INSTITUT
POLYTECHNIQUE
DE PARIS

NNT : 20XXIPPAXXXX

Thèse de doctorat

Decoding Algorithms for Lattices

Algorithmes de décodage pour les réseaux de points

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

École doctorale n°626 de l'Institut Polytechnique de Paris
Spécialité de doctorat: Réseaux, Informations et Communications

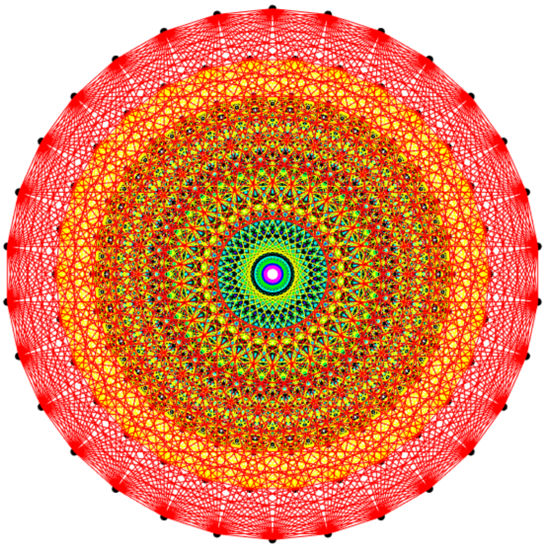
Thèse présentée et soutenue à XXX, le XXX, par

Vincent Corlay

Composition du Jury :

Charly Poulliat Professeur, INP-ENSEEIHT Toulouse	Président
Frédérique Elise Oggier Maître de conférences, Nanyang Technological University	Rapporteur
Jean-Pierre Tillich Directeur de recherche, INRIA Paris	Rapporteur
Jean-Claude Belfiore Docteur, Huawei France	Examinateur
Robert Calderbank Professeur, Duke University	Examinateur
Hans-Andrea Loeliger Professeur, ETH Zürich	Examinateur
Philippe Ciblat Professeur, Télécom Paris	Directeur de thèse
Loïc Brunel Docteur, Mitsubishi Electric R&D Centre Europe	Co-Directeur de thèse
Joseph Jean Boutros Professeur, Texas A&M University	Invité

"Euclidean space coding is to Hamming space coding
as classical music is to rock 'n' roll."
Neil Sloane



This figure depicts a 2-dimensional projection of the 240 lattice points composing the first shell of the Gosset lattice E_8. The edges join lattice points that are nearest neighbors.
See `http://www.math.lsa.umich.edu/~jrs/coxplane.html` for more information on this projection.

Acknowledgments

Contents

Presentation of the thesis and its main topics

A new framework for building and decoding group codes

Lattices and neural networks with and without learning

Conclusions and perspectives

Notations

- AWGN: additive white Gaussian noise.
- BDD: bounded-distance decoder.
- BW: Barnes-Wall.
- CPWL: continuous piecewise linear.
- CVP: closest vector problem.
- DNN: deep neural network.
- HLD: hyperplane logical decoder.
- MIMO: multiple-input multiple-output.
- MLD: maximum-likelihood decoding.
- OSD: ordered statistics decoder.
- QMLD: quasi-maximum-likelihood decoding.
- ReLU: rectified linear unit.
- SNR: signal-to-noise ratio.

Part

Presentation of the thesis and its main topics

Chapter 1

Introduction

1.1 Abstract

This thesis discusses two problems related to lattices, an old problem and a new one. Both of them are lattice decoding problems: Namely, given a point in the space, find the closest lattice point.

The first problem is related to channel coding in moderate dimensions. While efficient lattice schemes exist in low dimensions $n \leq 30$ and high dimensions $n \geq 1000$, this is not the case of intermediate dimensions. We investigate the decoding of interesting lattices in these intermediate dimensions. We introduce new families of lattices obtained by recursively applying parity checks. These families include famous lattices, such as Barnes-Wall lattices, the Leech and Nebe lattices, as well as new parity lattices. We show that all these lattices can be efficiently decoded with an original recursive list decoder.

The second problem involves neural networks. Since 2016 countless papers tried to use deep learning[1] to solve the decoding/detection problem encountered in digital communications. We propose to investigate the complexity of the problem that neural networks should solve. We introduce a new approach to the lattice decoding problem to fit the operations performed by a neural network. This enables to better understand what a neural network can and cannot do in the scope of this problem, and get hints regarding the best architecture of the neural network. Some computer simulations validating our analysis are provided.

1.2 Thesis outline

The first part of the document introduces the main topics covered in this thesis. We discuss the channel coding problem and propose an overview of lattices.

- Chapter 2 provides a brief history of the channel coding problem. We present the main strategies investigated by coding theorists to approach the Shannon limit with a reasonable decoding complexity. Reasons for failure or success of these strategies are discussed. We also explain the differences between the power-limited and the bandwidth-limited regimes. Lattice coding for the latter regime is introduced.

- Chapter 3 is an introduction to lattices. We first give in Section 3.1 an overview of the different absract meaning of the term lattice that one may encounter. Standard lattice parameters are then defined. Section 3.2 describes several categories of lattice decoders. Coset decomposition and constructions of lattices from codes are also presented. We discuss in Section 3.3 the two most famous lattice packing problems. Finally, the last section describes the main computational problems related to lattices. Hardness results as well as existing algorithms are presented. We also show applications in cryptography and digital communications.

The second part of the thesis investigates the construction and the decoding of a remarkable set of lattices and codes viewed as group codes: We treat in a unified framework the Leech lattice and the Golay code in dimension 24, the Nebe lattice in dimension 72, the Barnes-Wall lattices, and the Reed-Muller codes. We also present a new family of lattices called the parity lattices. The common aspect of these group codes is that they can be obtained as single parity-check k-groups or via the k-ing construction. We

[1] Sometimes more shallow than deep.

exploit these constructions to introduce a new efficient paradigm for decoding. This leads to efficient list decoders and quasi-optimal decoders on the Gaussian channel. This part of the thesis is divided in three chapters.

- The k-ing construction and the single parity-check k-groups are introduced in Section 4.2 of Chapter 4. The baseline decoding algorithms as well as the splitting strategy for efficient list decoding are described in Section 4.3.
- Chapter 5 is dedicated to the study of a new family of lattices named parity lattices. The chapter is divided into two main sections. In Section 5.1, we first define the parity lattices. Recursive versions of the algorithms of Chapter 4, to decode the parity lattices, are then presented. We also provide formulas to assess the performance of these algorithms on the Gaussian channel. In Section 5.2, we further investigate the recursive list-decoding algorithms for the parity lattices with $k = 2$ (Barnes-Wall lattices) and $k = n^{\frac{1}{\log \log n}}$.
- In Chapter 6, we study the decoding of famous group codes obtained via either the k-ing construction or as parity-check k-groups. Sections 6.1 and 6.2 focus on the construction and the decoding of the Leech and Nebe lattices as special cases of the k-ing construction. We also consider a sublattice of the Nebe lattice: a parity lattice with $k = 3$. Section 6.3 addresses the case of codes built with the parity-check construction or with the k-ing construction. Finally, Section 6.4 presents some additional numerical results: A benchmark of the performance of existing schemes is provided and a finite constellation designed.

In the third part of the thesis, we jointly study lattices and neural networks. We characterize the complexity of the lattice decoding problem from a neural network perspective. On the one hand, this problem is shown to be equivalent to computing a continuous piecewise linear (CPWL) function restricted to the fundamental parallelotope of the lattice. On the other hand, it is known that any function computed by a rectified linear unit (ReLU) feed-forward neural network is CPWL. As a result, we count the number of affine pieces in the CPWL decoding function to characterize the complexity of the decoding problem. We investigate strategies to efficiently compute this CPWL function. Besides, this CPWL function is also used to underline the advantage of depth over width in neural network architectures. This part of the thesis is composed of three chapters.

- In Chapter 7, we first show in Section 7.2 how the lattice decoding problem can be restricted to the fundamental parallelotope $\mathcal{P}(\mathcal{B})$. This new lattice decoding problem in $\mathcal{P}(\mathcal{B})$ induces a new type of lattice reduction basis. This category of basis, called Voronoi-reduced basis, is presented in Section 7.3. In Section 7.4, we introduce the decision boundary to decode componentwise. The discrimination with respect to this boundary can be implemented via the hyperplane logical decoder (HLD). It is also proved that, under some assumptions, this boundary is a CPWL function with an exponential number of pieces. Finally, we show in Section 7.5 that for some famous dense lattices this function can be computed at a reduced complexity via folding with deep neural networks. We also argue that the number of pieces to consider for quasi-optimal decoding is reduced with multiple-input multiple-output (MIMO) lattices on the Gaussian channel, which makes the problem easier.
- We show how lattice theory can be used to establish new results in the field of deep learning in Chapter 8. We prove a result which highlights the advantage of depth over width in neural network architectures.
- The last chapter reports some simulation results where deep learning is used to decode/detect. We first describe the training paradigm in Section 9.1. Then, we propose an original neural network for the problem of MIMO detection in Section 9.2: The architecture is based on the projected gradient descent, a multi-plateau sigmoid function is considered, and we propose a twin-network structure. Simulation results are reported. In the last section, we describe the decoding performance of standard feed-forward neural networks where the point to decode is aliased in $\mathcal{P}(\mathcal{B})$.

1.3 Main contributions

Chapter 2

- We argue that capacity approaching schemes are based on the "combining paradigm on a tree". This tree represents component codes, unlike the tree of codes studied before the advent of modern coding theory which represents codewords. This enables to avoid the decoding computational cutoff rate. We show how this paradigm can be adapted to moderate dimensions, a regime that has not been studied much in the literature. This naturally leads to a new class of codes presented in the second part of the thesis: the recursive parity-check k-groups (see in particular Chapter 5).

Chapter 4

- We establish a new abstract framework for group codes: It is composed of $\Gamma(V, \alpha, \beta, k)$, obtained via the k-ing construction, and its subgroup $\Gamma(V, \beta, k)_{\mathcal{P}}$ constructed as single parity-check k-group. See Section 4.2.
- A new decoding paradigm to decode any group code $\Gamma(V, \beta, k)_{\mathcal{P}}$ is summarized within Algorithm 4.1. Algorithms 4.2 and 4.3 are extensions of this first algorithm to decode $\Gamma(V, \alpha, \beta, k)$. A new technique, called the splitting strategy, is then introduced to reduce the complexity of these algorithms in the scope of list decoding. See Section 4.3.

Chapter 5

- We introduce new lattices reffered to as parity lattices. These lattices are recursively built as $\Gamma(V, \beta, k)_{\mathcal{P}}$. Recursive versions of the algorithms of Section 4.3 are presented to decode these new lattices. A modified list-decoding algorithm is proposed for the Gaussian channel: Our analysis reveals that it is easier to explore a non-spherical body for quasi-optimal decoding than the usual sphere considered for regular list decoding[2]. Analytic expressions to assess the performance are provided, along with examples. These examples demonstrate the potential of the modified list-decoding algorithm to approach the Poltyrev limit. The formulas also enable to establish design guidelines. See Section 5.1.
- The complexity of the recursive list decoder for the parity lattices is investigated for the cases $k = 2$, which includes Barnes-Wall (BW) lattices, and $k = n^{\frac{1}{\log \log n}}$. Regarding regular list decoding, the complexity is linear in the list size. For BW lattices, this allows to achieve a lower complexity than the one of existing regular list decoders. Moreover, while the coding gain is reduced for $k = n^{\frac{1}{\log \log n}}$, compared to $k = 2$, so is the list size and thus the list-decoding complexity. See Figure 5.4. Finally, the modified list-decoding algorithm yields quasi-optimal decoding performance for BW lattices over the Gaussian channel, at a reasonable complexity, up to dimension 128. See Section 5.2.

Chapter 6

- The Leech lattice Λ_{24} and the Nebe lattice $\mathscr{N}_{72}$ are built as instances of $\Gamma(V, \alpha, \beta, 3)$. New decoding algorithms for Λ_{24} and $\mathscr{N}_{72}$ are developed as an application of our decoding paradigm. See Section 6.2.
- We show that the parity lattice $L_{3 \cdot 24} = \Gamma(V, \beta, 3)_{\mathcal{P}}$, as sublattice of $\mathscr{N}_{72}$, has performance only 0.2 dB apart from $\mathscr{N}_{72}$ on the Gaussian channel. Moreover, the decoding complexity of $L_{3 \cdot 24}$ is significantly reduced. See Section 6.2.5. This is a remarkable result in finding a complexity-performance trade-off.
- Reed-Muller codes are recursively obtained from $\Gamma(V, \beta, 2)_{\mathcal{P}}$ and some famous short-length self-dual binary codes built as $\Gamma(V, \alpha, \beta, k)$. We establish a connection between idempotents and the component codes of the k-ing construction. See Section 6.3.

Chapter 7

- We first state a new closest vector problem (CVP), where the point to decode is restricted to the fundamental parallelotope $\mathcal{P}(\mathcal{B})$. See Problem 7.1. This problem naturally induces a new type of lattice basis reduction, called Voronoi-reduced basis. See Definition 7.1. In Section 7.3, we prove that some famous dense lattices admit a Voronoi-reduced basis. We also show that it is easy to get quasi-Voronoi-reduced bases for random MIMO lattices up to dimension $n = 12$.

[2] Even if the volume of the non-spherical body is larger than the one of the sphere required for quasi-optimal decoding.

- A new paradigm to address the CVP problem in $\mathcal{P}(\mathcal{B})$ is presented. We introduce the notion of decision boundary in order to decode componentwise in $\mathcal{P}(\mathcal{B})$. This decision boundary partition $\mathcal{P}(\mathcal{B})$ into two regions. The discrimination of a point with respect to this boundary enables to decode. The hyperplane logical decoder (HLD, see Algorithm 7.2) is a brute-force algorithm which computes the position of a point with respect to this decision boundary. The HLD can be viewed as a shallow neural network.
- In Section 7.4.5, we show that the number of affine pieces in the decision boundary grows exponentially with the dimension for some basic lattices such as A_n, D_n, and E_n (see e.g. Theorem 7.5). This induces both a HLD of exponential complexity and a shallow (one hidden layer) neural network of exponential size (Theorem 7.6).
- In Section 7.5.1, in order to compute the decision boundary function in polynomial time, the folding strategy is introduced (see e.g. Theorem 7.9). The folding strategy can be naturally implemented by a deep neural network.
- Regarding less structured lattices, such as the ones considered in the scope of MIMO, we argue that the decoding problem on the Gaussian channel, to be addressed by a neural network, is easier compared to decoding dense lattices (in low to moderate dimensions). Namely, only a small fraction of the total number of pieces in the decision boundary function should be considered for quasi-optimal decoding. As a result, smaller shallow neural networks can be considered, which makes the training easier and the decoding complexity reasonable.

Chapter 8

- We prove that there exists a function f that can be computed by a deep neural network of polynomial size where any function g computed by a shallow neural network of polynomial size induces an approximation error growing exponentially with the dimension. See Theorem 8.1. This highlights the advantage of depth over width in neural network architectures.

Chapter 9

- Two simulation settings for decoding/detection via deep learning are considered. With both neural networks, the complexity remains reasonable with MIMO lattices but not with dense lattices.

1.4 Publications

Peer-reviewed Journal

1. Vincent Corlay, Joseph J. Boutros, Philippe Ciblat, and Loïc Brunel, A new framework for building and decoding group codes, *In preparation for IEEE Transactions on Information Theory.*
   The paper corresponds to Chapters 4, 5, and 6. It will be submitted after the return of the rapporteurs' comments.
2. —, Neural network approaches to point lattice decoding, *In preparation for IEEE Transactions on Information Theory.*
   The paper corresponds to Chaper 7. It will be submitted after the return of the rapporteurs' comments.

International conferences

1. Vincent Corlay, Joseph J. Boutros, Philippe Ciblat, and Loïc Brunel, "Multilevel MIMO Detection with Deep Learning", *Asilomar Conference on Signals, Systems, and Computer*, Pacific Grove (USA), October 2018.
2. —, "Neural Lattice Decoders", *IEEE Global Conference on Signal and Image Processing (GlobalSIP)*, Anaheim (USA), November 2018.
3. —, "A Turyn-based neural Leech decoder," *2019 International Workshop on Coding and Cryptography (WCC)*, Saint-Jacut-de-la-Mer (France), March 2019.
4. —,"On the CVP for the root lattices via folding with deep ReLU neural networks," *IEEE International Symposium on Information Theory (ISIT)*, Paris (France), July 2019.

5. —,“On the decoding of Barnes-Wall lattices,” *IEEE International Symposium on Information Theory (ISIT)*, Los Angeles (USA), June 2020.

Other

1. Vincent Corlay, Joseph J. Boutros, Philippe Ciblat, and Loic Brunel, “A lattice-based approach to the expressivity of deep ReLU neural networks,” February 2019.

Chapter 2

The channel coding problem

2.1 A general presentation

2.1.1 Problem statement

Picture yourself standing in a noisy hall full of people with your friend on the other side of the hall. You want to communicate an information to your friend. Obviously, you should speak louder than if both of you were alone in a small room. In other words, you should increase the power of your voice because of the ambient noise in the hall. If your friend is not too far, perhaps he could hear you but not fully understand what you would say. In this case, instead of screaming you could try to repeat your message several times.

This example is an instance of the communication problem encountered in countless situations. The most famous model of this problem is called the discrete-time additive white Gaussian noise (AWGN) channel. It is expressed by the following equation

$$y = f(z) + w, \tag{2.1}$$

where:

- The vector $z = (z_1, z_2, ..., z_k)$, $z_i \in \{0, 1\}$, is the message. z is assumed to be uniformly distributed over $\{0, 1\}^k$.
- $f(z) \in \mathbb{R}^n$ is the channel input (the transmitted signal). The function $f(\cdot)$ is an encoding function.
- $y \in \mathbb{R}^n$ is the received message.
- $w = (w_1, w_2, ...w_n)$ is a Gaussian noise vector with an i.i.d components $\mathcal{N}(0, \sigma^2)$.

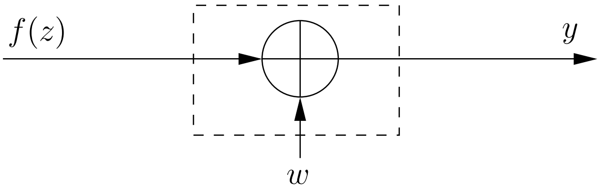


Figure 2.1: Discrete-time AWGN channel. The dashed square represents the communication channel.

The model is summarized on Figure 2.1. The three fundamental quantities of the communication problem are:

- The reliability of the communication: It is measured by the probability that the receiver, given y, is not able to recover the exact message z.
- The information rate (or speed) of communication: It refers to the average amount of information transmitted per use of the channel. Let us call one use of the channel a dimension. In our model, the message is sent over n dimensions. The information rate is denoted by R and is measured in bits per dimension (bits/dim).

- The quality of the communication channel: It is the average energy of the transmitted signal compared to the one of the noise. It is measured by the signal-to-noise ratio (SNR).

In our example of two people in a hall, two solutions are proposed to improve the reliability of the communications: Improve the SNR by increasing the power of your voice compared to the one of the noise or decrease the information rate by repeating several times the message. This latter option is achieved by coding the message: I.e. $f(z) \neq z$. For each element in $\{0,1\}^k$ we associate an element of a set $\mathcal{C}$ where $|\mathcal{C}| = |\{0,1\}^k| = 2^k$. This operation is called channel coding. The set $\mathcal{C}$ is called a code and its elements are called codewords. For instance, repeating the message means that a repetition code is used and the codeword $x \in \mathcal{C}$ corresponding to z is obtained as $x = f(z) = (z, z)$ and $n = 2k$.

The (optimal) decoding operation, for the receiver, consists in finding the most likely message z given the received y. This is called maximum a posteriori decoding. Since the message is chosen uniformly at random in $\{0,1\}^k$, it is equivalent to finding the codeword $x \in \mathcal{C}$ maximizing the probability of receiving y given z, called maximum likelihood decoding (MLD). Given our AWGN channel model, it simply consists in finding the closest codeword from y, i.e. performing minimum Euclidean distance decoding.

In 1948, Claude Shannon established the maximum information rate at which a reliable communication is possible with a given SNR [Sha48]. Here, reliable means that the error probability is vanishing for $n \to +\infty$. This quantity is called the capacity, denoted by C. For the discrete-time AWGN channel the capacity is

$$C = \frac{1}{2}\log_2(1 + SNR). \tag{2.2}$$

The SNR associated to the capacity C is named Shannon limit. Shannon proved his result with the random coding argument: I.e. he considered an ensemble[1] of random codes and showed that the average error probability of codes in the ensemble is vanishing when $n \to +\infty$ and $R \leq C$.

However, even if we admit that a code randomly chosen in this ensemble is also good, this result does not answer the main practical question: How do we encode and decode such a code? This question has been studied by generations of researchers for seven decades.

When looking for a good code for channel coding, the goal is twofold:

- The codewords of $\mathcal{C}$ should be packed in the space with some desirable properties: The probability that y is not in the decoding region of the transmitted codeword should be low. This is the coding problem.

- Given the received message y it should not be too hard to recover the transmitted message z. This is the decoding problem.

In fact, as noticed as early as 1955 by Elias [Eli55] and reported in [CF07], "almost all randomly chosen codes perform essentially as well as the best codes; that is, most codes are good codes". The channel coding problem is therefore essentially a channel decoding problem: It consists in "finding classes of codes that have some special structure, so as to simplify implementation, without sacrificing average performance over the class" [CF07]. For instance, the special class of codes with a group structure has the same average performance as the class of random codes, while being simpler to manipulate [Gal68].

We shall see in the sequel that depending on the information rate (and thus the SNR), different categories of codes are considered.

2.1.2 The power-limited and the bandwidth-limited regimes

Let $\mathcal{Z} = \{0, 1, ..., \mathcal{Q} - 1\}$. The discrete-input AWGN channel is a special case of the discrete-time AWGN channel model where the input of the channel is restricted to this equiprobable $\mathcal{Q}$-ary alphabet. The maximum information rate achievable with this model is given by the mutual information curves on Figure 2.2 [FU98]. Consequently, the encoding function $f : \{0,1\}^k \to \mathcal{Z}^n$ can be considered for high information rates. The function f is not surjective if the information rate is less than $\log_2 \mathcal{Q}$: Figure 2.2 shows that $\mathcal{C}$ should contain only a fraction of $R/\log_2 \mathcal{Q}$ elements of $\mathcal{Z}^n$ such that $|\mathcal{C}| = 2^{nR}$. While this approach can solve the coding problem on the discrete-time AWGN channel for low rates, there is still a gap of 1.53 dB for high rates. Note also that $\mathcal{Z}$ can be binary when $R < 1$, which makes many practical operations easier.

[1] I.e., a set

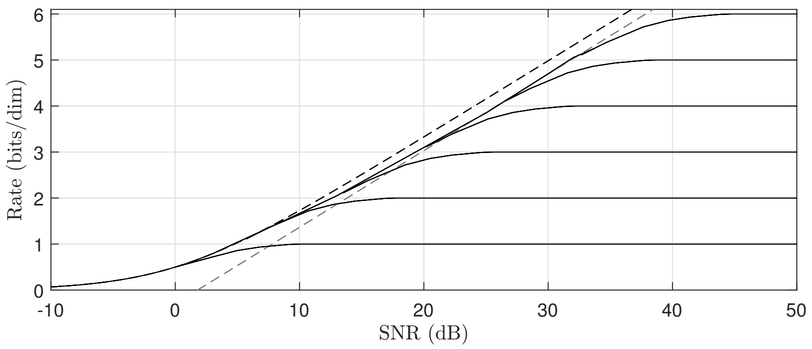


Figure 2.2: Maximum information rates achievable with low error probability [FU98]. The back dashed line represent the capacity of the discrete-time AWGN given by (2.2). The black lines represent the maximum information rate achievable on the discrete-input channel with different number of equiprobable levels (1, 2, 3, 4, 5, and 6). The grey dashed line is the curve $1/2 \log_2 SNR$ shifted by 1.53 dB to the right.

As a result, there are two distinct regimes for the channel coding problem: the power-limited and the bandwidth-limited regimes. In the former, the rate is low, typically $R < 1$ and binary codes are usually considered. Moreover, when $R << 1$ the SNR is low and the capacity C behaves like a linear function in the SNR. For the bandwidth-limited regime, $R > 1$, either multilevel schemes based on binary codes, such as bit-interleaved coded modulation, or lattice-based constellations[2] are generally used. Note that a code is often called a constellation in this second regime.

The lattice-based approach is the following. Consider the set $\mathscr{Z} = \mathbb{Z}$ (the integers). We remove points of this infinite constellation[3] $\mathscr{Z}^n$ to get a sub-constellation Λ with $\text{vol}(\Lambda) > 1$ points per volume unit. If we select the elements of Λ inside a hypercube of size $\mathscr{Q}^n$, we get a constellation $\mathcal{C}$ with $|\mathcal{C}| \approx \mathscr{Q}^n / \text{vol}(\Lambda)$ and the information rate is $R \approx \frac{1}{n} \log_2 |\mathcal{C}|$. Consequently, the constellation Λ, provided that it has the good properties for the Gaussian channel, can be used for coding at several information rates. The remaining 1.53 dB are obtained by choosing elements of the infinite constellation within a sphere rather than a cube. Indeed, the asymptotic second order moment of a sphere is 1.53 dB smaller than the one of a cube. This problem, called the shaping problem, can be solved independently of the coding problem when $|\mathcal{C}|$ is large enough[4]. We shall not discuss it further in this chapter.

Despite these differences between low rate and high rate regimes, the coding problem in both cases can be addressed with the same guidelines. As a result, for both regimes, the problem amounts to finding a code $\mathcal{C}$, finite or infinite, where the codewords are appropriately packed and where the decoding operation can be efficiently performed. We shall present these guidelines in the next sections.

For more information on the channel model considered, the reader can consult [PS08].

2.2 From the 50's to the 80's

2.2.1 The union bound and the minimum distance

In the 50's, the main goal was to find codes with a large minimum distance: The value

$$d(\mathcal{C}) = \min_{x \neq x' \in \mathcal{C}} d(x, x'), \tag{2.3}$$

should be the largest possible. Here $d(\cdot)$ denotes the Euclidean distance. This goal was motivated by an estimate of the MLD performance of a given code $\mathcal{C}$ obtained via the union bound. This estimate is based on the pairwise error probability between two codewords, defined as the probability that the received message y is closer to x' than to the transmitted x, $x' \neq x$, $x, x' \in \mathcal{C}$. We denote this probability

[2] Efficient lattices can also be obtained as multilevel schemes.

[3] While preserving the group structure of $\mathbb{Z}^n$.

[4] Note that on the receiver side, y is in general decoded in the infinite constellation, not restricted to $\mathcal{C}$. Namely, the decoder may find a closest element in the infinite constellation which is not a codeword in $\mathcal{C}$. Such a decoder is called a lattice decoder, to be opposed with a nearest-codeword decoder. While it was known since [deB75] that lattice decoding achieves $1/2 \log_2(SNR)$, it was not obvious if the approximation made by the lattice decoder is negligible such that one can get $1/2 \log_2(1 + SNR)$. It was finally proved in [EZ04] and [dZB18] that a lattice decoder can be optimal.

by $P(x'|x)$. Let σ^2 be the variance of the Gaussian noise. It is easily shown that, with a channel as (2.1), $P(x'|x) = Q(d(x,x')/2\sigma)$. The union bound yields

$$P_e(opt) \leq \frac{1}{|\mathcal{C}|} \sum_{x} \sum_{x' \neq x} Q\left(\frac{d(x,x')}{2\sigma}\right), \tag{2.4}$$

where $P_e(opt)$ denotes the MLD performance of the code. Let τ_i be the average number of points at a distance d_i from a codeword x. Equation (2.4) becomes

$$P_e(opt) \leq \sum_i \tau_i Q\left(\frac{d_i}{2\sigma}\right). \tag{2.5}$$

Let $\tau = \tau_1$ be the average number of codewords at the minimum distance of $\mathcal{C}$. Given that the terms τ_i are not too large, the error probability at large SNR can be estimated with the first term of the sum in (2.5). We get

$$P_e(opt) \approx \tau Q\left(\frac{d(\mathcal{C})}{2\sigma}\right), \tag{2.6}$$

which highlights the interest of finding codes with a large minimum distance.

As a result, the first approach to build codes was algebraic. The idea was to set strong deterministic constraints on the structure of the code to have a minimum distance as large as possible. Some of the most famous outcomes of this line of research are the $(8,4,4)$ Hamming code [Ham50], the $(24,12,8)$ Golay code [Gol49], the Leech lattice in dimension 24 [Lee67], the families of Reed-Muller codes [Ree54] [Mul54], BCH codes [BRC60] [Hoc59], and Barnes-Wall lattices [BW59] (in low dimensions), Reed-Solomon codes [RS60], and more recently the Nebe lattice in dimension 72 [Neb12]. Computer searches to find convolutional codes (presented in the following subsection) with a good minimum distance were also performed.

2.2.2 Hard-decision decoding, Viterbi algorithm, sequential decoding, and the cutoff rate

On the decoding side, a solution often used in practice was to quantize the received message in the discrete space of the code. Efficient algorithms to solve equations in the space of the code can then be used. A famous instance of this category of algorithms is the Berlekamp-Massey algorithm [Ber68] [Mas69]. This paradigm is called hard-decision decoding. While it makes the decoding easier, it also costs 2 to 3 dB compared to the MLD performance of the code on the AWGN channel. Note that a significant improvement in this line of work was made recently with the Guruswami-Sudan list decoder of Reed-Solomon codes [Sud97] [GS99].

When y is not quantized at the input of the decoder, the decoding algorithm is called a soft-decoder. As explained in the previous section, the optimal (soft) decoder should find the closest codeword to y. The brute-force approach is to compare the distance of y with all codewords in $\mathcal{C}$. The complexity is $\approx |\mathcal{C}|$ and intractable when the dimension increases. This observation motivated the invention of tree and trellis-based codes, where one codeword is a path in the tree/trellis. This enables to find the closest codeword to y at lower complexity. For instance, convolutional codes, introduced by Elias [Eli55], are finite-state Markov processes where the code is viewed as a time series. At each given time, the code is in a given state which depends only on a finite number of previous scalar values z_{i-j}, $0 < j \leq \nu$ of the message z. The quantity $\nu + 1$ is called the constraint length. Since the state of the code at time i depends only on the previous state of the code and z_i, these codes are efficiently represented by a trellis. They admit an efficient optimal decoding algorithm, the Viterbi algorithm [Vit67], the complexity of which depends on ν. Similarly, a sequential decoder [Woz57] operates on the trellis/tree of the code by looking for the most likely path/codeword. This decoder is dynamic as even if it has established one candidate path, it is allowed to change its hypothesis.

One way to understand these efficient decoders is the following. They look for the neighbor codewords of x in $\mathcal{C}$ that are close to y, similarly to a sphere decoder[5] which looks for the closest lattice point in a sphere around y. Indeed, as explained by Arikan in [Ari16], sequential decoders behave like the theoretical guessing decoder. The principles of this guessing decoder are the following. A candidate for

[5] See Chapter 3 for more information on the sphere decoder.

the transmitted x is submitted by the decoder and a "genie" tells the decoder whether it is correct or not. If not, a second candidate is submitted by the decoder and the process stops when x is found. The complexity is obtained as the average number of candidates that the decoder has to submit to the genie before finding the transmitted x. The best order for this genie-aided decoder is the MLD order: The closest codeword x_1 to y is first submitted, then the second closest codeword x_2 is submitted, and so on. Consequently, a codeword x' is submitted to the genie if a pairwise error occurs between x and x'. Let $E_y[\cdot]$ denote the expectation with respect to the probability distribution of y. The average complexity with respect to y, denoted by $E_y[\mathfrak{C}]$, is given by the average number of pairwise errors:

$$E_y[\mathfrak{C}] = E_x[\sum_{x' \neq x} P(x'|x)]. \tag{2.7}$$

Consider a random ensemble similar to the one used by Shannon to prove the capacity (see the text below (2.2)). The complexity of the decoder can be averaged over this random ensemble. It can be shown that the average pairwise error probability is (see e.g. [Ari16])

$$E_{\mathcal{C}}[P(x'|x)] = \sum_{\mathcal{C}} P(\mathcal{C}) P(x'|x) \leq 2^{-nR_0}, \tag{2.8}$$

where $P(\mathcal{C})$ denotes the probability of choosing the code $\mathcal{C}$ in the ensemble and the quantity R_0 is discussed below. The average complexity is thus given by

$$\begin{aligned} E_{\mathcal{C}} E_y[\mathfrak{C}] = \frac{1}{\mathcal{C}} \sum_{x} \sum_{x' \neq x} E_{\mathcal{C}}[P(x'|x)] \leq (2^{nR-1}) 2^{-nR_0}, \\ \leq 2^{n(R-R_0)}. \end{aligned} \tag{2.9}$$

Only the upper bound is shown here but the converse also holds [Ari16]. As a result, R_0 acts as a decoding computational cutoff rate: For $R > R_0$, the average decoding complexity is exponential in n. Note that (2.9) is also the union bound on the average performance of the random ensemble. Hence, we have

$$E_{\mathcal{C}}[P_e(opt)] \leq 2^{-n(R_0-R)}. \tag{2.10}$$

This latter quantity should be compared with the random-coding exponent $E(R)$, which quantifies the speed at which the error probability, given an optimal decoder, vanishes when the information rate is smaller than the capacity [Gal68]. Namely,

$$E_{\mathcal{C}}[P_e(opt)] \leq 2^{-nE(R)}, \tag{2.11}$$

where $E(R)$ is a positive convex function in $R \in [0, C]$ and $E(R) \to 0$ when $R \to C$. One has $R_0 < C$, which highlights that decoders behaving as the guessing decoder are not efficient in the range $R_0 < R \leq C$.

The fundamental limit encountered here is not the code but the decoding paradigm: Looking for the most likely codeword among all possible codewords. This phenomenon has a nice geometric interpretation. Consider the following lemma, which shows that as $n \to \infty$, the norm of the Gaussian noise concentrates around its mean. In other words, only noise vectors whose norm is equal to the mean are likely. We name these noise vectors typical noise.

