dépendant de s et de la base $\mathbf{B}$ telle que $\mu_s^\Lambda(\mathbf{x})\leqslant C_{\mathbf{B},s}\nu_s^\Lambda(\mathbf{x})$ pour tout $\mathbf{x}\in\Lambda$. Cet algorithme tourne en temps polynomial en n.*

Entrée : La base $\mathbf{B}$ d'un réseau Λ de $\mathbb{R}^n$.
Sortie : Un vecteur distribué selon une distribution proche de μ_s^Λ.

Calculer $\tilde{\mathbf{B}}$ l'orthogonalisée de Gram-Schmidt de $\mathbf{B}$
On initialise $\mathbf{u}_n$ à $\mathbf{0}$.
Pour $j=n,...,1$:
  Tirer x_j selon $\mu_s^{||\tilde{\mathbf{b}}_j||\mathbb{Z}+\langle\mathbf{u}_j,\overline{\mathbf{b}}_j\rangle}$
  Poser $\mathbf{u}_{j-1}:=\mathbf{u}_j+\big(x_j-\langle\mathbf{u}_j,\overline{\mathbf{b}}_j\rangle\big)\mathbf{b}_j/||\tilde{\mathbf{b}}_j||-x_j\overline{\mathbf{b}}_j$
Retourner $\mathbf{x}:=\sum x_j\overline{\mathbf{b}}_j$

Démonstration. Posons $t_j:=\langle\mathbf{u}_j,\overline{\mathbf{b}}_j\rangle$.

On montre tout d'abord que les $\mathbf{x}$ que l'on peut obtenir par cet algorithme sont exactement les vecteurs de Λ. Pour ce faire, on prouve que les vecteurs dont les coordonnées dans la base $\overline{\mathbf{B}}$ sont dans $||\tilde{\mathbf{b}}_j||\mathbb{Z}+t_j$ sont exactement ceux dont les coordonnées dans $\mathbf{B}$ sont entières. On se propose de détailler le calcul dans le cas $n=2$, les cas des dimensions supérieures étant analogues mais bien plus lourds. Soit donc $\mathbf{x}=k_1\mathbf{b}_1+k_2\mathbf{b}_2$ à coordonnées entières dans $\mathbf{B}$, et (x_1,x_2) ses coordonnées dans $\overline{\mathbf{B}}$. Alors, en rappelant que $\mathbf{b}_1=\tilde{\mathbf{b}}_1=||\tilde{\mathbf{b}}_1||\overline{\mathbf{b}}_1$:

$$\begin{aligned}\frac{x_1}{||\tilde{\mathbf{b}}_1||}\tilde{\mathbf{b}}_1+\frac{x_2}{||\tilde{\mathbf{b}}_2||}\tilde{\mathbf{b}}_2=k_1\mathbf{b}_1+k_2\mathbf{b}_2 &\iff \frac{x_1}{||\tilde{\mathbf{b}}_1||}\mathbf{b}_1+\frac{x_2}{||\tilde{\mathbf{b}}_2||}\left(\mathbf{b}_2-\frac{\langle\mathbf{b}_1,\mathbf{b}_2\rangle}{||\mathbf{b}_1||^2}\mathbf{b}_1\right)=k_1\mathbf{b}_1+k_2\mathbf{b}_2\\ &\iff\begin{cases}x_2=||\tilde{\mathbf{b}}_2||k_2\\ x_1=||\tilde{\mathbf{b}}_1||k_1+||\tilde{\mathbf{b}}_1||\frac{x_2}{||\tilde{\mathbf{b}}_2||}\langle\overline{\mathbf{b_1}},\mathbf{b}_2\rangle\overline{\mathbf{b}}_1=||\tilde{\mathbf{b}}_1||k_1+t_1\end{cases}\end{aligned}$$

On peut alors expliciter ν_s^Λ : si $\mathbf{x}\in\Lambda$, on note $\mathbf{x}=\sum_{j=1}^n x_j\overline{\mathbf{b}}_j$. Alors, en notant $\mathbf{X}$ la variable en sortie de l'algorithme et X_j la variable tirée selon $\mu_s^{||\tilde{\mathbf{b}}_j||\mathbb{Z}+t_j}$, la probabilité d'obtenir $\mathbf{x}$ est :

$$\begin{aligned}\mathbb{P}[\mathbf{X}=\mathbf{x}]&=\mathbb{P}\left[\sum_{j=1}^n X_j\overline{\mathbf{b}}_j=\sum_{j=1}^n x_j\overline{\mathbf{b}}_j\right]\\ &=\mathbb{P}\left[\bigcap_{1\leqslant j\leqslant n}X_j=x_j\right]\\ &=\prod_{j=1}^n\mathbb{P}\left[X_j=x_j\,\middle|\bigcap_{j<i\leqslant n}X_i=x_i\right]\end{aligned}$$

Or, par définition des variables X_j, $\mathbb{P}\left[X_j = x_j \middle| \bigcap_{j<i\leqslant n} X_i = x_i\right]$ est exactement $\rho_s(x_j)/\rho_s(\tilde{\mathbf{b}}_j||\mathbb{Z}+t_j)$. On trouve donc finalement :

$$\begin{aligned}\mathbb{P}[\mathbf{X}=\mathbf{x}] &= \prod_{j=1}^{n} \frac{\rho_s(x_j)}{\rho_s(||\tilde{\mathbf{b}}_j||\mathbb{Z}+t_j)} \\ &= \frac{\rho_s(\mathbf{x})}{\prod_{j=1}^{n}\rho_s(||\tilde{\mathbf{b}}_j||\mathbb{Z}+t_j)}\end{aligned}$$

Reste à déterminer $C_{\mathbf{B},s}$. On rappelle à cet effet que $\rho_s(\alpha\mathbb{Z}+t) \leqslant \rho_s(\alpha\mathbb{Z})$ pour tous $\alpha > 0$, $t \in \mathbb{R}$, et on a alors :

$$\begin{aligned}\mu_s^{\Lambda}(\mathbf{x}) &= \frac{\rho_s(\mathbf{x})}{\rho_s(\Lambda)} \\ &= \frac{\rho_s(\mathbf{x})}{\prod_{j=1}^{n}\rho_s(||\tilde{\mathbf{b}}_j||\mathbb{Z}+t_j)} \cdot \frac{\prod_{j=1}^{n}\rho_s(||\tilde{\mathbf{b}}_j||\mathbb{Z}+t_j)}{\rho_s(\Lambda)} \\ &\leqslant \nu_s^{\Lambda}(\mathbf{x}) \cdot \frac{\prod_{j=1}^{n}\rho_s(||\tilde{\mathbf{b}}_j||\mathbb{Z})}{\rho_s(\Lambda)}\end{aligned}$$

On pose donc $C_{\mathbf{B},s} = \frac{\prod_{j=1}^{n}\rho_s(||\tilde{\mathbf{b}}_j||\mathbb{Z})}{\rho_s(\Lambda)}$ et on a le résultat annoncé. □

Algorithme 2.8. *Soit Ω un ensemble dénombrable, p, q deux lois sur Ω. On suppose qu'il existe une constante $C > 0$ telle que $q(x) \leq Cp(x)$ pour tout $x \in \Omega$ (en particulier, si $p(x) = 0$, alors $q(x) = 0$). Supposons que l'on dispose d'un algorithme permettant de tirer selon p en temps moyen T : alors, l'algorithme suivant permet de tirer selon q en temps moyen CT.*

Entrée : Une distribution cible q, un algorithme qui tire selon une distribution source p
Sortie : Un élément de Ω distribué selon q

Initialiser b à 0, y à un élément quelconque de Ω.
Tant que $b = 0$
  Tirer x selon la distribution p.
  $y \leftarrow x$
  Tirer b' selon une Bernoulli de paramètre $\frac{q(y)}{Cp(y)}$
  $b \leftarrow b'$
Retourner y

Démonstration. On vérifie que la distribution en sortie est bien q : on introduit à cet effet la variable τ de temps d'arrêt de l'algorithme (en nombre d'itérations). En notant X la variable en sortie et $(X_n)_n$ les variables correspondant aux tirages selon p, on a alors :

$$\begin{aligned}
\mathbb{P}[X=x] &= \sum_{n\geqslant 1}\mathbb{P}[X=x\cap\tau=n]\\
&= \sum_{n\geqslant 1}\mathbb{P}[X_n=x\cap\tau=n]\\
&= \sum_{n\geqslant 1}\mathbb{P}[\tau=n|X_n=x]\,p(x)\\
&= \sum_{n\geqslant 1}\left(\sum_{x_1,\dots,x_{n-1}\in\Omega}\mathbb{P}[\tau=n|X_n=x,X_j=x_j]\prod_{j=1}^{n-1}p(x_j)\right)p(x) \qquad \text{(indépendance des } X_i)\\
&= \sum_{n\geqslant 1}\left(\sum_{x_1,\dots,x_{n-1}\in\Omega}\frac{q(x)}{Cp(x)}\prod_{j=1}^{n-1}\left(1-\frac{q(x_j)}{Cp(x_j)}\right)\prod_{j=1}^{n-1}p(x_j)\right)p(x)\\
&= \sum_{n\geqslant 1}C^{-1}q(x)\sum_{x_1,\dots,x_{n-1}\in\Omega}\prod_{j=1}^{n-1}\big(p(x_j)-C^{-1}q(x_j)\big)\\
&= \sum_{n\geqslant 1}C^{-1}q(x)\prod_{j=1}^{n-1}\left(\sum_{x_1,\dots,x_{n-1}\in\Omega}p(x_j)-C^{-1}q(x_j)\right)\\
&= \sum_{n\geqslant 1}C^{-1}q(x)\big(1-C^{-1}\big)^{n-1}=q(x)
\end{aligned}$$

Des calculs très similaires permettent de constater que la variable τ est géométrique de paramètre C^{-1} :

$$\begin{aligned}
\mathbb{P}[\tau=n] &= \sum_{x_1,\dots,x_n\in\Omega}\mathbb{P}[\tau=n|X_j=x_j]\prod p(x_j)\\
&= \sum_{x_1,\dots,x_n\in\Omega}C^{-1}q(x_n)\prod_{j=1}^{n-1}\big(p(x_j)-C^{-1}q(x_j)\big)\\
&= C^{-1}(1-C^{-1})^{n-1}
\end{aligned}$$

Ainsi, $\mathbb{E}[\tau]=C$, ce qui prouve le temps de l'algorithme. □

On ne peut en l'état pas utiliser l'algorithme 2.8 avec q = la gaussienne discrète et $p=\nu_s^\Lambda$ la distribution en sortie de l'algorithme 2.7, puisque la constante $C_{\mathbf{B},s}$ dépend a priori de s et de $\mathbf{B}$ (et dans une moindre mesure de Λ) d'une manière qu'il peut être difficile de quantifier. On va donc introduire le smoothing parameter de Λ, et montrer quelques propositions permettant d'obtenir des majorations sur $C_{n,s}$.