Lemma 2.1 (Typical noise)**.** *Let σ^2 be the variance of the noise on the unconstrained AWGN channel. As $n \to \infty$, for fixed $\epsilon > 0$ the probability that $(1-\epsilon)n\sigma^2 < ||w||^2 < (1+\epsilon)n\sigma^2$ goes to 1.*

As a result, good codes should have spherical decision regions. This leads to Figure 2.3.

We conclude that a different paradigm should be considered to decode beyond the cutoff rate. This problem was not solved until 1993. In fact, it took so long to solve this problem that, as explained in [FU98], many communications engineers believed that the cutoff rate R_0 was the "practical capacity".

2.3 From the 90's until today

2.3.1 A brief history of "modern" coding theory

The breakthrough happened in 1993 with turbo codes [BGT93]. The discovery was so unexpected, and made by researchers that were neither information theorists nor channel coding theorists, that many

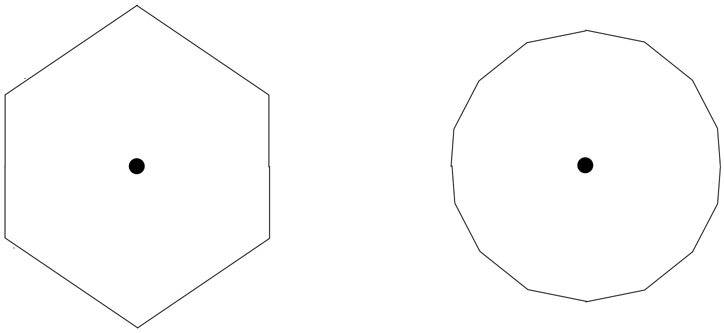


Figure 2.3: Decision region of two codes. The center of each decision region represents the transmitted codeword. In low dimensions the number of facets of decision regions is relatively low, as illustrated on the left, and the probability that the transmitted point leaves the decision region is well approximated by the union bound: The pairwise error events are approximately independent. However, good codes in high dimensions have spherical decision regions, as illustrated on the right, resulting from a high number of neighboring codewords. The number of facets is high and the pairwise error events are not independent anymore. From a decoding perspective, trying all neighboring codewords close y becomes intractable.

believed the existence of a factor 2 mistake, accounting for the 3 dB gain achieved by turbo codes over all other known practical codes. It was soon recognized that turbo codes do approach capacity. According to [Bou20], this breakthrough was the result of using jointly the four following principles, which had never been done before. Note that the inventors probably did not know the importance of each principle as implied by their motivations.

- Recursive convolutional codes. They were commonly used at this period as high rate systematic codes within trellis coded modulation (invented by Ungerboeck).
- Parallel concatenation instead of serial concatenation. Claude Berrou was an electrical engineer and his motivation was to use only one clock rather than two for hardware implementation. Indeed, serial concatenation requires a different clock for each code.
- The interleaver. It was added such that the two convolutional codes have different inputs.
- Soft-output decoding, as recommended by Gérard Battail.

In other words, they managed to make the code look random and powerful enough, similarly to the one used by Shannon for its theorem, while maintaining some structure suited to iterative decoding. This invention was quickly followed by the re-discovery of low density parity-check (LDPC) codes [MN96], invented by Gallager in his PhD thesis more than forty years earlier [Gal62]. The recent computer simulations have shown that LDPC codes approach the capacity within tenth of dB (when the information rate is smaller than 1).

Nevertheless, until 2008 there were only a few theoretical results on (practical) capacity achieving codes. There only existed a proof that LDPC codes, under iterative decoding, achieve the capacity of the binary erasure channel.

In 2008, polar codes were invented by Arikan [Ari09]. These codes provably achieve the capacity of any binary input memoryless channel and have a deterministic construction. The beauty of these codes is that the capacity proof explains how they should be constructed and decoded. Unfortunately, polar codes with successive-cancellation decoding (the low-complexity decoder used in the proof) are efficient only if the code length is very large. Indeed, the original idea of Arikan was initially to create several virtual channels to "boost" the cutoff rate such that it is easier to code each individual channel with an outer code. However, for large code length the channels become so good (or so bad) that the outer codes are not needed. This is not the case for smaller code length.

Finally, at the same period, it was proved in [KRU13] that spatial coupling, i.e. combining several LDPC codes with extra edges as for convolutional LDPC codes [FZ99], enables to achieve the capacity of any binary input memoryless channels under iterative decoding.

How good are these codes with respect to their minimum distance, to compare with the codes of Section 2.2.1? Polar codes and turbo codes have a relatively low minimum distance, increasing as $\sqrt{n}$ and $\log n$, respectively (e.g. with turbo codes an error-floor is often encountered at low to medium error probability). While it is possible to construct LDPC codes with a large minimum distance, increasing linearly with n, their iterative decoding threshold is in this case far from the capacity. Indeed, as discussed in the next section, the local neighborhood of a variable node should be a tree. Therefore, the Tanner graph of the code should not have too many cycles and it is known that cycle-free codes have a poor minimum distance [ETV99]. Hence, an efficient LDPC code should not have a too high minimum distance to avoid local cycles fatal to iterative decoding, but not too low to remain capacity approaching. With spatial-coupling, the minimum distance can be increased, but the required number of decoding iterations also significantly increases. As a result, to the best of our knowledge all known efficient capacity approaching schemes are not asymptotically good in terms of minimum distance.

The study of these capacity approaching/achieving codes is often named modern coding theory, as the name of the reference book by Urbanke and Richardson [UR08], even though it is now a thirty years old field. Many researchers refer to these codes as codes on graphs... even though many older codes were also modeled via graphs.

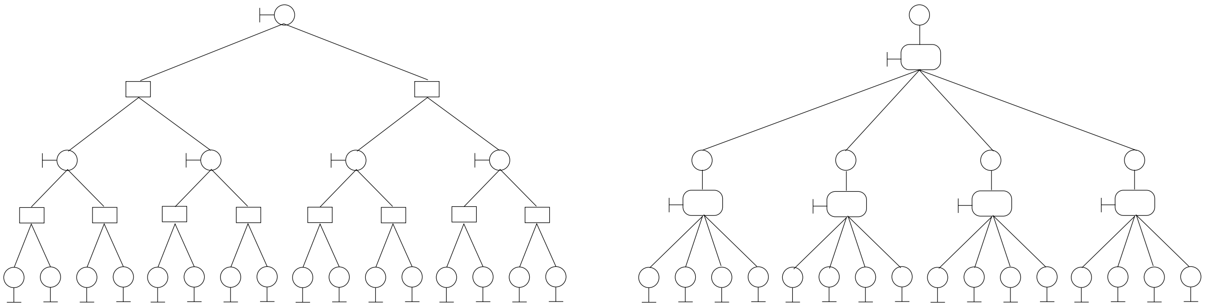
2.3.2 The combining paradigm on a tree

Whereas these coding techniques (turbo codes, LDPC codes, polar codes) may seem different at a first glance, we argue that they all involve the same underlying idea: Small component codes (e.g. single parity-check codes) should be stacked in a tree such that the reliability of the symbol estimates produced at each level of the tree increases as one goes up in the tree. The two key ingredients are:

- A tree.
- Combining operations at each level of the tree.

We call this idea the combining paradigm on a tree. Unlike the tree codes of the Section 2.2.1, the tree does not represent codewords but component codes.

Regarding LDPC codes, unfolding the Tanner graph around a variable node yields a local tree. On Figure 3.6a, this tree is shown with the usual variable nodes/check nodes representation. Figure 2.4b depicts a similar tree where the combining paradigm is emphasized: First, n channel observations are grouped in small sets of k elements. The k elements within one set are combined to produce an estimate of a symbol whose reliability is higher than the one of each individual element in the set. Then these n/k estimates are again grouped in sets of k elements and again combined to produce more reliable estimates. The process is repeated until the reliability is sufficiently high. Note that the estimates produced at each level are for different symbols than the child-node symbols.

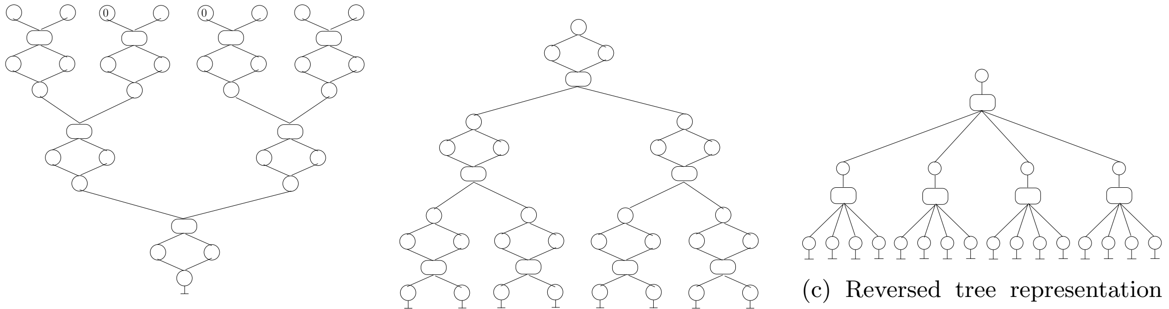


(a) Representation of the local neighborhood of a variable node in a LDPC code. The half edges on the variable nodes represent a channel observation. The small spheres represent variable nodes and the squares check nodes. Each check node and variable node are of degree 3.

(b) Alternative representation of the local neighborhood of a variable node in a LDPC code. At each level of the tree, four symbol estimates and one channel observation are combined to produce a more reliable estimate of a symbol. The rounded squares denote a combining operation.

Figure 2.4: Tree representation of the local neighborhood of a symbol for a LDPC code.

Polar codes, obtained via the Plotkin $(u, u+v)$ construction, can be represented with a similar (reversed) tree. This is illustrated on Figure 2.5a. Polar codes rely on a similar combining paradigm. A symbol y is observed from the channel. It is split into two symbols $y = (y_1, y_2)$. This two symbols are combined to produce two new symbols with different reliabilities. Unlike LDPC codes, the tree is explored in a depth-first manner. The combining operations are performed both in the forward and backward direction.



(a) Tree representation of a Polar code. Each symbol is split into two symbols. The tree is explored in a depth-first manner. The combining operations are performed both in the forward and backward directions. The leafs where the reliability is null are frozen.

(b) Reversed tree representation of a code obtained via the Plotkin $(u, u + v)$ construction. After a combining operation (rounded squares) the two symbols obtained are merged into one higher dimensional symbol.

(c) Reversed tree representation of Figure 2.5a, but where the single parity checks are performed on four symbols and the variable nodes above the rounded squares represent a symbol obtained from the four child node variables.

Figure 2.5: How the polar code tree can be transformed in a LDPC code-like tree.

2.3.3 From LDPC codes and polar codes to the single parity-check k-groups

What is specific to asymptotic dimensions for these codes among the two key ingredients of the combining paradigm on a tree? For LDPC codes, the local tree structure around each symbol of the code is obtained by choosing randomly a large enough sparse parity-check matrix. Hence, the deep-enough tree structure is obtained with large dimensions. However, the combining operations have nothing specific to large dimensions. On the contrary, the tree for polar codes is obtained deterministically. The large dimension is necessary for the combining operations, which become very simple and efficient thanks to the polarization phenomenon. Indeed, only successive-cancellation operations are involved.

This combining paradigm on a tree has been extensively studied for large dimensions (typically $n \geq 1000$) but not much in low and moderate dimensions. LDPC codes are inefficient when n is too small. Some studies consider polar codes in moderate dimensions. They focus on "fixing" the combining principles: Indeed, "polar" codes do not totally polarize if n is not large and the successive-cancellation combining operation is not efficient. As a result, low dimensional polar codes should be concatenated with an outer codes (e.g. a cyclic redundancy check) and list decoding should be considered [TV15]. Note that these principles can also be used with Reed-Muller codes as explained by Dumer in [DS06]. Indeed, they admit the same decoding tree[6] as polar codes since they are also constructed via the recursive $(u, u + v)$ construction.

For moderate dimensions, instead of "fixing" the combining operations of polar codes, one could consider the combining operations of LDPC codes based on single parity checks. Nevertheless, since the LDPC code tree can not be obtained, the deterministic tree of polar codes should be used. If we reverse the tree of Figure 2.5a, we get a tree similar to the one of LDPC codes shown on Figure 2.4b. This is illustrated on Figure 2.5b. Unlike LDPC codes, two symbols are concatenated at each level of the tree and the parity checks involve higher dimensional symbols. I.e. the parity checks are performed on $\mathcal{Q}$-ary alphabets rather than binary alphabets, where $\mathcal{Q}$ increases with the depth of the tree. Of course, instead of considering parity checks on two symbols, we could also consider parity checks on k symbols, e.g. $k = 4$ to match Figure 2.4b. This yields the tree on Figure 2.5c. This latter tree represents the recursive version of a new class of codes, called single parity-check k-groups. These codes are studied in the second part of the thesis. Since the paradigm is different than sequential decoding, the cutoff rate is unlikely to appear as a fundamental complexity barrier. We shall see that these codes are very efficient in moderate dimensions.

2.4 Is the channel coding problem really solved?

Equation (2.2) holds only when the code length $n \to \infty$. The maximum rate achievable with low error probability, say $C'(n, SNR)$, is smaller if the code length is not asymptotically large. Nevertheless,

[6] In fact, this decoding method was introduced for Reed-Muller codes before the invention of polar codes.

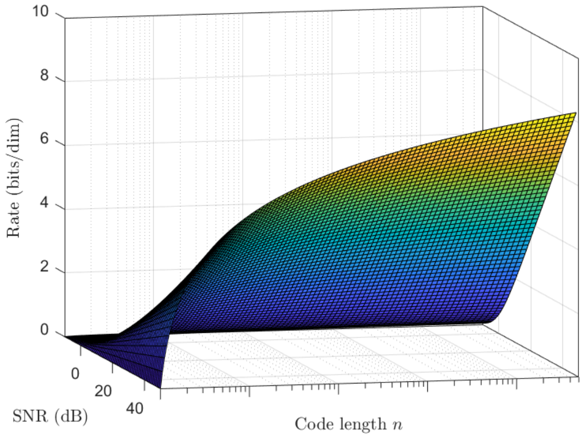


Figure 2.6: Shape of the maximum information rates achievable with low error probability on the discrete-time AWGN channel as a function of the SNR and the code length n.

$C'(n, SNR)$ increases fastly towards the capacity C when n is not too large [PPV10]. It means that relatively large rates, e.g. above the cutoff rate, can be achieved with small block lengths. The shape of the the maximum information rates achievable $C'(n, SNR)$ is plotted on Figure 2.6. Solving the channel coding problem can be understood as having efficient schemes for any code length n and SNR.

As discussed in the previous section, capacity achieving/approaching schemes based on iterative decoding are only efficient for large n. Moreover, very little schemes being both practical and quasi-optimal have been proposed for moderate dimensions since the advent of modern coding theory.

Note however that the new 5G communications standard targets some ultra-reliable low-latency communications (often named URLLC) applications. In this scope, there was recently a renewed interest in short-length codes. For instance [WABM16] investigates the optimal performance of known small-length binary codes with the ordered statistics decoder. The decoder considered is however too complex to be practical. The schemes in [DS06] and [TV15] (discussed in the previous section) are efficient in moderate dimensions, but they are also for low information rates as they are based on binary codes. Schemes for information rates $R > 1$ in moderate dimensions have not been studied a lot in the literature. Parity lattices, constructed as single parity-check k-groups, enable to fill this gap.

Chapter 3

An overview of lattices

This thesis is about lattices. The first question we should address is therefore: Wh In "common" language, it usually refers to an infinite set of points with a periodic structure in $\mathbb{R}^n$. Such a periodic structure is illustrated on Figure 3.1 for $\mathbb{R}^2$. In this scope, a lattice is also called a (regular) sphere packing: The lattice points are the centers of spheres whose radius is half the smallest distance between any two points of the set.

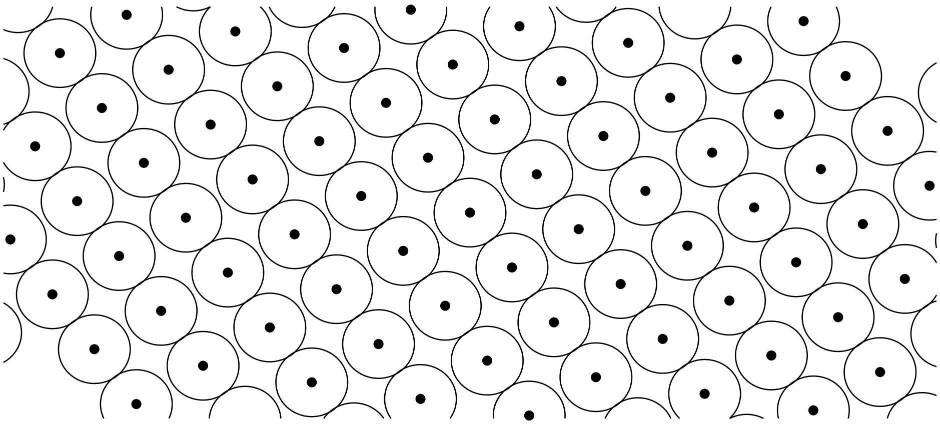


Figure 3.1: A lattice sphere packing in $\mathbb{R}^2$.

Nevertheless, lattices have a long history and have been studied by scientists from many different schools of thoughts. One may encounter several perspectives. Arguably, one can group the scientists who have been working on lattices into the three following fields, with the following distinguished representatives:

- Pure mathematicians such as Conway, Sloane, Nebe, Rains, Quebbemann, Martinet, Elkies, Viazovska, Cohn...
- Digital communications engineers (and information theorists) such as Forney, Ungerboeck, Calderbank, Feder, Zamir...
- Computer scientists, especially cryptographers, such as Goldwasser, Micciancio, Regev, Nguyen, Stehlé...

Which references should consult a newcomer in the field of lattices? The mathematician should choose the SPLAG [CS99], considered by many as the holy bible of lattices. [Mar03] and [Ebe99] may be complementary. The book of Zamir [Zam14] is suited to the information theorist and the papers of Forney (e.g. [For88a] [For88b] [FTS00]) to digital communications engineers. Computer scientists, with interest in computational complexity, should consult the book by Micciancio and Goldwasser [MG02], or the survey by Micciancio and Regev [MR09] for an overview of lattice-based cryptography. Finally, if the reader enjoys history, he should consult [Tho83] to find how the discovery of simple groups is connected to lattices.

Before diving into lattices, we start with an abstract family of codes which we call group codes. We shall see that lattices can be seen as special instances of group codes. The notion of group codes is helpful to link lattices with other class of codes: E.g. binary linear codes are also group codes. Consequently, any lattice-result or algorithm based on the properties inherited from the group structure should be translatable to other group codes.

Given an additive group $\mathcal{G}$, we define a group code $\mathcal{C}$ as a subgroup of $\mathcal{G}^k$, where $\mathcal{G}^k$ is the Cartesian products of G with itself, k times. We may consider that these group codes admit a distance metric which is endowed with the group property and additive over k-tuples, i.e.

$$\begin{aligned} &d(s,s') = d(0, s-s'),\ \ s,s' \in \mathcal{G}, \\ &\text{Given } x=(s_1,...,s_k) \in \mathcal{G}^k,\ \ d(0^k, x) = \sum_i d(0, s_i), \end{aligned} \tag{3.1}$$

where the notation s^k means $(s,...,s,s)$, with $s \in \mathcal{G}$ repeated k times. All groups considered in this thesis are Abelian. $\mathcal{G}$ may be richer than a group and be a ring. In this case, we shall consider that $\mathcal{C}$ is a $\mathcal{G}$-module. This implies that $\mathcal{C}$ is linear as, given $\alpha, \alpha' \in \mathcal{G}$ and $x, x' \in \mathcal{C}$, then $\alpha \cdot x + \alpha' \cdot x' \in \mathcal{C}$. A $\mathcal{G}$-module is free if it has a basis; all elements can be obtained as linear combinations of the basis elements.

3.1 What is a lattice?

3.1.1 A general presentation

Definition 3.1 (Real lattice)**.** *A real lattice Λ of rank-n is a set composed of all integer linear combinations of n linearly independent vectors in $\mathbb{R}^n$:*

$$\Lambda = \{x \in \mathbb{R}^n \ : \ x = \sum_{i=1}^n z_i g_i = zG, z \in \mathbb{Z}^n\}, \tag{3.2}$$

where G is a generator matrix of Λ whose rows are the basis vectors in $\mathcal{B} = \{g_i\}_{i=1}^n$.

A real lattice Λ is an additive rank-n discrete subgroup of $\mathbb{R}^n$: If 0^n is in Λ, $x \in \Lambda$ then $-x \in \Lambda$, and $x_1 + x_2 \in \Lambda$ where $x_1, x_2 \in \Lambda$. See [Des86] for the discrete part. Moreover, since $\mathbb{Z}$ is a ring, a real lattice Λ is a free $\mathbb{Z}$-module. Of course, the rank-n lattice can also be described in $\mathbb{R}^m$, $m \geq n$, by a $n \times m$ generator matrix G. Namely, the lattice lies in an $(m-n)$-dimensional subspace of $\mathbb{R}^m$. In this case, their always exists a (unitary) rotation matrix $Q \in \mathbb{R}^{m \times m}$ such that $G' = GQ$ describes the lattice in $\mathbb{R}^n$. The lattice is said full-rank if $m = n$.

In other words, a real lattice is a group sphere packing in $\mathbb{R}^n$. The periodic structure in Figure 3.1 is a real lattice. Note that real lattices are naturally embedded in the Euclidean vector space $\mathbb{R}^n$. As a result, the lattice points are sometimes called vectors by abuse of language. Moreover, it is now obvious that real lattices are special cases of group codes.

In general, the distance between two points of a real lattice is evaluated as $d(x_1, x_2) = ||x_1 - x_2||$, $x_1, x_2 \in \Lambda$ ($|| \cdot ||$ stands for the Euclidean norm). The norm of an element $x = zG \in \Lambda$ is thus obtained as $d(x,x) = zGG^Tz^T$ and the distance between two elements is $d(x_1, x_2) = (z_1 - z_2)GG^T(z_1 - z_2)^T$. The map $d : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ is a symmetric bilinear form and the map $q : \mathbb{R}^n \rightarrow \mathbb{R}$, $q(x) = d(x,x)$ is a positive definite quadratic form. The matrix $\Gamma = GG^T$ is called the Gram matrix of Λ and it can be used to define the lattice (up to a rotation). Lattices may thus be called positive definite quadratic forms in some contexts.

Two lattices are equivalent $\Lambda' \cong \Lambda$ if their generator matrices, G' and G respectively, are related by $G' = cUGB$, where c is a non zero constant, U a unimodular matrix, and B an orthogonal matrix. If the constant c should be explicit, we write $\Lambda' \cong c\Lambda$. Hence, two Gram matrices Γ and Γ' define equivalent lattices if and only if there exists a unimodular matrix U such that $\Gamma' = c^2 U \Gamma U^T$.

A real lattice is said to be integer if it is a subgroup of $\mathbb{Z}^n$ (or $\mathbb{Z}^n$ itself), i.e. all the coefficients of the basis vectors are integers. It is said to be integral if Γ has integer entries.

Definition 3.2 (Complex lattice)**.** *A complex lattice of rank-n is a set composed of all (complex) integer linear combinations of n linearly independent vectors in $\mathbb{C}^n$.*

Similarly, a complex lattice is a discrete subgroup of $\mathbb{C}^n$ and a J-module, where J is a ring in $\mathbb{C}$: Any $x \in \Lambda$ is obtained as $x = \sum_{i=1}^n z_i g_i$, where the integer $z_i \in J$ and $\{g_i\}_{i=1}^n$ is a basis of Λ. A complex lattice is said to be integer if it is a subgroub of J^n.

One could go on by defining a quaternionic lattice as a rank-n discrete subgroup of $\mathcal{H}^n$ generated by n linearly independent vectors, where $\mathcal{H}$ is the Hurwitz ring of quaternionic integers. These examples highlight that a general definition of a lattice could be the following. Note however that the term lattice refers most of the time to a real lattice in this thesis.

Definition 3.3 (Lattice). *A lattice is a free J-module, where J is a ring, together with a positive definite quadratic form q.*

One must be careful when choosing J: If it is not a principal ideal domain, the J-module may not be free. Nevertheless, some authors (e.g. [CS99, Chapter 2.6]) still consider these objects as lattices (non-principal lattices). This emphasizes that the notion of lattice has several meanings.

Most of the time one is interested in the properties of real lattices and the more abstract lattices are used to simplify the constructions of real lattices. Indeed, given a rank-n lattice in $\mathbb{C}^n$ one can get a generator matrix for the corresponding real lattice in $\mathbb{R}^{2n}$. As an example, let J be either the ring of Gaussian integers $\mathbb{Z}[i]$, or the ring of algebraic integers $\mathbb{Z}[\lambda]$, $\lambda = \frac{1+i\sqrt{7}}{2}$. A real generator matrix is obtained as follows: Map each component $a + ib$ of the complex generator matrix to

$$\begin{bmatrix} a & b \\ -b & a \end{bmatrix} \text{ or } \begin{bmatrix} a & b \\ (a-\sqrt{7}b)/2 & (b+\sqrt{7}a)/2 \end{bmatrix}, \tag{3.3}$$

if J is respectively $\mathbb{Z}[i]$ and $\mathbb{Z}[\lambda]$. For instance, finding a real lattice with a coding gain[1] equal to 8 in dimension 72 was a long–standing open problem. Such a lattice was recently found by Nebe via its complex $\mathbb{Z}[\lambda]$ structure [Neb12].
Additionally, complex J structures of a real lattice enables to easily find rotations $R : \Lambda \rightarrow \Lambda$, as illustrated by the following example. We first establish a notation: Given a complex J-lattice Λ with generator matrix G, we denote by $\theta\Lambda$ the lattice generated by $\theta \cdot G$, $\theta \in J$. We have $\theta\Lambda \subseteq \Lambda$.

Example 3.1 (Gosset lattice E_8). *The Gosset lattice is famous because it yields the densest sphere packing*[2] *in dimension 8. Let $\phi = 1 + i \in \mathbb{Z}[i]$. E_8 can be generated both over $\mathbb{Z}[i]$ and $\mathbb{Z}[\lambda]$. As a result, the lattices ϕE_8 and λE_8 are two sublattices of E_8, equivalent to E_8, scaled by a factor $\sqrt{2}$, and rotated by an angle $\frac{\pi}{4}$ and arctan$\sqrt{7}$, respectively.*

3.1.2 Additional definitions

We define standard lattice parameters in this section.

Definition 3.4 (Fundamental region). *A set $F \subseteq \mathbb{R}^n$ is called a fundamental region of a latice Λ if the union of all "lattice shifts" of this region covers the entire space with no overlap between any two regions.*

- $\mathbb{R}^n = \cup_{x \in \Lambda}(x + F)$,
- $\text{vol}((x_1 + F) \cap (x_2 + F)) = 0$, *for all $x_1, x_2 \in \Lambda$, $x_1 \neq x_2$.*

Definition 3.5 (Fundamental parallelotope). *The set obtained by taking any linear combinations of the basis vectors in $\mathcal{B}$ with coefficients in $[0, 1)$ is called the fundamental parallelotope of Λ:*

$$\mathcal{P}(\mathcal{B}) = \{y \in \mathbb{R}^n : y = \sum_{i=1}^n \alpha_i g_i, \ 0 \leq \alpha_i < 1\}. \tag{3.4}$$

By abuse of notation, we may write $\mathcal{P}(\Lambda)$ to denote one of the parallelotopes of Λ.

Definition 3.6 (Voronoi region). *The Voronoi region of $x \in \Lambda$ is the set of lattice points of $\mathbb{R}^n$ closer to x than any other lattice point:*

$$\mathcal{V}(x) = \{y \in \mathbb{R}^n : \|y - x\| \leq \|y - x'\|, \forall x' \neq x \textit{ where } x, x' \in \Lambda\}. \tag{3.5}$$

[1] See Definition 3.9.

[2] The fact that E_8 is densest lattice sphere packing in dimension 8 is not a new result. However, the general result, i.e. that it is the densest among lattice and non-lattice sphere packings, was only obtained recently by Viazovska [Via17].

A Voronoi facet denotes a subset of the points

$$\{y \in \mathbb{R}^n : \|y - x\| = \|y - x'\|, \forall x' \neq x \text{ where } x, x' \in \Lambda\}, \tag{3.6}$$

which are in a common hyperplane.
The additive group property of the lattice yields $\mathcal{V}(0) + x = \mathcal{V}(x)$. Similarly, the parallelotope of a lattice point x is $\mathcal{P}(\mathcal{B}) + x$. Hence, all Voronoi regions and parallelotopes have the same volume, do not overlap[3], and cover the space. These two regions are fundamental regions of the lattice.

In many situations, it is useful to represent the basis of the lattice in the orthogonal coordinate system of the Gram-Schmidt vectors $\{g_i^*\}_{i=1}^n$: Starting with $g_1^* = g_1$, the Gram-Schmidt vectors are recursively computed from 1 to n as

$$g_i^* = g_i - \sum_{j=1}^{i-1} \underbrace{\frac{\langle g_i, g_j^* \rangle}{||g_j^*||^2}}_{\mu_{i,j}} \cdot g_j^*. \tag{3.7}$$

This yields[4] a lower triangular generator matrix R of Λ:

$$\underbrace{\begin{pmatrix} g_1 \\ g_2 \\ \cdot \\ \cdot \\ g_n \end{pmatrix}}_{G} = \underbrace{\begin{pmatrix} ||g_1^*|| & 0 & 0 & \dots & 0 \\ \mu_{2,1}||g_1^*|| & ||g_2^*|| & 0 & \dots & 0 \\ \mu_{3,1}||g_1^*|| & \mu_{3,2}||g_2^*|| & ||g_3^*|| & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \mu_{n,1}||g_1^*|| & \mu_{n,2}||g_2^*|| & \dots & \mu_{n,n-1}||g_{n-1}^*|| & ||g_n^*|| \end{pmatrix}}_{R} \cdot \underbrace{\begin{pmatrix} \frac{g_1^*}{||g_1^*||} \\ \frac{g_2^*}{||g_2^*||} \\ \cdot \\ \cdot \\ \frac{g_n^*}{||g_n^*||} \end{pmatrix}}_{Q}, \tag{3.8}$$

where Q is an orthogonal matrix.
With this Gram-Schmidt representation, we see that $\text{vol}(\mathcal{P}(\mathcal{B})) = \prod_i ||g_i^*|| = \det(R)$. Since $\det(Q) = \pm 1$, then $\det(G) = \pm\det(R)$ and $\text{vol}(\mathcal{P}(\mathcal{B})) = |\det(G)|$. Note that $\mathcal{P}(\mathcal{B})$ depends on the lattice basis $\mathcal{B}$. Nevertheless, any other basis of Λ is obtained by multiplying G by a unimodular matrix U (on the left) where $\det(U) = \pm 1$. This operation does not change the volume of the parallelotope. One can also notice that $\text{vol}(\mathcal{P}(\mathcal{B})) = \text{vol}(\mathcal{V}(x))$ where $\mathcal{V}(x)$ is independent of the basis $\mathcal{B}$. This highlights that $\text{vol}(\mathcal{P}(\mathcal{B}))$ is a constant of Λ. It is called the fundamental volume.

Definition 3.7 (Fundamental volume)**.** *Let G be any generator matrix of a lattice Λ. The fundamental volume of Λ is defined as*

$$\text{vol}(\Lambda) = |\det(G)|. \tag{3.9}$$

Definition 3.8 (Minimum distance and packing radius)**.** *The minimum distance $d(\Lambda)$ of a latice Λ is defined as the minimal norm between any two points of Λ. The packing radius is the radius of the largest sphere contained in $\mathcal{V}(0)$, i.e. $\rho(\Lambda) = d(\Lambda)/2$.*

Warning: In the second part of this thesis, dedicated to group codes, $d(\cdot)$ refers to the **squared** minimal norm. So we have $\rho^2(\Lambda) = d(\Lambda)/4$.

An easy way to increase the minimum distance of the lattice is to scale it. However, scaling also increases the fundamental volume of the lattice. Therefore, the quantity of interest is often the fundamental coding gain rather than the minimum distance.

Definition 3.9 (Fundamental coding gain)**.** *The fundamental coding gain of a lattice Λ is the ratio of the minimum distance and the normalized fundamental volume of the lattice*

$$\gamma(\Lambda) = \frac{d^2(\Lambda)}{\text{vol}(\Lambda)^{\frac{2}{n}}}. \tag{3.10}$$

By abuse of language, in this thesis we use the term sphere to refer both to a sphere and a ball. Let $B_r(y)$ denote a sphere of radius r centered at $y \in \mathbb{R}^n$. If the center is irrelevant the sphere is denoted B_r. Let us recall that the volume of a n-dimensional unit sphere is

$$\text{vol}(B_1) = V_n = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} \approx \frac{1}{\sqrt{n\pi}} \left(\frac{2\pi e}{n}\right)^{\frac{n}{2}}. \tag{3.11}$$

[3] Neglecting the boundaries for the Voronoi regions.

[4] Note that this is an instance of a QR decomposition.

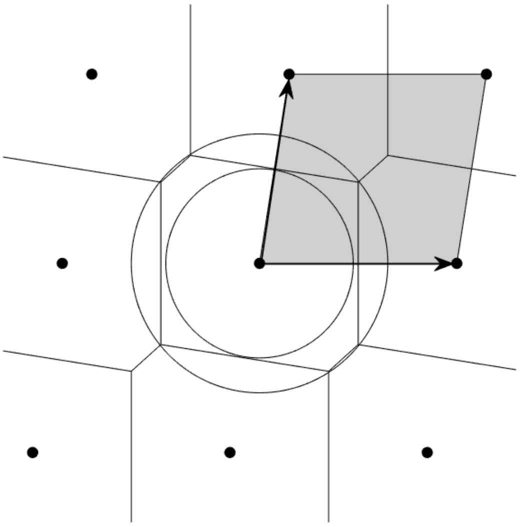


Figure 3.2: Parameters of a lattice. The black arrows represent a basis $\mathcal{B}$. The shaded area is the parallelotope $\mathcal{P}(\mathcal{B})$. The packing radius is the radius of the smallest circle and the covering radius the one of the largest circle. The kissing number τ of this lattice is 2 and the Voronoi number τ_f is 6. In this case, all Voronoi vectors are relevant.