Définition 2.9 (Paramètre de lissage / smoothing parameter)**.** *Pour un réseau Λ, on définit le paramètre de lissage $\eta_\varepsilon(\Lambda)$ comme le plus petit paramètre $s>0$ vérifiant :*

$$\rho_{\frac{1}{s}}(\Lambda^*)\leqslant 1+\varepsilon$$

Une faible masse de Λ^* signifie que le réseau est "peu dense" dans l'espace, et Λ serait donc à l'inverse assez dense. Cette vision est justifiée par la proposition suivante :

Proposition 2.10. *Si $s \geqslant \eta_\varepsilon(\Lambda)$, alors pour tout $\mathbf{u} \in \mathbb{R}^n$:*

$$\frac{\rho_s(\Lambda + \mathbf{u})}{\rho_s(\Lambda)} \geqslant \frac{1-\varepsilon}{1+\varepsilon}$$

Démonstration. On utilise la formule sommatoire de Poisson :

$$\begin{aligned}
\frac{\rho_s(\Lambda + \mathbf{u})}{\rho_s(\Lambda)} &= \frac{\mathbf{covol}(\Lambda^*) s^n \sum_{\mathbf{y}\in\Lambda^*} \rho_{\frac{1}{s}}(\mathbf{y}) e^{2i\pi \mathbf{y}\cdot\mathbf{u}}}{\mathbf{covol}(\Lambda^*) s^n \rho_{\frac{1}{s}}(\Lambda^*)} \\
&= \frac{1 + \sum_{\mathbf{y}\in\Lambda^*\setminus\{\mathbf{0}\}} \rho_{\frac{1}{s}}(\mathbf{y}) e^{2i\pi \mathbf{y}\cdot\mathbf{u}}}{\rho_{\frac{1}{s}}(\Lambda^*)} \\
&\geqslant \frac{1 - \left|\sum_{\mathbf{y}\in\Lambda^*\setminus\{\mathbf{0}\}} \rho_{\frac{1}{s}}(\mathbf{y}) e^{2i\pi \mathbf{y}\cdot\mathbf{u}}\right|}{1+\varepsilon}
\end{aligned}$$

Finalement, on remarque que $\left|\sum_{\mathbf{y}\in\Lambda^*\setminus\{\mathbf{0}\}} \rho_{\frac{1}{s}}(\mathbf{y}) e^{2i\pi \mathbf{y}\cdot\mathbf{u}}\right| \leqslant \rho_{\frac{1}{s}}(\Lambda^*) - 1 \leqslant \varepsilon$ pour terminer la preuve. □

Ce résultat peut s'interpréter dans le sens suivant : pour que la masse soit à peu près constante par décalage de n'importe quel $\mathbf{u} \in \mathbb{R}^n$, il faut que les points du réseau décalé soient très proches des points du réseau initial, et ce pour tout $\mathbf{u}$ - le réseau doit donc être très "dense".

Proposition 2.11. *Soit $\varepsilon > 0$, $\alpha > 0$. Alors :*

$$\eta_\varepsilon(\alpha\mathbb{Z}) \leqslant \alpha \sqrt{\frac{\log\left(2\left(1+\frac{1}{\varepsilon}\right)\right)}{\pi}}$$

Démonstration. Puisque $\rho_s(\Lambda)$ est croissante en s, il suffit de montrer que pour $s = \alpha\sqrt{\frac{\log\left(2+\frac{2}{\varepsilon}\right)}{\pi}}$, on a $\rho_{\frac{1}{s}}(\alpha^{-1}\mathbb{Z}) \leqslant 1+\varepsilon$.

$$\begin{aligned}
\rho_{\frac{1}{s}}(\alpha^{-1}\mathbb{Z}) &= \sum_{k\in\mathbb{Z}} e^{-\pi\alpha^{-2}s^2k^2} \\
&= 1 + 2\sum_{k\geq 1} e^{-\log\left(2+\frac{2}{\varepsilon}\right)k^2} \\
&\leqslant 1 + 2\sum_{k\geq 1} \left(2+\frac{2}{\varepsilon}\right)^{-k} \\
&\leqslant 1 + 2\frac{\frac{\varepsilon}{2+2\varepsilon}}{1-\frac{\varepsilon}{2+2\varepsilon}} \leqslant 1 + \frac{\varepsilon}{1+\varepsilon/2} \leqslant 1+\varepsilon
\end{aligned}$$

□

Cette proposition traduit en réalité un fait général sur les réseaux de dimension quelconque, pour peu que l'on remplace 2 par $2n$ dans le log. Pour autant, la preuve en dimension supérieure n'est pas aussi simple et demande entre autres de considérer les normes ℓ^p et ℓ^∞ sur Λ.

Corollaire 2.12 (Tirage DGS à grand paramètre)**.** *Soit Λ un réseau de rang n. Il existe un algorithme probabiliste permettant, étant donné une base $\mathbf{B}$ de Λ, de tirer en temps moyen polynomial en n selon μ_s^Λ si $s \geqslant ||\tilde{\mathbf{B}}||\sqrt{\log(2n+4)/\pi}$.*

Démonstration. On utilise l'algorithme 2.7 - la distribution en sortie est ν_s^Λ, qui vérifie $\mu_s^\Lambda \leqslant C_{\mathbf{B},s}\nu_s^\Lambda$.

On va alors montrer que pour le choix de s proposé, $C_{\mathbf{B},s}$ est inférieur à e^2.

On montre tout d'abord que $\rho_s(\Lambda) \geqslant \inf_{\mathbf{u}=u_1,\dots,u_n\in\mathbb{R}^n} \prod_{j=1}^n \rho_s(||\tilde{\mathbf{b}}_j||\mathbb{Z}+u_j)$. En effet, la loi de probabilité ν_s^Λ en sortie de l'algorithme 2.7 vérifie :

$$\begin{aligned}
1 &= \sum_{\mathbf{x}\in\Lambda} \nu_s^\Lambda(\mathbf{x}) \\
&= \sum_{\mathbf{x}\in\Lambda} \frac{\rho_s(\mathbf{x})}{\prod_{j=1}^n \rho_s\big(||\tilde{\mathbf{b}}_j||\mathbb{Z}+t_j(\mathbf{x})\big)} \\
&\geqslant \sum_{\mathbf{x}\in\Lambda} \frac{\rho_s(\mathbf{x})}{\inf_{\mathbf{u}} \prod_{j=1}^n \rho_s(||\tilde{\mathbf{b}}_j||\mathbb{Z}+u_j)} \\
&\geqslant \frac{\rho_s(\Lambda)}{\inf_{\mathbf{u}} \prod_{j=1}^n \rho_s(||\tilde{\mathbf{b}}_j||\mathbb{Z}+u_j)}
\end{aligned}$$

Notons qu'on peut ramener l'infiniumum sur $\mathbb{R}^n$ à un infinimum sur un pavé fondamental $\mathcal{P}_\mathbf{B}$, qui est précompact : il existe donc un $\mathbf{v}\in\mathcal{P}_\mathbf{B}$ qui l'atteint. On peut alors majorer $C_{\mathbf{B},s}$:

$$\begin{aligned}
C_{\mathbf{B},s} &= \frac{\prod_{j=1}^n \rho_s(||\tilde{\mathbf{b}}_j||\mathbb{Z})}{\rho_s(\Lambda)} \\
&\leqslant \prod_{j=1}^n \frac{\rho_s(||\tilde{\mathbf{b}}_j||\mathbb{Z})}{\rho_s(||\tilde{\mathbf{b}}_j||\mathbb{Z}+v_j)} \ .
\end{aligned}$$

Comme $s \geqslant \eta_{\frac{1}{n+1}}(||\tilde{\mathbf{b}}_j||\mathbb{Z})$ pour $j=1,\dots,n$ par (2.11), on applique (2.10) et on obtient :

$$\begin{aligned}
\frac{\rho_s(||\tilde{\mathbf{b}}_j||\mathbb{Z}+v_j)}{\rho_s(||\tilde{\mathbf{b}}_j||\mathbb{Z})} &\geqslant \frac{1-\frac{1}{n+1}}{1+\frac{1}{n+1}} \\
&\geqslant \frac{n}{n+2} = 1-\frac{2}{n+2} \ .
\end{aligned}$$

D'où finalement :

$$C_{\mathbf{B},s} \leqslant \left(1-\frac{2}{n+2}\right)^{-n} \leqslant e^2 \ .$$

On peut donc appliquer 2.8, avec T polynomial en n. □

Remarque. *Cette procédure permet en fait de tirer selon la gaussienne μ_s^Λ pour tous s, mais dès lors que le contrôle sur $C_{\mathbf{B},s}$ est perdu, on perd également le contrôle sur le temps de l'algorithme. Rappelons en effet que l'intérêt principal de l'algorithme que nous aimerions présenter est sa rapidité : il est impératif de garder à l'esprit le contrôle du temps.*

Remarquons cependant que cet algorithme nous demande de contrôler la taille de la base d'un réseau : rappelons en effet qu'un réseau avec un très faible λ_1, ou même un très faible covolume, peut avoir des vecteurs de base très longs. On aura donc finalement besoin du lemme suivant pour trouver des bases raisonnablement petites :

Algorithme 2.13. *Soit Λ un réseau de rang n et de base $\mathbf{B}$. On présente un algorithme qui, étant donné $r>0$, et un paramètre $a\geq 2$, renvoie une base $\mathbf{B}'$ d'un sous-réseau $\Lambda'\subseteq\Lambda$, contenant tous les vecteurs de Λ de longueur au plus $ra^{-n/a}$. La base $\mathbf{B}'$ est alors petite au sens où $||\tilde{\mathbf{B}}'||\leq r$.*

Cet algorithme tourne en temps $\mathrm{poly}(n)\cdot 2^{O(a)}$.