Definition 3.10 (Lattice density and packing efficiency)**.** *The density Υ of a lattice Λ represents the fraction of the space covered by spheres of radius ρ. It is given by the ratio between the volume of a sphere of radius ρ and the fundamental volume of Λ:*

$$\Upsilon = \frac{V_n \cdot \rho^n}{\mathrm{vol}(\Lambda)}. \tag{3.12}$$

The centered density, another conventional parameter, is $\Upsilon/V_n = \gamma^{\frac{n}{2}}/2^n$.
Let ρ_e be the radius of a sphere with volume vol(Λ)*. The packing efficiency is*

$$\rho_{pack}(\Lambda) = \frac{\rho}{\rho_e}. \tag{3.13}$$

We clearly see that the packing efficiency and the density are maximized if the Voronoi region has a spherical shape.

Definition 3.11 (Kissing number)**.** *The number of lattice points located at a distance $d(\Lambda)$ from the origin is the kissing number τ.*

Definition 3.12 (Voronoi vector)**.** *A vector $v \in \Lambda$ is called Voronoi vector if the hyperplane*

$$\{y \in \mathbb{R}^n \ : \ y \cdot v = \frac{1}{2}||v||^2\} \tag{3.14}$$

has a non empty intersection with $\mathcal{V}(0)$. The vector is said relevant if the intersection includes a $n-1$-dimensional face of $\mathcal{V}(0)$.

An example of Voronoi vectors which are not relevant is obtained with the lattice $\mathbb{Z}^2$. The corners of the Voronoi region (a square) are 0-dimensional faces located between two lattice points. We denote by τ_f the number of relevant Voronoi vectors, referred to as the Voronoi number. For root lattices [CS99], the Voronoi number is equal to the kissing number τ. It is proved in [CS92] that all non-zero short vectors in a coset[5] $x + 2\Lambda$ of Λ, $x \in \Lambda$, are Voronoi vectors. There are $|\Lambda/2\Lambda| - 1$ cosets of Λ ($\neq \Lambda$) and thus at least $2(2^n - 1)$ Voronoi vectors. For random lattices, there are only two short vectors per coset, all relevant, and $\tau_f = 2^{n+1} - 2$ [CS92].

[5] See the beginning of Section 3.2.2 for a definition of the term coset.

Definition 3.13 (Theta series and lattice shell)**.** *The theta series of a lattice Λ is*

$$\Theta_{\Lambda}(q) = \sum_{x \in \Lambda} q^{\|x\|^2}, \tag{3.15}$$

$$= \sum_{i=0}^{\infty} \tau_i q^i, \tag{3.16}$$

where τ_i represents the number of lattice points of norm i in Λ. Moreover, a lattice shell denotes the set of τ_i lattice points at a distance i from the origin (with $\tau_{4\rho^2} = \tau$).

For instance, the first non-zero term of the series is $\tau q^{4\rho^2}$ as there are τ lattice points at a distance $d(\Lambda)$ from the origin. These lattice points constitute the first lattice shell.

Definition 3.14 (Covering radius)**.** *The covering radius of Λ is defined as*

$$R(\Lambda) = \underset{y \in \mathbb{R}^n}{max} \ \underset{x \in \Lambda}{min} \ d(y, x), \tag{3.17}$$

Definition 3.15 (Dual lattice)**.** *For any lattice Λ the dual lattice Λ^* is defined as follows:*

$$\Lambda^* = \{u \in \mathbb{R}^n : u \cdot x \in \mathbb{Z}, \ \forall \ x \in \Lambda\}. \tag{3.18}$$

Note that if G is a square generator matrix for Λ, then $(G^{-1})^T$ is a generator matrix for Λ^*. Moreover, if a lattice is equivalent to its dual, it is called a self-dual (or unimodular) lattice. For instance, E_8 and Λ_{24} are self-dual.

Definition 3.16 (KZ basis)**.** *Consider a lower triangular generator matrix G of a lattice Λ where the rows of G are the basis vectors. The basis is said Korkin-Zolotarev (KZ) reduced if it has the following property:*

- $g_{11} = d(\Lambda)$,
- *The lattice generated by the $n-1 \times n-1$ right lower part of G is KZ reduced.*

The main lattice parameters are depicted on Figure 3.2.

3.2 Standard lattice decoders and lattice constructions

3.2.1 BDD, list decoding, optimal and quasi-optimal decoding

Given a lattice Λ, a radius $r > 0$, and any point $y \in \mathbb{R}^n$, the task of a list decoder is to determine all points $x \in \Lambda$ satisfying $d(x, y) \leq r$: i.e. compute the set $\Lambda \cap B_r(y)$. If $r < \rho(\Lambda)$, there is either no point or a unique point found and the decoder is known as a bounded-distance decoder (BDD). In this thesis, BDD means that we consider a decoding radius $r = \rho(\Lambda)$ where in case of a tie between several lattice points, one of them is randomly chosen by the decoder. When $d(x, y) < \rho(\Lambda)$, we say that y is within the guaranteed (or unique) error-correction radius of the lattice. If $r \geq \rho(\Lambda)$, there may be more than one point in the sphere. In this case, the process is called list decoding rather than BDD.
Note that a modified list decoder may output a set of lattice points $\mathcal{T} \neq \Lambda \cap B_r(y)$. Therefore, we may refer to list decoders where $\mathcal{T} = \Lambda \cap B_r(y)$ as "regular" list decoders.
Optimal decoding simply refers to finding the closest lattice point in Λ to any point $y \in \mathbb{R}^n$. In the literature, optimal decoding is usually said that an optimal decoder solves the closest vector problem (CVP). If regular list decoding is used, it is equivalent to choosing a decoding radius equal to $R(\Lambda)$ and keeping the closest point to y in the list outputted by the list decoder.

Let $x \in \Lambda$ and w be a Gaussian vector where each component is i.i.d with distribution $\mathcal{N}(0, \sigma^2)$. Consider the point y obtained as

$$y = x + w. \tag{3.19}$$

Since this model is similar to the discrete-time AWGN channel (see (2.1)), x is referred to as the transmitted point, y the received point, and the process described by (3.19) is also called a Gaussian channel. It is discussed in Section 3.3.2. The point error probability under optimal decoding is $P_e(opt, \sigma^2) = P(y \notin \mathcal{V}(x))$. On the Gaussian channel, given equiprobable symbols, optimal decoding is also referred to as maximum likelihood decoding (MLD). Moreover, we say that a decoder is quasi-MLD (QMLD) if $P_e(dec, \sigma^2) \leq P_e(opt, \sigma^2) \cdot (1 + \epsilon)$, for $\epsilon > 0$.

3.2.2 Coset decomposition and construction of lattices from codes

Coset decomposition of a discrete group

Let S and T be discrete sets endowed with the group property (i.e. group codes), where $T \subseteq S$. If the order of the quotient group S/T is q, then S can be expressed as the union of q cosets of T. We denote by $[S/T]$ a system of coset representatives for this quotient group. It follows that

$$S = \bigcup_{x_i \in [S/T]} T + x_i = T + [S/T]. \tag{3.20}$$

Of course, the groups S and T will be mainly lattices in this thesis. Let Λ be a lattice and $\Lambda' \subseteq \Lambda$ be one of its sublattices. The lattice Λ can be expressed as $\Lambda = \Lambda' + [\Lambda/\Lambda']$.

Theorem 3.1. *Let $\Lambda' \subseteq \Lambda$ be lattices. The order of the quotient group is equal to the ratio of the fundamental volumes*

$$|\Lambda/\Lambda'| = \frac{\mathrm{vol}(\Lambda')}{\mathrm{vol}(\Lambda)}. \tag{3.21}$$

Proof. Since $\Lambda = \Lambda' + [\Lambda/\Lambda']$, it is the union of $|\Lambda/\Lambda'|$ cosets of Λ'. The space $\mathbb{R}^n$ can be expressed as

$$\mathbb{R}^n = \Lambda + \mathcal{P}(\Lambda) = \Lambda' + [\Lambda/\Lambda'] + \mathcal{P}(\Lambda) = \Lambda' + \mathcal{P}(\Lambda'). \tag{3.22}$$

This shows that $\mathcal{P}(\Lambda')$ is the union of $|\Lambda/\Lambda'|$ instances of $\mathcal{P}(\Lambda)$ and thus has a volume $|\Lambda/\Lambda'|$ times larger. □

Example 3.2 (Gaussian heuristic)**.** *The Gaussian heuristic enables to approximate the number of lattice points within a sphere B. Similarly to the proof of Theorem 3.1, the idea is to count the number of parallelotopes in B. If the volume of the sphere is large enough (compared to the one of $\mathcal{P}(\Lambda)$), one can neglect the boundaries and the number of points in B is*

$$|\Lambda \cap B| \approx \frac{\mathrm{vol}(B)}{\mathrm{vol}(\Lambda)}. \tag{3.23}$$

For large dimensions, the Gaussian heuristic can also be used to estimate the average minimum distance of "typical" random n-dimensional lattices of volume vol*. We estimate the average minimum distance of these random lattices as the radius of the n-sphere B of volume* vol *(i.e. this is the case $|\Lambda \cap B| = 1$ in* (3.23)*):*

$$d(\Lambda) \approx \frac{\mathrm{vol}^{\frac{1}{n}}}{V_n^{\frac{1}{n}}} \approx \sqrt{\frac{n}{2\pi e}} \cdot vol^{\frac{1}{n}}, \tag{3.24}$$

where we used (3.11)*. Note however that the heuristic* (3.23) *is not always accurate as emphasized in [MO90], especially when there is a low number of points within the set.*

Example 3.3 ($\mathbb{Z}/2\mathbb{Z}$ and $\mathbb{Z}^2/R\mathbb{Z}^2$)**.** *Figure 3.3 shows two lattices Λ represented as $\Lambda = \cup_{x \in [\Lambda/\Lambda']} \Lambda' + x$. We let $R\mathbb{Z}^2$ denote the real version of $\phi\mathbb{Z}[i]$ (obtained e.g. with* (3.3)*).*

Construction of lattices from codes

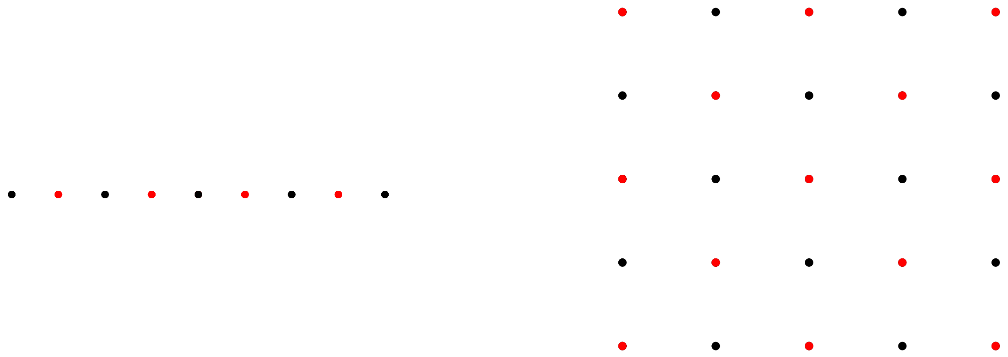
Throughout the recent history of lattices, a popular method to construct lattices with interesting properties (and often easily decodable) has been to use the coset decomposition of a lattice along with a code [CS99].

Definition 3.17 (Construction A)**.** *Let $\Lambda_1 \subset \Lambda_0$ be J-lattices. Let $\mathcal{C}$ be a linear code of length m over $[\Lambda_0/\Lambda_1]$, denoted as $\mathcal{C}_{[\Lambda_0/\Lambda_1]}$. A new lattice $\Lambda \subseteq \Lambda_0^m$ of dimension n is obtained as*

$$\Lambda = \Lambda_1^m + \mathcal{C}_{[\Lambda_0/\Lambda_1]} = \{x \in \Lambda_0^m : x \mod \Lambda_1^m \in \mathcal{C}_{[\Lambda_0/\Lambda_1]}\}. \tag{3.25}$$

Given a good code, this method enables to build lattices with a large coding gain. For real lattice, we have

$$\gamma = \frac{d^2(\Lambda)}{\mathrm{vol}(\Lambda)^{\frac{2}{n}}} = \frac{\min\{d^2(\Lambda_1), d^2(\mathcal{C}_{[\Lambda_0/\Lambda_1]})\}}{\left(\frac{\mathrm{vol}(\Lambda_1)^m}{|\mathcal{C}_{[\Lambda_0/\Lambda_1]}|}\right)^{\frac{2}{n}}}. \tag{3.26}$$



(a) The real integers $\mathbb{Z}$ represented as $2\mathbb{Z} \cup 2\mathbb{Z}+1$, (where $[\mathbb{Z}/2\mathbb{Z}] = \{0,1\}$).

(b) $\mathbb{Z}[i]$ represented as $(\phi\mathbb{Z}[i]+0) \cup (\phi\mathbb{Z}[i]+1)$, (where $[\mathbb{Z}[i]/\phi\mathbb{Z}[i]] = \{0,1\}$). The real version is $(R\mathbb{Z}^2+(0,0)) \cup (R\mathbb{Z}^2+(1,0))$.

Figure 3.3: Examples of coset decomposition of lattices.

An advantage of this construction is that theses lattices admit an obvious decoding algorithm, called a multistage decoder. It is illustrated in Figure 3.4 and works as follows. Given a point $y = x_0 + x_1 + w$ to decode, where $x_0 \in \mathcal{C}_{[\Lambda_0/\Lambda_1]}$ and $x_1 \in \Lambda_1^m$:

1. Compute $y' = y \mod \Lambda_1^m$.
2. Decode y' in $\mathcal{C}_{[\Lambda_0/\Lambda_1]}$ as $\hat{x}_0$.
3. Decode $y - x_0$ in Λ_1^m as $\hat{x}_1$.
4. The decoded lattice point is $\hat{x} = \hat{x}_0 + \hat{x}_1$.

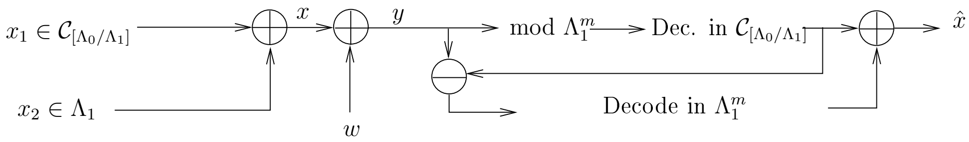

$x_1 \in \mathcal{C}_{[\Lambda_0/\Lambda_1]}$
x
y
mod Λ_1^m
Dec. in $\mathcal{C}_{[\Lambda_0/\Lambda_1]}$
$\hat{x}$
$x_2 \in \Lambda_1$
w
Decode in Λ_1^m

Figure 3.4: Multistage decoder.

However, it may be too complicated to decode in $\mathcal{C}_{[\Lambda_0/\Lambda_1]}$ or to find a code $\mathcal{C}_{[\Lambda_0/\Lambda_1]}$ with the desired properties. Consequently, several levels may be considered to reduce the order of $[\Lambda_0/\Lambda_1]$.

Definition 3.18 (Construction by code formula)**.** *Let $\Lambda_a \subset \Lambda_{a-1} \subset ... \subset \Lambda_1 \subset \Lambda_0$ be J-lattices. Consider several linear codes, one for each quotient group Λ_{i-1}/Λ_i: $\mathcal{C}^i_{[\Lambda_{i-1}/\Lambda_i]}$. We say that a lattice Λ can be described by a code formula if the following holds.*

$$\begin{aligned}\Lambda =& \Lambda_a^m + \mathcal{C}^a_{[\Lambda_{a-2}/\Lambda_{a-1}]} + ... + \mathcal{C}^2_{[\Lambda_1/\Lambda_2]} + \mathcal{C}^1_{[\Lambda_0/\Lambda_1]}, \\ =& \Big\{ x_a + \sum_{i=1}^{a} x_i \in \Lambda_0^m : \ x_a \in \Lambda_a^m, x_i \in \mathcal{C}^i_{[\Lambda_{i-1}/\Lambda_i]} \Big\}.\end{aligned} \tag{3.27}$$

The above definition means that the ith level of the lattice is an element of the code $\mathcal{C}^i_{[\Lambda_{i-1}/\Lambda_i]}$. If the set described by (3.27) is not constrained to be a lattice, Construction by code formula is called the multilevel construction. This latter construction was introduced by Imai and Hirakawa in 1977 [IH77]. These constructions have been largely exploited in the scope of binary lattices [For88a] [For88b] [Loe97] [FTS00] [Reg05] [dZB18].

Definition 3.19 (Real binary lattice)**.** *A real lattice Λ of dimension n is binary if there exists an integer a such that $2^a\mathbb{Z}^n \subset \Lambda \subset \mathbb{Z}^n$.*

Examples. We provide several examples of binary lattices:

- Construction A based on one dimensional real integers: Let $\Lambda_0 = \mathbb{Z}$ and $\Lambda_1 = q\mathbb{Z}$. Let $\mathcal{C}_{\mathbb{Z}_q}$ be a linear code of length n and dimension k over $\mathbb{Z}_q = [\mathbb{Z}/q\mathbb{Z}]$, generated by a matrix $A \in \mathbb{Z}_q^{k\times n}$ (i.e. we assume that the code is free). A new lattice $\Lambda(A)$ is obtained as

$$\Lambda(A) = q\mathbb{Z}^n + \mathcal{C}_{\mathbb{Z}_q} = \{x \in \mathbb{Z}^n : x \mod q \in \mathcal{C}_{\mathbb{Z}_q}\} = \{x \in \mathbb{Z}^n : x = zA + q\mathbb{Z}^n,\ z \in \mathbb{Z}^k\}. \quad (3.28)$$

Given a systematic generator matrix of $\mathcal{C}_{\mathbb{Z}_q}$ of the form $A = (I_k\ P)$, a generator matrix for $\Lambda(A)$ is

$$G = \begin{pmatrix} qI_{n-k} & 0 \\ P & I_k \end{pmatrix}. \quad (3.29)$$

Note that the same lattice can be defined via the parity-check matrix $H \in Z_q^{n\times k}$ of $\mathcal{C}_{\mathbb{Z}_q}$:

$$\Lambda = \{x \in \mathbb{Z}^n : xH \mod q = 0\}. \quad (3.30)$$

Two instances of this construction are the following.

  - Let $q = p$ be a prime and $\mathcal{C}_{\mathbb{F}_p}$ be linear code of length n over the finite field $\mathbb{F}_p$.The group Λ_0/Λ_1 injects naturally into $\mathbb{F}_p$ via an additive group isomorphism. Then the lattice is

$$\Lambda = p\mathbb{Z}^n + \mathcal{C}_{\mathbb{F}_p}. \quad (3.31)$$

  - Let A be the generator matrix of the (linear) quaternary Golay code $\hat{Q}_{24}$ over $\mathbb{Z}_4$. It is shown in [BSC95] that $\Lambda(A)$ is the Leech lattice.

- Construction D based on one dimensional real integers: Note that (3.27) does not always yield a lattice, in contrary to the following construction D (as in [SI17]).

Definition 3.20 (Construction D). *Choose $\Lambda_0 = \mathbb{Z}$ and decompose it as $\Lambda_0 = q^a\mathbb{Z} + [q^{a-1}\mathbb{Z}/q^a\mathbb{Z}] + ... + [\mathbb{Z}/q\mathbb{Z}]$. Let $\mathcal{C}^1_{\mathbb{Z}_q} \subseteq \mathcal{C}^2_{\mathbb{Z}_q} \subseteq ... \subseteq \mathcal{C}^a_{\mathbb{Z}_q}$ be nested linear codes of length m over $\mathbb{Z}_q$. Let the vectors $g^i_1, .., g^i_{k_i}$ span $\mathcal{C}^i$. The lattice Λ consists of all points of the form*

$$\Lambda = \Big\{x_a + \sum_{i=1}^{a} q^i \sum_{j=1}^{k_i} \alpha^i_j g^i_j,\ x_a \in \Lambda_a^m, \alpha^i_j \in \{0, 1, ..., q-1\}\Big\}. \quad (3.32)$$

In some cases, it can be shown that Construction by code formula and Construction D are equivalent. See e.g. [KO14] and the following example.

- Construction by code formula based on one dimensional real integers and binary quotient group: Again, choose $\Lambda_0 = \mathbb{Z}$, decompose it as $\Lambda_0 = 2^a\mathbb{Z} + [2^{a-1}\mathbb{Z}/2^a\mathbb{Z}] + ... + [\mathbb{Z}/2\mathbb{Z}]$ and choose binary codes $\mathcal{C}^1_{\mathbb{Z}_2} \subseteq \mathcal{C}^2_{\mathbb{Z}_2} \subseteq ... \subseteq \mathcal{C}^a_{\mathbb{Z}_2}$ of length n, such that the Schur product of any two codewords of $\mathcal{C}_i$ is contained in $\mathcal{C}_{i+1}$, for all i. In this case the Construction by code formula always yield a lattice, which is the same as the one obtained with Construction D [KO14]. For instance, nested Reed-Muller codes do have this properties. The lattice is

$$\Lambda = 2^a\mathbb{Z}^n + 2^{a-1}\mathcal{C}^a_{\mathbb{Z}_2} + ... + 2\mathcal{C}^2_{\mathbb{Z}_2} + \mathcal{C}^1_{\mathbb{Z}_2}. \quad (3.33)$$

- Construction by code formula based on one dimensional complex integers: Choose $\Lambda_0 = \mathbb{Z}[i]$, decompose it as $\Lambda_0 = \phi^a\mathbb{Z}[i] + [\phi^{a-1}\mathbb{Z}[i]/2^a\mathbb{Z}[i]] + ... + [\mathbb{Z}[i]/\phi\mathbb{Z}[i]]$ (where $|\mathbb{Z}[i]/\phi\mathbb{Z}[i]| = 2$) and choose binary codes $\mathcal{C}^1_{[\mathbb{Z}[i]/\phi\mathbb{Z}[i]]} \subseteq \mathcal{C}^2_{[\mathbb{Z}[i]/\phi\mathbb{Z}[i]]} \subseteq ... \subseteq \mathcal{C}^a_{[\mathbb{Z}[i]/\phi\mathbb{Z}[i]]}$ of length $n/2$ such that the Schur product of any two codewords of $\mathcal{C}_i$ is contained in $\mathcal{C}_{i+1}$, for all i. The (complex) lattice is

$$\Lambda = \phi^a\mathbb{Z}[i]^n + \phi^{a-1}\mathcal{C}^a_{[\mathbb{Z}[i]/\phi\mathbb{Z}[i]]} + ... + \mathcal{C}^1_{[\mathbb{Z}[i]/\phi\mathbb{Z}[i]]}. \quad (3.34)$$

The real version of this lattice Λ^r is a binary lattice since $2^a\mathbb{Z}^n \subset \Lambda^r \subset \mathbb{Z}^n$.

3.3 Lattice packing problems

We present two famous lattice packing problems. These problems could also be established for any discrete sets in the Euclidean space. Nevertheless, researchers have been focusing on lattices as their group structure is very helpful for manipulations. Moreover, in many situations the group structure of lattices does not prevent them from being good with respect to these problems.

For instance, it was proved recently that the lattice A_3 (the "Orange in a market" packing), E_8, and Λ_{24} are the best sphere packings possible in their respective dimension [Via17] [CKM$^+$17]. The result for the A_3 lattice (alos known has the face-centered cubic lattice) was one of the first conjecture related to lattices. The conjecture was made in the seventeenth century by the astronomer Johannes Kepler. Gauss showed that no other lattice sphere packing is better but it was only in 1998 that Hales proved that no non-lattice packing is better, thus proving the conjecture[6].

3.3.1 Sphere packing

Problem 3.1 (Lattice sphere packing). *The lattice sphere packing problem asks to find lattices with the highest coding gain, i.e. with the highest radius of the packing spheres relatively to the fundamental volume.*

This problem is equivalent to maximizing the density Υ. Remember that the packing spheres are included in the Voronoi regions. Consequently, since the space not covered by the packing spheres should be minimized, a good lattice packing should have "spherical" Voronoi regions.
How large can the fundamental coding gain be? Is it bounded? The answer to the latter question is positive, and it was shown by Minkowski that for any lattice Λ, $d(\Lambda) \leq \sqrt{n} \cdot \text{vol}(\Lambda)^{\frac{1}{n}}$.

Theorem 3.2 (Minkowski's first theorem). *Let Λ be any lattice in $\mathbb{R}^n$. For any lattice $\Lambda \subset \mathbb{R}^n$, we have*

$$\gamma(\Lambda) \leq n. \tag{3.35}$$

In a given dimension n, the highest possible coding gain is called the Hermite constant. It is known for $n \leq 8$ and $n = 24$ (and achieved by the lattice shown in Figure 3.5).

Theorem 3.3 (Hermite constant). *Let Λ be any lattice in $\mathbb{R}^n$. The Hermite constant is the highest achievable coding gain by a lattice in a given dimension n.*

$$\gamma_n = \max_{\Lambda} \gamma(\Lambda). \tag{3.36}$$

What about a lower bound on the highest density achievable by a lattice in dimension n? This can be obtained via the Minkowski-Hlawaka theorem. The theorem was introduced by [deB75] in this scope (see also [Zam14, Chapter 7]). Intuitively, the Gaussian heuristic (see Example 3.2) should become accurate if the point density is uniform: I.e. if the number of lattice points per unit volume is almost everywhere $\zeta = \frac{1}{\text{vol}(\Lambda)}$. The number of lattice points in any sphere B would then be $\zeta \cdot \text{vol}(B)$. More generally, such a lattice (if it exists) enables to replace sampling by integration in a body S: i.e. $\sum_{x \in \mathcal{S}} f(x) = \zeta \int_{\mathcal{S}} f(x)dx$. The result of Minkowski (but proved by Hlawaka and Siegel) proves that there exists such a random set of lattices. In particular if $f(x) = 1_{x \in \mathcal{S}}$, it implies that the Gaussian heuristic becomes exact for this set of lattices.

Theorem 3.4 (Minkowski-Hlawaka). *For each dimensions $n > 1$, there exists a random ensemble of lattice $\mathbb{L}$, called the Minkowski-Hlawaka ensemble, such that for any bounded set S:*

$$E_{\mathbb{L}}[|S \cap \Lambda|] = \frac{\text{vol}(S)}{\text{vol}(\Lambda)}. \tag{3.37}$$

Now, choose S as a sphere of radius ρ_e, centered on 0, such that $\text{vol}(B_{\rho_e}(0)) = \text{vol}(\Lambda) - \epsilon$. Equation (3.37) shows that (on average) there is no non-zero lattice point in $B_{\rho_e}(0)$. The minimum distance of at least one lattice of the ensemble (random coding argument), say Λ, must be at least as large as the radius of this sphere: i.e. $d(\Lambda) \geq \rho_e$. This yields the following theorem.

Theorem 3.5. *For any $n \geq 1$, there exists a lattice $\Lambda \in \mathbb{R}^n$ with packing efficiency $\rho_{pack} > \frac{1}{2}$.*

Figure 3.5 presents the densest lattice sphere packings known[7] in dimension $n \leq 80$. For large n, it is known [CS99, Chapter 1] that

$$\frac{n}{2\pi e} \leq \gamma_n \leq \frac{1.744n}{2\pi e}. \tag{3.38}$$

[6] Note that the proof of Hales is computer based: The result was a big achievement for the field of computer based proof verification but it has also drawn criticism as this proof technique provides less understanding of the problem than usual ones, and is thus prone to error.

[7] Based on the data of: http://www.math.rwth-aachen.de/∼Gabriele.Nebe/LATTICES/density.html

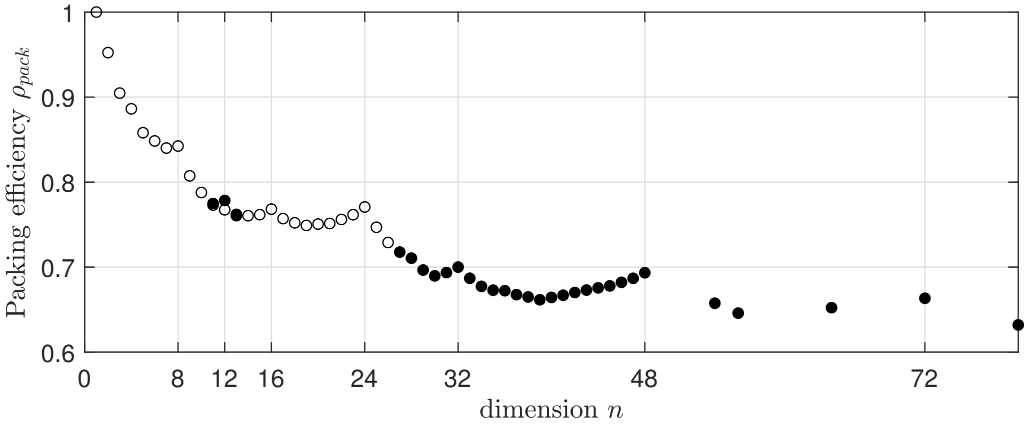


Figure 3.5: The densest lattice sphere packings in dimensions $n \leq 80$. The laminated lattices (see [CS99]) are shown by the white dots. Note that the dimensions 8, 16, 24, 32, 48, and 72 are "special" as they achieve a local maximum. The names of the dense lattices in these special dimensions are E_8, BW_{16}, Λ_{24}, Q_{32}, $P_{48\{n,p,q\}}$, and $\mathscr{N}_{72}$, respectively.

3.3.2 Lattices with additive Gaussian noise

The second lattice problem is the following.

Problem 3.2 (Lattice coding on the unconstrained AWGN channel)**.** *Let $x \in \Lambda$ and w be a Gaussian vector with i.i.d components $\mathcal{N}(0, \sigma^2)$. Let $y \in \mathbb{R}^n$ such that $y = x + w$. Find a lattice Λ such that $P_e(opt) = P(y \notin \mathcal{V}(x)) < \epsilon$.*

This unconstrained AWGN channel is to be opposed with the power constrained AWGN channel, where the average norm of the lattice points x should not be greater than a given value. This problem is trivial if the variance of the noise is small relatively to the volume of the lattice. Therefore, the error probability (of wrong decoding) is in general evaluated with respect to the volume-to-noise ratio (VNR).

Definition 3.21 (Volume-to-noise ratio)**.** *The volume-to-noise ratio, denoted by Δ, is*

$$VNR = \Delta = \frac{\text{vol}(\Lambda)^{\frac{2}{n}}}{2\pi e \sigma^2}. \tag{3.39}$$

Thanks to the additive group structure of lattices, the error probability does not depend on the lattice point transmitted. We shall therefore consider that the transmitted point x is the origin. The probability of correct decoding $P_c = P(y \in \mathcal{V}(x))$ (with an optimal decoder) is $P_c = \int_{\mathcal{V}(0)} f(t)dt$, where

$$f(t) = \frac{1}{(\sqrt{2\pi}\sigma)^n} e^{-\frac{||t||}{2\sigma^2}}. \tag{3.40}$$

Trivially, the error probability is then $P_e = 1 - P_c$.

The union bound, presented in Section 2.2, can be used with the Theta series to estimated the MLD performance of a lattice Λ. The probability of error per lattice point of a lattice Λ is bounded from above by

$$P_e(opt) \leq P_e(ub), \tag{3.41}$$

where

$$P_e(ub) = \frac{1}{2}\Theta_\Lambda\left(\exp(-\frac{1}{8\sigma^2})\right) - \frac{1}{2}. \tag{3.42}$$

It can be easily shown that $\frac{\rho^2}{2\sigma^2} = \frac{\pi e \Delta \gamma}{4}$. For $\Delta \to \infty$, the term $\tau q^{4\rho^2}$ dominates the sum in $\Theta_\Lambda(q)$, then

$$P_e(opt) \leq \frac{\tau}{2}\exp(-\frac{\pi e \Delta\gamma}{4}) + o\left(\exp(-\frac{\pi e \Delta\gamma}{4})\right). \tag{3.43}$$

For moderate Δ, only the lattice shells within ≈ 3 dB of the first lattice shell (i.e. at a squared distance of $2 \cdot d^2(\Lambda)$ from the origin), need to be considered to get a good estimate of the MLD performance.

How small can $P_e(opt)$ be? There is a simple lower bound for this problem (which applies to any lattice). The lower bound is based on the geometric fact that no decoding region of a given volume can be better than a sphere:

$$P_c \leq \int_{B_{\rho_e(0)}} f(t)dt, \tag{3.44}$$

where ρ_e is the radius of a sphere of volume vol(Λ). It is therefore often called the sphere lower bound. A method to compute this bound for a given n is provided in [TVZ99]. Remember Lemma 2.1 which states that the squared norm of the noise vector concentrates around it mean $n\sigma^2$. Asymptotically, the error probability can be made small if the decoding sphere radius is larger than the one of the "noise sphere" of squared radius $n\sigma^2$, where its volume is $\approx (2\pi e\sigma^2)^{\frac{n}{2}}$ for n large (see (3.11)). It follows that one must have $\text{vol}(\Lambda)^{\frac{n}{2}} \geq 2\pi e\sigma^2$ if the error probability is to be small (see e.g. [FTS00] or [IZF13]).

Theorem 3.6 (Asymptotic Sphere bound)**.** *For large n, given any lattice Λ, the error probability can not be small if the noise variance is greater than*

$$\sigma^2 \geq \frac{\text{vol}(\Lambda)^{\frac{2}{n}}}{2\pi e}. \tag{3.45}$$

Poltyrev proved in 1997 that there exist lattices achieving this bound.

Theorem 3.7 (Poltyrev limit)**.** *Let $\sigma_{max}^2 = \text{vol}(\Lambda)^{\frac{2}{n}}/2\pi e$. For the unconstrained AWGN channel with noise variance σ^2, there exists a sequence of lattice Λ_n such that for $\sigma \leq \sigma_{max}$, the error probability decreases exponentially in n.*

The following VNR value is called the Poltyrev limit.

$$\frac{\text{vol}(\Lambda)^{\frac{2}{n}}}{2\pi e\sigma_{max}^2} = 1. \tag{3.46}$$

We present three different methods to prove that a family of lattices achieves the Poltyrev limit. The ensemble studied by the first proof (which was the first discovered) is not practical as we do not know how to decode these lattices. However, the ensembles considered by the two other methods admit efficient decoding algorithms.