Entrée : La base $\mathbf{B}$ d'un réseau Λ, une longueur r et un paramètre $a \geqslant 2$.
Sortie : Une petite base $\mathbf{B}'$ d'un sous-réseau de Λ.

Initialiser $\mathbf{b}$ à $\mathbf{0}$, i à 1 et $\mathbf{B}'$ à la liste vide.
Utiliser le théorème 1.15 pour obtenir une base $(\mathbf{x}_i)_{1\leqslant i\leqslant n}$ $a^{n/a}-$réduite.
Tant que $||\mathbf{b}|| \leqslant r$:
  $\mathbf{b} \leftarrow \mathbf{x}_i$
  Ajouter $\mathbf{b}$ à la base $\mathbf{B}'$
  $i \leftarrow i+1$
Retourner $\mathbf{B}'$

Démonstration. La complexité de l'algorithme est la même que celle de (1.15), modulo les vérifications de taille sur les $\mathbf{x}_i$ qui sont faites en temps polynomial.

Reste alors à prouver que tous les petits vecteurs sont bien dans Λ'. Notons $\Lambda'' := \pi_{\Lambda'^{\perp}}(\Lambda) = \pi_{(\mathbf{x}_1,\dots,\mathbf{x}_k)^{\perp}}(\Lambda) = \pi_{(\tilde{\mathbf{x}}_1,\dots,\tilde{\mathbf{x}}_k)^{\perp}}(\Lambda)$. Le premier vecteur dans la base de Λ'' est $\pi_{(\tilde{\mathbf{x}}_1,\dots,\tilde{\mathbf{x}}_k)^{\perp}}(\mathbf{x}_{k+1}) = \tilde{\mathbf{x}}_{k+1}$, ce qui permet d'affirmer que $\lambda_1(\Lambda'') \geqslant a^{-n/a}||\tilde{\mathbf{x}}_{k+1}|| > a^{-n/a}r$ (car la base $\mathbf{B}$ est $a^{n/a}$-réduite).

Soit $\mathbf{x} \in \Lambda\backslash\Lambda'$: alors $\mathbf{x} = \pi_{\Lambda'}(\mathbf{x}) + \pi_{\Lambda'^{\perp}}(\mathbf{x})$. Le deuxième terme est non-nul puisque $\mathbf{x} \notin \Lambda'$. Une application du théorème de Pythagore suffit à conclure que :

$$||\mathbf{x}|| \geqslant ||\pi_{\Lambda'^{\perp}}(\mathbf{x})|| \geqslant \lambda_1(\Lambda'') \geqslant a^{-n/a}r \ .$$

Tout petit vecteur est donc effectivement dans Λ'. □

2.4 Réduction du paramètre

Grâce à nos efforts jusqu'à ce point, nous devrions être en mesure de résoudre DGS en temps raisonnable pour des paramètres assez grands. Il est maintenant question d'être capable de réduire le paramètre s - c'est là qu'intervient un lemme crucial :

Lemme 2.14. *Soit Λ un réseau de rang n. On considère le réseau de rang $2n$ suivant $\overline{\Lambda} := \{(\mathbf{x},\mathbf{y}) \in \Lambda\oplus\Lambda : \mathbf{x} \equiv \mathbf{y} \pmod{2\Lambda}\}$. C'est un sous-réseau de $\Lambda\oplus\Lambda$, où la norme est définie orthogonalement : en notant $(\mathbf{x},\mathbf{y})$ un élément de $\Lambda_{\mathbb{R}}\oplus\Lambda_{\mathbb{R}}$, on a $||(\mathbf{x},\mathbf{y})||^2 = ||\mathbf{x}||^2 + ||\mathbf{y}||^2$. On définit la rotation orthogonale*

$$\boldsymbol{\varrho} = \frac{1}{\sqrt{2}}\begin{bmatrix}\mathbf{I}_n & -\mathbf{I}_n\\ \mathbf{I}_n & \mathbf{I}_n\end{bmatrix} \ .$$

Cette rotation envoie (isométriquement) $\overline{\Lambda}$ sur $\sqrt{2}\big(\Lambda\oplus\Lambda\big)$

Démonstration. Le fait que $\boldsymbol{\varrho}\overline{\Lambda} \subseteq \sqrt{2}\big(\Lambda\oplus\Lambda\big)$ découle de la définition de $\overline{\Lambda}$. On a par ailleurs :

$$\boldsymbol{\varrho}^{-1} = \frac{1}{\sqrt{2}}\begin{bmatrix}\mathbf{I}_n & \mathbf{I}_n\\ -\mathbf{I}_n & \mathbf{I}_n\end{bmatrix}$$

Soit $\sqrt{2}(\mathbf{x},\mathbf{y}) \in \sqrt{2}\big(\Lambda\oplus\Lambda\big)$. Alors, $\boldsymbol{\varrho}^{-1}\big(\sqrt{2}\mathbf{x},\sqrt{2}\mathbf{y}\big) = (\mathbf{x}+\mathbf{y},\mathbf{x}-\mathbf{y}) \in \overline{\Lambda}$.
Ce fait achève de prouver que $\boldsymbol{\varrho}$ induit effectivement une isométrie entre $\overline{\Lambda}$ et $\sqrt{2}\big(\Lambda\oplus\Lambda\big)$ □

Ce fait géométrique a des conséquences sur nos tirages aléatoires − en particulier, il permet de déduire comment et dans quelles conditions il est possible de diminuer le paramètre d' une gaussienne discrète.

Corollaire 2.15. *Soit Λ un réseau de rang n, et $\mathbf{X}, \mathbf{Y}$ deux vecteurs indépendants dont la distribution est exactement la gaussienne sur Λ de paramètre s. Alors la distribution de $\frac{\mathbf{X}+\mathbf{Y}}{2}$ conditionnée à $\mathbf{X} \equiv \mathbf{Y} \pmod{2\Lambda}$ est une gaussienne sur Λ de paramètre $\frac{s}{\sqrt{2}}$:*

$$\text{Pour } \mathbf{y} \in \Lambda, \mathbb{P}\left[\frac{\mathbf{X}+\mathbf{Y}}{2} = \mathbf{y} \middle| \mathbf{X}, \mathbf{Y} \in \overline{\Lambda}\right] = \mu^{\Lambda}_{s/\sqrt{2}}(\mathbf{y})$$

Démonstration. Comme le couple $(\mathbf{X}, \mathbf{Y})$ est indépendant, on peut le considérer comme un vecteur aléatoire distribué selon la gaussienne de paramètre s sur $\Lambda \oplus \Lambda$.

Montrons tout d'abord que la gaussienne de paramètre s sur $\Lambda \oplus \Lambda$ conditionnée à $(\mathbf{X}, \mathbf{Y}) \in \overline{\Lambda}$ est une gaussienne de paramètre s sur $\overline{\Lambda}$. Soient $(\mathbf{x}, \mathbf{y}) \in \overline{\Lambda}$:

$$\begin{aligned}
\mathbb{P}\left[(\mathbf{X}, \mathbf{Y}) = (\mathbf{x}, \mathbf{y}) \middle| (\mathbf{X}, \mathbf{Y}) \in \overline{\Lambda}\right] &= \frac{\mathbb{P}\left[(\mathbf{X}, \mathbf{Y}) = (\mathbf{x}, \mathbf{y}) \cap (\mathbf{X}, \mathbf{Y}) \in \overline{\Lambda}\right]}{\mathbb{P}[(\mathbf{X}, \mathbf{Y}) \in \overline{\Lambda}]} \\
&= \frac{\mathbb{P}\left[(\mathbf{X}, \mathbf{Y}) = (\mathbf{x}, \mathbf{y})\right]}{\mathbb{P}[\mathbf{X}, \mathbf{Y} \in \overline{\Lambda}]} \\
&= \frac{\mu^{\Lambda}_s(\mathbf{x}, \mathbf{y})}{\mu^{\Lambda}_s(\overline{\Lambda})} = \mu^{\overline{\Lambda}}_s(\mathbf{x}, \mathbf{y}) \ .
\end{aligned}$$

Alors $\frac{1}{\sqrt{2}}(\mathbf{X}, \mathbf{Y})$ conditionné suit une loi gaussienne de paramètre $s/\sqrt{2}$ sur $\frac{1}{\sqrt{2}}\overline{\Lambda}$. En appliquant $\boldsymbol{\varrho}$, on trouve que $\left(\frac{\mathbf{X}-\mathbf{Y}}{2}, \frac{\mathbf{X}+\mathbf{Y}}{2}\right)$ suit une gaussienne de même paramètre $s/\sqrt{2}$ sur $\frac{1}{\sqrt{2}}\boldsymbol{\varrho}\overline{\Lambda} = \Lambda \oplus \Lambda$. En prenant seulement la deuxième coordonnée, on a le résultat voulu.

□

Il s'agit maintenant de trouver une manière d'utiliser ce résultat pour diminuer le paramètre. En effet, la loi du couple $(\mathbf{X}, \mathbf{Y})$ conditionné à $(\mathbf{X}, \mathbf{Y}) \in \overline{\Lambda}$ est exactement $\mu^{\overline{\Lambda}}_s$, et on ne sait pas a priori tirer selon cette loi.

3 Implémentation de l'algorithme et résolution de SVP

Nous allons à présent étudier les obstacles à la réduction de paramètre, ainsi que les différentes manières de les contourner.

Certains résultats dans cette partie n'ont pas pu être vérifiés par M. F. Charles par manque de temps. Ils seront marqués par une astérisque.

3.1 Algorithmes de moyennage avec ou sans rejet

On présente dans ce paragraphe les différentes procédures utilisées dans l'algorithme final et dans sa preuve. On a vu que le fait de moyenner des vecteurs dans la même classe d'équivalence permet de diminuer le paramètre, ce qui nous pousse à introduire la procédure 1 :

Procédure 1 : Appairage et moyenne
Entrée : Un réseau Λ de dimension n représenté par une base et des vecteurs $(\mathbf{x}_1, \ldots, \mathbf{x}_M)$ de Λ
Sortie : Des vecteurs $\mathbf{x}'_1, \ldots, \mathbf{x}'_N$ de Λ

Créer $(\mathbf{y}_c)_{c \in \Lambda/2\Lambda}$ des variables, initialement vides / sans valeur, et L une liste vide
Pour $1 \leqslant i \leqslant M$:
  $c \leftarrow$ la classe de $\mathbf{x}_i$ mod 2Λ

si $\mathbf{y}_c$ est vide :
$\mathbf{y}_c \leftarrow \mathbf{x}_i$
sinon :
ajouter $\frac{\mathbf{x}_i+\mathbf{y}_c}{2}$ à L puis vider $\mathbf{y}_j$
renvoyer le contenu de L (sous la forme d'un M'-uplet par exemple)

Ici, on ne fait la moyenne que de vecteurs dans la même classe d'équivalence mod 2Λ pour obtenir des vecteurs qui sont toujours dans le réseau.
Cette procédure d'appairage et de moyenne naïve ne donne cependant pas, sur une entrée gaussienne de paramètre s, une sortie gaussienne de paramètre $s/\sqrt{2}$, puisque le nombre de vecteurs dans une classe c mod 2Λ sera approximativement $M\rho_s(c)/\rho_s(\Lambda)$, là où la probabilité de se trouver dans la classe c pour un couple dans $\overline{\Lambda}$ est $\rho_s(c)^2/\rho_{s/\sqrt{2}}(\Lambda)^2$. On voudrait corriger cette erreur en rejetant certains vecteurs.

La deuxième procédure va uniquement servir à la preuve du fonctionnement de l'algorithme final, mais n'a pas vocation à être implémentée.

On fera appel à une fonction (possiblement aléatoire) de rejet qui, aux classes de nos vecteurs nous donne un sous ensemble d'indices des vecteurs à conserver. Typiquement, dans la partie suivante, la fonction de rejet est donnée par (3.1)

Dans le cas ou la fonction de rejet est aléatoire, elle ne doit utiliser en plus de son entrée, que des sources d'aléa indépendantes de tout autre tirage (comme par exemples les échantillons gaussiens)

Procédure 2 : moyennage avec rejet
Entrée : Un réseau Λ de dimension n représenté par une base, des vecteurs $(\mathbf{x}_1, \dots, \mathbf{x}_M)$ de Λ et une fonction f de rejet.
Sortie : Des vecteurs $(\mathbf{x}'_1, \dots, \mathbf{x}'_N)$ de Λ.

$(i_1, \dots, i_N) \leftarrow f([\mathbf{x}_1], \dots, [\mathbf{x}_M])$ (ou $[\mathbf{x}]$ est la classe de $\mathbf{x}$ mod 2Λ)
$(\mathbf{y}_1, \dots, \mathbf{y}_{M'}) \leftarrow$ *appairage et moyenne*$(\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_N})$
renvoyer $(\mathbf{y}_1, \dots, \mathbf{y}_{M'})$

On aura aussi besoin d'une version modifiée de la procédure 1, qui servira dans la preuve du théorème 3.8. La seule différence est l'ordre dans lequel les moyennes sont faites, on s'arrange ici pour que les vecteurs obtenus par cette procédure contiennent toujours les vecteurs obtenus par la procédure 2 même après plusieurs étapes.

Procédure 1bis : Appairage et moyenne avec réarrangement
Entrée :Un réseau Λ représenté par une base, une fonction f de rejet, des vecteurs $(\mathbf{x}_1, \dots, \mathbf{x}_M)$ de Λ et une sous famille $(\mathbf{u}_1, \dots, \mathbf{u}_q)$ de ces vecteurs.

$(i_1, \dots, i_N) \leftarrow f([\mathbf{u}_1], \dots, [\mathbf{u}_q])$
$(\mathbf{y}_1, \dots, \mathbf{y}_{M'}) \leftarrow$ *appairage et moyenne*$(\mathbf{u}_{i_1}, \dots, \mathbf{u}_{i_N})$
On note $(\mathbf{v}_1, \dots, \mathbf{v}_r)$ tous les vecteurs qui n'ont pas été appairés (y compris les $\mathbf{u}$ qui ne l'ont pqs été à l'étape précédente)
$(\mathbf{z}_1, \dots, \mathbf{z}_{N'}) \leftarrow$ *appairage et moyenne*$(\mathbf{v}_1, \dots, \mathbf{v}_r)$
renvoyer $(\mathbf{y}_1, \dots, \mathbf{y}_{M'})$ et $(\mathbf{y}_1, \dots, \mathbf{y}_{M'}, \mathbf{z}_1, \dots, \mathbf{z}_{N'})$

Quand on utilisera cette procédure plusieurs fois d'affilée on prendra toujours comme vecteurs la deuxième sortie et comme sous famille la première sortie. Cela garantit que après un certain nombres d'étapes, la

procédure 1bis avec en entrée une fonction f, un réseau Λ et une famille $(\mathbf{x}_1, \dots, \mathbf{x}_M)$ en prenant comme sous famille la famille entière contient le résultat de la procédure 2 avec le même réseau, les mêmes vecteurs et la même fonction.

3.2 Le square sampler

Le corollaire 2.15 permet de comprendre, comment on compte résoudre DGS à paramètre arbitraire en temps $2^{n+o(n)}$: on va d'abord le résoudre à "haut" paramètre, ce qu'on sait fair rapidement, puis réduire s par appairage et moyenne.

On se heurte cependant à un problème : comme soulevé précédemment, l'appairage et moyenne naïf (Procédure 1) ne donne pas une gaussienne en sortie.

Pour résoudre ce problème, on peut mettre au point un "échantillonneur du carré" qui donne une fonction de rejet qui permet de rectifier la distribution pour la ramener à ce qu'on aurait

Notons finalement que le square sampler tel que nous le présentons est impossible à implémenter. En effet, tant que l'on ne connaît pas $\max p_i$, on peut difficilement implémenter quoi que ce soit ([ADRS15] propose d'ailleurs un algorithme d'estimation de cette probabilité, mais nous choisissons de présenter une solution plus élégante). Ce serait un problème si notre algorithme final utilisait cette procédure - mais ce n'est pas le cas : comme nous le verrons, les résultats sur la domination (3.11) permettent de conclure même si l'on domine une fonction qu'il est impossible d'implémenter.

Nous citons donc, sans le prouver en détail, ce théorème de [ADRS15].

Théorème 3.1 (échantillonneur du carré/square sampler)**.** *Il existe un algorithme qui, étant donné un paramètre de confiance $\kappa \geqslant 2$, et un $M-$uplet d'éléments de $\{1, ..., N\}$, a pour sortie un uplet d'éléments de ce même ensemble tels que :*

(i) Chaque $j \in \{1, ..., N\}$ apparaît au moins deux fois plus souvent en entrée qu'en sortie

(ii) Si l'entrée est à distance statistique au plus d de $M \geqslant 10\kappa^2/\max p_i$ tirages indépendants selon la loi (supposée connue de l'algorithme) $\mathbf{p} = (p_1, ..., p_N)$ sur $\{1, ..., N\}$, alors il existe des constantes C_1, C_2 indépendantes de M, $\mathbf{p}$ et κ telles que la sortie est à distance statistique au plus $d + C_1 N M e^{-C_2\kappa}$ de M' tirages indépendants selon la loi $\mathbf{p}^2$, où M' est une variable aléatoire vérifiant la borne $M' \geqslant M||\mathbf{p}||^2/(8\kappa \max p_i)$

La preuve de ce théorème n'est pas complètement inaccessible, mais elle est longue et demande bon nombre de lemmes. La grande difficulté est qu'on veut une distribution en sortie qui soit relativement proche de l'entrée, pour ne pas perdre trop de vecteurs dans le moyennage, mais pas trop proche car cela tuerait l'indépendance des vecteurs gardés. L'idée de la preuve est la suivante : on commence par diviser la liste obtenue en $M \max p_i/4$ sous-listes, de taille aléatoire suivant une loi de Poisson $\mathcal{P}(1/\max(p_i))$, ce qui laisse un "reste" libre. On compte ensuite le nombre d'occurences de i dans la j-ème liste (sauf la dernière) - que l'on note $a_{i,j}$, puis l'on tire pour chaque (i, j) associé à un $a_{i,j}$ une Bernoulli $b_{i,j}$ de paramètre $\min(1, a_{i,j}/\kappa)$: les $b_{i,j}$ seront alors raisonnablement proches de Bernoulli indépendantes de paramètre p_i.

La phase finale de cet algorithme consiste à regarder les $M/3$ premiers éléments restants (quitte à échouer s'il n'en reste pas autant) et, en lisant i, chercher le plus petit j tel que $b_{i,j}$ est inutilisé et le retourner seulement si $b_{i,j} = 1$ (quitte à échouer si tous les $b_{i,j}$ ont été utilisés)

Grâce à d'ingénieuses bornes sur les queues des lois de Poisson, on montre que l'algorithme échoue en fait très rarement pourvu que κ soit assez grand.