Minkowski-Hlawaka based proof

We investigate the average performance of a Minkowski-Hlawaka ensemble of lattices $\mathbb{L}$ (see Theorem 3.4): $P_e = E_{\Lambda \in \mathbb{L}}[P_e(\Lambda)]$. Consider regular list decoding where r is the radius of a decoding sphere around the received point y. P_e is bounded from above by the probability that the norm of the noise is greater than the radius of the decoding sphere plus the probability that another point than 0 is in the decoding sphere:

$$\begin{aligned} P_e \leq& P(||w|| > r) + E_{\mathbb{L}}[P(y \notin \mathcal{V}_\Lambda(x))], \\ \leq& P(||w|| > r) + E_{\mathbb{L}} P(|B_r(y) \cap \Lambda \backslash 0| > 1), \\ \leq& P(||w|| > r) + E_{\mathbb{L}} E_y[|B_r(y) \cap \Lambda \backslash 0|]. \end{aligned} \tag{3.47}$$

Using the uniformity property of a Minkowski-Hlawaka ensemble (see Theorem 3.4), the expected number of lattice points depends only on the volume of the sphere and not on its location:

$$E_{\mathbb{L}} E_y[|B_r(y) \cap \Lambda \backslash 0|] = E_y E_{\mathbb{L}}[|B_r(y) \cap \Lambda|] = E_{\mathbb{L}}[|B_r(0) \cap \Lambda|]. \tag{3.48}$$

Moreover, with Equation (3.37) we get

$$E_{\mathbb{L}}[|B_r(0) \cap \Lambda|] = \left(\frac{r}{\rho_e}\right)^n, \tag{3.49}$$

where ρ_e is the radius of a sphere of volume vol(Λ). Lemma 2.1 indicates that if we choose r slightly greater than the typical norm of the noise $\sqrt{n\sigma^2}$, say $r = \sqrt{n\sigma^2}(1+\epsilon_1)$ then $P(||w|| > r) \to 0$. Moreover, if $r < \rho_e$ then $(r/\rho_e)^n \to 0$. Hence, if $\rho_e = \sqrt{n\sigma^2}(1+\epsilon_2)$, $\epsilon_2 > \epsilon_1$, then $P_e \to 0$ when $n \to 0$. The VNR obtained with this lattice is

$$\frac{(V_n(\sqrt{n\sigma^2}(1+\epsilon_2))^n)^{\frac{2}{n}}}{2\pi e\sigma^2} \to 1 \text{ when } n \to +\infty \ (\epsilon_2 \to 0). \tag{3.50}$$

This proof works for any lattice ensemble with the uniformity property of the Minkowski-Hlawaka ensemble. For instance, Loeliger proved that random Construction A lattices[8] do have this property [Loe97] and therefore inherits the result.

Remark 3.1. *While being a useful theoretical tool, random Construction A lattices can be seen as the worst choice from a decoding complexity view point (and hence for practical communications system). Indeed, Regev proved in 2005 that decoding random Construction A lattices are as hard as decoding any lattice (see Section 3.4.4).*

Remark 3.2. *The above proof implies that the coding gain of these sphere bound achieving lattices is $\Theta(n)$: Normalize the lattice such that $\text{vol}(\Lambda) = 1$, then $\sigma^2_{max} = \frac{1}{2\pi e}$, and the noise sphere has radius $n\sigma^2_{max}$. Since there is only one lattice point in this noise sphere (the center of the sphere), the minimum distance of the lattice is $\Omega(n)$. Does it means that $P_e(opt)$ can not be small if $\gamma(\Lambda) = o(n)$? No: If $\text{vol}(B_{n\sigma^2_{max}}(0)\backslash\mathcal{V}(0))$ is negligible, the error probability can be small despite a small coding gain. For instance, the construction presented in the next subsection achieves the Poltyrev limit despite having a coding gain of the form n^λ, $\lambda < 1$.*

Low-density Construction A lattices

Nicolas di Pietro studied recently an ensemble similar to the one of Loeliger but where the random code over $\mathbb{Z}_q^n$ is replaced by a random low density parity-check code $\mathcal{C}_{\mathbb{F}_p}$ over the finite field $\mathbb{F}_p$. This ensemble of lattices is called low density Construction A (LDA) lattices [dZB18]. The proof technique relies on the same idea as the one above: Prove that there is only the transmitted lattice point in the decoding sphere. More specifically, it consists in computing the probability that the integer points (except the origin) in the decoding sphere $B_{n\sigma^2_{max}(1-\epsilon)}(y)$ are in the lattice (i.e. belong to $\mathcal{C}_{\mathbb{F}_p}$). Namely, compute the quantity (assuming that the integer points in $p\mathbb{Z}^n$ are not in the decoding sphere):

$$\sum_{x \in \mathbb{Z}^n \backslash 0 \backslash p\mathbb{Z}^n \cap B_{n\sigma^2_{max}(1-\epsilon)}(y)} P(x \in \Lambda), \tag{3.51}$$

where $P(x \in \Lambda) = P(xH = 0 \mod p)$, and where each column of H has m non zero value (chosen uniformly at random) with each of these non-zero value is chosen uniformly at random in $\{0, ..., p-1\}$. Showing that (3.51) goes to 0 when $n \to +\infty$ proves that the ensemble achieves the Poltyrev limit. Note that with this construction it is critical that the decoding sphere is not centered at the origin as the minimum distance of the lattices in this ensemble is $o(n)$.

The advantage of this construction is that there exists an efficient (but suboptimal) decoder: The iterative message passing algorithm. It is shown in [dPBZB12] that performance within 0.8 dB of the Poltyrev limit can be achieved with this decoding algorithm.

Information theoretic proof

While the proofs for the two previous constructions rely on averaging and geometric arguments, this last result, established in [FTS00], is based on the chain rule of information theory. Consider the channel defined by the random variables $Y = A + B + W$, where A,B are inputs and W is the noise. The chain rule of information theory states that

$$I(AB;Y) = I(A;Y) + I(B;Y|A), \tag{3.52}$$

where $I(\cdot;\cdot)$ denotes the mutual information.
This equations means that one should code A at the information rate $I(A;Y)$ and B at $I(B;Y|A)$. On the receiving side, on should first decode A with negligible error probability and then decode B given A. This principle applies as follows to lattices build via Construction A or D.

Let the transmitted message be $x = a + b'$, where $a \in \mathcal{P}(\Lambda_0)$ and $b' \in \Lambda_0$. The received message is $y = a + b' + w$ where w is the usual Gaussian noise. The $\mod \Lambda_0$ operation (as on Figure 3.4) creates the

[8] Where the generator matrix of the code is chosen uniformly at random over $\mathbb{Z}_q^{k\times n}$, a specific scaling is applied to $\Lambda(A)$, and $q \to \infty$.

$\mod \Lambda_0$ channel $y' = a + w \mod \Lambda_0$. Let $w' = w \mod \Lambda_0$. Then, $y' = a + w'$. It is shown in [FTS00] that the capacity of this channel can be computed has $C(\Lambda_0, \sigma^2) = \log V(\Lambda_0) - h(\Lambda_0, \sigma^2)$ where $h(\Lambda_0, \sigma^2)$ is the differential entropy of the $\mod \Lambda_0$ aliased noise. This capacity is achieved when the distribution of a is uniform in $\mathcal{P}(\Lambda_0)$.

The Λ_0/Λ_1 channel is a special $\mod \Lambda_1$ channel where the input b is restricted to the discrete set $\Lambda_0/\Lambda_1 \cap \mathcal{P}(\Lambda_1) = [\Lambda_0/\Lambda_1] \mod \Lambda_1 = [\Lambda_0/\Lambda_1]$ (if the coset representatives are properly chosen), instead of $\mathcal{P}(\Lambda_1)$ for the regular $\mod \Lambda_1$ channel. Since $[\Lambda_0/\Lambda_1] + \mathcal{P}(\Lambda_0) = \mathcal{P}(\Lambda_1)$, given a transmitted message $x = a + b + c$, $a \in \mathcal{P}(\Lambda_0)$, $b \in [\Lambda_0/\Lambda_1]$, $c \in \Lambda_1$, then $y' = x + w \mod \Lambda_1 = a + b + w'$ is a proper $\mod \Lambda_1$ channel.

Now, let A be a continuous random variable uniformly distributed in $\mathcal{P}(\Lambda_0)$ and B a discrete random variable uniformy distributed in $[\Lambda_0/\Lambda_1]$. Then, $I(AB; Y') = C(\Lambda_1, \sigma^2)$. By the chain rule of information theory (3.52):

$$C(\Lambda_1, \sigma^2) = C(\Lambda_0, \sigma^2) + C(\Lambda_0/\Lambda_1, \sigma^2). \tag{3.53}$$

Consequently, if the noise variance is high relatively to the volume of Λ_0, then $C(\Lambda_0) \approx 0$ and $C(\Lambda_1) \approx C(\Lambda_0/\Lambda_1)$. This can also be understood as follows: If Λ_1 is fixed and Λ_0 is chosen such that $\operatorname{vol}\Lambda_0 \to 0$, then the variable B over $[\Lambda_0/\Lambda_1]$ tends to a continuous variable over the entire $\mathcal{P}(\Lambda_1)$, where the continuous uniform variable over $\mathcal{P}(\Lambda_1)$ is precisely the capacity achieving scheme.

Assume that $\Lambda_1 \in \mathbb{R}^N$ and that a code of length m over $[\Lambda_0/\Lambda_1]$ is used. Considering that $\operatorname{vol}\Lambda_0$ is negligible, one should code $[\Lambda_0/\Lambda_1]$ with with a capacity achieving code $\mathcal{C}_{[\Lambda_0/\Lambda_1]}$ of information rate $R = C(\Lambda_0/\Lambda_1, \sigma^2) = \frac{2}{mN} \log |\mathcal{C}_{[\Lambda_0/\Lambda_1]}|$ bits per two dimensions. With (3.21), we get

$$\log \operatorname{vol}(\Lambda)^{\frac{2}{mN}} = \log \left(\frac{\operatorname{vol}(\Lambda_1)^m}{|\mathcal{C}_{[\Lambda_0/\Lambda_1]}|} \right)^{\frac{2}{mN}} = \log \frac{\operatorname{vol}(\Lambda_1)^{\frac{2}{N}}}{|\mathcal{C}_{[\Lambda_0/\Lambda_1]}|^{\frac{2}{mN}}} = \log \operatorname{vol}(\Lambda_1)^{\frac{2}{N}} - R. \tag{3.54}$$

Then, if $\operatorname{vol}(\Lambda_1)$ is large enough such that $P_e(\Lambda_1, \sigma^2) = 0$, then it can be shown that

$$C(\Lambda_0/\Lambda_1, \sigma^2) \approx \log \frac{\operatorname{vol}(\Lambda_1)^{\frac{2}{N}}}{2\pi e \sigma^2}. \tag{3.55}$$

As a result, $VNR(\Lambda, \sigma^2) \approx 1$ and the error probability with multistage decoding (see Figure 3.4) is

$$P_e \leq P_e(\mathcal{C}_{[\Lambda_0/\Lambda_1]}) + P_e(\Lambda_1) \approx 0. \tag{3.56}$$

Nevertheless, as explained in Section 3.2.2, binary codes are easier to build and to decode than non-binary codes. Therefore, several levels of partitioning can are can be considered and $\Lambda_a \subset ... \subset \Lambda_1 \subset \Lambda_0$ can simply be chosen as $2^{a-1}\mathbb{Z} \subset ..., \subset 2\mathbb{Z} \subset \mathbb{Z}$. Then via the chain rule of informations theory:

$$C(\Lambda_a/\Lambda_0, \sigma^2) = C(\Lambda_0/\Lambda_1, \sigma^2) + ... + C(\Lambda_{a-1}/\Lambda_a, \sigma^2), \tag{3.57}$$

and each level should be coded via a capacity achieving binary code of rate $C(\Lambda_{i-1}/\Lambda_i, \sigma^2)$ to get a capacity achieving lattice.

The disadvantage of this multilevel construction under multistage decoding is the error propagation of the successive-cancellation type decoder: If the information rates are not correctly chosen or the dimension n is not large enough, significant loss in performance are observed in practice. For instance, [YLW13] used polar codes [Ari09] to code each level of a Construction D lattice. The advantage of polar codes over other codes is that it is easy to build a capacity achieving code with a specific information rate (but smaller than 1). Hence, it is theoretically possible to code each level at its capacity. Nevertheless, [YLW13] reports disappointing performance since the error probability is vanishing at more than 1.75 dB from the Poltyrev limit for a block-length $n = 8000$.

Nevertheless, [WH95] [WFH99] exhibit very satisfactory performance, on the discrete-time AWGN channel, using binary turbo codes at each level of a (non-lattice) multilevel finite constellation.

3.4 Main computational problems

We describe the main computational problems related to lattices. We state some hardness results and present some techniques to solve these problems.

3.4.1 Description of the problems

Problem 3.3 (Search-SVP). *Given a real lattice Λ represented by a given basis, find $x \in \Lambda$ such that $||x|| = d(\Lambda)$, i.e. find a shortest vector.*

Search-SVP is in general simply called SVP. Approximate-SVP$_\delta$ (or Search-SVP$_\delta$) is a relaxed version of Search-SVP where one asks for a vector x whose size is bounded away from the non-zero shortest vector: $||x|| \leq \delta \cdot d(\Lambda)$. Approximate-SVP$_\delta$ is often simply called SVP$_\delta$. Hermite-SVP asks to find a non-zero lattice vector whose norm is only a function of the volume of the lattice (and not the minimum distance): $||x|| \leq \delta \cdot \mathrm{vol}(\Lambda)^{\frac{1}{n}}$. The quantity $\delta^{\frac{1}{n}}$ is called the root approximation factor for Approximate-SVP$_\delta$ and the root Hermite factor for Hermite-SVP. Finally, Decisional-SVP asks whether $d(\Lambda) \leq r$ or $d(\Lambda) > r$ and GapSVP$_\delta$ (or Promise SVP$_\delta$) wether $d(\Lambda) \leq r$ or $d(\Lambda) \geq \delta \cdot r$ where $r \in \mathbb{Q}$.

Problem 3.4 (Search-CVP). *Given a real lattice Λ in $\mathbb{R}^n$ represented by a given basis and a point $y \in \mathbb{R}^n$, find the closest lattice point $x \in \Lambda$ to y.*

Similarly, this problem is often called CVP. Decisional-CVP asks if $d(y, \Lambda) \leq r$, $r \in \mathbb{Q}$. CVP$_\delta$ and GapCVP$_\delta$ are the analog of SVP$_\delta$ and GapSVP$_\delta$, respectively.

Problem 3.5 (Search BDD$_\delta$). *Given a lattice Λ represented by a given basis and a point $y \in \mathbb{R}^n$ such that $d(y, \Lambda) < \delta \cdot d(\Lambda)$, find the closest lattice point $x \in \Lambda$ to y.*

These problems have been intensively studied by both computer scientists and communications engineers. However, the hardness of these problems have been almost exclusively investigated by computer scientists. Indeed, hardness results hold for arbitrary lattices: they are worst-case hardness. Communications engineers are not interested in arbitrary lattices but in structured lattices used for a specific goal. Their "trick" is to build a family of lattices easily decodable while having the desired properties. As a rule of thumb, the more structured a lattice is, the easier to decode it is, but the less likely to have the desired property it is. Moreover, communications engineers sometimes care of non-asymptotic dimensions, which is a case not handled by complexity theory. Of course, knowing that a specific problem is hard (via a connection with another hard problem) may be a useful information in order not to waste time at looking for an unlikely efficient solution. We present hardness results for arbitrary lattices in the next subsection.

3.4.2 Hardness results

The following theorems report connections between the problems and some hardness results. Note that these theorems represent only a small subset of existing results on these problems, see e.g. Figure 3 in [LvdPdW12]. We refer to the lecture notes of Regev [Reg09] and the book [MG02] for the proofs of the results presented in this section.

Theorem 3.8 (Connections between the problems). *The following statements hold:*

- *SVP$_\delta$ is not harder[9] than CVP$_\delta$.*
- *Solving Hermite-SVP with a Hermite factor δ can be used linearly many times to solve Approximate-SVP with an approximation factor δ^2 in polynomial time.*
- *GapSVP$_\delta$ can be polynomially reduced to BDD$_{\frac{1}{\delta}\sqrt{n \log n}}$.*

Theorem 3.9 (Complexity). *The following statements hold:*

- *Search-SVP is NP-complete[10].*
- *Decisional CVP is NP-complete.*
- *GapCVP$_\delta$ is NP-hard for $\delta \leq n^{O(\frac{1}{\log \log n})}$.*
- *GapCVP$_\delta$ is not NP-hard for $\delta \geq \frac{\sqrt{n}}{\log n}$.*

As example, we provide the proof showing that Decisional-CVP is NP-hard.

[9] In other words, given oracle access to a subroutine which returns approximate closest vectors in a lattice, one may find in polynomial time approximate shortest vectors in a lattice with the level of approximation maintained.

[10] Under randomized reduction.

Proof. The proof is based on a reduction from the subset-sum problem (which is NP-hard): Given $n+1$ integers a_1,...,a_n, S, the goal is to decide whether there is a set $A \subset \{1, ..., n\}$ such that $\sum_{i \in A} a_i = S$. Given a subset-sum instance, we reduce it to a Decisional-CVP instance where $r = \sqrt{n}$ and

$$G = \begin{pmatrix} a_1 & 2 & 0 & \dots & 0 \\ a_2 & 0 & 2 & \dots & 0 \\ \vdots & \vdots & \ddots & \dots & 0 \\ a_n & 0 & \dots & 0 & 2 \end{pmatrix}, \ y = (S, 1, ..., 1). \tag{3.58}$$

First, we show that a solution for the subset-sum problem is a vector $x \in \Lambda$ such that $d(x, y) = \sqrt{n}$. Assume that there is a set $A \subseteq \{1, ..., n\}$ such that $\sum_{i \in A} a_i = S$ and consider the vector obtained by summing the row vectors of G. The first coordinate is S and the remaining ones are either 0 or 2. As a result, $d(x, y) = \sqrt{n}$. Then, we show that x is the closest vector in Λ to y. Clearly, due to the form of G, the last n coordinates of any point $x' \in \Lambda$ have to be even. This implies that $d(y, x') \geq \sqrt{n}$. Hence, x is the closest vector to y. As a result, given an oracle for Decisional-CVP, one can solve the subset-sum problem in polynomial time. □

All these problems do not admit polynomial time algorithms for an approximation factor $\delta = poly(n)$; the best known algorithms have exponential complexity. Moreover, it is conjectured that the quantum paradigm does not enable to improve (significantly) existing classical algorithms[11]. This does not mean that these algorithms are NP-hard for any $\delta = poly(n)$. They are believed to be (exponentially) hard as no efficient classical and quantum algorithms exist for these problems despite being intensively studied. Moreover, they are connected to other problems wich are also believed to be hard such as learning parity with noise and the subset sum problem.

The SVP and CVP problems are deeply related. Many algorithms for the CVP are significantly faster if they are provided with a good basis, i.e. a basis composed of relatively short vectors that are "as orthogonal as possible". Hence, Approximate-SVP is often solved as preprocessing of the CVP. And Approximate-SVP is solved using several... k-dimensional CVP ($k << n$). As a result, these algorithms are recursively calling themselves in a well determined manner. Similarly, exact and approximation algorithms (i.e. solving the problem without or with an approximation factor δ) are complementary: Approximation algorithms in high dimension use exact algorithms in lower dimension. And these exact algorithms use approximation algorithms as preprocessing.

We discuss some of the most famous algorithms for CVP and SVP in the next subsection.

3.4.3 Enumeration and basis reduction algorithms

The enumeration algorithm

The goal of this algorithm is to enumerate the lattice points located in a sphere centered at a given $y \in \mathbb{R}^n$, as illustrated in Figure 3.6a. The algorithm can be represented as a depth-first search on a tree, as shown on Figure 3.6b.

Let G be a lower triangular generator matrix of the lattice (i.e. the matrix R in (3.8)). Let $\{g_i^*\}_{i=1}^n$ be the Gram-Schmidt vectors, see (3.7). Let $y \in \mathbb{R}^n$ and $x = zG$ be a lattice point, $z \in \mathbb{Z}^n$. The enumeration algorithm outputs all vectors $z \in \mathbb{Z}^n$ such that $||zG - y||^2 < r^2$. It is based on the following representation of $||zG - y||^2$:

$$||zG - y||^2 = \sum_{j=1}^{n} \left(z_j ||g_j^*|| + \sum_{i=j+1}^{n} z_{i,j} \mu_{i,j} ||g_j^*|| - y_j \right)^2 = \sum_{j=1}^{n} ||g_j^*||^2 \left(z_j + \sum_{i=j+1}^{n} z_i \cdot \mu_{i,j} - \frac{y_j}{||g_j^*||} \right)^2. \tag{3.59}$$

This equation shows that a lattice point x is in a sphere of radius r only if the lattice point represented by the last k components of x, which is in the lattice generated by the $k \times k$ right lower part of G, is also in the sphere of radius r (of dimension k). Consequently, the paths leading to a node at the kth level of the tree on Figure 3.6b correspond to lattice points inside the sphere of radius r in dimension k. The n equations, one for each level of the tree, have the form (from Equation 3.59)

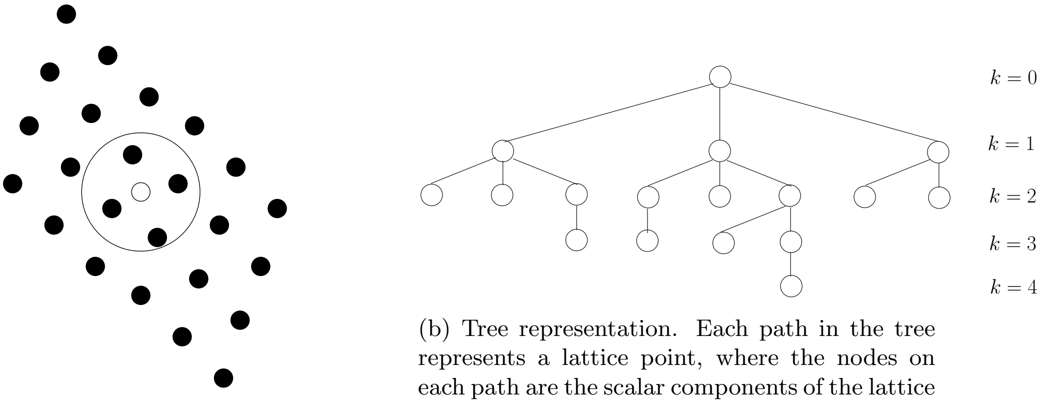
$$||g_{n-k}^*|| \cdot |z_{n-k} + \sum_{i=n-k+1}^{n} z_i \mu_{i,n-k} - \frac{y_{n-k}}{||g_{n-k}^*||}| \leq \underbrace{\sqrt{r^2 - \sum_{j=n-k+1}^{n} ||g_j^*||^2 \left(z_j + \sum_{i=j+1}^{n} z_{i,j} \mu_{i,j} - \frac{y_j}{||g_j^*||} \right)^2}}_{\nu}, \tag{3.60}$$

[11] In contrary to the problem of factoring integers, where an efficient quantum algorithm exists but no classical one.

where $0 \leq k < n$. The range for each z_{n-k} becomes

$$-\frac{\nu}{||g^*_{n-k}||} - \sum_{i=n-k+1}^{n} z_i\mu_{i,n-k} + \frac{y_{n-k}}{||g^*_{n-k}||} \leq z_{n-k} \leq \frac{\nu}{||g^*_{n-k}||} - \sum_{i=n-k+1}^{n} z_i\mu_{i,n-k} + \frac{y_{n-k}}{||g^*_{n-k}||}. \tag{3.61}$$

This last equation explains why the enumeration algorithm can be seen as enumerating the lattice points of $\mathbb{Z}^n$ in an ellipsoid.



(a) Geometrical representation. The black points represent lattice points and the white one is the point y to decode.

(b) Tree representation. Each path in the tree represents a lattice point, where the nodes on each path are the scalar components of the lattice point.

Figure 3.6: Representations of the enumeration algorithm.

Example 3.4. *Let*

$$G = \begin{pmatrix} ||g^*_{n-2}|| & 0 & 0 \\ \mu_{n-1,n-2}||g^*_{n-2}| & ||g^*_{n-1}|| & 0 \\ \mu_{n,n-2}||g^*_{n-2}|| & \mu_{n,n-1}||g^*_{n-1}|| & ||g^*_n|| \end{pmatrix} \tag{3.62}$$

Given a point $y \in \mathbb{R}^3$, the enumeration algorithm processes as follows.

- *First, the points of the projected 1-dimensional lattice generated by $G = ||g^*_n||$, located in the 1-dimensional sphere of radius r, are: $\{\hat{z}_n : (\hat{z}_n||g^*_n|| - y_n)^2 < r^2\}$. The $\{\hat{z}_n\}$ constitutes the nodes at the level $k = 1$ of the tree on Figure 3.6b.*

- *The points in the projected 2-dimensional lattice, located in the 2-dimensional sphere of radius r, are computed as: $\{\hat{z}_{n-1} : ([\hat{z}_{n-1}||g^*_{n-1}|| + \hat{z}_n\mu_{n,n-1}||g^*_{n-1}||] - y_{n-1})^2 < r^2 - (\hat{z}_n||g^*_n|| - y_n)^2\}$, for the $\hat{z}_n$ found at the previous step. The $\{\hat{z}_{n-1}\}$ constitutes the nodes at the level $k = 2$ of the tree on Figure 3.6b.*

- *Finally, the lattice points within the sphere are found by computing the last coordinate $\hat{z}_n$: $\{\hat{z}_n : ([\hat{z}_n||g^*_n|| + \hat{z}_{n-1}\mu_{n-1,n-2}||g^*_{n-2}|| + \hat{z}_{n-2}\mu_{n,n-2}||g^*_{n-3}||] - y_{n-2})^2 < r^2 - ([\hat{z}_{n-1}||g^*_{n-1}|| + \hat{z}_n\mu_{n,n-1}||g^*_{n-1}||] - y_{n-1})^2 - (\hat{z}_n||g^*_n|| - y_n)^2\}$.*

How is the tree explored ? Compute $r^2 \leftarrow r^2 - (\hat{z}_n||g^*_n|| - y_n)^2$ and recursively call the algorithm for each $\hat{z}_n$ in the range. When a leaf is reached at depth n, or all $\hat{z}_i$ have been tried at a given level, go up from one level and try another z_{i+1} in the range.

If one is interested in the closest point to y rather than enumerating all the points, the search radius r can be updated each time a point is found (i.e. a leaf is reached at depth n).

The enumeration algorithm was first presented in [Poh81] and further analyzed and improved by the same authors in [FP85]. We shall refer to their strategy as the Fincke-Pohst (FP) strategy (name of the authors). An improvement was proposed in [SE94], called the Schnorr-Euchner (SE) strategy. There are two main differences between the two strategies. The first difference lies in the way (3.61) is computed. The SE strategy performs more operations in the higher dimensions (lower part of the tree) whereas the FP strategy performs more operations in the lower dimensions. This is an advantage as some paths in the tree do not lead to a leaf of depth n. The second difference is the enumeration order: The FP strategy investigates each dimension from the lower bound on the z_i onward (given by (3.61)) whereas the SE strategy starts with the integer the closest from the median of the lower and upper bound of z_i. For instance, the components of the first point $x^* = zG$ outputted by the enumeration algorithm with the SE strategy are computed as

$$z_{n-k} = \lfloor \frac{y_{n-k}}{||g^*_{n-k}||} - \sum_{i=n-k+1}^{n} z_i \mu_{i,n-k} \rceil, \tag{3.63}$$

where $\lfloor \cdot \rceil$ denotes the round function. This point is called the Babai point. It is easily seen that the error at each level is upper bounded by half the norm $||g^*_i||$. Hence, we have:

$$d(x^*, y) \leq \frac{1}{2} \sum_i ||g^*_i||. \tag{3.64}$$

Finally, pruning techniques on the enumeration tree are often considered to speed up the enumeration. Extreme prunning, introduced in [GNR10], is often used in practice in the field of cryptography. It enables to solve the SVP for most lattices, with the enumeration algorithm and up to dimension 110, in about 62 CPU days.

Complexity analysis of the enumeration algorithm

Obviously, the enumeration algorithm is a sequential algorithm similarly to the algorithms discussed in Section 2.2.2: each path in the tree represents a lattice point. As a result, if the algorithm is used to solve the CVP, one should expect an exponential complexity in many situations (see the discussion on the cutoff rate in Section 2.2.2). We report the main existing complexity analyses of the enumeration algorithm.

To begin with, FP analyzed the complexity of their algorithm with a fixed decoding radius. They proved in [FP85] that the complexity of the enumeration algorithm with a radius r is at most[12]

$$O\left(n^2 \cdot \left(1 + \frac{n-1}{\lfloor 4r^2 d \rfloor} \right)^{\lfloor 4r^2 d \rfloor} \right), \tag{3.65}$$

where $\frac{1}{d}$ is a lower bound of $\min\{||g^*_i||^2 \; | 1 < i < n\}$.
FP suggests the following lower bound: If $||g^*_i||$ is an element of the diagonal of G, then $1/||g^*_i||$ is an element of the diagonal of G^{-1}. Hence, d can be chosen as the norm of the longest column vector[13] of G^{-1}. Note that a lower bound on $||g^*_i||$ is also the smallest singular value of G. Nevertheless, the bound (3.65) is exponential in the decoding radius. In general, the radius should increase with n: the volume of a sphere goes to zero if r^2 does not increase linearly with n (see Equation (3.11)). As a result, the Gaussian heuristic (3.23) predicts that there would be no point in the sphere if the radius does not increase with n.

[HV05] proposed a finer analysis than the one of FP for a given radius: They performed an average-case complexity analysis for lattice decoding on the unconstrained AWGN channel (see Problem 3.2), where the generator matrix G of the lattice has i.i.d. $\mathcal{N}(0, 1)$ components. The main ideas of their analysis are as follows. The complexity of the enumeration algorithm depends on the number of nodes in the explored tree. The number of nodes at a given depth corresponds to the number of lattice points in a k-dimensional sphere of radius r. Consequently, the expected complexity $\mathfrak{C}(n, \sigma^2, r)$ is the sum of the expected number of points in k-dimensional sphere of radius $r = \alpha n \sigma^2$ centered at y:

$$\mathfrak{C}(n, \sigma^2, r) = \sum_{k=1}^{n} (\text{number of operations/nodes}) \cdot E^k_{y,G}[|B^k_r(y) \cap \Lambda^k|]. \tag{3.66}$$

The probability that a lattice point $\hat{x} = \hat{z}G$ lies in a sphere centered at $y = zG + w$ is

$$\gamma\left(\frac{r^2}{\sigma^2 + ||\hat{z} - z||^2}, \frac{k}{2} \right), \tag{3.67}$$

where $\gamma(p, q)$ denotes the incomplete gamma function of argument p and q degrees of freedom. Notice that $\hat{x} - x = x' = z'G$ is another lattice point. Hence, we get

$$E_{y,G}[|B^k_r(y) \cap \Lambda^k|] = \sum_{l=0}^{\infty} \gamma\left(\frac{r^2}{\sigma^2 + l}, \frac{k}{2} \right) \cdot (\text{number of lattice points in } \mathbb{Z}^k \text{ with } ||z'||^2 = l). \tag{3.68}$$

[12] Simulations show that this bound is often pessimistic.

[13] FP also use this obervation to justify that the LLL algorithm [LLL82], discussed in the next subsection, should be applied on G^{-1} as preprocessing.

The number of lattice points in $\mathbb{Z}^k$ with $||z'||^2 = l$ is simply the coefficients of the Theta series of $\mathbb{Z}^k$, wich we call $r_k(l)$. Hence, we get

$$\mathfrak{C}(n, \sigma^2, r) = \sum_{k=1}^{n} (\text{number of operations/nodes}) \sum_{l=0}^{\infty} r_k(l) \gamma \left(\frac{r^2}{\sigma^2 + l}, \frac{k}{2} \right). \tag{3.69}$$

Unfortunately, this function is neither trivial to analyze nor to compute. Its asymptotic behavior is studied in [JO05] when each $z_i \in \{-L, L+1, ...L\}$ (in the scope of digital communications). It is shown to behave exponentially as $L^{\delta n}$, where δ decreases when the SNR increases.

Kannan was the first to analyze the enumeration algorithm to solve the CVP [Kan83], i.e. not with a fixed decoding radius. It is a worst-case analysis. The algorithm analyzed is the enumeration algorithm combined with a long preprocessing; the KZ-reduced basis is computed before running the enumeration algorithm. The whole algorithm is named Kannan's algorithm. The three main ingredients of his analysis are the following:

- The distance between the point to decode y and its closest lattice point is smaller or equal than the one with the Babai point (see (3.64)).
- The number of lattice points in the decoding sphere is upper bounded by the one in the circumscribed hypercube: I.e. in Equation (3.60), ν is not updated and has a fixed value equals to r.
- It it is possible to bound the number of lattice points in this circumbscribed hypercube given the worst-case KZ basis: I.e. the worst-case rate of decrease of the length of the Gram-Schmidt vectors.

Of course, this analysis implies knowing the cost of computing the KZ basis (and thus solving the SVP). It turns out that a similar analysis can be performed to estimate this cost: Kannan's KZ reduction algorithm involves KZ reducing the $n-1$ projected lattice, and using the enumeration algorithm at $y = 0$, where the radius of the sphere is upper bounded in this case by $\min ||g_i||$. Hence, the analysis of these two algorithms enables to upper bound the complexity of the CVP and SVP.

Recently, this analysis was improved in [HS10] where the authors managed to compute directly the number of lattice points in the decoding sphere (instead of the circumscribed hypercube). The complexity of the SVP and CVP with Kannan's algorithms are respectively $2^{O(n)} \cdot n^{\frac{n}{2e}}$ and $2^{O(n)} \cdot n^{n/2}$.

Lattice reduction algorithms: LLL and BKZ

Lattice reduction algorithms are used to solve Approximate-SVP. They are also often implemented as preprocessing algorithms in the scope of the CVP to improve the lattice basis quality.

In Equation (3.61), notice that the diagonal coefficients of G (i.e. the norm of the Gram-Schmidt vectors) are dividing coefficients in the lower and upper bound of the possible values of z_{n-k}. If these coefficients are small, the range for z_{n-k} becomes very large. Intuitively, the goal of a reduction algorithm should be to make these diagonal coefficients large. Nevertheless, the product of the diagonal coefficients $\Pi_{i=1}^{n} ||g_i^*||$ is a constant: It is the determinant of the generator matrix which is equal to the fundamental volume of the lattice. As a result, avoiding small coefficients means making them more balanced: the maximum should be smaller and the minimum larger. Assuming that the coefficients are in decreasing order, it means avoiding the decrease to be too fast.

The standard algorithm to address Approximate-SVP is the LLL algorithm, introduced in 1982 by Lenstra, Lenstra, and Lovasz [LLL82]. LLL finishes in polynomial time but achieves a (worst-case) approximation factor of $\delta = 2^{O(n)}$. Despite this bad approximation constant, the LLL tends to perform better than its worst-case performance [GN08]. E.g. the LLL often finds several shortest vectors when used in small dimensions (up to $n \approx 30$). The LLL algorithm is composed of two main subalgorithms. The first one is called size reduction. It makes the off-diagonal coefficients smaller than the coefficients on the diagonal. The second one is a swapping algorithm: It swaps two row vectors if the decrease between two succesive values on the diagonal (i.e. the norm of the Gram-Schmidt vectors) is too fast, and a new triangular matrix is then computed. The swapping operations are performed sequentially going from the top coefficient to the bottom one, several times, until a design criterion is achieved.

This algorithm was generalized in [Sch87] [SE94]. Instead of swapping two rows, the Blockwise-KZ (BKZ)-k algorithm sequentially reduces blocks of size $k \leq n$ around the diagonal of G: For all $1 < i < n$ (neglecting the boundaries) a KZ basis is computed for the k-dimensional lattice generated by the $k \times k$ submatrix around the $i, ..., i+k-1$ diagonal coefficients of G using a k-dimensional SVP solver. BKZ-k is the algorithm used in practice for Approximate-SVP in large dimensions.

BKZ does not have the best theoretical bound among reduction algorithms. In fact, its performance is in general assessed via numerical simulations or via simulation models [CN11]. In the latter case, KZ bases are not computed in dimension k but the results are estimated: For blocks of size $\gtrapprox 40$, most k-dimensional lattices behave as random lattices and it is possible to predict the length of the vectors of the KZ-reduced basis.

The approximation factor for Approximate-SVP achieved by BKZ-k diminishes with k but the runtime of BKZ-k is dominated by the complexity of the k-dimensional SVP solver. Often, the k-dimensional SVP is solved using enumeration-based algorithms. For Hermite-SVP, BKZ-k outputs a vector of norm

$$\approx (k^{\frac{1}{2k}})^n \cdot \operatorname{vol}(\Lambda)^{\frac{1}{n}} \tag{3.70}$$

in time $\approx k^{\frac{k}{2e}}$ when n is sufficiently large compared to k [ABF+20]. Using the connection between Hermite-SVP and Approximate-SVP (see Theorem 3.8), the root approximation factor is estimated as $(k^{\frac{1}{2k}})^2$ and the norm of the vector outputted by BKZ-k is $(k^{\frac{1}{k}})^n \cdot d(\Lambda)$. Notice that using the Gaussian heuristic given by (3.24), $d(\Lambda)/\operatorname{vol}(\Lambda)^{\frac{1}{n}} \approx \sqrt{\frac{n}{2\pi e}}$ and the approximation factor is simply the Hermite factor divided by $\sqrt{\frac{n}{2\pi e}}$. In general, the root approximation factor is smaller than the root Hermite factor.

To summarize, a rule of thumb is: If one pays a complexity[14] $2^{O(k)}$, the resulting approximation factor is $k^{O(n/k)}$.

Implementations of the BKZ algorithm are available via publicly available libraries such as FPLLL [dt19]. Extensive numerical simulations were performed in 2008 [GN08] to characterize the practical performance of BKZ. The authors of [GN08] made the following obervations on BKZ:

- The Hermite factor of a BKZ-reduced lattice basis does not depend on the lattice unless it has an exceptional structure.
- For BKZ-28 and BKZ-100, the root Hermite factor obtained is 1.0109 and 1.009, respectively (slightly better than $k^{\frac{1}{2k}}$). Based on these approximations and the current computing power, the authors estimated that "a root Hermite factor of 1.005 in dimension 500 looks totally out of reach".

The Darmstadt SVP challenge [MG10] enables to test the efficiency of a reduction algorithm. Currently (in 2020), lattice sieving algorithm are used for the k-dimensional SVP solver in BKZ (rather than enumeration-based algorithms) despite the memory cost that grows exponentially with the lattice dimension n [ABF+20].

3.4.4 Lattice-based cryptography

The dawn of lattice-based cryptography dates back to 1996, when Ajtai's discovered that it is possible possible to construct cryptographic schemes whose security are based on the worst-case hardness of lattice problems [Ajt96]. It means that if one manages to break the cryptographic scheme (even with a low success rate), then one can solve any instance of the underlying lattice problem. This is to be opposed with average-case hardness based cryptographic schemes. In this latter setting, one knows that the average instance of a problem is hard, but one should be careful when choosing a specific instance of the problem: It should not be easier than the average case.

The cryptographic scheme proposed by Ajtai is a family of one-way functions. However, we shall present lattice-based cryptography via a more recent problem called learning with errors (LWE). This problem was introduced by Regev in 2005 [Reg05]. Its success is mainly due to the worst-case to average-case reduction combined with the computational and memory efficiency of the associated cryptosystems[15] as well as a low decryption error rate. The worst-case to average-case reduction is based on a (both quantum and classical) reduction from worst-case GapSVP$_\delta$ to LWE, which was established in [Reg05] [Pei09] [BLP+13].

Problem 3.6 (Search-LWE). *Let $\Lambda(A)$, $A \in \mathbb{Z}^{k \times n}$, be a Construction A lattice. Let $y = zA + w$ where z is selected uniformly at random in $\mathbb{Z}_q^k$ and the error w chosen from a normal distribution with standard deviation αq (and rounded to the nearest integer). Let $\beta < \frac{d(\Lambda(A))}{2}$. The LWE problem asks to find $\hat{z} \in \mathbb{Z}_q^k$, such that $d(y, \hat{z}A) \leq \beta$.*

[14] The bound with Kannan's algorithm is $2^{O(k)} \cdot k^{\frac{k}{2e}}$ but there exists other algorithms to compute a KZ basis where the complexity is reduced but the space required increased.

[15] E.g. in the case of public key encryption, the public key size is relatively small, a low number of operations is required per encrypted bit, and the cryptosystem does not expand much each encrypted bit.

There is also a decisional version of LWE which is related to the dual lattice of $\Lambda(A)$. At first glance, search LWE seems to be hard since it is an instance of a BDD problem. The worst-case to average-case reduction states that under any discrete Gaussian error distribution (and over some non-restrictive choice of parameters), LWE is at least as hard as solving GapSVP$_\delta$. Moreover, since GapSVP$_\delta$ is as hard as $BBD_{1/\delta}$ (up to a polynomial factor, see Theorem 3.8) LWE is as hard as the hardest BDD$_{1/\delta}$ problem.

The practical security of LWE can be established based on the best known algorithm for Approximate-SPV$_\delta$. Consequently, the parameters of cryptosystems based on LWE should be chosen such that Approximate-SPV$_\delta$ is hard, i.e. based on the length of the shortest vector outputted by the best known SVP$_\delta$ algorithm :

$$\min\{q, \delta \cdot \mathrm{vol}(\delta)^{\frac{1}{n}}\} = \min\{q, \delta q^{1-\frac{k}{n}}\}, \tag{3.71}$$

where δ, the Hermite factor, is estimated with (3.70).

Remark 3.3. *An immediate consequence of these results is that lattices from the Loeliger ensemble (see Section 3.6) are as hard to decode as any lattice. In other words, it is a worst-case choice of family from a practical view point.*

3.4.5 The enumeration algorithm in digital communications

In the field of digital communications, lattices can efficiently model the communication channels encountered in diverse situations. The detection problem consists in finding the closest lattice point to the received point where the lattice depends on the channel characteristics. Consequently, versions of the enumeration algorithm are often used in practice for detection. They are called sphere decoders in this field.

Sphere decoding, based on the FP strategy, was first introduced by [VB99] in digital communications for fading channels. The semitutorial paper [AEVZ02] then presented the SE strategy [SE94]. Moreover, [Lin11] analyzes how basis reduction affects sub-optimal detection decoders and [BK98] considers Kannan's algorithm for communications. This small survey is far from being exhaustive.

An important channel where lattices are used as model is the MIMO channel. Consider a flat quasi-static MIMO channel with n transmit antennas and n receive antennas. The channel coefficients between the transmit antennas and the receive antennas are represented by a $n \times n$ matrix G. For simplicity, it is assumed that G has real entries. Any complex matrix of size $n/2$ can be transformed into a real matrix of size n with (3.3). Let $z \in \mathbb{Z}^n$ be the channel input, i.e., z is the uncoded information sequence. The input message yields the output $y \in \mathbb{R}^n$ via the following flat MIMO channel equation

$$y = \underbrace{z \cdot G}_{x} + w. \tag{3.72}$$

We call a lattice generated by a matrix G representing a MIMO channel, a MIMO lattice. Sphere decoding was introduced by [DCB00] [DGC03] in this scope. We shall investigate MIMO lattices several times in this thesis.

Part

A new framework for building and decoding group codes

Chapter 4

The k-ing groups and the single parity-check k-groups

The Leech lattice was discovered at the dawn of the communications era [Lee67]. Recently, it was proved that the Leech lattice is the densest packing of congruent spheres in 24 dimensions [CKM^{+}17]. Between these two major events, it has been subject to countless studies. This 24-dimensional lattice is exceptionally dense for its dimension and, unsurprisingly, has a remarkable structure. For instance, it contains the densest known lattices in many lower dimensions and it can be obtained in different ways from these lower dimensional lattices. In fact, finding the simplest structure for efficient decoding of the Leech lattice has become a challenge among engineers. Forney even refers to the performance of the best algorithm as a world record [For89a]. Of course, decoding the Leech lattice is not just an amusing game between engineers as it has many practical interests: Its high fundamental coding gain of 6 dB makes it a good candidate for high spectral efficiency short block length channel coding and its spherical-like Voronoi region of 16969680 facets [CS99] enables to get state-of-the-art performance for operations such as vector quantization or lattice shaping. The Leech lattice is also related to the discovery of some of the most exciting objects of pure mathematics; please, refer to [Tho83] to find how the history of simple groups is connected to this lattice.

Recently, Nebe solved a long standing open problem when she found an extremal even unimodular lattice in dimension 72 [Neb12]. The construction she used to obtain this new lattice involves the Leech lattice and Turyn's construction [AMT67] [MS77, Chap. 18, Sec 7.4]. This 72-dimensional extremal lattice (referred to as the Nebe lattice) is likely to have better property than the Leech lattice for the operations mentioned above. However, unlike the Leech lattice, its decoding aspect has not been studied much and, to the best of our knowledge, no efficient decoding algorithm is known. Moreover, none of the existing decoding algorithms for the Leech lattice seems to scale to the Nebe lattice. The primary motivation of this work was to propose a new decoder for this lattice.

In this thesis, these two lattices are presented as special instances of general constructions: We introduce a generic framework for two constructions of group codes. These families of group codes are named the k-ing groups $\Gamma(V, \alpha, \beta, k)$ and the single parity-check k-groups $\Gamma(V, \beta, k)_{\mathcal{P}}$, where $\Gamma(V, \beta, k)_{\mathcal{P}} \subseteq \Gamma(V, \alpha, \beta, k)$. In the literature, the k-ing construction was either studied for some fixed k or jointly for several k with codes. As examples, the groups $\Gamma(V, \alpha, \beta, k)$ for $k = 3$ (known as Turyn's construction) include the Leech lattice and the Nebe lattice. Regarding the single parity-check k-goups, Barnes-Wall lattices and Reed-Muller codes are part of the case $k = 2$. Nevertheless, there is no paper studying the general properties of single parity-check k-goups in the literature. The parity lattices, a new family of lattices studied in this thesis, are instances of single parity-check k-goups, recursively constructed as $\Gamma(V, \beta, k)_{\mathcal{P}}$.

This framework enables to jointly investigate the construction of many group codes, existing ones and new ones, and to present a new decoding paradigm for all of them. The paradigm can be either used for bounded-distance decoding (BDD), for list decoding, or for (quasi or exact)-maximum likelihood decoding (MLD) on the additive white Gaussian channel. For regular list decoding (i.e. enumerating all lattice points in a sphere whose radius is greater than half the minimum distance[1] of the lattice), the paradigm is combined with a technique called the splitting strategy which enables to reduce the complexity. Regarding quasi-optimal decoding on the Gaussian channel, our analysis reveals that regular

[1] In this part, we consider squared distances. Therefore, for consistency we should have stated: Greater than a *quarter* of the minimum distance.

list decoding is not the best choice with our decoding paradigm from a complexity point of view. A modified version of the regular list decoder is therefore presented. Formulas to predict the performance of these algorithms on the Gaussian channel are provided. Throughout this part of the thesis, we discuss similarities and differences between our decoding paradigm and exisiting algorithms, such as (among others) the ordered statistics decoder (OSD) [FL95], the successive-cancellation decoder of Reed-Muller codes and Polar codes [DS06] [TV15], and the iterative decoder of LDPC codes.

These new decoding algorithms uncover the performance of several lattices on the Gaussian channel. For instance, Barnes-Wall lattices, the Nebe lattice, and the 3-parity-Leech lattice (established in the thesis) are very competitive in their respective dimension: They have performance similar to known lattices whose dimension is an order of magnitude larger.

While the emphasis is put on lattices in this thesis, the main decoding paradigm is presented for arbitrary group codes. Indeed, based on our current understanding of the approach, we see no fundamental obstacle that prevents this paradigm to be extended to any group code built via the parity construction or the k-ing construction. Some examples with binary linear codes are provided at the end of the part (in Chapter 6), but they are not as developed as the ones with lattices. These studies are left for future work. Even for lattices, many cases are not covered and left for future work.

4.1 Preliminaries

Warning: In this part of the thesis, the distances $d(\cdot)$, $d(\cdot,\cdot)$, and the decoding radius r refer to squared norms when used with real lattices.

Group code and (regular) codes. A group code $\mathcal{G}^k$ is defined as in Section 3.1. All groups in this part admit a distance metric as (3.1). A (regular) code is a subspace of a vector space over a finite field. We use $d(\mathcal{G})$ to denote the minimum distance of the group $\mathcal{G}$. Let $\mathcal{G}'$ be a group with $\mathcal{G} \subset \mathcal{G}'$ and let $B_r(y)$ be a ball of radius r centered at $y \in \mathcal{G}$. The set $\mathcal{G} \cap B_r(y)$ represents the elements $x \in \mathcal{G}$ where $d(x,y) \leq r$. Let $L(\mathcal{G},r,y) = |\mathcal{G} \cap B_r(y)|$, $y \in \mathcal{G}'$ be the number of elements in the set $\mathcal{G} \cap B_r(y)$. The quantity

$$L(\mathcal{G},r) = \max_{y \in \mathcal{G}'} |\mathcal{G} \cap B_r(y)| \tag{4.1}$$

denotes the maximum number of elements in the set $\mathcal{G} \cap B_r(y)$, for any $y \in \mathcal{G}'$. In most situations it will be convenient to consider the relative radius $\delta = r/d(\mathcal{G})$, which enables to define $l(\mathcal{G},\delta,y) = L(\mathcal{G}, r = \delta d(\mathcal{G}), y)$ and $l(\mathcal{G},\delta) = L(\mathcal{G},r)$. By abuse of notations, we set $B_r(y) = B_\delta(y)$; it should be clear from the context whether the radius or relative radius is used. We also define the relative distance: $\delta(x,y) = \frac{d(x,y)}{d(\mathcal{G})}$.

Lattice. We consider lattices as a free J-module, where the possible rings J considered in this part are $\mathbb{Z}$, $\mathbb{Z}[i]$, and $\mathbb{Z}[\lambda]$, $\lambda = \frac{1+i\sqrt{7}}{2}$. We recall that given a complex lattice $\Lambda^{\mathbb{C}}$ with generator matrix $G^{\mathbb{C}}$, the lattice generated by $\theta \cdot G^{\mathbb{C}}$ is denoted $\theta\Lambda^{\mathbb{C}}$, $\theta \in J$. Let Λ, with generator matrix G, be the real lattice obtained via (3.3) from the complex lattice $\Lambda^{\mathbb{C}}$. The real version of $\theta\Lambda^{\mathbb{C}}$ can be either obtained using (3.3) on $\theta \cdot G^{\mathbb{C}}$ or from G as follows. Let $R(2,\theta)$ be the 2×2 matrix obtained from θ via (3.3), e.g.

$$R(2,\lambda) = \begin{bmatrix} 1/2 & \sqrt{7}/2 \\ -\sqrt{7}/2 & 1/2 \end{bmatrix} \text{ and } R(2,\phi) = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}. \tag{4.2}$$

where $\phi = 1+i$. The scaling-rotation operator $R(n,\theta)$ in even dimension n is defined by the application of $R(2,\theta)$ on each pair of components. I.e. the scaling-rotation operator is $R(n,\theta) = I_{n/2} \otimes R(2,\theta)$, where I_n is the $n \times n$ identity matrix and $\otimes$ the Kronecker product. Then, the real version of $\theta\Lambda^{\mathbb{C}}$ is generated by $G \cdot R(n,\theta)$. We name it either $R_\theta\Lambda$ or $\theta\Lambda$.

We say that an integral lattice (i.e. the Gram matrix has integer entries) is even if $\|x\|^2$ is even for any x in Λ. Moreover, an integral lattice with $\text{vol}(\Lambda) = 1$ is called unimodular or self-dual lattice. The following Johnson-type bound on the list size for arbitrary lattices is proved in [MG02, Chapter. 5].

Theorem 4.1. *Let Λ be a lattice in $\mathbb{R}^n$. The list size $L(\Lambda,r)$, defined by* (4.1)*, is bounded as:*

- $L(\Lambda,r) \leq \frac{1}{2\epsilon}$ *if* $r \leq d(\Lambda)(1/2-\epsilon)$, $0 < \epsilon \leq 1/4$.
- $L(\Lambda,r) \leq 2n$ *if* $r = d(\Lambda)/2$.

Let $\Lambda_n \in \mathbb{R}^n$ be part of a family of lattices with instances in several dimensions n. If we want to specify the list size for the lattice in a given dimension n, we simplify the notations as follows: We let $L(n,r) = L(\Lambda_n, r)$ and $l(n,\delta) = l(\Lambda_n, \delta)$.

Complexity analysis. The complexity of the proposed algorithms is denoted by $\mathfrak{C}$ or $\mathfrak{C}_{A.i}$, where i represents the number of the algorithm. The decoding complexity of a group code $\mathcal{G}$ is expressed as $\mathfrak{C}(\mathcal{G})$, where the considered decoding technique is clear from the context.

In general, $\mathfrak{C}$ denotes the worst-case running time. By abuse of notation, we use equalities (e.g. $\mathfrak{C} = X$) even though we only provide upper-bounds on the worst-case running time. We adopt this approach to characterize the complexity of the proposed algorithms, which does not take into account the position of the point y to be decoded. However, to assess the complexity of the algorithms on the Gaussian channel, we take advantage of the distribution of the point y to be decoded and assess the average complexity $E_y[\mathfrak{C}]$ (warning: $E_y[\mathfrak{C}]$ does not denote the average worst-case complexity but the average complexity).

The complexity of decoding in a group $\mathcal{G}$ with a specific decoder is denoted by $\mathfrak{C}^{\mathcal{G}}_{dec}$, where "dec" should be replaced by the name of the decoder: E.g. if $\mathcal{G}$ is a lattice Λ, the complexity of BDD, optimal decoding, MLD, and quasi-MLD are $\mathfrak{C}^{\Lambda}_{BDD}$, $\mathfrak{C}^{\Lambda}_{opt}$, $\mathfrak{C}^{\Lambda}_{MLD}$, $\mathfrak{C}^{\Lambda}_{QMLD}$, respectively. Moreover, we denote by $\mathfrak{C}_{\mathcal{G}\cap B_\delta(y)}$, $\mathfrak{C}^{\mathcal{G}}_{stor.}$, and $\mathfrak{C}_{clos.}(n)$, the complexity of computing the set $\mathcal{G} \cap B_\delta(y)$, storing an element belonging to $\mathcal{G}$, and finding the closest element to y among n elements, respectively. If not specified, the set $\Lambda \cap B_\delta(y)$ can be computed via the sphere decoding algorithm [VB99]. In this case $\mathfrak{C}_{\Lambda\cap B_\delta(y)} = \mathfrak{C}^{\Lambda}_{Sph.dec.,\delta}$.

In general, we assume that $\mathfrak{C}^{\mathcal{G}}_{dec} >> \mathfrak{C}^{\mathcal{G}}_{stor.}$ and that $k\mathfrak{C}^{\mathcal{G}}_{dec} >> \mathfrak{C}_{clos.}(k)$. Hence, we have

$$k\mathfrak{C}^{\mathcal{G}}_{dec} + k\mathfrak{C}^{\mathcal{G}}_{stor.} + \mathfrak{C}_{clos.}(k) \approx k\mathfrak{C}^{\mathcal{G}}_{dec}. \tag{4.3}$$

Similarly, we also have:

$$\mathfrak{C}_{\mathcal{G}\cap B_\delta(y)} + l(\mathcal{G},\delta)\mathfrak{C}^{\mathcal{G}}_{stor.} + \mathfrak{C}_{clos.}(l(\mathcal{G},\delta)) \approx \mathfrak{C}_{\mathcal{G}\cap B_\delta(y)}. \tag{4.4}$$

By abuse of notations, we may write $k\mathfrak{C}^{\mathcal{G}}_{dec} + k\mathfrak{C}^{\mathcal{G}}_{stor.} + \mathfrak{C}_{clos.}(k) = k\mathfrak{C}^{\mathcal{G}}_{dec}$ (e.g. if $\mathcal{G} \in \mathbb{R}^{\frac{n}{k}}$, we sometimes write $k\mathfrak{C}^{\mathcal{G}}_{dec} + O(n) = k\mathfrak{C}^{\mathcal{G}}_{dec}$ if the $O(n)$ is not relevant in the context).

When recursively decoding a lattice $\Lambda_n \in \mathbb{R}^n$, we simplify the notation $\mathfrak{C}(\Lambda_n, \delta)$ by writing $\mathfrak{C}(n,\delta)$.

The $\widetilde{O}$ notations is used to ignore logarithmic factors. The notation $f(n) = \widetilde{O}(h(n))$ is equivalent to: $\exists k$ such that $f(n) = O(h(n)\log^k(h(n)))$ (since $\log^k(n) = O(n^\epsilon)$ for any $\epsilon > 0$).

The complexity exponent is defined as $\frac{\log \mathfrak{C}(n,\delta)}{\log n}$. Finally, when analyzing the number of real operations performed by an algorithm we use the unit *flop*, where 1 flop corresponds to 1 addition/subtraction/division/multiplication.

Extremal lattice. The fundamental coding gain of an even unimodular lattice of dimension n is at most $2\lfloor \frac{n}{24} \rfloor + 2$. Lattices achieving this coding gain are called extremal.

4.2 Construction of group codes

We begin by defining two group codes of length k over an arbitrary discrete group $\mathcal{G}$. We have the repetition code

$$(k,1)_{\mathcal{G}} = \{(m_1, m_2, ..., m_k), m_1 = m_2 = ... = m_k \in \mathcal{G}\}, \tag{4.5}$$

and the single parity-check code

$$(k, k-1)^{\mathcal{G}'}_{\mathcal{G}} = \{(n_1, n_2, ..., n_k), n_i \in \mathcal{G}, \sum_{i=1}^{k} n_i \in \mathcal{G}'\}, \tag{4.6}$$

where $\mathcal{G}' \subset \mathcal{G}$. $\mathcal{G}' = \{0\}$ corresponds to the standard parity check.

Definition 4.1. *Consider discrete Abelian groups S, T, V, where $V \subset T \subset S$. Let us denote $\alpha = [S/T]$ and $\beta = [T/V]$, two groups of coset representatives. The k-ing construction of a group Γ is defined as*

$$\Gamma(V, \alpha, \beta, k) = V^k + (k, k-1)^{\{0\}}_{\beta} + (k,1)_{\alpha}. \tag{4.7}$$

Note the this k-ing construction can be seen as a non-binary instance of Construction B **??**. Since any coordinate of V^k can be decomposed as the sum of two elements of V where the first element is constant over all coordinates, (4.7) can be re-written as

$$\begin{aligned}\Gamma(V,\alpha,\beta,k) =&\{(m'+n'_1, m'+n'_2, ..., m'+n'_k), m' \in V+\alpha, n'_i \in V+\beta, \sum_{i=1}^{k} n'_i \in V\},\\ =&\{(m'+t_1, m'+t_2, ..., m'+t_k), m' \in V+\alpha, t_i \in T, \sum_{i=1}^{k} t_i \in V\},\end{aligned} \tag{4.8}$$

since $V+\beta$ is the group T, regardless of the choice of coset representatives β. We denote the group $V+\alpha$ by T^*. The k-ing construction has the following form using T^* and T instead of α and β:

$$\Gamma(V,T^*,T,k) = V^k + (k,k-1)_T^V + (k,1)_{T^*} = \Gamma(V,\alpha,\beta,k). \tag{4.9}$$

Figure 4.1 illustrates the relationships between S, T, T^*, V, α, and β.

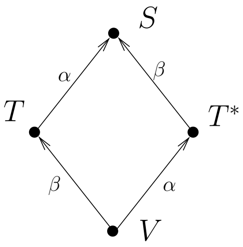


Figure 4.1: Relationships between S, T, T^*, V, α, and β.

Two obvious subgroups of $\Gamma(V,\alpha,\beta,k)$ are:

- The single parity-check k-group in T^k

$$\begin{aligned}\Gamma(V,\beta,k)_{\mathcal{P}} =&V^k + (k,k-1)_{\beta}^{\{0\}},\\ =&\{(v_1+n_1, v_2+n_2, ..., v_k - \sum_{i\neq k} n_i), v_i \in V,\ n_i \in \beta\},\\ =&\{(t_1, t_2, ..., v'_k - \sum_{i\neq k} t_i), v'_k \in V,\ t_i \in T\},\end{aligned} \tag{4.10}$$

  where the last expression is the most useful in practice. This group code can be viewed as a generalized Construction A group [FV96], since a code is applied on the quotient group T/V.

- The repetition k-group in ${T^*}^k$

$$\Gamma(V,\alpha,k)_{\mathcal{R}} := V^k + (k,1)_{\alpha}. \tag{4.11}$$

Using these subgroups, we introduce two alternative representations of the k-ing construction, which will be useful for decoding. First, we have

$$\begin{aligned}\Gamma(V,\alpha,\beta,k) &= \Gamma(\Gamma(V,\beta,k)_{\mathcal{P}},\alpha,k)_{\mathcal{R}},\\ &= \bigcup_{m\in\alpha}\{\Gamma(V,\beta,k)_{\mathcal{P}} + m^k\},\end{aligned} \tag{4.12}$$

where m^k is $(m, ..., m)$.
Second, we present $\Gamma(V,\alpha,\beta,k)$ as a parity-check-like group. The following equation shows that the k-ing construction can be seen as a parity-check k-group since p_k can be recovered from $p_1, ..., p_{k-1}$, but "enhanced" as an additional constraint is put on the $p_{i\neq k}$: All m's should be equal in the decomposition $p_i = m + n_i$.

$$\begin{aligned}\Gamma(V,\alpha,\beta,k) =&\{(v_1+p_1, v_2+p_2, ..., v_k+p_k),\\ &v_i \in V,\ p_{i\neq k} = m + n_i \in [S/V],\ p_k = m - \sum_{i\neq k} n_i, m \in \alpha, n_i \in \beta\}.\end{aligned} \tag{4.13}$$

Regarding the notations, following (4.8) and (4.13), the letters m, n, and p are used to denote elements of the quotient groups α, β, and S/V respectively. The letters m', n', and p' denotes elements of $V+\alpha$, $V+\beta$, and $V+S/V$ respectively.

With (4.10), we easily see that $d(\Gamma(V,\beta,k)_{\mathcal{P}}) = \min\{d(V), 2d(T)\}$. The minimum distance of $\Gamma(V,\alpha,\beta,k)$ is provided by the next theorem.

Theorem 4.2. *The minimum distance of* $\Gamma(V, \alpha, \beta, k)$ *satisfies*

$$\min\{d(V), 2d(T), kd(S)\} \leq d(\Gamma(V, \alpha, \beta, k)) \leq \min\{d(V), 2d(T)\}. \tag{4.14}$$

Proof. Using (4.7), x can be decomposed as $x = v + c_1 + c_2$ with $v \in V^k, c_1 \in (k, 1)_\alpha, c_2 \in (k, k-1)_\beta^{\{0\}}$. There are two cases to be addressed:

1. $c_1 = 0^k$. Then, $x \in \Gamma(V, \beta, k)_\mathcal{P}$ where $d(\Gamma(V, \beta, k)_\mathcal{P}) = \min\{d(V), 2d(T)\}$ from (4.10).
2. $c_1 \neq 0^k$. Then, given $x = (x_1, ..., x_k)$, every $x_i \neq 0$ because V, α, and β have 0 as the unique common element. Considering $x_i \in S$ all of minimum squared norm, we get $d(\Gamma(V, \alpha, \beta, k)) \geq kd(S)$.

□

Theorem 4.2 is not fundamentally new: It was presented for several fixed k in [For88b] for group codes and for arbitrary k in [DB04], but only in the scope of codes.

The main idea behind the k-ing construction is to find groups such that $d(V) \approx 2d(T) \approx kd(S)$. E.g. if $d(V) = 4d(S)$, it is pointless to choose a large $k >> 4$: the bottleneck on $d(\Gamma(V, \alpha, \beta, k))$ would be $d(V)$ regardless of the value $kd(S)$.

When using Theorem 4.2, there are two main cases to consider:

- First, the case $d(V) = 2d(T) = kd(S)$. Then, the inequalities in Theorem 4.2 become equalities.
- Second, the case $d(V) = 2d(T) > kd(S)$. Then, Theorem 4.2 yields $d(\Gamma(V, \alpha, \beta, k)) \geq kd(S)$. Can we get $d(\Gamma(V, \alpha, \beta, k)) > kd(S)$? The degree of freedom, in order to achieve this inequality, is the choice of coset representatives α (or equivalently T^*). In some cases, a technique called polarisation [Neb12] enables to choose T^* (i.e. α) such that $d(\Gamma(V, \alpha, \beta, k)) > kd(S)$.

As we shall see in the sequel, a lot of famous lattices and codes can be obtained via the k-ing construction and single parity-check k-groups, including:

- The Leech lattice and the Nebe lattice.
- The Reed-Muller codes, the hexacode, and the Golay code.
- The Barnes-Wall lattices.
- Many quasi-cyclic codes (see [LS01]).
- New interesting group codes (presented in this thesis).

Most papers study the k-ing construction for a fixed k. As a result, various names exist for this construction given a fixed k:

- If $k = 3$, it is Turyn's construction [MS77, Chap. 18, sec 7.4] (also known as the cubing construction [For88b]).
- If $k = 4$, it is the two-level squaring construction [For88b].
- If $k = 5$, it is the quinting construction [LS01].
- If $k = 7$, it is the septing construction [LS01].

Note also that $\Gamma(V, \beta, 2)_\mathcal{R} \subset T^2$ is the squaring construction [For88b] (equivalent to the Plotkin $(u, u+v)$ construction [Plo60]).

Some examples. While α and β are always finite groups, in these examples we show that S, T, V can be either finite (i.e. with codes) or infinite (i.e. with lattices).

1. Take $V = 2\mathbb{Z}$ and $T = \mathbb{Z}$. Then, $\beta = [\mathbb{Z}/2\mathbb{Z}]$. The checkerboard lattice is $D_n = \Gamma(V, \beta, n)_\mathcal{P}$.
2. Take $S = (\mathbb{Z}/2\mathbb{Z})^2$, $T = (2,1)_{\mathbb{Z}/2\mathbb{Z}}$ (the binary repetition code), and $V = \{0^2\}$. Then, $\alpha = [(\mathbb{Z}/2\mathbb{Z})^2/(2,1)_{\mathbb{Z}/2\mathbb{Z}}]$, $\beta = [(2,1)_{\mathbb{Z}/2\mathbb{Z}}/0^2]$. The first order Reed-Muller code of length 8, $RM(1,3)$, is $\Gamma(V, \alpha, \beta, 4)$. More generally, any Reed-Muller code of length 2^m is obtained as $RM(r,m) = \Gamma(V, \alpha, \beta, 4)$, where $V = RM(r-2, m-2)$, the first quotient group is $\alpha = [RM(r, m-2)/RM(r-1, m-2)]$, and the second quotient group is $\beta = [RM(r-1, m-2)/RM(r-2, m-2)]$ [For88b]. These codes can also be obtained as single parity-check group. See (6.32).

4.3 Decoding algorithms

In this section, we introduce the main decoding algorithms. The decoding paradigm is first presented in Subsection 4.3.2. We then explain how to combine this paradigm with list decoding, with and without the splitting strategy, in the following subsection. A detailed complexity analysis is also provided.