On va alors immédiatement utiliser cet échantillonneur pour corriger la distribution des classes mod 2Λ :

Théorème 3.2. *Soit Λ un réseau de dimension inférieure ou égale à n. Si l'on applique la procédure 2 avec $f =$ le square sampler sur une entrée constituée de $M \geqslant 10\kappa^2\rho_s(\Lambda)/\rho_s(2\Lambda)$ vecteurs à distance statistique au plus d de M échantillons indépendants de μ_s^Λ, alors la sortie est constituée de M' vecteurs de Λ à distance statistique au plus $d + C_1 M 2^n e^{-C_2\kappa}$ de M' échantillons indépendants de $\mu_{s/\sqrt{2}}^\Lambda$, avec*

$$M' \geqslant \frac{M}{8\kappa} \cdot \frac{\rho_{s/\sqrt{2}}(\Lambda)^2}{\rho_s(\Lambda)\rho_s(2\Lambda)}.$$

Démonstration. La distribution des classes de l'échantillon initial sera à distance statistique au plus d de M tirages indépendants des classes mod 2Λ pour la loi μ_s^Λ. On utilise alors (3.1) pour obtenir un échantillon selon la loi carré correspondante. On a par ailleurs $\max p(c) = \rho_s(2\Lambda)/\rho_s(\Lambda)$ et $\sum_{c\in\Lambda/2\Lambda} p(c)^2 = \frac{\rho_{s/\sqrt{2}}(\Lambda)^2}{\rho_s(\Lambda)^2}$ en vertu de (2.14).

Pour chaque classe c obtenue en sortie, on choisit alors $X_j, X_k \in c$ inutilisés et on ajoute leur moyenne aux vecteurs en sortie. Ceux-ci sont alors indépendants, et distribués correctement en vertu de (2.15). □

A partir de ce théorème, on peut réduire le paramètre autant que l'on veut - et l'on va s'attacher à démontrer que l'on peut le faire en gardant au moins $2^{n/2}$ vecteurs en sortie, ce qui, rappelons-le, est le minimum requis pour avoir une bonne probabilité de trouver un petit vecteur en sortie d'après le théorème 2.5.

Corollaire 3.3. *Soit Λ un réseau de dimension $k \leqslant n$. Pour $\ell \in \mathbb{N}$, $\kappa \geqslant 2$ et $M = (8\kappa)^{\ell+1}2^n$, il existe un algorithme qui, étant donné M vecteurs en entrée distribués selon μ_s^Λ, donne en sortie $M^{(\ell)} \geqslant 2^{n/2}$ vecteurs de Λ dont la distribution est à distance statistique au plus $2^n C_1 \ell M e^{-C_2\kappa}$ de $M^{(\ell)}$ échantillons indépendants de distribution $\mu_{s/2^{\ell/2}}^\Lambda$.*

Démonstration. On va itérer la procédure donnée par (3.2). Il suffit de vérifier qu'après j étapes ($j < \ell$), on a bien $M^{(j)} \geqslant 10\kappa^2\rho_{s/2^{j/2}}(\Lambda)/\rho_{s/2^{j/2}}(2\Lambda)$, et que l'on a bien plus de $2^{n/2}$ vecteurs après ℓ étapes.

$$\begin{aligned} M^{(j)} &\geqslant \frac{M}{(8\kappa)^j} \prod_{i=1}^{j} \frac{\rho_{s/2^{i/2}}(\Lambda)^2}{\rho_{s/2^{(i-1)/2}}(\Lambda)\rho_{s/2^{(i+1)/2}}(\Lambda)} \\ &\geqslant 2^n (8\kappa)^{\ell-j+1} \frac{\rho_{s/\sqrt{2}}(\Lambda)\rho_{s/2^{j/2}}(\Lambda)}{\rho_s(\Lambda)\rho_{s/2^{(j+1)/2}}(\Lambda)}. \qquad \text{(Télescopage)} \end{aligned}$$

Par (1.2), on a $\rho_s(\Lambda) \leqslant 2^{k/2}\rho_{s/\sqrt{2}}(\Lambda) \leqslant 2^{n/2}\rho_{s/\sqrt{2}}(\Lambda)$. On obtient alors, pour $j < \ell$, l'inégalité $M^{(j)} \geqslant 2^{n/2}(8\kappa)^2 \frac{\rho_{s/2^{j/2}}(\Lambda)}{\rho_{s/2^{(j+1)/2}}(\Lambda)}$. Encore par (1.2), on a $\frac{\rho_{s/2^{j/2}}(2\Lambda)}{\rho_{s/2^{(j+1)/2}}(\Lambda)} \geqslant 2^{-k/2} \geqslant 2^{-n/2}$, et alors :

$$\begin{aligned} M^{(j)} &\geqslant 2^{n/2}(8\kappa)^2 \frac{\rho_{s/2^{j/2}}(\Lambda)}{\rho_{s/2^{(j+1)/2}}(\Lambda)} \\ &\geqslant (8\kappa)^2 2^{n/2} \frac{\rho_{s/2^{j/2}}(2\Lambda)}{\rho_{s/2^{(j+1)/2}}(\Lambda)} \cdot \frac{\rho_{s/2^{j/2}}(\Lambda)}{\rho_{s/2^{j/2}}(2\Lambda)} \\ &\geqslant (8\kappa)^2 2^{n/2} 2^{-n/2} \frac{\rho_{s/2^{j/2}}(\Lambda)}{\rho_{s/2^{j/2}}(2\Lambda)} 2^{n/2} \geqslant 10\kappa^2 \frac{\rho_{s/2^{j/2}}(\Lambda)}{\rho_{s/2^{j/2}}(2\Lambda)} 2^{n/2}. \end{aligned}$$

On réutilise le fait que $\frac{\rho_{s/\sqrt{2}}(\Lambda)}{\rho_s(\Lambda)} \geqslant 2^{-n/2}$, conjointement avec la croissance de $\rho_s(\Lambda)$ en s, pour déterminer que :

$$M^{(\ell)} \geqslant 2^n \frac{\rho_{s/\sqrt{2}}(\Lambda)\rho_{s/2^{\ell/2}}(\Lambda)}{\rho_s(\Lambda)\rho_{s/2^{(\ell+1)/2}}(\Lambda)} \geqslant 2^{n-n/2} \frac{\rho_{s/2^{\ell/2}}(\Lambda)}{\rho_{s/2^{(\ell+1)/2}}(\Lambda)} \geqslant 2^{n/2}.$$

□

Insistons sur le fait que si l'on était en mesure d'implémenter le square sampler, on disposerait d'un algorithme qui résout SVP en le temps désiré. C'est d'ailleurs à ça que s'arrête [ADRS15] : ils fournissent une version entièrement fonctionnelle du square sampler. L'apport de [ADRS18] peut se résumer à « si ça marche avec rejet, ça marche sans », ce qui fournit un algorithme à la fois plus élégant, plus simple conceptuellement et plus rapide.

Les sections qui suivent formalisent cette idée : nous allons à présent nous atteler à montrer qu'avec le même nombre de vecteurs en entrée, une sortie probable en rejetant des vecteurs à chaque étape l'est au moins autant si l'on ne rejette rien (3.11).

3.3 Mélanges de gaussiennes

L'outil principal pour résoudre le problème est la notion de mélanges de gaussiennes indépendantes, développé dans [ADRS18]. On ne va désormais plus supposer que les tirages sont indépendants puisque la phase d'appareillage ne nous permet pas d'appliquer cette hypothèse a deux variables choisies (Les v.a. $\mathbf{X}_1, \dots, \mathbf{X}_M$ sont indépendantes, mais si on note $i = i(\mathbf{X}_1, \dots, \mathbf{X}_M)$ et j les variables qui déterminent les indices des deux premiers échantillons appareillés, alors $\mathbf{X}_i$ et $\mathbf{X}_j$ ne sont pas indépendantes donc on ne peut pas appliquer le lemme 2.14).

On donne donc :

Définition 3.4 (Mélange de gaussiennes indépendantes)**.** *Soit Λ un réseau de dimension n.*
Une famille de variables aléatoires $(\mathbf{X}_i)_{i\in\mathbb{N}}$ à valeurs dans Λ munie d'une variable aléatoire M à valeurs dans $\mathbb{N}$ est appelée mélange de gaussiennes de paramètre s sur Λ si
$\forall p \in \mathbb{N}, \forall (c_1, \dots, c_p) \in \left(\Lambda/2\Lambda\right)^p$*, si on note*

$$B = \bigcap_{i=1}^{p} (\mathbf{X}_i^{-1}(c_i)) \cap (M^{-1}(\{p\}))$$

alors les variables $\mathbf{X}_1, \dots, \mathbf{X}_p$ vérifient :

(i) $\forall 1 \leqslant i \leqslant p, \forall \mathbf{y} \in c_i, \mathbb{P}[\mathbf{X}_i = \mathbf{y}|B] = \frac{\rho_s(\mathbf{y})}{\rho_s(c_i)}$
(Elles suivent des lois gaussiennes sur les classes mod 2Λ)

(ii) $\forall (A_i)_{1\leqslant i\leqslant p}, A_i \subset c_i, \mathbb{P}[\mathbf{X}_1^{-1}(A_1) \cap \dots \cap \mathbf{X}_p^{-1}(A_p)|B] = \mathbb{P}[X_1^{-1}(A_1)|B] \dots \mathbb{P}[\mathbf{X}_p^{-1}(A_p)|B]$
(Elles conservent une certaine notion d'indépendance dans les classes mod 2Λ)

On notera $\mathcal{X}$ le mélange de gaussiennes.

Remarque. *M permet de prendre en compte le nombre de vecteurs obtenus après l'étape d'appairage et moyennage, qui peut varier légèrement selon l'entrée. Même s'il est plus facile de se donner formellement une infinité de variable, nous ne traiterons en pratique qu'un nombre fini de variable, dans le sens où M aura toujours un support fini.*

En fait cette définition se prête bien à l'opération d'appairage et moyennage, comme le souligne le théorème suivant :

Théorème 3.5. *Soit Λ un réseau de rang n, $\mathcal{X} = \big((\mathbf{X}_i), M\big)$ un mélange de gaussiennes de paramètre s sur Λ*
le resultat $\big((\mathbf{Y}_j), M'\big)$ du procédé d'appairage et moyennage appliqué à $(\mathbf{X}_1, \dots, \mathbf{X}_M)$ (où M' est le nombre de vecteurs obtenus) est un mélange de gaussiennes sur Λ de paramètre $s/\sqrt{2}$

Démonstration. (i) On se donne $q \in \mathbb{N}, (c'_1, \dots, c'_q) \in \left(\Lambda/2\Lambda\right)^q$ et on note

$$A = \bigcap_{i=1}^{q} \left(\mathbf{Y}_i^{-1}(c'_i)\right) \cap \left(M^{-1}(\{q\})\right)$$

On note $\mathcal{B}$ l'ensemble des évènements de cette forme en remplaçant les $\mathbf{Y}_i$ par les $\mathbf{X}_i$ et M' par M. On a donc, avec $\mathbf{y} \in c'_1$,

$$\mathbb{P}[\mathbf{Y}_1 = \mathbf{y}|A] = \frac{1}{\mathbb{P}[A]} \sum_{B \in \mathcal{B}} \mathbb{P}\big[(\mathbf{Y}_1 = \mathbf{u}) \cap B \cap A\big]$$

Donnons-nous un $B \in \mathcal{B}$, dans ce B, il existe i, i' tels que $\mathbf{Y}_1 = \frac{\mathbf{X}_i + \mathbf{X}_{i'}}{2}$ on supposera sans perte de généralités qu'il s'agit de $\mathbf{X}_1$ et $\mathbf{X}_2$, on a donc $c_1 = c_2 = c$
On a alors

$$\begin{aligned}
\mathbb{P}[(\mathbf{Y}_1 = \mathbf{y}) \cap A \cap B] &= \mathbb{P}[(\mathbf{X}_1 + \mathbf{X}_2 = 2\mathbf{y}) \cap B]\mathbb{P}\left[\bigcap_{i=2}^{p} (\mathbf{Y}_i^{-1}(c'_i))\right] && \text{(par indépendance)} \\
&= \alpha \sum_{\mathbf{x} \in c + 2\Lambda} \rho_s(\mathbf{x})\rho_s(2\mathbf{y} - \mathbf{x}) && (\alpha \text{ constante indépendante de } \mathbf{y}) \\
&= \alpha\rho_{s/\sqrt{2}}(\mathbf{y}) \sum_{\mathbf{x} \in c} \rho_{s/\sqrt{2}}(\mathbf{y} - \mathbf{x}) && \text{(voir 1.24)} \\
&= \alpha\rho_{s/\sqrt{2}}(\mathbf{y})\rho_{s/\sqrt{2}}(c + \mathbf{y}) \\
&= \alpha\rho_{s/\sqrt{2}}(\mathbf{y})\rho_{s/\sqrt{2}}(c + c'_1)
\end{aligned}$$

Et donc $\mathbb{P}[(\mathbf{Y}_1 = \mathbf{y})|A] = \beta\rho_{s/\sqrt{2}}(\mathbf{y})$, β constante indépendante de $\mathbf{y}$. La loi de probabilité étant à support dans c'_1, on a forcément $\mathbb{P}[(\mathbf{Y}_1 = \mathbf{y})|A] = \frac{\rho_{s/\sqrt{2}}(\mathbf{y})}{\rho_{s/\sqrt{2}}(c'_1)}$

(ii) Pour l'indépendance, on se donne un $A = \bigcap A_j$ de la même forme que dans (i) et on se donne $(\mathbf{y}_j)_{1 \leqslant j \leqslant q}, \mathbf{y}_j \in \mathbf{Y}(A_j)$

On remarque alors, en procédant de la même manière que pour la preuve de la loi que $\mathbb{P}[(\mathbf{Y}_1 = \mathbf{y}_1, \dots, \mathbf{Y}_q = \mathbf{y}_q)|A]$ est égal à une constante multiplicative près à

$$\prod_{j=1}^{q} \rho_{s/\sqrt{2}}(\mathbf{y}_j)$$

Or $\mathbb{P}[(\mathbf{Y}_1 = \mathbf{y}_1, \dots, \mathbf{Y}_q = \mathbf{y}_q)|A]$ définit une loi de probabilité sur $c'_1 \times \cdots \times c'_q$, on a donc :

$$\mathbb{P}\big[(\mathbf{Y}_1 = \mathbf{y}_1, \dots, \mathbf{Y}_q = \mathbf{y}_q)|A\big] = \mathbb{P}[(\mathbf{Y}_1 = \mathbf{y}_1)|A] \dots \mathbb{P}[(\mathbf{Y}_q = \mathbf{y}_q)|A].$$

□

On peut faire de même avec la procédure 1bis en vertu de :

Proposition 3.6. *(*) Soit Λ un réseau, f une fonction de rejet, $s \in \mathbb{R}_+^*$ et $\mathcal{X}$ un mélange de gaussiennes de paramètre s sur Λ.*
Si $m \in \mathbb{N}$, on note $(S^m, \mathcal{X}^m)$ le résultat de la procédure 1bis appliquée m fois a ΛX en prenant comme fonction de rejet f et comme sous-famille de vecteurs $\mathcal{X}$ au départ. $\mathcal{X}^m$ désigne le mélange de gaussiennes et S^m les indices de la sous famille. On note M^m la longueur de $\mathcal{X}^m$

On a alors : $\forall m \in \mathbb{N}, \forall p \in \mathbb{N}, \forall (c_i)_{1 \leqslant i \leqslant N} \in (\Lambda/2\Lambda)^p, \forall S \subset \{1 \dots p\}$ si on note

$$B = \bigcap_{i=1}^{p} (\mathbf{X}_i^{-1}(c_i)) \cap ((M^m)^{-1}\{p\} \cap (S^m)^{-1}\{S\})$$

(i) $\forall 1 \leqslant i \leqslant p, \forall y \in c_i, \mathbb{P}[\mathbf{X}_i^m = \mathbf{y}|B] = \frac{\rho_{s/2^{m/2}}(\mathbf{y})}{\rho_{s/2^{m/2}}(c_i)}$

(ii) $\forall (A_i)_{1\leqslant i\leqslant p}, A_i \subset c_i, \mathbb{P}[(\mathbf{X}_1^m)^{-1}(A_1)\cap\cdots\cap(\mathbf{X}_p^m)^{-1}(A_p)|B] = \mathbb{P}[(\mathbf{X}_1^m)^{-1}(A_1)|B]\dots\mathbb{P}[(\mathbf{X}_p^m)^{-1}(A_p)|B]$

Démonstration. Pour cela, on procède par récurrence sur m. Les calculs sont très similaires à la preuve de 3.5, on doit cependant rajouter un conditionnement sur la valeur de S^{m-1} qui rajoute un terme multiplicatif qui dépend de f. Cependant tout tirage utilisé pour la calculer est indépendant du mélange donc cela ne nous dérange pas. □

On peut avec ceci calculer les lois des X_i^m conditionnées par l'appartenance aux classes d'équivalence mod 2Λ et on obtient que $\mathcal{X}^m$ est un mélange de gaussiennes de paramètre $\frac{s}{2^{m/2}}$

3.4 Domination de variables aléatoires

Le dernier élément nécessaire pour passer de l'algorithme avec rejet à celui sans est la domination. C'est un moyen de formaliser le fait que l'on aura une probabilité plus élevée de trouver un certain vecteur dans le résultat de l'algorithme simplifié.

Définition 3.7 (Domination de mélanges de gaussiennes)**.** *Soit Λ un réseau, et $\mathcal{X} = (X_1, \dots, X_N)$, $\mathcal{Y}(Y_1, \dots, Y_M)$ deux mélanges de gaussiennes sur Λ.*
On note $c_1, \dots c_q$ les classes mod 2Λ dans Λ et on note aussi, pour $1 \leqslant i \leqslant q, |\mathcal{X}|_i$ le nombre de vecteurs de $\mathcal{X}$ à l'intérieur de c_i. On dit que $\mathcal{X}$ domine $\mathcal{Y}$ si

$$\forall (k_1, \dots, k_q) \in \mathbb{N}^q, \mathbb{P}\left[\bigcap_{i=1}^{q}(|\mathcal{X}|_i \geqslant k_i)\right] \geqslant \mathbb{P}\left[\bigcap_{i=1}^{q}(|\mathcal{Y}|_i \geqslant k_i)\right]$$

C'est la manière de formaliser que "Il y a plus de vecteurs dans $\mathcal{X}$ que dans $\mathcal{Y}$". La domination est alors une relation d'ordre (partielle) sur les mélanges de gaussiennes sur Λ.

Le lemme suivant exprime le fait que cette définition se comporte bien avec les mélanges de gaussiennes et la procédure d'appairage et moyenne.

Proposition 3.8. *(*) Soit Λ un réseau, Soit $\mathcal{X}$ un mélange de gaussiennes et $m \in N$. Le résultat de la procédure 1 appliqué m fois à $\mathcal{X}$ domine le résultat de la procédure 2 appliquée m fois à $\mathcal{X}$ avec n'importe quelle fonction de rejet.*

Démonstration. On note $\mathcal{X}^m$(resp. $\tilde{\mathcal{X}}^m$ et $\mathcal{Y}^m$) le résultat de la procédure 1 (resp. procédure 1bis et 2) appliquée à $\mathcal{X}$ m fois avec une certaine fonction de rejet f et comme sous famille de départ tout $\mathcal{X}$.