4.3.1 Existing decoding algorithm for $\Gamma(V,\alpha,\beta,k)$ (and $\Gamma(V,\beta,k)_{\mathcal{P}}$)

To the best of the author's knowledge, there exists only one "efficient" optimal algorithm for the k-ing construction called trellis decoding. This decoding algorithm uses a graph-based representation of (4.7) to efficiently explore all the cosets of V^k in $\Gamma(V,\alpha,\beta,k)$. As an example, the trellis of a $\Gamma(V,\alpha,\beta,3)$ group is illustrated on Figure 4.2 with $|\alpha| = 3$ and $|\beta| = 2$. Each path in this three sections trellis corresponds to a coset of V^3 in $\Gamma(V,\alpha,\beta,3)$. Each edge is associated with a coset of V in S: E.g. given $\alpha = \{m_1, m_2, m_3\}$ and $\beta = \{n_1, n_2\}$, the two upper edges on the left should be labeled $m_1 + n_1$ and $m_1 + n_2$, respectively. All the edges in the upper part of the trellis correspond to the same m_1 and the sub-trellis formed by these edges is a standard single parity-check trellis. This sub-trellis is repeated three times for m_1,m_2, and m_3. Standard trellis algorithms, such as the Viterbi algorithm, can then be used to decode the group.

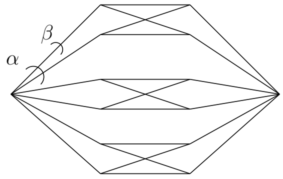


Figure 4.2: Trellis representing a $\Gamma(V,\alpha,\beta,3)$ with $|\alpha| = 3$ and $|\beta| = 2$. The edges labeled with the same α are associated with the same $m_1 \in \alpha$.

Trellis decoding of $\Gamma(V,\alpha,\beta,k)$ involves decoding in V for each edge in the trellis. The number of edges in the trellis is:

$$2|\alpha||\beta| + (k-2)|\alpha||\beta|^2. \tag{4.15}$$

Therefore, the complexity is dominated by the quantity $|\alpha||\beta|^2\mathfrak{C}^V_{dec}$. For more information on trellis decoding of group codes, the reader should refer to [For88b] or [DBNS08].

Of course, trellis decoding can also be used to decode the single parity-check k-group $\Gamma(V,\beta,k)_{\mathcal{P}}$. The number of edges in the standard single parity-check trellis is $2|\beta| + (k-2)|\beta|^2$.

4.3.2 Decoding paradigm for $\Gamma(V,\beta,k)_{\mathcal{P}}$ and $\Gamma(V,\alpha,\beta,k)$

Let $\mathcal{G}$ be a group with $V \subset T \subset S \subset \mathcal{G}$. Set $y \in \mathcal{G}^k$ and let $x = (t_1, t_2, ..., t_k) \in \Gamma(V,\beta,k)_{\mathcal{P}}$ be the closest element to y (with respect to $d(\cdot,\cdot)$).

The minimum distance of V is (in general) larger than the one of T. Hence, decoding y_j in the coset of V to which the element t_j belongs is safer than decoding in T. Moreover, remember that $\Gamma(V,\beta,k)_{\mathcal{P}}$ is a Construction A over V^k with a single parity-check code over a $\mathcal{Q}$-ary alphabet, where $\mathcal{Q} = |\beta|$. Therefore, any set of $k-1$ t_j's is enough to know in which coset of V^k in $\Gamma(V,\beta,k)_{\mathcal{P}}$ is located x. Hence, given $t_1, t_2, ...t_{k-1}$, the element t_k can be recovered by decoding $y_k - (-\sum_{j=1}^{k-1} t_j)$ in V (and adding back $-\sum_{j=1}^{k-1} t_j$ on the decoded element), as shown by Algorithm 4.1. All algorithms presented in this part of the thesis are variations of Algorithm 4.1.

It is easily seen that the complexity of Algorithm 4.1 is

$$\mathfrak{C}_{A.4.1} = k\mathfrak{C}^T_{dec} + k\mathfrak{C}^V_{dec}, \tag{4.16}$$

where we used the simplification of Equation (4.3).

Then, we introduce two algorithms to decode $\Gamma(V,\alpha,\beta,k)$. The first algorithm uses representation (4.12) of the k-ing construction: $\Gamma(V,\alpha,\beta,k)$ is decoded via $|\alpha|$ use of Algorithm 4.1, as described in Algorithm 4.2. The second algorithm avoids the investigation of the $|\alpha|$ cosets of $\Gamma(V,\beta,3)_{\mathcal{P}}$ while keeping a similar strategy: Algorithm 4.1 is slightly modified to address the additional constraints of an "enhanced" parity check (see (4.13)). This yields Algorithm 4.3. Note that the first step of this latter

algorithm involves decoding in S instead of T.
The complexity of Algorithm 4.2 is

$$\mathfrak{C}_{A.4.2} = |\alpha|(k\mathfrak{C}^{T}_{dec} + k\mathfrak{C}^{V}_{dec}), \tag{4.17}$$

and the one of Algorithm 4.3 is

$$\mathfrak{C}_{A.4.3} = k\mathfrak{C}^{S}_{dec} + k\mathfrak{C}^{V}_{dec}. \tag{4.18}$$

Algorithm 4.1 Decoder for $\Gamma(V, \beta, k)_{\mathcal{P}}$

Input: $y = (y_1, y_2, ..., y_k) \in \mathcal{G}^k$.

```
Decode $y_1, y_2, ..., y_k$ in $T$ as $t_1, t_2, ..., t_k$.
for $1 \leq i \leq k$ do
    Decode $y_i - (-\sum_{j \neq i} t_j)$ in $V$ as $v_i$.
    Add $(t_1, ..., t_{i-1}, v_i + (-\sum_{j \neq i} t_j), t_{i+1}, ..., t_k)$ to the list $\mathcal{T}$.
end for
Return the closest element of $\mathcal{T}$ to $y$.
```

Algorithm 4.2 Decoder 1 for $\Gamma(V, \alpha, \beta, k)$

Input: $y = (y_1, y_2, ..., y_k) \in \mathcal{G}^k$.

```
for $m \in \alpha$ do
    $y' \leftarrow y - m^k$
    Decode $y'_1, y'_2, ..., y'_k$ in $T$ as $t_1, t_2, ..., t_k$.
    for $1 \leq i \leq k$ do
        Decode $y'_i - (-\sum_{j \neq i} t_j)$ in $V$ as $v_i$.
        Add $(t_1, ..., t_{i-1}, v_i + (-\sum_{j \neq i} t_j), t_{i+1}, ..., t_k)$ to the list $\mathcal{T}$.
    end for
end for
Return the closest element of $\mathcal{T}$ to $y$.
```

Algorithm 4.3 Decoder 2 for $\Gamma(V, \alpha, \beta, k)$

Input: $y = (y_1, y_2, ..., y_k) \in \mathcal{G}^k$.

```
Decode $y_1, y_2, ..., y_k$ in $S$ as $t_1, t_2, ..., t_k$.
Decompose each $t_j$ as $t_j = v_j + m_j + n_j$, $v_j \in V$, $m_j \in \alpha$, $n_j \in \beta$.
for $1 \leq i \leq k$ do
    if $m_1 = ... = m_{j \neq i} = ... = m_k$ then
        Compute $p_i = m_j - \sum_{j \neq i} n_j$.
        Decode $y_i - p_i$ in $V$ as $v_i$. Add $(t_1, ..., t_{i-1}, v_i + p_i, t_{i+1}, ..., t_k)$ to the list $\mathcal{T}$.
    end if
end for
Return the closest element of $\mathcal{T}$ to $y$ or declare failure if $\mathcal{T}$ is empty.
```

Similarities with some existing algorithms. The decoding paradigm of these algorithms is similar to the OSD [FL95]. The main idea of the OSD, to decode a (n, k) binary code, is to use the k most reliable bits to re-encode the codeword and correct errors. The order of the OSD refers to the number of bits that are flipped among the k most reliable bits: E.g. in the case of binary codes, one candidate is generated with the order 0, $\binom{k}{1}$ candidates with the order 1, $\binom{k}{2}$ candidates with the order 2, etc... For Algorithm 4.1, $k-1$ coordinates are used to "re-encode" (here it means finding the coset of V) the remaining coordinate via the parity constraint. Unlike the regular OSD, we do not know the $k-1$ most reliable coordinates. Therefore, all k possibilities are considered: We use k instances of the OSD of order 0.

The paradigm also presents similarities with the algorithm of [SA06]. In this paper, binary lattices, i.e. lattices that are subset of a 2^m-scaled version of $\mathbb{Z}^n$, are constructed by employing the Kronecker product. They consider for instance the lattice $D_4 \otimes D_4$, which is a sublattice of $D_4 \oplus D_4 \oplus D_4 \oplus D_4$. Let $x = (a, b, c, d) \in D_4 \otimes D_4$. Then, $a, b, c, d \in D_4$. The main observation made in [SA06] for decoding is that from any set of three blocks of coordinates (e.g. $\{a, b, c\}$), it is "simple" to decode the whole lattice point (in the paper this is called "decodability" under "subset constraint"). This is the same idea as the one exploited by Algorithm 4.1, where given any set of $k-1$ t_j it is simple to decode the lattice point.

4.3.3 List decoding with and without the splitting strategy

We recall that regular list decoding consists in computing the set $\Gamma(V, \beta, k)_{\mathcal{P}} \cap B_\delta(y)$: i.e. finding all lattice points $x \in \Gamma(V, \beta, k)_{\mathcal{P}}$ where $d(y, x) \leq r$, $y \in \mathbb{R}^{kn}$. The parameter $\delta = r/d(\Gamma(V, \beta, k)_{\mathcal{P}})$ is the relative decoding radius.

For the sake of simplicity, we present list decoding with and without the splitting strategy in the scope of lattices and we assume that $V \cong \sqrt{2}T$: The groups T, V, and thus $\Gamma(V, \beta, k)_{\mathcal{P}}$ are lattices in $\mathbb{R}^n$ and $\mathbb{R}^{kn}$, respectively. We do not consider list decoding beyond the minimum distance, i.e. $\delta < 1$.

We now explain how Algorithm 4.1 can be adapted to list decoding. We shall see that the splitting strategy is a useful "trick" to reduce the complexity of list decoding without the splitting strategy, where this latter version is the natural generalization of Algorithm 4.1.

Without the splitting strategy

Remember that $d(\Gamma(V, \beta, k)_{\mathcal{P}}) = d(V) = 2d(T)$. List decoding $\Gamma(V, \beta, k)_{\mathcal{P}}$ with a radius r without the splitting strategy consists in list decoding each y_j (Step 1 of Algorithm 4.1) in T with a radius $r/2$ and each $y_i - (-\sum_{j \neq i} t_j)$ (Step 3) in V with a radius r. In both cases the relative radius is $\delta = \frac{r}{2d(T)} = \frac{r}{d(V)}$ and the maximum number of elements in each list is $l(T, \delta) = l(V, \delta) = L(T, \frac{r}{2}) = L(V, r)$ (see Section 4.1 for the definitions of $L(\cdot, \cdot)$ and $l(\cdot, \cdot)$). As a result, Step 3 (of Algorithm 4.1), for a given i, should be executed for any of the combinations of candidates (for each $t_{j \neq i}$) in the $k-1$ lists: I.e. $l(T, \delta)^{k-1}$ times. The resulting maximum number of stored elements (for this given i) is $l(T, \delta)^{k-1} \cdot l(V, \delta)$. Consequently, the number of elements in $\mathcal{T}$ is bounded from above by

$$k \cdot l(T, \delta)^{k-1} \cdot l(V, \delta) = k \cdot l(T, \delta)^k. \tag{4.19}$$

This list decoding version of Algorithm 4.4 is presented in Algorithm 4.4.

Algorithm 4.4 List dec. for $\Gamma(V, \beta, k)_{\mathcal{P}} \in \mathbb{R}^{kn}$ **without the splitting strategy**

Input: $y = (y_1, y_2, ..., y_k) \in \mathbb{R}^{kn}$, $0 \leq \delta$.

```
 1: Compute the sets $\mathcal{T}_1 = T \cap B_\delta(y_1), \mathcal{T}_2 = T \cap B_\delta(y_2), ..., \mathcal{T}_k = T \cap B_\delta(y_k)$.
 2: for $1 \leq i \leq k$ do
 3:   Set $j_1 < j_2 < ... < j_{k-1}$, where $j_1, j_2, ..., j_{k-1} \in \{1, 2, ..., k\} \backslash \{i\}$.
 4:   for each $(t_{j_1}, ..., t_{j_{k-1}}) \in \mathcal{T}_{j_1} \times \mathcal{T}_{j_2} \times ... \times \mathcal{T}_{j_{k-1}}$ do
 5:     Compute the set $\mathcal{V}_i = V \cap B_\delta(y_i - (-\sum_{j'} t_{j_{j'}}))$.
 6:     for $v_i \in \mathcal{V}_i$ do
 7:       Add $(t_{j_1}, ..., t_{j_{i-1}}, v_i + (-\sum_{j'} t_{j_{j'}}), t_{j_i}, ..., t_{j_{k-1}})$ to the list $\mathcal{T}$.
 8:     end for
 9:   end for
10: end for
11: Return $\mathcal{T}$.
```

Lemma 4.1 (Complexity without the splitting strategy)**.** *Algorithm 4.4 outputs the set* $\Gamma(V, \beta, k)_{\mathcal{P}} \cap B_\delta(y)$ *in worst-case time*

$$\mathfrak{C}_{A.4.4} = k\mathfrak{C}_{T \cap B_\delta(y)} + k \cdot l(T, \delta)^{k-1} \mathfrak{C}_{V \cap B_\delta(y)}. \tag{4.20}$$

Proof. We first prove that all points $x = (x_1, x_2, ..., x_k) \in \Gamma(V, \beta, k)_{\mathcal{P}} \cap B_\delta(y)$ are outputted by Algorithm 4.4. If $d(y_i, x_i) > r/2$ then $d(y_j, x_j) < r/2$ for all $j \neq i$. Hence, among the k lists $\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_k$ computed at Step 1 of the algorithm, at least $k-1$ of them contain the correct $t_j^* = x_j$.

Assume (without loss of generality) that all $\mathcal{T}_j$, $1 \leq j \neq i \leq n$, contain t_j^*. Since $d(y,x) \leq r$, one has $d(y_i, x_i) \leq r$. Therefore, $\mathcal{V}_i = V \cap B_\delta(y_i - (-\sum_{j \neq i} t_j^*))$ contains $v_i^* = x_i - (-\sum_{j \neq i} t_j^*)$.
As a result, all $x \in \Gamma(V, \beta, k)_\mathcal{P} \cap B_\delta(y)$ are outputted by the algorithm.
The complexity is obtained by reading Algorithm 4.4, with the simplification of Equation (4.3). □

Note that if $\delta < \frac{1}{4}$ (the relative packing radius), there is only one element in each of the set computed in Algorithm 4.4. Algorithm 4.4 is in this case equivalent to Algorithm 4.1 (adapted to lattices).

Similarities with the OSD. We established in Section 4.3.2 similarities between Algorithm 4.1 and the OSD of order 0. With Algorithm 4.4, lists of candidates are generated for the $k-1$ coordinates used for "re-encoding". This algorithm can thus be seen as k instances of the OSD of order $k-1$. Note that if all possibilities were used to generate each list of candidates (i.e. trying all $|\beta|^{k-1}$ cosets to compute $\sum_{j'} t_{j_{j'}}$ at Step 5 and then optimally decoding in V), the algorithm would be optimal but with a much higher complexity: $\approx |\beta|^{k-1}\mathfrak{C}_{opt}^V$ instead of $\approx k \cdot l(T,\delta)^{k-1}\mathfrak{C}_{opt}^V$.

Moreover, the "order" of Algorithm 4.4 could be reduced: At Step 4 of the algorithm, the combinations with only ν lists could be considered (and the $k-1-\nu$ remaining elements could be BDD lattice points), where this operation would be repeated $\binom{k}{\nu}$ times. The maximum number of outputted element would be

$$k \cdot \binom{k}{\nu} \cdot l(T,\delta)^\nu. \tag{4.21}$$

We did not study this latter approach. This is left for future work. Nevertheless, the splitting strategy can be seen as a method to reduce the order of Algorithm 4.4.

With the splitting strategy

The complexity of Algorithm 4.4 can be reduced via two splitting strategies. The first splitting strategy exploits the following observation: Assume (without loss of generality) that $d(y_i, x_i) > \frac{r}{2}$ (and thus $\sum_{j \neq i} d(x_j, y_j) \leq \frac{r}{2}$). This case can be split into two sub-cases. Let $0 \leq a' \leq \frac{r}{2}$.

- If $a' \leq \sum_{j \neq i} d(x_j, y_j) \leq \frac{r}{2}$ then $\frac{r}{2} < d(x_i, y_i) \leq r - a'$:
  Then, each y_j should be list decoded in T with a radius $\frac{r}{2}$ and $y_i - (-\sum_{j \neq i} t_j)$, for all resulting combinations of t_j, list decoded in V with a radius $r - a'$.
- Else $0 \leq \sum_{j \neq i} d(x_j, y_j) < a'$ and $\frac{r}{2} < d(x_i, y_i) \leq r$:
  Then, each y_j should be list decoded in T with a radius a', and $y_i - (-\sum_{j \neq i} t_j)$, for all resulting combinations of t_j, list decoded in V with a radius r.

These two sub-cases are illustrated in Figure 4.3. The number of stored elements (when computing each sub-case) is bounded by

- $l(T,\delta)^{k-1} \cdot l(V, a_1 = \frac{r-a'}{d(V)})$, for the first sub-case,
- $l(T, a_2 = \frac{a'}{d(T)})^{k-1} \cdot l(V,\delta)$ for the second sub-case.

Consequently, if we choose $a_1 = a_2 = \frac{2}{3}\delta$, the number of elements in $\mathcal{T}$ is bounded from above by

$$k\big[l(T,\delta)^{k-1} l(V, \frac{2}{3}\delta) + l(T, \frac{2}{3}\delta)^{k-1} l(V,\delta)\big], \tag{4.22}$$

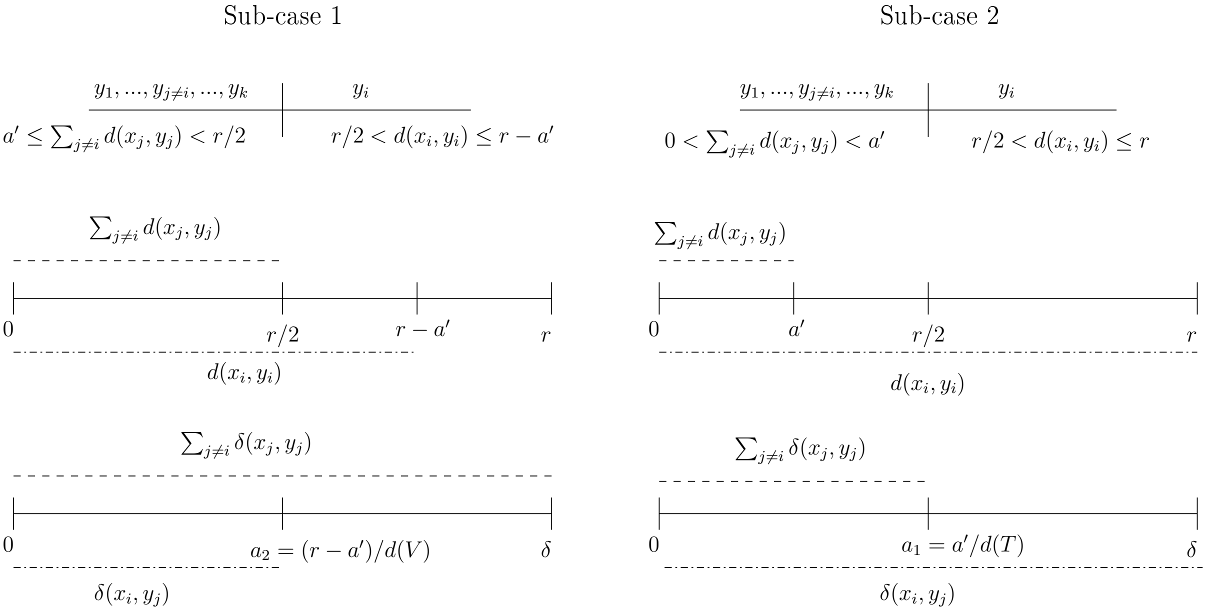
which is likely to be smaller than $k \cdot l(T,\delta)^k$, the bound obtained without the splitting strategy.

Lemma 4.2 (Complexity using the first splitting strat. with two sub-cases, no second splitting strat.)**.** *Algorithm 4.5 with the subroutine listed in Algorithm 4.6, outputs the set* $\Gamma(V,\beta,k)_\mathcal{P} \cap B_\delta(y)$ *in worst-case time*

$$\mathfrak{C}_{A.4.5} = k\mathfrak{C}_{T \cap B_\delta(y)} + k\Big[l(T,\delta)^{k-1}\mathfrak{C}_{V \cap B_{\frac{2}{3}\delta}(y)} + l(T, \frac{2}{3}\delta)^{k-1}\mathfrak{C}_{V \cap B_\delta(y)}\Big]. \tag{4.23}$$

Note: The set $T \cap B_{\frac{2}{3}\delta}(y)$ is obtained by removing the lattice points too far from y in the set $T \cap B_\delta(y)$. Hence, we made the approximation $\mathfrak{C}_{T \cap B_\delta(y), T \cap B_{\frac{2}{3}\delta}(y)} \approx \mathfrak{C}_{T \cap B_\delta(y)}$.

Similarly, the second splitting strategy splits the case $0 \leq d(x_j, y_j) \leq \frac{r}{2}$, $j \neq i$, into several sub-cases. Let $0 \leq a' \leq \frac{r}{2}$. We recall that we have $\sum_{j \neq i} d(x_j, y_j) \leq \frac{r}{2}$.



Sub-case 1

Sub-case 2

$y_1, ..., y_{j\neq i}, ..., y_k$
y_i
$a' \leq \sum_{j\neq i} d(x_j, y_j) < r/2$
$r/2 < d(x_i, y_i) \leq r - a'$
$\sum_{j\neq i} d(x_j, y_j)$
0
$r/2$
$r - a'$
r
$d(x_i, y_i)$
$\sum_{j\neq i} \delta(x_j, y_j)$
0
$a_2 = (r - a')/d(V)$
δ
$\delta(x_i, y_j)$
$y_1, ..., y_{j\neq i}, ..., y_k$
y_i
$0 < \sum_{j\neq i} d(x_j, y_j) < a'$
$r/2 < d(x_i, y_i) \leq r$
$\sum_{j\neq i} d(x_j, y_j)$
0
a'
$r/2$
r
$d(x_i, y_i)$
$\sum_{j\neq i} \delta(x_j, y_j)$
0
$a_1 = a'/d(T)$
δ
$\delta(x_i, y_j)$

Figure 4.3: Possible noise repartition, in the scope of the first splitting strategy. From the top to the bottom: Representation of y as $(y_1, ..., y_{j\neq i}, ..., y_k,\ y_i)$ with the associated possible noise levels; Unnormalized gauge representing the two noise levels; Normalized gauge representing the two noise levels.

Algorithm 4.5 List dec. for $\Gamma(V, \beta, k)_{\mathcal{P}} \in \mathbb{R}^{kn}$ **with the splitting strategy** (first splitting strategy with two sub-cases)

Input: $y = (y_1, y_2, ..., y_k) \in \mathbb{R}^{kn}$, $0 \leq \delta$.

```
for $\eta \in \{\delta, \frac{2}{3}\delta, \frac{\delta}{2}^*, \frac{\delta}{3}^*, \frac{\delta}{4}^*\}$ do
    Set $\mathcal{T}_1^{\eta}, \mathcal{T}_2^{\eta}, ..., \mathcal{T}_k^{\eta}$ as global variables.
    Compute the sets $\mathcal{T}_1^{\eta} = T \cap B_{\eta}(y_1), \mathcal{T}_2^{\eta} = T \cap B_{\eta}(y_2), ..., \mathcal{T}_k^{\eta} = T \cap B_{\eta}(y_k)$.
end for
for $1 \leq i \leq k$ do
    $\mathcal{T}_1 \leftarrow SubR(y_1, y_2, ..., y_k, \delta, 2/3\delta, i)$.
    $\mathcal{T}_2 \leftarrow SubR(y_1, y_2, ..., y_k, 2/3\delta, \delta, i)$.
end for
Return $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2\}$.
*The sets $\mathcal{T}_i^{\frac{\delta}{2}}$, $\mathcal{T}_i^{\frac{\delta}{3}}$, and $\mathcal{T}_i^{\frac{\delta}{4}}$ are computed only if used by the subroutine.
```

- If $a' \leq d(x_j, y_j) \leq \frac{r}{2}$ then $0 \leq d(x_l, y_l) \leq \frac{r}{2} - a'$, $\forall l$ where $1 \leq l \neq j \neq i \leq k$:
  Then, y_j should be list decoded in T with a radius $\frac{r}{2}$ and each y_l list decoded in T with a radius $\frac{r}{2} - a'$.
- Else $0 \leq d(x_j, y_j) < a'$ and for one l, $1 \leq l \neq j \neq i \leq k$, one may have $a' \leq d(x_l, y_l) \leq \frac{r}{2}$:
  Then, y_j should be list decoded in T with a radius a', y_l list decoded in T with a radius $\frac{r}{2}$, and all the remaining $y's$ list decoded in T with a radius a'.

Of course, since it is not possible to know the index l where[2] $a' \leq d(x_l, y_l) \leq \frac{r}{2}$, all $k-2$ possibilities should be computed (which yields $k-1$ possibilities if we include the first sub-case $a' \leq d(x_j, y_j) \leq \frac{r}{2}$). If we choose $a' = \frac{r}{2} - a' = \frac{r}{4}$, the product of the maximum list size of each $k-1$ case is $l(T, \delta)l(T, \frac{\delta}{2})^{k-2}$. As a result, the maximum number of possibilities to consider for $\sum_{j\neq i} t_j$ is

$$(k-1)l(T, \delta)l(T, \frac{\delta}{2})^{k-2}, \tag{4.24}$$

instead of $l(T, \delta)^{k-1}$ without this second splitting strategy.

[2] One may not have $a' \leq d(x_l, y_l)$, $\forall l$, $1 \leq l \neq i \leq k$. It is not an issue as we would then simply decode with a radius greater than necessary.

Algorithm 4.6 Subroutine of Algorithm 4.5 (without the second splitting strategy)

Function $SubR(y_1, y_2, ..., y_k, \delta_1, \delta_2, i)$
Input: $y = (y_1, y_2, ..., y_k) \in \mathbb{R}^{kn}$, $1 \leq t$, $0 \leq \delta_1, \delta_2$, $1 \leq i \leq k$.

```
Set $j_1 < j_2 < ... < j_{k-2}$, where $j_1, j_2, ..., j_{k-1} \in \{1, 2, ..., k\}\backslash\{i\}$.
for each $(t_{j_1}, ..., t_{j_{k-1}}) \in \mathcal{T}_{j_1}^{\delta_1} \times \mathcal{T}_{j_2}^{\delta_1} ... \times \mathcal{T}_{j_{k-1}}^{\delta_1}$ do
    Compute the sets $\mathcal{V}_i = V \cap B_{\delta_2}\big(y_i - (-\sum_{j'} t_{j_{j'}})\big)$
    for $v_i \in \mathcal{V}_i$ do
        Add $(t_{j_1}, ..., t_{j_{i-1}}, v_i + (-\sum_{j'} t_{j_{j'}}), t_{j_i}, ..., t_{j_{k-1}})$ to the list $\mathcal{T}$.
    end for
end for
Return $\mathcal{T}$.
```

Algorithm 4.7 Subroutine of Algorithm 4.5 (with once the second splitting strategy)

Function $SubR(y_1, y_2, ..., y_k, \delta_1, \delta_2, i)$
Input: $y = (y_1, y_2, ..., y_k) \in \mathbb{R}^{kn}$, $1 \leq t$, $0 \leq \delta_1, \delta_2$, $1 \leq i \leq k$.

```
for $1 \leq l \neq i \leq k$ do
    Set $j_1 < j_2 < ... < j_{k-2}$, where $j_1, j_2, ..., j_{k-2} \in \{1, 2, ..., k\}\backslash\{i, l\}$.
    for each $(t_l, t_{j_1}, ..., t_{j_{k-2}}) \in \mathcal{T}_l^{\delta_1} \times \mathcal{T}_{j_1}^{\delta_1/2} \times \mathcal{T}_{j_2}^{\delta_1/2} ... \times \mathcal{T}_{j_{k-2}}^{\delta_1/2}$ do
        Compute the sets $\mathcal{V}_i^{\delta_2} = V \cap B_{\delta_2}\big(y_i - (-t_l - \sum_{j'} t_{j_{j'}})\big)$
        for $v_i \in \mathcal{V}_i^{\delta_2}$ do
            Add $(t_{j_1}, ..., t_l, ...t_{j_{i-1}}, v_i + (-t_l - \sum_{j'} t_{j_{j'}}), t_{j_i}, ..., t_{j_{k-1}})$ to the list $\mathcal{T}$.
        end for
    end for
end for
Return $\mathcal{T}$.
```

The combination of the two splitting strategies is implemented via the association of the subroutine listed in Algorithm 4.7 with Algorithm 4.5. Substituting (4.24) in (4.22), the number of element in $\mathcal{T}$ is bounded from above by

$$\begin{aligned} & k(k-1)\big[l(T,\delta)l(T,\frac{\delta}{2})^{k-2}l(V,\frac{2}{3}\delta) + l(T,\frac{2}{3}\delta)l(T,\frac{\delta}{3})^{k-2}l(V,\delta)\big], \\ & = k(k-1)l(T,\delta)l(T,\frac{2}{3}\delta)\big[l(T,\frac{\delta}{2})^{k-2} + l(T,\frac{\delta}{3})^{k-2}\big], \end{aligned} \tag{4.25}$$

where we used $l(T,\delta) = l(V,\delta)$.

Lemma 4.3 (Complexity using the first splitting strat. with two sub-cases and once the second splitting strat.). *Algorithm 4.5, with the subroutine listed in Algorithm 4.7, outputs the set $\Gamma(V,\beta,k)_{\mathcal{P}} \cap B_\delta(y)$ in worst-case time*

$$\mathfrak{C}_{A.4.5} = k\mathfrak{C}_{T \cap B_\delta(y)} + (k^2 - k)\Big[l(T,\delta)l(T,\frac{\delta}{2})^{k-2}\mathfrak{C}_{V \cap B_{\frac{2}{3}\delta}(y)} + l(T,\frac{2}{3}\delta)l(T,\frac{\delta}{3})^{k-2}\mathfrak{C}_{V \cap B_\delta(y)}\Big]. \tag{4.26}$$

Example 4.1. *Let $\delta < \frac{1}{2}$. Using Theorem 4.1 we have: $l(T,\delta) < 2n$, $l(T,\frac{2}{3}\delta) = 2$, and $l(T,\frac{\delta}{2}) = l(T,\frac{\delta}{3}) = 1$. On the one hand, the complexity of Algorithm 4.4 (without the splitting strategy), given by Equation* (4.20)*, becomes:*

$$\begin{aligned} \mathfrak{C}_{A.4.4} = & k\mathfrak{C}_{T \cap B_\delta(y)} + k(2n)^{k-1}\mathfrak{C}_{V \cap B_\delta(y)}, \\ \approx & k(2n)^{k-1}\mathfrak{C}_{V \cap B_\delta(y)}. \end{aligned} \tag{4.27}$$

On the other hand, the complexity of Algorithm 4.5, with the subroutine listed in Algorithm 4.7, given by Equation (4.26)*, becomes:*

$$\begin{aligned} \mathfrak{C}_{A.4.5} & = k\mathfrak{C}_{T \cap B_\delta(y)} + (k^2 - k)[2n\mathfrak{C}_{V \cap B_{\frac{2}{3}\delta}(y)} + 2\mathfrak{C}_{V \cap B_\delta(y)}], \\ & \approx (k^2 - k)2n\mathfrak{C}_{V \cap B_{\frac{2}{3}\delta}(y)}. \end{aligned} \tag{4.28}$$

As a result, the worst-case complexity is quasi-linear in $l(T,\delta)$ with the splitting strategies, which is the space dimension n in this case, whereas it is polynomial of order $k-1$ in $l(T,\delta)$ without the splitting strategy.

The second splitting strategy can be applied several times. Assume that $d(x_1, y_1) > \frac{r}{2}$ and $d(x_2, y_2) > \frac{r}{4}$. Then, $\sum_{j=3}^{k} d(x_j, y_j) \leq \frac{r}{4}$. Hence, instead of decoding each y_j with a relative radius $\frac{\delta}{2}$, we apply (again) the second splitting strategy which yields

$$(k-2)l(T, \frac{\delta}{2})l(T, \frac{\delta}{4})^{k-3} \tag{4.29}$$

possibilities instead of $l(T, \frac{\delta}{2})^{k-2}$. Substituting (4.29) in (4.24) yields

$$(k-1)(k-2)l(T, \delta)l(T, \frac{\delta}{2})l(T, \frac{\delta}{4})^{k-3}. \tag{4.30}$$

It is not necessary to further apply this second splitting strategy as we do not consider list decoding beyond the minimum distance, i.e. $\delta < 1$ and $l(T, \frac{\delta}{4}) < l(T, \frac{1}{4}) = 2$. Hence, (4.30) becomes

$$(k-1)(k-2)l(T, \delta)l(T, \frac{\delta}{2}). \tag{4.31}$$

Substituting (4.31) in (4.22), the number of elements in $\mathcal{T}$ is bounded from above by (when applying twice the second splitting strategy)

$$k(k-1)(k-2)l(T, \delta)l(T, \frac{2}{3}\delta)\big[l(T, \frac{\delta}{2}) + l(T, \frac{1}{3}\delta)\big]. \tag{4.32}$$

Algorithm 4.8 Subroutine of Algorithm 4.5 (with twice the second splitting strategy)

Function $SubR(y_1, y_2, ..., y_k, \delta_1, \delta_2, i)$
Input: $y = (y_1, y_2, ..., y_k) \in \mathbb{R}^{kn}$, $1 \leq t$, $0 \leq \delta_1, \delta_2$, $1 \leq i \leq k$.

```
for $1 \leq l \neq i \leq k$ do
    for $1 \leq m \neq l \neq i \leq k$ do
        Set $j_1 < j_2 < ... < j_{k-3}$, where $j_1, j_2, ..., j_{k-3} \in \{1, 2, ..., k\} \backslash \{i, l, m\}$.
        for each $(t_l, t_m, t_{j_1}, ..., t_{j_{k-3}}) \in \mathcal{T}_l^{\delta_1} \times \mathcal{T}_m^{\frac{\delta_1}{2}} \times \mathcal{T}_{j_1}^{\frac{\delta_1}{4}} \times \mathcal{T}_{j_2}^{\frac{\delta_1}{4}} ... \times \mathcal{T}_{j_{k-2}}^{\frac{\delta_1}{4}}$ do
            Compute the sets $\mathcal{V}_i^{\delta_2} = V \cap B_{\delta_2}\big(y_i - (-t_l - t_m - \sum_{j'} t_{j_{j'}})\big)$
            for $v_i \in \mathcal{V}_i^{\delta_2}$ do
                Add $(t_{j_1}, ..., t_l, ..., t_{j_{i-1}}, v_i + (-t_l - t_m - \sum_{j'} t_{j_{j'}}), t_{j_i}, ..., t_m, ...t_{j_{k-1}})$ to the list $\mathcal{T}$.
            end for
        end for
    end for
end for
Return $\mathcal{T}$.
```

Lemma 4.4 (Complexity using the first splitting strat. with two sub-cases and twice the second splitting strat.)**.** *Algorithm 4.5, with the subroutine listed in Algorithm 4.8, outputs the set* $\Gamma(V, \beta, k)_{\mathcal{P}} \cap B_\delta(y)$ *in worst-case time*

$$\mathfrak{C}_{A.4.5} = k\mathfrak{C}_{T\cap B_\delta(y)} + k(k-1)(k-2)\Big[l(T, \delta)l(T, \frac{\delta}{2})\mathfrak{C}_{V\cap B_{\frac{2}{3}\delta}(y)} + l(T, \frac{2}{3}\delta)l(T, \frac{\delta}{3})\mathfrak{C}_{V\cap B_\delta(y)}\Big]. \tag{4.33}$$

Finally, an alternative to the first splitting strategy is to split the analysis into three cases instead of two. Let $0 \leq a' \leq a'' \leq \frac{r}{2}$.

- If $a'' \leq \sum_{j\neq i} d(x_j, y_j) \leq \frac{r}{2}$ then $\frac{r}{2} < d(x_i, y_i) \leq r - a''$:
  Then, each y_j should be list decoded in T with a radius $\frac{r}{2}$ and $y_i - (-\sum_{j\neq i} t_j)$, for all resulting combinations of t_j, list decoded in V with a radius $r - a''$.

- Else if $a' \leq \sum_{j\neq i} d(x_j, y_j) \leq a''$ then $\frac{r}{2} < d(x_i, y_i) \leq r - a'$:
  Then, each y_j should be list decoded in T with a radius a'' and $y_i - (-\sum_{j\neq i} t_j)$, for all resulting combinations of t_j, list decoded in V with a radius $r - a'$.

- Else $0 \leq \sum_{j\neq i} d(x_j, y_j) < a'$ and $\frac{r}{2} < d(x_i, y_i) \leq r$:
  Then, each y_j should be list decoded in T with a radius a' and $y_i - (-\sum_{j\neq i} t_j)$, for all resulting combinations of t_j, list decoded in V with a radius r.

The number of stored elements is bounded by

- $l(T,\delta)^{k-1}\cdot l(V,\delta-\frac{a_1}{2}=\frac{r-a''}{d(V)})$, for the first sub-case,
- $l(T,a_1=\frac{a''}{d(T)})^{k-1}\cdot l(V,\delta-\frac{a_2}{2}=\frac{r-a'}{d(V)})$ for the second sub-case,
- $l(T,a_2=\frac{a'}{d(T)})^{k-1}\cdot l(V,\delta)$ for the third sub-case.

The number of candidates in $\mathcal{T}$ is bounded from above by

$$k\big[l(T,\delta)^{k-1}l(V,\delta-\frac{a_1}{2})+l(T,a_1)^{k-1}l(V,\delta-\frac{a_2}{2})+l(T,a_2)^{k-1}l(V,\delta)\big]. \tag{4.34}$$

Substituting (4.31) in (4.34), the maximum number of elements to process at the last step of the algorithm is upper bounded by (when applying the first splitting strategy with three cases and twice the second splitting strategy)

$$k(k-1)(k-2)\Big[l(T,\delta)l(T,\frac{\delta}{2})l(V,\delta-\frac{a_1}{2})+l(T,a_1)l(T,\frac{a_1}{2})l(V,\delta-\frac{a_2}{2})+l(T,a_2)l(T,\frac{a_2}{2})l(V,\delta)\Big]. \tag{4.35}$$

Algorithm 4.9 List dec. for $\Gamma(V,\beta,k)_{\mathcal{P}}\in\mathbb{R}^{kn}$ **with the splitting strategy** (first splitting strategy with three sub-cases)

Input: $y=(y_1,y_2,...,y_k)\in\mathbb{R}^{kn}$, $3/4\leq\delta$, $0<a_2<a_1<\delta$.

```
for $\eta\in\{\delta,a_1,a_2,a_1/2,a_2/2\}$ do
    Set $\mathcal{T}_1^\eta,\mathcal{T}_2^\eta,...,\mathcal{T}_k^\eta$ as global variables.
    Compute the sets $\mathcal{T}_1^\eta=T\cap B_\eta(y_1),\mathcal{T}_2^\eta=T\cap B_\eta(y_2),...,\mathcal{T}_k^\eta=T\cap B_\eta(y_k)$.
end for
for $1\leq i\leq k$ do
    $\mathcal{T}_1\leftarrow SubR(y_1,y_2,...,y_k,\delta,\delta-\frac{a_1}{2},i)$.
    $\mathcal{T}_2\leftarrow SubR(y_1,y_2,...,y_k,a_1,\delta-\frac{a_2}{2},i)$.
    $\mathcal{T}_3\leftarrow SubR(y_1,y_2,...,y_k,a_2,\delta,i)$.
end for
Return $\mathcal{T}=\{\mathcal{T}_1,\mathcal{T}_2,\mathcal{T}_3\}$.
```

Lemma 4.5 (Complexity using the first splitting strat. with three sub-cases and once or twice second splitting strat.)**.** *Algorithm 4.9 outputs the set $\Gamma(V,\beta,k)_{\mathcal{P}}\cap B_\delta(y)$ in worst-case time:*

- *If one uses the subroutine listed in Algorithm 4.7 (once the second splitting strategy)*

$$\begin{aligned}\mathfrak{C}_{A.4.9}=k\mathfrak{C}_{T\cap B_\delta(y)}+(k^2-k)\Big[&l(T,\delta)l(T,\frac{\delta}{2})^{k-2}\mathfrak{C}_{V\cap B_{\delta-\frac{a_1}{2}}(y)}+l(T,a_1)l(T,\frac{a_1}{2})^{k-2}\mathfrak{C}_{V\cap B_{\delta-\frac{a_2}{2}}(y)}\\&+l(T,a_2)l(T,\frac{a_2}{2})^{k-2}\mathfrak{C}_{V\cap B_\delta(y)}\Big].\end{aligned} \tag{4.36}$$

- *If one uses the subroutine listed in Algorithm 4.8 (twice the second splitting strategy)*

$$\begin{aligned}\mathfrak{C}_{A.4.9}=k\mathfrak{C}_{T\cap B_\delta(y)}+k(k-1)(k-2)\Big[&l(T,\delta)l(T,\frac{\delta}{2})\mathfrak{C}_{V\cap B_{\delta-\frac{a_1}{2}}(y)}+l(T,a_1)l(T,\frac{a_1}{2})\mathfrak{C}_{V\cap B_{\delta-\frac{a_2}{2}}(y)}\\&+l(T,a_2)l(T,\frac{a_2}{2})\mathfrak{C}_{V\cap B_\delta(y)}\Big].\end{aligned} \tag{4.37}$$

Similarities with [GP17]. The first splitting strategy is used in [GP17] to prove that the worst-case list size of the Barnes-Wall lattices (which belong to the family of parity lattices with $k=2$, see Section 5.2.1) is polynomial in the lattice dimension for any decoding radius bounded away from the minimum distance. For instance, Equation (4.22) for $k=2$ and $\delta=5/8$ becomes the same equation as the one at the end of the proof of Lemma 2.5 in [GP17]. While [GP17] also investigates the decoding aspect of Barnes-Wall lattices, a different technique is used, resulting in a complexity quadratic in the list size. Finding an algorithm with quasi-linear dependence in the list size is stated as an open problem in [GP17]. As shown in this section, the splitting strategy for decoding enables to get this quasi-linear complexity in the list size. See Section 5.2.1 for more details on [GP17].

Chapter 5

Parity lattices

This chapter is dedicated to the study of parity lattices. It is divided into two sections. The first section discusses the parity lattices for arbitrary k. The second section is about parity lattices obtained with $k = 2$ and $k = n^{\frac{1}{\log \log n}}$.

5.1 Algorithms and theorems

5.1.1 Properties of the parity lattices

Families of single parity-check groups can be built by recursively applying the single parity-check construction (see Equation (4.10)). For instance, a new family of lattices is obtained as follows.

First, given $d(\Gamma(V, \beta, k)_{\mathcal{P}}) = \min\{2d(T), d(V)\}$, we shall consider only lattices where $d(V) = 2d(T)$. In order to find two lattices having this property, with $V \subset T$, we consider a lattice T over a complex ring J, and rotate it by an element $\theta \in J$, with $|\theta| = \sqrt{2}$, to get V: i.e. $V = \theta \cdot T$. This yields $V \cong \sqrt{2}T$ and $d(\Gamma(\theta T, \beta, k)_{\mathcal{P}}) = 2d(T)$. The ring J can for instance be $\mathbb{Z}[i]$ or $\mathbb{Z}[\lambda]$.

More formally, let $\Lambda_c^{\mathbb{C}}$ be a lattice over a complex ring J, where J is either $\mathbb{Z}[i]$ or $\mathbb{Z}[\lambda]$. We denote by $\Lambda_c \in \mathbb{R}^c$ the corresponding real lattice, with real dimension c. Let L_n be the real lattice obtained via (3.3) from a complex lattice $L_n^{\mathbb{C}}$ and θL_n be the real lattice obtained from $\theta \cdot L_n^{\mathbb{C}}$. Also, $\beta = [L_n/\theta L_n]$.

Definition 5.1. *Let $n = c \cdot k^t$, $t \geq 0$. The parity lattices in dimension kn are defined by the following recursion:*

$$\begin{aligned} L_{kn} &= \Gamma(\theta L_n, \beta, k)_{\mathcal{P}}, \\ &= \{(v_1 + n_1, v_2 + n_2, ..., v_k - \sum_{i \neq k} n_i), v_i \in \theta L_n, n_i \in \beta\}, \\ &= \{(t_1, t_2, ..., v_k' - \sum_{i \neq k} t_i), v_k' \in \theta L_n, t_i \in L_n\}, \end{aligned} \tag{5.1}$$

with initial condition $L_c = \Lambda_c$. The number t of recursive steps should not be confused with $t_i \in L_n$.

The fundamental coding gain of these lattices is given by the following theorem. It shows that a small k maximizes the asymptotic density as a function of n but a large k minimizes the number of recursive steps t needed to reach a given density.

Lemma 5.1. *Let $n = c \cdot k^t$, $t \geq 0$ and g be the coding gain of L_c. The fundamental coding gain of the parity lattices is*

$$\gamma(L_n) = \frac{g \cdot 2^t}{2^{\frac{t}{k}}} = n^{\frac{1}{\log_2 k} - \frac{1}{k \log_2 k}} 2^{\log_2 g - \log_k(c)(1 - \frac{1}{k})}. \tag{5.2}$$

Proof. Scale L_c such that $\mathrm{vol}(L_c) = 1$. First, we compute the minimum distance: $d(L_{kn}) = \min\{d(\theta L_n), 2d(L_n)\} = 2d(L_n)$, with initial condition $d(L_c) = g$. Hence, $d(L_n) = g \cdot 2^t$. Second, we have

$$\begin{aligned} \mathrm{vol}(L_{kn})^{\frac{2}{kn}} &= \left(\frac{\mathrm{vol}(\theta L_n)^k}{|\beta|^{k-1}}\right)^{\frac{2}{kn}} = \left(\frac{(\mathrm{vol}(L_n) \cdot 2^{\frac{n}{2}})^k}{2^{(\frac{n}{2})(k-1)}}\right)^{\frac{2}{kn}}, \\ &= \mathrm{vol}(L_n)^{\frac{2}{n}} \cdot 2^{\frac{1}{k}}, \end{aligned} \tag{5.3}$$

with initial condition $\mathrm{vol}(L_c)^{2/kn} = 1$. Hence, $\mathrm{vol}(L_n)^{2/kn} = 2^{t/k}$ and the fundamental coding gain becomes:

$$\gamma(L_n) = \frac{g \cdot 2^t}{2^{t/k}} = 2^{t-\frac{t}{k}} 2^{\log_2 g} = \left(\frac{n}{c}\right)^{\frac{1}{\log_2 k} - \frac{1}{k \log_2 k}} 2^{\log_2 g}.$$

□

For instance, taking $J = \mathbb{Z}[i]$, $\theta = \phi = 1+i$, $k = 3$, and an initial condition $L_2 = \mathbb{Z}^2$, yields a series of lattices with $\gamma(L_n) = O(n^{0.4206})$. Another example: $J = \mathbb{Z}[\lambda]$, $k = 3$, and an initial condition $L_{24} = \Lambda_{24}$, one gets a denser series but with the same asymptotic behavior with respect to the coding gain $O(n^{0.4206})$.

A generator matrix for L_{kn}, say $G_{L_{kn}}$, as a function of G_{L_n} has the following form:

$$G_{L_{kn}} = \begin{bmatrix} G_{\theta L_n} & 0 & 0 & \dots & 0 \\ G_{L_n} & -G_{L_n} & 0 & \dots & 0 \\ 0 & G_{L_n} & -G_{L_n} & \dots & 0 \\ 0 & 0 & G_{L_n} & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & G_{L_n} & -G_{L_n} \end{bmatrix}. \tag{5.4}$$

5.1.2 Recursive decoding

Presentation of the recursive algorithms

The decoding algorithms of the previous chapter can be adapted to decode groups recursively built from the single parity-check construction. As examples, we adapt Algorithms 4.1, 4.4, and 4.5 in the recursive Algorithms 5.1, 5.2, and 5.3, respectively, to decode the parity lattices $L_{kn} = \Gamma(\theta L_n, \beta, k)$ (presented in Definition 5.1). Hence, we have $T = L_n$ and $V = \theta L_n$. Since $l(L_n, \delta) = l(\theta L_n, \delta)$, we set $l(n, \delta) = L(n, r) = l(L_n, \delta)$ to simplify the notations. Moreover, we also write $\mathfrak{C}(\delta)$ for $\mathfrak{C}(\frac{n}{k}, \delta)$ and $l(\delta)$ for $l(\frac{n}{k}, \delta)$.

Algorithm 5.1, a recursive BDD, is obtained by replacing Steps 1-3 of Algorithm 4.1 by the recursive Steps 4-7. Regarding Step 6: The point $y_i - (-\sum_{j \neq i} t_j)$ should be decoded in the lattice $R_\theta L_{c \cdot k^t}$. It is equivalent to decode $(y_i - (-\sum_{j \neq i} t_j)) \cdot R(c \cdot k^t, \theta)^{-1}$ in $L_{c \cdot k^t}$ and then rotate the output lattice point by $R(c \cdot k^t, \theta)$.

The recursive list decoders involve the above modifications as well as the following one: The "removing" steps (Steps 14 and 15 in Algorithms 5.2 and Steps 11 and 12 in Algorithm 5.3) are added to ensure that a list with no more than $l(n, \delta)$ elements is returned by each recursive call. This enables to control the complexity of the algorithm. However, we shall see that the step in bold is not always necessary for the Gaussian channel.

The recursive version of all algorithms presented in the previous chapter are obtained in a similar manner.

Algorithm 5.1 Recursive BDD for $L_{kn} = \Gamma(\theta L_n, \beta, k)$, $n = c \cdot k^t$

Function $RecL(y, t)$
Input: $y = (y_1, y_2, ..., y_k) \in \mathbb{R}^{kn}$, $0 \leq t$.

```
1: if $t = 0$ then
2:     $x_{opt} \leftarrow$ Decode (BDD) $y$ in $\Lambda_c$.
3: else
4:     $t_1 \leftarrow RecL(y_1, t-1)$, $t_2 \leftarrow RecL(y_2, t-1)$,...,
       $t_k \leftarrow RecL(y_k, t-1)$.
5:     for $1 \leq i \leq k$ do
6:         $v_i \leftarrow RecL([y_i - (-\sum_{j'} t_{j_{j'}})] \cdot R(c \cdot k^t, \theta)^T, t-1) \cdot R(c \cdot k^t, \theta)$.
7:         $\hat{x}_i \leftarrow (t_{j_1}, ..., t_{j_{i-1}}, v_i + (-\sum_{j'} t_{j_{j'}}), t_{j_i}, ..., t_{j_{k-1}})$
8:     end for
9: end if
10: Return $x_{opt} = \arg\min_{1 \leq i \leq k} ||y - \hat{x}_i||$.
```

Algorithm 5.2 Recursive list dec. for $L_{kn} = \Gamma(\theta L_n, \beta, k)_{\mathcal{P}} \in \mathbb{R}^{kn}$ **without the splitting strategy**

Function $ListRecL(y, t, \delta)$
Input: $y = (y_1, y_2, ..., y_k) \in \mathbb{R}^{kn}$, $0 \leq t$, $0 \leq \delta$.

```
if $t = 0$ then
  $\mathcal{T}_0 \leftarrow$ The set $\Lambda_c \cap B_\delta(y)$.
else
  $\mathcal{T}_1 \leftarrow ListRecL(y_1, t-1, \delta)$, $\mathcal{T}_2 \leftarrow ListRecL(y_2, t-1, \delta)$,..., $\mathcal{T}_k \leftarrow ListRecL(y_k, t-1, \delta)$.
  for $1 \leq i \leq k$ do
    Set $j_1 < j_2 < ... < j_{k-1}$, where $j_1, j_2, ..., j_{k-1} \in \{1, 2, ..., k\} \backslash \{i\}$.
    for each $(t_{j_1}, ..., t_{j_{k-1}}) \in \mathcal{T}_{j_1} \times \mathcal{T}_{j_2} ... \times \mathcal{T}_{j_{k-1}}$ do
      $\mathcal{V}_i \leftarrow ListRecL([y_i - (-\sum_{j'} t_{j_{j'}})] \cdot R(c \cdot k^t, \theta)^T, t-1, \delta) \cdot R(c \cdot k^t, \theta)$.
      for $v_i \in \mathcal{V}_i$ do
        Add $(t_{j_1}, ..., t_{j_{i-1}}, v_i + (-\sum_{j'} t_{j_{j'}}), t_{j_i}, ..., t_{j_{k-1}})$ in the list $\mathcal{T}$.
      end for
    end for
  end for
  Remove all elements in $\mathcal{T}$ at a relative distance $> \delta$ from $y$.
  Sort the remaining elements in $\mathcal{T}$ in a lexicographic order and remove all duplicates.
end if
Return $\mathcal{T}$.
```

Preliminary complexity analysis

The following theorem presents the complexity of the recursive BDD.

Theorem 5.1. *Let $n = c \cdot k^t$ and $y \in \mathbb{R}^n$. If $d(y, L_n) < \rho^2(L_n)$, then Algorithm 5.1 outputs the closest lattice point to y in time*

$$\mathfrak{C}_{A.5.1}(n, \frac{1}{4}) = O(n^{1 + \frac{1}{log_2 k}}). \tag{5.5}$$

Proof.

$$\begin{aligned} \mathfrak{C}(n, \frac{1}{4}) =& 2k\mathfrak{C}(\frac{n}{k}, \frac{1}{4}) + O(n) = O(n) \sum_{i=0}^{\log_k n} \left(\frac{2k}{k}\right)^i, \\ =& O(n^{1+\frac{1}{log_2 k}}). \end{aligned}$$

□

Regarding the recursive list decoders, we address the following question. Among the proposed list-decoding strategies enumerated in the previous chapter, which one should be used to minimize the complexity with respect to the space dimension n?
We assume that $k = o(n)$. For $\frac{1}{4} < \delta < 1$, list decoding without the splitting strategy induces a complexity bounded from below by $l(n/k, \delta)^{k-1} \cdot \mathfrak{C}(n/k, \delta)$. Unwinding the recurrence yields a complexity $\geq n^{(k-1)\log_k \frac{1}{2\epsilon}}$, if $\delta = \frac{1}{2} - \epsilon, 0 < \epsilon < \frac{1}{4}$ and

$$\mathfrak{C}(n, \delta) \geq n^{\Omega(\log_k n)}, \tag{5.6}$$

if $\delta > \frac{1}{2}$ (assuming $l(n, \delta > \frac{1}{2}) = \Omega(n)$). Consequently, we study the different splitting strategies to break this super-polynomial complexity.

First, when should one consider the second splitting strategy? We recall that this strategy linearizes $l(n, \delta)^{k-1}$ in: $(k-1)l(n, \delta)l(n, \delta/2)^{k-1}$ if used once and $(k-1)(k-2)l(n, \delta)l(n, \delta/2)$ if used twice. Obviously, if $k = 2$ the strategy is pointless and if $k = 3$ it should be considered at most once. However, for any $k > 3$, the second splitting strategy should be used (to avoid a super-polynomial complexity):

- Once if $l(n, \delta) > 1$ and $l(n, \frac{\delta}{2}) = 1$, i.e. for $\frac{1}{4} \leq \delta < \frac{1}{2}$ (using Theorem 4.1).
- Twice if $l(n, \frac{\delta}{2}) > 1$, i.e. for $\frac{1}{2} \leq \delta < 1$.

Algorithm 5.3 Recursive list dec. for $L_{kn} = \Gamma(\theta L_n, \beta, k)_{\mathcal{P}} \in \mathbb{R}^{kn}$ **with the splitting strategy** (first splitting strategy with two sub-cases)

```
Function $ListRecL(y, t, \delta)$
Input: $y = (y_1, y_2, ..., y_k) \in \mathbb{R}^{kn}$, $0 \leq t$, $0 \leq \delta$.

if $t = 0$ then
    $\hat{x} \leftarrow$ The set $\Lambda_c \cap B_\delta(y)$.
else
    for $\eta \in \{\delta, 2/3\delta, 1/2\delta, 1/3\delta\}$ do
        $\mathcal{T}_1^\eta \leftarrow ListRecL(y_1, t-1, \eta)$, $\mathcal{T}_2^\eta \leftarrow ListRecL(y_2, t-1, \eta), ..., \mathcal{T}_k^\eta \leftarrow ListRecL(y_k, t-1, \eta)$.
    end for
    for $1 \leq i \leq k$ do
        $\mathcal{T}_1 \leftarrow SubR(y_1, y_2, ..., y_k, t, \delta, 2/3\delta, i)$.
        $\mathcal{T}_2 \leftarrow SubR(y_1, y_2, ..., y_k, t, 2/3\delta, \delta, i)$.
    end for
    Remove all elements in $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2\}$ at a relative distance $> \delta$ from $y$.
    Sort the remaining elements in $\mathcal{T}$ in a lexicographic order and remove all duplicates.
end if
Return $\mathcal{T}$.
```

Algorithm 5.4 Subroutine of the recursive Algorithm 5.3 (with once the second splitting strategy)

```
Function $SubR(y_1, y_2, ..., y_k, t, \delta_1, \delta_2, i)$
Input: $y = (y_1, y_2, ..., y_k) \in \mathbb{R}^n$, $1 \leq t$, $0 \leq \delta_1, \delta_2$, $1 \leq i \leq k$.

for $1 \leq l \neq i \leq k$ do
    Set $j_1 < j_2 < ... < j_{k-2}$, where $j_1, j_2, ..., j_{k-2} \in \{1, 2, ..., k\} \backslash \{i, l\}$.
    for each $(t_l, t_{j_1}, ..., t_{j_{k-2}}) \in \mathcal{T}_l^{\delta_1} \times \mathcal{T}_{j_1}^{\delta_1/2} \times \mathcal{T}_{j_2}^{\delta_1/2} ... \times \mathcal{T}_{j_{k-2}}^{\delta_1/2}$ do
        $\mathcal{V}_i^{\delta_2} \leftarrow ListRecL([y_i - (-t_l - \sum_{j'} t_{j_{j'}})] \cdot R(c \cdot k^t, \theta)^T, t-1), t-1, \delta_2) \cdot R(c \cdot k^t, \theta)$.
        for $v_i \in \mathcal{V}_i^{\delta_2}$ do
            Add $(t_{j_1}, ..., t_{j_{i-1}}, v_i + (-t_l - \sum_{j'} t_{j_{j'}}), t_{j_i}, ..., t_{j_{k-1}})$ in the list $\mathcal{T}$.
        end for
    end for
end for
Return $\mathcal{T}$.
```

Second, should one use the first slitting strategy with two sub-cases or three? We analyze the behavior of the first splitting strategy with two sub-cases. Depending on the second splitting strategy, there is a multiplicative term k^{x_1}, $1 \leq x_1 \leq 3$, in the complexity formulas. Moreover, assume that $\mathfrak{C}(n, \frac{2}{3}\delta)$ has polynomial complexity $O(n^{x_2})$, where x_2 is a positive constant. The complexity has the form (see Equations (4.23), (4.26), and (4.33))

$$\begin{aligned}\mathfrak{C}(n, \delta) &\approx k^{x_1}\left[l(\frac{2}{3}\delta)\mathfrak{C}(\delta) + \frac{l(\delta)}{k^{x_2}}O(n^{x_2})\right], \\ &\leq k^{x_1 - x_2} l(\delta) O(n^{x_2}) \sum_{i=0}^{\log_k n} \left(\frac{k^{x_1} l(\frac{2}{3}\delta)}{k^{x_2}}\right)^i, \\ &= k^{x_1 - x_2} l(\delta) O(n^{x_1} n^{\log_k l(\frac{2}{3}\delta)}),\end{aligned} \tag{5.7}$$

where we assumed that $k^{x_1 - x_2} l(\frac{2}{3}\delta) > 1$.

Equation (5.7) illustrates the critical aspect of $l(n, \frac{2}{3}\delta)$. If $\frac{1}{4} < \delta < \frac{3}{4}$, we have $l(n, \frac{2}{3}\delta) < l(n, \frac{1}{2}) = O(1)$ (using Theorem 4.1). But if $\frac{3}{4} \leq \delta < 1$, then $l(n, \frac{2}{3}\delta) = \Omega(n)$. Hence, the first splitting strategy with two-cases induces a super-polynomial complexity for $\delta \geq \frac{3}{4}$, but not for $\delta < \frac{3}{4}$ (assuming that $l(n, \delta)$ is not super-polynomial).

On the other hand, the splitting strategy with three cases yields a complexity (see Equation (4.37))

$$\mathfrak{C}(n, \delta) \approx k^{x_1}\left[l(\delta)\mathfrak{C}(a_1) + l(2(\delta - a_1))\mathfrak{C}(\delta - \frac{a_2}{2}) + l(a_2)\mathfrak{C}(\delta)\right]. \tag{5.8}$$

To avoid the super-polynomial behavior, we need $l(a_2) = O(1)$. We set $a_2 = \frac{1}{2} - \epsilon_1'$ (see Theorem 4.1) which implies $\delta - \frac{a_2}{2} = \delta - \frac{1}{4} + \frac{\epsilon_1'}{2}$. It is then possible to show by induction that (5.8) is polynomial in n if $l(\delta)$ is also polynomial (see the next sections).

As a result, one should choose the first splitting strategy with two cases if $\delta < \frac{3}{4}$, but with three cases otherwise.

Note: When we address the complexity of these recursive list decoders, the left-hand term of (4.3) should be updated to include the complexity of the removing steps performed at each recursion. The most expensive operation is the sorting step. It can be realized with the classical Merge Sort algorithm in time $O(n \cdot l(\frac{n}{k}, \delta) \log l(\frac{n}{k}, \delta))$ (see Appendix 5.3.1). This term should be added to the first line of (5.7) and thus compared with $k^{x_1} \frac{l(\delta)}{k^{x_2}} O(n^{x_2})$ for the recursion. In our analyses, we assume that the former term is negligible compared to the latter one. Potential errors induced by this approximation only involve a $\log l(\delta)$ term which has no impact on the conclusions we draw.

5.1.3 Decoding performance on the Gaussian channel

Similarly to the previous section, we also restrict the analysis to the lattices $\Gamma(V, \beta, k)_{\mathcal{P}}$ where $V \cong \sqrt{2}T$. Nevertheless, it should be possible to extend it to other group codes (such as binary linear codes) built via the single parity-check construction.

Lemma 5.2. *Let $x \in \Lambda \subset \mathbb{R}^n$ and let $y \in \mathbb{R}^n$ be the point to decode. Let $\mathcal{T}$ denote the list outputted by a list-decoding algorithm. The point error probability under list decoding is bounded from above by:*

$$P_e(dec) \leq P_e(opt) + P(x \notin \mathcal{T}). \tag{5.9}$$

Proof.

$$\begin{aligned} P_e(dec) =& P(y \notin \mathcal{V}(x)) + P(x \notin \mathcal{T} \cap y \in \mathcal{V}(x)), \\ \leq& P(y \notin \mathcal{V}(x)) + P(x \notin \mathcal{T}). \end{aligned} \tag{5.10}$$

□

In the sequel, we derive formulas to estimate the term $P(x \notin \mathcal{T})$ obtained with several versions of the recursive list decoders presented in the previous section.

Choosing the decoding radius for regular list decoding on the Gaussian channel

Consider the Gaussian channel where $y = x + w$, with $y \in \mathbb{R}^n$, $x \in \Lambda \subset \mathbb{R}^n$, and $w \in \mathbb{R}^n$ with i.i.d $\mathcal{N}(0, \sigma^2)$ components. With a regular list decoder $\mathcal{T} = \Lambda \cap B_\delta(y)$ and

$$P(x \notin \mathcal{T}) = P(||w||^2 > r). \tag{5.11}$$

Since $||w||^2$ is a Chi-square random variable with n degrees of freedom, $P(||w||^2 > r) = F(n, r, \sigma^2)$, where, for n even :

$$F(n, r, \sigma^2) = e^{-\frac{r}{2\sigma^2}} \sum_{k=0}^{n/2-1} \frac{1}{k!} \left(\frac{r}{2\sigma^2} \right)^k. \tag{5.12}$$

Theorem 5.2. *Consider Algorithm 5.2 with the following input parameters. The point $y = x + w$, where $y \in \mathbb{R}^{kn}$, $x \in L_{kn}$, and $w \in \mathbb{R}^{kn}$ with i.i.d $\mathcal{N}(0, \sigma^2)$ components. Moreover, $t \geq 0$ and $\delta = r/d(L_{kn})$. We have*

$$P(x \notin \mathcal{T}) = F(kn, r, \sigma^2). \tag{5.13}$$

Based on (5.9), quasi-optimal performance with regular list decoding is obtained by choosing a decoding radius $r = E[||w||^2](1+\epsilon) = n\sigma^2(1+\epsilon)$ such that $F(n, r, \sigma^2) < \eta \cdot P_e(opt, \sigma^2)$ (in practice $\eta = 1/2$ is good enough). Moreover, it is easy to show that $\epsilon \to 0$ when $n \to +\infty$. We denote by δ^* the relative decoding radius corresponding to this specific r:

$$\delta^* = \frac{n\sigma^2(1+\epsilon)}{d(\Lambda)}. \tag{5.14}$$

Of course, the greater δ^*, the greater the list-decoding complexity.

A modified list-decoding algorithm

Notice that due to the "removing step" (Steps 14 and 11, in bold, of Algorithms 5.2 and 5.3, respectively), some points that are correctly decoded by Algorithm 5.1 (the BDD) are not in the list outputted by Algorithms 5.2 and 5.3 (if $\delta = 1/4$ or even slightly greater): The decoding radius is r in V and $r/2$ in T, but only the points at a distance less than r from y are kept. Hence, if a point found at the last recursive step is at a distance greater than r from y, even if it is the unique point found, it is not kept and an empty list is returned. On the contrary, the BDD outputs a point even if it is further than $r = \frac{1}{4}d(L_{kn})$ from y. Of course, the relative decoding radius δ can be increased to avoid this situation, but at the cost of a greater decoding complexity.

To avoid the situation mentioned above, we remove Step 14 in Algorithm 5.2. We will see in the rest of the chapter that this enables to choose smaller decoding radii for QMLD than with regular list decoding and reduce the complexity despite the absence of the removing step. In terms of error probability, decoding in a sphere is the best choice given a finite decoding volume around the received point y. However, there may be larger non-spherical volumes that achieve satisfactory performance but that are less complex to explore. This is the main idea behind the this modified list-decoding algorithm. This subsection concentrates on the analysis of the error probability of the modified algorithm. The analysis of the asymptotic complexity is deferred to the next sections.

Theorem 5.3. *Consider Algorithm 5.2 without Step 14 with the following input parameters. The point y is obtained on a Gaussian channel with VNR $\Delta = \text{vol}(L_{kn})^{2/kn}/2\pi e\sigma^2$ as $y = x + w$, where $y \in \mathbb{R}^{kn}$, $x \in L_{kn}$, and $w \in \mathbb{R}^{kn}$ with i.i.d $\mathcal{N}(0, \sigma^2)$ components. Moreover, $t \geq 0$ and δ is the relative decoding radius. We have*

$$P(x \notin \mathcal{T}) \leq U_{kn}(\delta, \Delta), \tag{5.15}$$

where

$$U_n(\delta, \Delta) = \min\Big\{\binom{k}{2} U_{\frac{n}{k}}(\delta, \frac{\Delta}{2^{\frac{1}{k}}})^2 + kU_{\frac{n}{k}}(\delta, 2^{\frac{k-1}{k}}\Delta)(1 - U_{\frac{n}{k}}(\delta, \frac{\Delta}{2^{\frac{1}{k}}}))^{k-1},\ 1\Big\}. \tag{5.16}$$

The initial condition $U_c(\delta, \Delta)$ corresponds to the decoding performance in L_c: $U_c(\delta, \Delta) = P(x \notin \mathcal{T}_c)$, where $\mathcal{T}_c$ denotes the list of candidates obtained when list decoding in L_c.