On va en fait montrer par récurrence sur m que

$$\forall m \in \mathbb{N}, \forall (k_1, \dots, k_q) \in \mathbb{N}^q, \mathbb{P}\left[\bigcap_{i=1}^{q}(|\mathcal{X}^m|_i \geqslant k_i)\right] = \mathbb{P}\left[\bigcap_{i=1}^{q}(|\tilde{\mathcal{X}}^m|_i \geqslant k_i)\right]$$

Cela suffit car il est clair que $\tilde{\mathcal{X}}^m$ domine $\mathcal{Y}^m$ car $\mathcal{Y}^m$ est une sous famille de $\tilde{\mathcal{X}}^m$

Pour $m = 0$ le résultat est clair car $\mathcal{X}^0 = \tilde{\mathcal{X}}^0 = \mathcal{X}$

Supposons le résultat prouvé pour $m-1$, on a alors $\forall (k_1, \dots, k_q) \in \mathbb{N}^q$:

$$\mathbb{P}\left[\bigcap_{i=1}^{q}(|\mathcal{X}^m|_i \geqslant k_i)\right] =$$

$$\sum_{(d_1,\dots,d_q)\in\mathbb{N}^q} \mathbb{P}\left[\bigcap_{i=1}^{q}(|\mathcal{X}^m|_i \geqslant k_i)\Big|\bigcap_{j=1}^{q}(|\mathcal{X}^{m-1}|_j \in \{2d_j, 2d_j+1\})\right]\mathbb{P}\left[\bigcap_{j=1}^{q}(|\mathcal{X}^{m-1}|_i \in \{2d_j, 2d_j+1\})\right]. \quad (*)$$

Or pour un certain $d=(d_1,\dots,d_q)\in\mathbb{N}^q$ on a

$$\mathbb{P}\left[\bigcap_{i=1}^{q}(|\mathcal{X}^m|_i \geqslant k_i)\Big|\bigcap_{j=1}^{q}(|\mathcal{X}^{m-1}|_j \in \{2d_j, 2d_j+1\})\right] = \sum_{A\in\mathcal{A}_d}\prod_{j=1}^{q} d_j!\prod_{i=1}^{q}\frac{1}{a_{i,j}!}p_{i,j}^{a_{i,j}}. \quad (**)$$

Où $\mathcal{A}_d = \left\{A\in\mathcal{M}_q(\mathbb{N})|\forall i, \sum_{j=1}^{q} a_{i,j}\geqslant k_i, \forall j, \sum_{i=1}^{q} a_{i,j} = d_j\right\}$ et $p_{i,j} = \mathbb{P}\left[\frac{\mathbf{X}_1+\mathbf{X}_2}{2}\in c_j\big|\mathbf{X}_1,\mathbf{X}_2\in c_j\right]$ avec $\mathbf{X}_1,\mathbf{X}_2$ des variables dans un mélanges de gaussiennes. Cette probabilité ne dépend pas du mélange d'après le même calcul que (3.5).

On voit ceci car une partition de l'évènement $\bigcap_{i=1}^{q}(|\mathcal{X}^m|_i\geqslant k_i)\cap\bigcap_{j=1}^{q}(|\mathcal{X}^{m-1}|_j\in\{2d_j,2d_j+1\})$ est donnée par les évènements :

$$\bigcap_{i=1}^{q}\bigcap_{j=1}^{q}(|\mathcal{X}^m\wedge\mathcal{X}^{m-1}|_{i,j}=a_{i,j}), A\in\mathcal{A}$$

où $|\mathcal{X}^m\wedge\mathcal{X}^{m-1}|_{i,j}$ désigne le nombre de vecteurs de $\mathcal{X}^m$ dans c_i qui proviennent de la moyenne de deux vecteurs de $\mathcal{X}^{m-1}$ dans c_j.

Si on note $\mathcal{C} = \left\{(p,\alpha_1,\dots,\alpha_p), \alpha_k\in {}^{\Lambda}\!/_{2\Lambda}|\forall j, \left(\sum_{k=1}^{p}\mathbb{1}_{(\alpha_k\in c_j)} = c_j\right)\in\{2d_j,2d_j+1\}\right\}$
on a, pour $A\in\mathcal{A}$:

$$\begin{aligned}
&\mathbb{P}\left[\bigcap_{i=1}^{q}\bigcap_{j=1}^{q}(|\mathcal{X}^m\wedge\mathcal{X}^{m-1}|_{i,j}=a_{i,j})\right]\\
&=\sum_{C\in\mathcal{C}}\mathbb{P}\left[\bigcap_{i=1}^{q}\bigcap_{j=1}^{q}(|\mathcal{X}^m\wedge\mathcal{X}^{m-1}|_{i,j}=a_{i,j})\big|(M^{m-1},\mathbf{X}_1^{m-1},\dots,\mathbf{X}_p^{m-1})=C\right]\mathbb{P}\left[(M^{m-1},\mathbf{X}_1^{m-1},\dots,\mathbf{X}_p^{m-1})=C\right]\\
&=\left(\prod_{j=1}^{q} d_j!\prod_{i=1}^{q}\frac{1}{a_{i,j}!}p_{i,j}^{a_{i,j}}\right)\sum_{C\in\mathcal{C}}\mathbb{P}\left[(M^{m-1},\mathbf{X}_1^{m-1},\dots,\mathbf{X}_p^{m-1})=C\right]\\
&=\left(\prod_{j=1}^{q} d_j!\prod_{i=1}^{q}\frac{1}{a_{i,j}!}p_{i,j}^{a_{i,j}}\right)\mathbb{P}\left[\bigcap_{j=1}^{q}(|\mathcal{X}^{m-1}|_j\in\{2d_j,2d_j+1\})\right].
\end{aligned}$$

Cela implique $(**)$ car $\bigcap_{j=1}^{q}(|\mathcal{X}^{m-1}|_j\in\{2d_j,2d_j+1\})\subset\bigcap_{j=1}^{q}(|\mathcal{X}^{m-1}|_j\in\{2d_j,2d_j+1\})$

Le même calcul pour $(**)$ reste valide, a quelques modifications près pour $\tilde{\mathcal{X}}$. De plus comme

$$\mathbb{P}\left[\bigcap_{j=1}^{q}(|\mathcal{X}^{m-1}|_i\in\{2d_j,2d_j+1\})\right]$$

s'exprime à partir des

$$\mathbb{P}\left[\bigcap_{j=1}^{q}(|\mathcal{X}^{m-1}|_i\geqslant 2d_j)\right], (d_1,\dots,d_q)\in\mathbb{N}^q.$$

On obtient alors avec l'hypothèse de récurrence que $(*)$ est égale pour $\mathcal{X}$ et pour $\tilde{\mathcal{X}}$.

□

La proposition 3.8 est en fait une version affaiblie du résultat de [ADRS18] (Corollaire 3.3.3), que nous n'avons pas réussi à prouver tel quel.

Lemme 3.9. *Soit Λ un réseau, $\mathcal{X}, \mathcal{Y}$ deux mélanges de gaussiennes sur Λ. Si $\mathcal{X}$ domine $\mathcal{Y}$, alors $\mathcal{X}'$ domine $\mathcal{Y}'$ avec $\mathcal{X}'$ et $\mathcal{Y}'$ les résultats de la procédure d'appairage et moyennage.*

On peut aussi, comme le montre le lemme suivant, réduire la taille des ensembles considérés pour regarder uniquement les petits vecteurs :

Proposition 3.10. *Soit Λ un réseau, $\mathcal{X}$ et $\mathcal{Y}$ deux mélanges de gaussiennes avec $\mathcal{X}$ qui domine $\mathcal{Y}$ Si $A \subseteq c_i$ pour une certaine classe d'équivalence mod 2Λ c_i alors :*

$$\forall k \in \mathbb{N}, \mathbb{P}[|\mathcal{X}|_A \geqslant k] \geqslant \mathbb{P}[|\mathcal{Y}|_A \geqslant k]$$

Démonstration.

$$\begin{aligned}
\mathbb{P}[|\mathcal{X}|_A \geqslant k] &= \sum_{r=k}^{\infty} \mathbb{P}[|\mathcal{X}|_A = r] \\
&= \sum_{r=k}^{\infty} \sum_{m=n}^{\infty} \mathbb{P}[|\mathcal{X}|_A = r \big| |\mathcal{X}|_i = m] \mathbb{P}[|\mathcal{X}|_i = m] \\
&= \sum_{m=k}^{\infty} \sum_{r=k}^{m} \binom{m}{r} \mathbb{P}[\mathbf{X} \in A \big| \mathbf{X} \in c_i]^r \mathbb{P}[\mathbf{X} \in c_i \backslash A \big| \mathbf{X} \in c_i]^{m-r} \Big(\mathbb{P}[|\mathcal{X}|_i \geqslant m] - \mathbb{P}[|\mathcal{X}|_i \geqslant m+1] \Big)
\end{aligned}$$

La finitude du support des sommes permet de les échanger, et on peut se contenter d'écrire $\mathbb{P}[\mathbf{X} \in A \big| \mathbf{X} \in c_i]$ (où $\mathbf{X}$ est une composante de $\mathcal{X}$) car cette probabilité ne dépend pas de la variable choisie, car on la regarde a l'intérieur d'une classe.

On constate alors, pour tout $m \geqslant k$, que le coefficient devant $\mathbb{P}[|\mathcal{X}|_i \geqslant m]$ est

$$\sum_{r=k}^{m} \binom{m}{r} \mathbb{P}[\mathbf{X} \in A \big| \mathbf{X} \in c_i]^r \mathbb{P}[\mathbf{X} \in c_i \backslash A \big| \mathbf{X} \in c_i]^{m-r} - \sum_{r=k}^{m-1} \binom{m-1}{r} \mathbb{P}[\mathbf{X} \in A \big| \mathbf{X} \in c_i]^r \mathbb{P}[\mathbf{X} \in c_i \backslash A \big| \mathbf{X} \in c_i]^{m-r-1}$$

Il s'agit de la différence $\mathbb{P}[B_m \geqslant k] - P[B_{m-1} \geqslant k]$ où $B_m \sim \mathcal{B}(m,p), B_{m-1} \sim \mathcal{B}(m-1,p)$ avec $p = \mathbb{P}[X \in A \big| X \in c_i]$

Il est alors clair que ce coefficient est positif puisque B_l peut être vu comme la somme de B_{l-1} avec une variable de Bernouilli indépendante.

On conclut en disant que les $\mathbb{P}[|\mathcal{X}|_i \geqslant l]$ sont les seules grandeurs qui changent entre $\mathcal{X}$ et $\mathcal{Y}$ □

Remarque. *On peut en fait, par la même preuve itérée, montrer que l'on peut passer sur des plus petits ensembles simultanément sur toutes les classes mod 2Λ*

3.5 Simplification de l'algorithme

Les résultats du paragraphe précédent donnent qu'un algorithme sans rejet fonctionnera toujours au moins aussi bien qu'un algorithme avec rejet, et ce, quelle que soit la procédure de rejet.

Théorème 3.11. *Soit Λ un réseau.*

Si on montre que un algorithme utilisant la procédure 2 avec une certaine fonction f a une certaine probabilité d'obtenir un certain petit vecteur, alors le même algorithme sans rejet aura au moins autant de chance de trouver ce vecteur.

Démonstration. Ce théorème est une conséquence du théorème 3.8 et de la proposition 3.10 appliquée à $A = \{x\}$ avec x un petit vecteur du réseau. □

3.6 Algorithme final

Nous présentons finalement notre résultat principal :

Algorithme 3.12 (SVP en $2^{n+o(n)}$)**.** *L'algorithme suivant résout le problème* SVP *en temps* $2^{n+o(n)}$*, excepté avec probabilité* $1 - \exp\big(-\Omega(n)\big)$

Entrée : la base $\mathbf{B}$ d'un réseau $\Lambda \subset \mathbb{R}^n$
Sortie : Un vecteur $\mathbf{x} \in \Lambda$ avec $||\mathbf{x}|| = \lambda_1(\Lambda)$

Utiliser l'algorithme LLL pour obtenir $\hat{\lambda}$ vérifiant $\lambda_1 \leqslant \hat{\lambda} \leqslant 2^{n/2}\lambda_1$
Pour $j = 1, ..., pn$:
$s_j := e^{-j\theta}\sqrt{\frac{2\pi e}{\beta^2 n}}\hat{\lambda}$
Soit $\Lambda' \subset \Lambda$ le résultat de l'algorithme 2.13 avec entrée a et $r_j := \sqrt{\pi}\frac{s_j}{\sqrt{\log(2n+4)}}$.
Tirer $\mathbf{X}_1, ..., \mathbf{X}_M$ selon $\mu_{s_j}^{\Lambda'}$ grâce à l'algorithme (2.12).
Pour $k = 1, ..., \ell$:
$(\mathbf{X}_1, ..., \mathbf{X}_{M'}) \leftarrow$ appairage et moyenne$(\mathbf{X}_1, ..., \mathbf{X}_M)$
$M' \leftarrow M$
$\mathbf{Y}_j := \text{argmin}_{\mathbf{X}_i \neq 0}||\mathbf{X}_i||$
Retourner le vecteur $\text{argmin}_j(||\mathbf{Y}_j||)$

(Les paramètres a, M et ℓ, ainsi que les constantes p et θ seront spécifiés dans la preuve)

Démonstration. La complexité en temps de l'algorithme est claire. En effet, LLL tourne en temps poly(n), et les opérations suivantes sont répétées pn fois :

- réduire le réseau : temps poly$(n) \cdot 2^{O(a)}$
- tirage de M vecteurs selon μ_s^Λ : temps en $A_1\text{poly}(n)M$
- appairage et moyenne, ℓ fois : temps inférieur à $\ell \cdot \text{poly(n)} \cdot M$
- Recherche du plus petit vecteur en sortie : temps inférieur à $A_2\text{poly(n)} \cdot M$ (moins de M vecteurs + calcul de la norme d'un vecteur de taille n)
- rajout de $A_3 pn^2$ pour trouver le plus petit $\mathbf{Y}_j$

Ce qui fait globalement un temps total inférieur à $\ell \cdot \text{poly}(n) \cdot (2^{O(a)} + M)$. Comme on posera a de sorte que $O(a) \leqslant n$ et $M = 2^{n+o(n)}$, on aura effectivement la complexité annoncée.

On prouve simplement que le même algorithme en remplaçant "appairage et moyenne" par "appairage et moyenne avec rejet" fonctionne également, et on concluera grâce au théorème (3.11). Commençons par une "chasse au constantes" pour contrôler le temps de l'algorithme.

Soit $\theta > 0$ un réel vérifiant l'inégalité $e^{\frac{\beta^2}{2e}+\theta} < \sqrt{2}$, et p un entier tel que $e^{p\theta} > \sqrt{2}$.

Le premier paramètre auquel il faut prêter attention est a : notre algorithme tourne en $2^{O(a)}$, donc 2^{Ca} pour un certain C - on va donc poser $a = C^{-1}n$. Ainsi, $a^{n/a}$ est de la forme Bn^C.

La raison pour laquelle on balaie les s_j a été plus ou moins expliquée avant (2.5) : on ne connaît pas exactement le paramètre $\hat{s}$, mais si on en a une bonne approximation, on aura des vecteurs de longueur minimale en sortie avec une probabilité élevée. Notre algorithme va dominer un tirage gaussien de paramètre $s_j/2^{\ell/2}$ sur le réseau Λ' : on veut alors que si $s_j/2^{\ell/2}$ vérifie l'inégalité $\hat{s} \leqslant s_j/2^{\ell/2} \leqslant e^\theta \hat{s}$, alors $r_j a^{-n/a} \geqslant \lambda_1$ (afin de remplir les conditions de (2.13) pour que $\lambda_1(\Lambda) = \lambda_1(\Lambda')$). Il suffit de demander $2^{\ell/2}\hat{s} \geqslant B^{-1}n^C\lambda_1 \cdot \sqrt{\frac{\log(2n+4)}{\pi}}$. Puisque $\hat{s}^{-1}$ est en $\sqrt{n}$, cela se traduit par $\ell/2 \geqslant 2(C+1)\log_2(n)$.

Reste à déterminer M, qui doit être suffisant pour faire fonctionner le tirage avec rejet. Pour ce faire, on aimerait poser $\kappa = Dn$, avec D assez grand pour que la distance statistique dans (3.3) soit faible - et celle-ci est de la forme $\exp[C_1 n - C_2\kappa]$ (le facteur ℓ est négligeable car d'ordre logarithmique).

On pose donc $M = (8\kappa)^{\ell+1}2^n = 2^{n+\Theta(\log^2(n))} = 2^{n+o(n)}$

On se fixe donc j tel que $\hat{s} \leqslant s_j/2^{\ell/2} \leqslant e^{\theta}\hat{s}$ - rappelons qu'alors $\mu^{\Lambda'}_{s_j/2^{\ell/2}}(\mathbf{x}) \geqslant \mu^{\Lambda}_{s_j/2^{\ell/2}}(\mathbf{x}) \geqslant \left(e^{\frac{\beta^2}{2\epsilon}+\theta}\right)^n := \alpha_\theta^n > 2^{-n/2}$ pour $||\mathbf{x}|| = \lambda_1$, par (2.4).

Si l'on exécute l'algorithme de tirage avec rejet sur Λ', qui est de dimension inférieure à n, on obtient au moins $2^{n/2}$ vecteurs à distance statistique $\exp\big(-\Omega(n)\big)$ de $2^{n/2}$ tirages indépendants de $\mu^{\Lambda'}_{s_j/2^{\ell/2}}$ par (3.3).

Mais alors, pour $||\mathbf{x}|| = \lambda_1$, on peut estimer la probabilité que $\mathbf{x}$ apparaisse dans $2^{n/2}$ tirages indépendants de $\mathbf{Z}_i \sim \mu^{\Lambda'}_{s_j/2^{\ell/2}}$ (en fait, on va estimer la probabilité que $\mathbf{x}$ n'apparaisse pas, ce qui est bien plus pratique) :

$$\begin{aligned}
\mathbb{P}[\forall i, \mathbf{Z}_i \neq \mathbf{x}] &= \prod_{i=1}^{2^{n/2}} \mathbb{P}[\mathbf{Z}_i \neq \mathbf{x}] \\
&= \big(1 - \mathbb{P}[\mathbf{Z}_i = \mathbf{x}]\big)^{2^{n/2}} \\
&\leqslant (1-\alpha_\theta^n)^{2^{n/2}} \\
&\leqslant \exp\Big(2^{n/2}\log(1-\alpha_\theta^n)\Big) \\
&\leqslant \exp\Big(-(\sqrt{2}\alpha_\theta)^n\Big) && (\log(1+x) \leqslant x) \\
&\leqslant \exp\big(-\Omega(n)\big) && (\sqrt{2}\alpha_\theta > 1)
\end{aligned}$$

Ainsi, un petit vecteur donné se trouvera en sortie avec probabilité $1-\exp\big(-\Omega(n)\big)$ pour $2^{n/2}$ tirages de $\mu_s^{\Lambda'}$. En soustrayant la distance statistique de $\exp\big(-\Omega(n)\big)$ due à l'erreur introduite dans le processus de réduction, on reste en $1-\exp\big(-\Omega(n)\big)$, ce qui permet d'assurer, à exponentielle décroissante près, la présence d'un petit vecteur en sortie.

□

Références

[MG] Complexity of lattice problems, a cryptographic perspective, D. Micciancio, S. Goldwasser

[ADRS15] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the shortest vector problem in 2^n time via discrete gaussian sampling. http ://arxiv.org/abs/1412.7994

[ADRS18] Divesh Aggarwal and Noah Stephens-Davidowitz. Just take the average ! An embarrassingly simple 2^n-Time algorithm for SVP (and CVP). https ://arxiv.org/abs/1709.01535

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. *STOC*, pages 575–584, 2013

[Sch87] C.P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53(23) : 201 – 224, 1987.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. *STOC*, pages 601–610, 2001

[PS09] Xavier Pujol and Damien Stehlé. Solving the shortest lattice vector problem in time $2^{2.465n}$. *IACR Cryptology ePrint Archive*, 2009 : 605, 2009

[MV13] Daniele Micciancio et Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. *SIAM Journal on Computing*, 42(3) : 1364–1391, 2013.

[GGH] Oded Goldreich, Shafi Goldwasser and Shai Halevi. Public-key cryptosystems from lattice reduction problems. *CRYPTO '97 : Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*. London : Springer-Verlag 112–131

[OR85] Andrew Odlyzko and Herman te Riele. Disproof of the Mertens conjecture. Journal für die Reine und angewandte Mathematik 357, 138–160 (1985)