The proof is provided in Appendix 5.3.2. For instance, a regular list decoder for L_c with relative decoding radius δ is used in Algorithm 5.2. Consequently, the initial condition is

$$U_c(\delta, \Delta) = F(c, f(\delta), f(\Delta)), \tag{5.17}$$

with $f(\delta) = \delta \cdot d(L_c)$, and $f(\Delta) = \text{vol}(L_c)^{\frac{2}{c}}/(2\pi e\Delta)$.

As illustrated by the next example, (5.16) means that the lattices of smaller dimensions are decoded with the same relative radius but with a VNR that is either greater, $2^{\frac{k-1}{k}}\Delta$, or smaller, $\Delta/2^{\frac{1}{k}}$. This result is a consequence of the following properties of the parity lattices: $\text{vol}(L_n)^{\frac{2}{n}} = \text{vol}(L_{kn})^{\frac{2}{kn}}/2^{\frac{1}{k}}$ and $d(L_n) = d(L_{kn})/2$.

Example 5.1. *Let Λ_{24} be the Leech lattice and let $\beta = [\Lambda_{24}/\lambda\Lambda_{24}]$, where $\lambda = \frac{1+i\sqrt{7}}{2}$ (see Section 6.1.1 for more details on $\lambda\Lambda_{24}$). The k-parity-Leech lattices are defined as $L_{kn} = \Gamma(\lambda L_n, \beta, k)_{\mathcal{P}}$ with initial condition $\Lambda_c = \Lambda_{24}$. On the Gaussian channel and with Algorithm 5.2 without Step 14, the probability that the transmitted lattice point is not in the outputted list is given by* (5.16). *We let the initial condition $U_{24}(\Delta)$ be the performance of the optimal decoder for Λ_{24} (δ is thus irrelevant in this case). It means that Step 2 of Algorithm 5.2 is modified by using a MLD decoder for Λ_{24}.*

For the lattice $L_{k\cdot 24}$, the value of $U_{k\cdot 24}(\Delta)$ is obtained by adding the performance curves representing

- $\binom{k}{2} \cdot (P_e^{\Lambda_{24}}(opt, \Delta))^2$*, shifted by* $10\log_{10}(2^{\frac{1}{k}})$ *dB to the left,*
- *and $kP_e^{\Lambda_{24}}(opt, \Delta)$ shifted by $10\log_{10}(2^{\frac{k-1}{k}})$ dB to the right (assuming that $(1 - U_{\frac{n}{k}}(\frac{\Delta}{2^{\frac{1}{k}}}))^{k-1} \approx 1$).*

The results for $k = 3$ and $k = 7$ are shown by the dashed line in Figure 5.1. Since there is only one recursive step, Algorithm 5.2 is equivalent to Algorithm 4.4. The associated decoding complexity is obtained from (4.20) *where the term $l(T, \delta)$ is set to 1 since the MLD decoder of Λ_{24} returns only one candidate (note that* (4.20) *reduces to* (4.16) *in this case). Consequently, we get*

$$\mathfrak{C}(L_{k\cdot 24}) = 2k\mathfrak{C}_{MLD}^{\Lambda_{24}} + O(k \cdot 24). \tag{5.18}$$

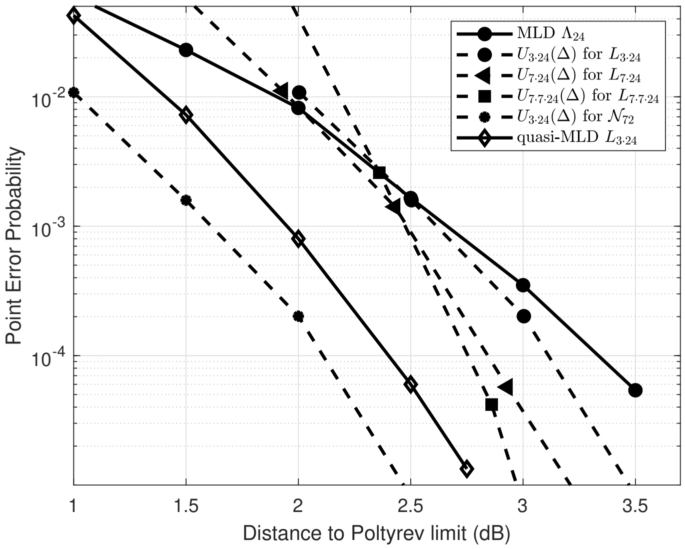


Figure 5.1: Performance curves for Example 5.1.

The probability $U_{k\cdot k\cdot 24}(\Delta)$ is obtained in a similar manner from $U_{k\cdot 24}(\Delta)$. For instance, $U_{7\cdot 7\cdot 24}(\Delta)$ is also plotted in the figure. The corresponding decoding complexity of $L_{k\cdot k\cdot 24}$ (without using the splitting strategy) is obtained from (4.20) *(with $l(T,\delta)$ set to k, the number of candidates obtained at the previous recursive step):*

$$\begin{aligned}\mathfrak{C}(L_{k\cdot k\cdot 24}) =& k(1+k^{k-1})\mathfrak{C}(L_{k\cdot 24}) + O(k\cdot k\cdot 24),\\ \approx& 2k^{k+1}\mathfrak{C}_{MLD}^{\Lambda_{24}} + O(24k^2).\end{aligned} \tag{5.19}$$

Figure 5.1 depicts the quasi-MLD performance of $L_{3\cdot 24}$ (obtained in Section 6.2.5) for comparison.

We shall see in the next chapter that the Nebe lattice $\mathscr{N}_{72}$, constructed as $\Gamma(\lambda\Lambda_{24},\alpha,\beta,3)$, has the following properties: $\text{vol}(\mathscr{N}_{72})^{\frac{2}{n=72}} = \text{vol}(T=\Lambda_{24})^{\frac{2}{n/3}}$ *and* $d(\mathscr{N}_{72}) = 2d(\Lambda_{24})$*. Hence,* (5.16) *becomes*

$$U_{n=72}(\Delta) = \min\Big\{3U_{\frac{n}{3}}(\Delta)^2 + 3U_{\frac{n}{3}}(2\Delta)(1-U_{\frac{n}{3}}(\delta,\Delta))^2,\ 1\Big\}. \tag{5.20}$$

Taking $U_{\frac{n}{3}}(\Delta) = P_e^{\Lambda_{24}}(opt,\Delta)$*, we get a similar curve as $U_{3\cdot 24}$ for $L_{3\cdot 24}$ but shifted by* $10\cdot\log_{10}(2^{\frac{1}{3}}) = 1$ *dB to the left. The curve $U_{72}(\Delta)$ is shown on Figure 5.1. See Section 6.2.4 for more details.*

If the splitting strategies are considered, as in Algorithm 5.3 (without Step 11), the error probability is slightly greater due to specific cases, such as having simultaneously $\frac{2}{3}\frac{r}{2} < ||w_j|| < \frac{r}{2}$ and $\frac{2}{3}r < ||w_i|| < r$, which are not correctly decoded (whereas they were without the splitting strategy).

For the case $k=2$, it is shown in Appendix 5.3.2 that with the splitting strategy we get the recursion

$$\begin{aligned}U_n(\delta,\Delta) = \min\Big\{&U_{\frac{n}{2}}(\delta,\frac{\Delta}{\sqrt{2}})^2 + 2\Big[\big(U_{\frac{n}{2}}(\frac{2}{3}\delta,\frac{\Delta}{\sqrt{2}}) - U_{\frac{n}{2}}(\delta,\frac{\Delta}{\sqrt{2}})\big)U_{\frac{n}{2}}(\frac{2}{3}\delta,\sqrt{2}\Delta)+\\ &(1-U_{\frac{n}{2}}(\frac{2}{3}\delta,\frac{\Delta}{\sqrt{2}}))U_{\frac{n}{2}}(\delta,\sqrt{2}\Delta)\Big],\ 1\Big\}.\end{aligned} \tag{5.21}$$

Assume that the parity lattices are Poltyrev capacity approaching[1]. In other words, in (5.9) for any $\Delta > 1$ we assume that $P_e(opt,\Delta) \approx 0$ and $P_e(dec,\Delta) \approx P(x \notin \mathcal{T})$. Can we decode at 0 dB with the proposed decoding paradigm (without considering the complexity issue)? To answer this question, we theoretically analyse the behavior of (5.16) to characterize the performance the modified decoding algorithm (without the splitting strategy). The theoretical analysis of (5.21) is left for future work.

Without loss of generality, let $c=2$, $L_2 = \mathbb{Z}^2$, and $n = 2\cdot k^t$. Assume that a regular list decoder is used for the lattice L_2. Consider Δ expressed in dB. We approximate the log log behavior of $U_2(\delta,\Delta)$ by an affine function (which is reasonable if the range for Δ is not too large, see e.g. the curve $t=0$ on Figure 5.2a). We have

[1] It is still to be proven, but we conjecture that they are.

$$
\begin{aligned}
U_2(\delta,\Delta) &= F(2, f(\delta), f(\Delta)) = e^{-r\pi e 10^{\frac{\Delta}{10}}},\\
&\approx \min\{10^{-a\Delta-b}, 1\}
\end{aligned}
\tag{5.22}
$$

The quantity $P(x \notin \mathcal{T}) = U_{2\cdot k^t}(\Delta)$ has the following behavior.

Theorem 5.4. *Let $L_c = \mathbb{Z}^2$ and $n = 2\cdot k^t$. Assume that $U_2(\Delta) \approx \min\{10^{-a\Delta-b}, 1\}$, where Δ is expressed in dB and a is large enough. Then, $U_{2\cdot k^t}(\Delta)$, given by (5.16), has the following behavior.*

$$
\begin{aligned}
&U_{2\cdot k^t}(\Delta) \approx 1 \ \textit{if}\ \Delta < \mathscr{T}_t,\\
&\log_{10}(U_{2\cdot k^t}(\Delta)) \approx -2^t a\Delta - b_t \ \textit{if}\ \Delta > \mathscr{T}_t.
\end{aligned}
\tag{5.23}
$$

with $-b_t = 2^t a \mathscr{T}_t$ and where the threshold $\mathscr{T}_t$ increases at each recursive step as

$$
\mathscr{T}_t = \mathscr{T}_{t-1} + \frac{10\log_{10}(2)}{k} + \frac{\log_{10}(\frac{k^2-k}{2})}{2^t a},
\tag{5.24}
$$

with

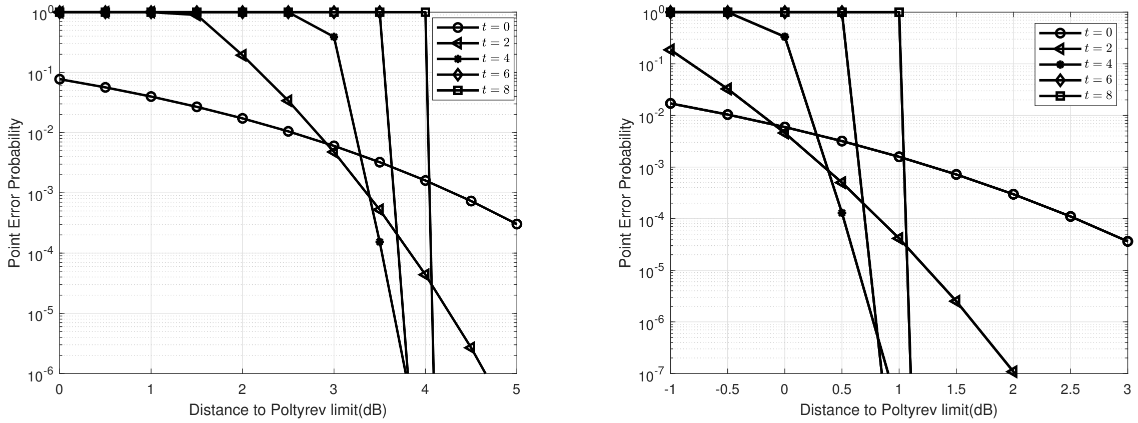
$$
\mathscr{T}_0 \approx -\frac{b}{a}.
\tag{5.25}
$$

Consequently, the main parameters are a, b, and k where:

- a determines the number of recursive steps needed to reach the desired slope (error exponent) $2^t a$.
- The initial threshold $\mathscr{T}_0$ is a function of b and a.
- k determines the speed at which the threshold moves to the right.

Moreover, we see that the $\binom{k}{2}$ coefficient may have a significant effect on the threshold for the first recursion, but after a few recursive steps it becomes negligible as the slope of the curve $2^t a$ increases.

As examples, Figures 5.2 show (5.16), with $L_c = \mathbb{Z}^2$, $n = 2\cdot k^t$, $U_2(\delta,\Delta) = F(2, f(\delta), f(\Delta))$, and where $k = 12$ with two different δ. The value of δ determines the parameters a and b, and thus the location of the thresholds. Note that a and thus the asymptotic slope of the curves does not vary significantly with δ.



(a) $\delta = 0.3$

(b) $\delta = 0.6$

Figure 5.2: Numerical evaluation of (5.16) for $k = 12$, the initial condition as (5.17), $\Lambda_c = \mathbb{Z}^2$, and with two different values of δ. The location of the threshold depends on δ. On the contrary, the asymptotic slope of the curves does not vary significantly with δ.

Proof. Assuming that a is large enough, the second element of (5.16) is negligible. Therefore, $U_n(\Delta)$ is estimated as

$$\begin{aligned} U_n(\Delta) &\approx \prod_{i=0}^{t-1} \left(\frac{k^2-k}{2}\right)^{2^i} \cdot \left(U_2(\delta, \frac{\Delta}{2^{\frac{t}{k}}})\right)^{2^t} \\ &= \prod_{i=0}^{t-1} \left(\frac{k^2-k}{2}\right)^{2^i} \cdot 10^{2^t \cdot [-a(\Delta - \frac{t \cdot 10 \cdot \log_{10}(2)}{k}) - b]}. \end{aligned} \tag{5.26}$$

and thus

$$\log_{10}(U_n(\Delta)) \approx -a2^t\Delta + 2^t \left[\frac{a \cdot t \cdot 10 \cdot \log_{10}(2)}{k} + \log_{10}\left(\frac{k^2-k}{2}\right) - b\right]. \tag{5.27}$$

This first model does not take into account the fact that $U_n(\Delta) \leq 1$. Indeed, there is a threshold located at

$$\Delta = \mathscr{T}_0 \approx -\frac{b}{a}. \tag{5.28}$$

Consequently, the function is more accurately represented by a piecewise affine function with two pieces: A first piece which is equal to 1 and a second piece which decreases with Δ.

$$\begin{aligned} &\log_{10}(U_2(\Delta)) \approx 0 \text{ if } \Delta < \mathscr{T}_0, \\ &\log_{10}(U_2(\Delta)) \approx -a\Delta - b \text{ if } \Delta > \mathscr{T}_0, \end{aligned} \tag{5.29}$$

If follows that (first without considering the effect of the $\binom{k}{2}$ coefficient)

$$\begin{aligned} &\log_{10}(U_{2\cdot k^t}(\Delta)) \approx 0 \text{ if } \Delta < \mathscr{T}_t', \\ &\frac{\log_{10}(U_{2\cdot k^t}(\Delta))}{\log_{10}(\frac{k^2-k}{2})} \approx -2^t a\Delta - b_t' \text{ if } \Delta > \mathscr{T}_t', \end{aligned} \tag{5.30}$$

with $-b_t' = 2^t a \mathscr{T}_t'$ and where the threshold $\mathscr{T}_t'$ evolves as

$$\mathscr{T}_t' = \mathscr{T}_{t-1} + \frac{10\log_{10}(2)}{k}. \tag{5.31}$$

Finally, we add the effect of the $\binom{k}{2}$ coefficient to get the behavior of $U_{2\cdot 2^t}(\Delta)$:

$$\begin{aligned} &\log_{10}(U_{2\cdot k^t}(\Delta)) \approx 0 \text{ if } \Delta < \mathscr{T}_t, \\ &\log_{10}(U_{2\cdot k^t}(\Delta)) \approx -2^t a\Delta - b_t \text{ if } \Delta > \mathscr{T}_t. \end{aligned} \tag{5.32}$$

with $-b_t = 2^t a \mathscr{T}_t$ and where the threshold $\mathscr{T}_t$ is

$$\mathscr{T}_t = \mathscr{T}_t' + \frac{\log_{10}(\frac{k^2-k}{2})}{2^t a}. \tag{5.33}$$

□

Unfortunately, the (worst-case) decoding complexity in this situation, i.e. without the splitting strategy, is super-polynomial in the dimension: It has the form $n^{c\cdot k \log n}$, where c is a constant. This underlines the interest of the splitting strategy.

Finally, one could wonder if the rate at which slope increases could be improved. One could for instance consider a stronger code than a single parity check such that at least three lists, instead of two, should not contain the good candidate for the decoding to fail. The slope would then increase as $3^t a$.

Similarities with LDPC codes and polar codes. We propose a brief summary of the analysis in Sections 2.3.3 and 2.3.2, where we argue that LDPC codes, polar codes, and parity lattices involve the same underlying idea: Namely, small component codes (e.g. single parity-check codes) are stacked in a tree such that the reliability of the symbol estimates produced at each level of the tree increases as one goes up in the tree. The two key ingredients are:

- A tree.
- Combining operations at each level of the tree.

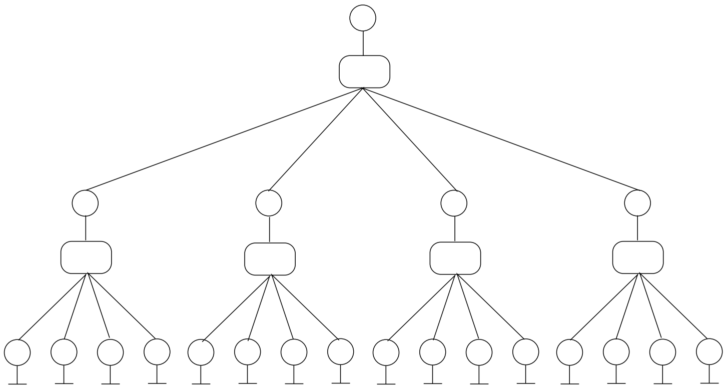


Figure 5.3: Tree representation of the parity lattices with $k = 4$. The half edges on the variable nodes represent a channel observation. At each level of the tree, four symbol estimates are combined to produce a more reliable estimate. The rounded squares denote a combining operation via a parity check.

Whereas one of the two elements is obtained via asymptotic dimensions for LDPC and polar codes, they are obtained in non-asymptotic dimensions with the parity lattices. Regarding LDPC codes, unfolding the Tanner graph around a variable node yields a local tree. However, this tree structure is obtained by choosing randomly a large enough sparse parity-check matrix in large dimensions. Nevertheless, the combining operations based on parity checks have nothing specific to large dimensions. On the contrary, the tree for polar codes, based on the $(u, u + v)$ recursion, is obtained deterministically. The large dimension is necessary for the combining operations, which become very simple and efficient thanks to the polarization phenomenon. As a result, if one wants an efficient code in moderate dimensions, one can consider the combining operations of LDPC codes based on single parity checks and a deterministic tree similar to the one of polar codes. This yields the tree on Figure 5.3. This latter tree represents the parity lattices.

5.2 Parity lattices with $k = 2$ and $k = n^{\frac{1}{\log \log n}}$

We further study two families of parity lattices: We treat the case $k = 2$ (BW lattices) and the case $k = n^{\frac{1}{\log \log n}}$. The recursive decoders yield regular list decoders with a complexity linear in the list size for both family of lattices. With $k = n^{\frac{1}{\log \log n}}$, the list size is reduced and so is the regular list-decoding complexity, but at the cost of a lower coding gain : $O(\log n)$ versus $O(\sqrt{n})$ for BW lattices. Our results are summarized in Figure 5.4.

The main idea to bound the list size of the parity lattices, when $\delta > 1/2$, is to compute the maximum number of elements outputted by the recursive algorithms presented in Sections 5.1.2 without the removing step. As a result, when a recursive call is made with $\delta > 1/2$ the algorithm do not need to include the removing step (Step 11 in Algorithm 5.3), but it should if $\delta \leq 1/2$ (because we use the bounds of Theorem 4.1 in this latter case).

The behavior of the algorithm on the Gaussian channel is also investigated for $k = 2$. We analytically estimate the average complexity for quasi-optimal decoding of BW_n. It is shown to be quadratic in the dimension up to $n = 64$ and quartic for $n = 128$. Moreover, we also highlight that the average complexity of the proposed BDD on the Gaussian channel is $O(n \log n)$.

5.2.1 Parity lattices with $k = 2$ (BW lattices)

We study the parity lattices obtained with $k = 2$, $L_c = \mathbb{Z}^2$ and $\theta = \phi = 1+i$. They are called Barnes-Wall lattices in the literature. These lattices are expressed as

$$BW_{2n} = \Gamma(\phi BW_n, \beta, 2) \text{ where } BW_2 = \mathbb{Z}^2, \ \theta = \phi. \tag{5.34}$$

In general, the lattice $\phi BW_n = R_\phi BW_n$ is denoted by RBW_n [For88b]. We adopt this notation for the rest of the document.

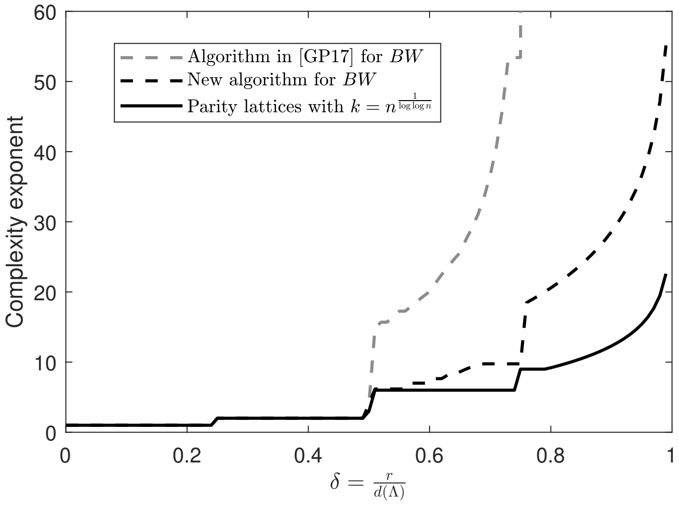


Figure 5.4: Bounds on the (worst-case) regular list decoders complexity with respect to the decoding radius. Three curves are displayed for the complexity of: The list decoder in [GP17] for BW lattices, the new algorithm for BW lattices, and the new algorithm for the parity lattices with $k = n^{\frac{1}{\log\log n}}$.

Note that for any parity-check k-group with $k = 2$ we have

$$\begin{aligned}\Gamma(V,\beta,2)_{\mathcal{P}} &= \{(v_1' + n_1, v_2' - n_1),\ v_i' \in V, n_1 \in \beta\},\\ &= \{(v_1' + m, v_3' + m),\ v_i' \in V, m \in \beta\},\\ &= \Gamma(V,\beta,2)_{\mathcal{R}},\end{aligned} \tag{5.35}$$

where we set $n_1 = m$. This representation goes under the name squaring construction [For88b]. Hence, this squaring construction defines BW lattices as

$$BW_{2n} = \{(\underbrace{v_1' + m}_{u_1 \in BW_n}, \underbrace{v_2' + m}_{u_2 \in BW_n}), v_i' \in RBW_n, m \in [BW_n/RBW_n]\}. \tag{5.36}$$

The squaring construction can be expressed under the form of the Plotkin $(u, u+v)$ construction [Plo60]:

$$\begin{aligned}\Gamma(V,\beta,2)_{\mathcal{R}} &= \{(v_1' + m, v_3' + m),\ v_i' \in V, m \in \beta\},\\ &= \{(v_1' + m, \underbrace{v_1' + v_2}_{v_3'} + m),\ v_i' \in V, m \in \beta\},\\ &= \{(u_1, u_1 + v_2), u_1 \in T, v_2 \in V\}.\end{aligned}$$

Finally, BW lattices can also be presented via the two-level squaring construction (see [For88b, Section IV.B]):

$$BW_{2n} = \Gamma(V,\alpha,\beta,4), V = RBW_{n/2},\ \alpha = [BW_{n/2}/RBW_{n/2}],\ \beta = [RBW_{n/2}/2BW_{n/2}]. \tag{5.37}$$

These lattices were one of the first series discovered with an infinitely increasing fundamental coding gain [BW59]: It increases as $\gamma(BW_n) = \sqrt{2}\cdot\gamma(BW_{n/2}) = \sqrt{n/2}$ (using Lemma 5.1). This series includes dense lattices in lower dimensions such as D_4, E_8, Λ_{16} [CS99], and is deeply related to Reed-Muller codes [For88b] [MS77]: BW lattices admit a Construction D based on these codes.

Several algorithms have been proposed to decode BW lattices. [For88b] uses the trellis representation of the two-level squaring construction to introduce an efficient MLD algorithm for the low dimension instances of BW_n. Nevertheless, the complexity of this algorithm is intractable for $n > 32$: Equation (4.15) (which gives the number of edges in the trellis) applied to (5.37) yields $2 \cdot 2^{2n/8} + 2 \cdot 2^{3n/8}$, e.g. decoding in BW_{128} involves $2 \cdot 2^{48} + 2 \cdot 2^{32}$ decoders of BW_{32}. Forney states in [For88b] : "The first four numbers in this sequence[2], i.e., 2, 4, 16, and 256, are well behaved, but then a combinatorial explosion occurs: 65 536 states for BW_{64}, which achieves a coding gain of 7.5 dB, and more than four billion states for BW_{128}, which achieves a coding gain of 9 dB. This explosion might have been expected from capacity and R_0 (cutoff rate) considerations".

[2] Forney refers to the number of states per section of the trellis, which is $2^{2n/8}$

Later, [MN08] proposed the first BDDs running in polynomial time; a parallelisable one of complexity $O(n^2)$ and a sequential one of complexity $O(n \log^2 n)$. The parallelisable decoder was generalized in [GP17] to work beyond the packing radius, still in polynomial time. It is discussed later in the chapter. The sequential decoder uses the BW multilevel construction to perform multistage decoding: Each of the $\approx \log n$ levels is decoded with a Reed-Muller decoder of complexity $n \log n$. This decoder was also further studied, in [HVB13], to design practical schemes for communications over the AWGN channel. However, the performance of this sequential decoder is far from MLD. A simple information-theoretic argument explains why multistage decoding[3] of BW lattices cannot be efficient: The rates of some component Reed-Muller codes exceed the channel capacities of the corresponding levels [FTS00] [YLW13].

As a result, no BW decoders, being both practical and quasi-optimal on the Gaussian channel, have been designed and executed for dimensions greater than 32.

A new BDD

We adapt Algorithm 4.1 to Barnes-Wall lattices to get Algorithm 5.5.

Algorithm 5.5 Double-sided $(u, u+v)$ decoder of BW_{2n}

```
Input: $y = (y_1, y_2) \in \mathbb{R}^{2n}$.
Decode (BDD) $y_1, y_2$ in $BW_n$ as $u_1, u_2$.
Decode (BDD) $y_2 - u_1$ in $RBW_n$ as $v_2$. Store $\hat{x} \leftarrow (u_1, u_1 + v_2)$.
Decode (BDD) $y_1 - u_2$ in $RBW_n$ as $v_1$. Store $\hat{x}' \leftarrow (u_2 + v_1, u_2)$.
Return $x_{dec} = \underset{x \in \{\hat{x}, \hat{x}'\}}{\text{argmin}} ||y - x||$
```

Given an element $y = (y_1, y_2)$ to be decoded, a trivial algorithm for a code obtained via the $(u, u+v)$ construction is to first decode y_1 as u_1, and then decode $y_2 - u_1$ as v_2 (see the end of Section 5.2.1 for more details). This algorithm could be called a single-sided decoder or a successive-cancellation decoder. Algorithm 5.5 can be seen as a double-sided $(u, u+v)$ decoder since we also decode y_2 as u_2 and then $y_1 - u_2$ as v_2. Theorem 5.5 shows that Algorithm 5.5 is a BDD with an error-correction radius equal to $\rho^2(BW_{2n})$.

Theorem 5.5. *Let y be a point in $\mathbb{R}^{2n}$ such that $d(y, BW_{2n})$ is less than $\rho^2(BW_{2n})$. Then, Algorithm 5.5 outputs the closest lattice point $x \in BW_{2n}$ to y.*

Proof. The result follows from Lemma 4.1 since Algorithm 5.5 is a special case of Algorithm 4.4. □

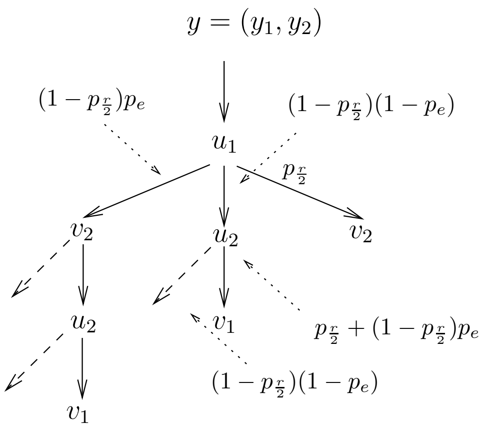
Algorithm 5.1, with $k = 2$ and $\theta = \phi$, is the recursive version of Algorithm 5.5. This algorithm is similar to the parallel decoder of [MN08]. The main difference is that [MN08] uses the automorphism group of BW_{2n} to get four candidates at each recursion whereas we generate only two candidates. Nevertheless, both our algorithm and [MN08] have four recursive calls at each recursive section and have the same asymptotic complexity. We apply Theorem 5.1 for the case $k = 2$.

Theorem 5.6. *Let $n = 2^{t+1}$ and $y \in \mathbb{R}^n$. If $d(y, BW_n) < \rho^2(BW_n)$, then Algorithm 5.1 outputs the closest lattice point to y in time $O(n^2)$.*

Could a similar algorithm have a quasi-linear complexity? The quadratic complexity is due to the four recursive calls. Having only two recursive calls would make it quasi-linear ($O(n \log n)$). Therefore, if we knew beforehand whether we should go for $(u_1, u_1 + v_2)$ or $(u_2 + v_1, u_2)$ the algorithm would be of quasi-linear complexity. Of course, it is not possible to have this information before decoding. Nevertheless, we can re-organize the recursive calls in order to minimize the average number of recursive calls.

Let the point to be decoded be $y = x + w$, where $y \in \mathbb{R}^{2n}$, $x = (x_1, x_2) \in BW_{2n}$, $w = (w_1, w_2) \in \mathbb{R}^{2n}$, and $w_1, w_1 \sim \mathcal{D}$, and where $\mathcal{D}$ is an arbitrary probability distribution and where w_1 and w_2 are i.i.d. Let $r = 1/4 \cdot d(BW_{2n})$ ($\delta = 1/4$). We denote by $p_r : P(||w_i||^2 < r)$ and $p_e : P(d(y_1, u_1) < \frac{r}{2} \mid ||w_1|| > \frac{r}{2})$. An overview of the modified algorithm is presented in Figure 5.5. See Appendix 5.3.3 for more details and the proof of the next Theorem.

[3] Where only one candidate is decoded at each level.



$y = (y_1, y_2)$
$(1 - p_{\frac{r}{2}})p_e$
$(1 - p_{\frac{r}{2}})(1 - p_e)$
u_1
$p_{\frac{r}{2}}$
v_2
u_2
v_2
u_2
v_1
$p_{\frac{r}{2}} + (1 - p_{\frac{r}{2}})p_e$
$(1 - p_{\frac{r}{2}})(1 - p_e)$
v_1

Figure 5.5: Probabilistic tree representing the possible behavior of the BDD with the re-organized recursive calls. To be read from the top to the bottom. y_1 is first decoded in BW_n as u_1. If $||w_1|| < r/2$ (with a probability $p_{\frac{r}{2}}$) then v_2 is decoded (right arrow). Otherwise, either $d(y_1, u_1) > r/2$ or $d(y_1, u_1) < r/2$ (but where $u_1 \neq x_1$, the point sent) and the tree presents the induced events. As a result, a black arrow represents one recursive call. The number of recursive calls to reach one outcome is obtained by counting the black arrows in the given path. The average number of recursive calls made wihtin one recursive section depends on the noise statistics.

Lemma 5.3. *Let $n = 2^{t+1}$ and $y \in \mathbb{R}^n$. If $d(y, BW_n) < \rho^2(BW_n)$, a modified version of Algorithm 5.1 outputs the closest lattice point to y in time:*

$$\begin{aligned}\mathfrak{C}(n,r) <& \mathfrak{C}(\frac{n}{2},\frac{r}{2})[\mathbf{2}(p_{\frac{r}{2}} + [(1-p_{\frac{r}{2}})(1-p_e)]^2)+ \\ & \mathbf{3}(1-p_{\frac{r}{2}})(1-p_e)(p_{\frac{r}{2}} + (1-p_{\frac{r}{2}})p_e) + \mathbf{4}(1-p_{\frac{r}{2}})p_e] + O(n).\end{aligned} \tag{5.38}$$

Consequently, the complexity depends on the noise statistics. See Section 5.2.1 for the case of Gaussian noise.

List decoding BW lattices

As mentionned at the beginning of the section, [GP17] adapts the parallel BDD of [MN08], which uses the automorphism group of BW_n, to output a list of all lattice points lying at a distance $r = d(BW_n)(1-\epsilon)$, $0 < \epsilon \leq 1$, from any $y \in \mathbb{R}^n$ in time

$$O(n^2) \cdot L(n, r^2)^2. \tag{5.39}$$

A critical aspect regarding the complexity of this decoder is the list size. Theorem 4.1 provides bounds on the list size when $r \leq d(BW_n)/2$. The following lemma, addressing $r > d(BW_n)/2$, is proved in [GP17].

Lemma 5.4 (Results from [GP17])**.** *The list size of BW_n lattices is bounded as [GP17]:*

- *$L(n,r) = O(n^{2\log_2 24})$ if $r = \frac{3}{4}d(BW_n)$.*
- *And*

$$L(n,r) = O(n^{16\log_2 \frac{1}{\epsilon}}), \tag{5.40}$$

  if $r \leq d(BW_n)(1-\epsilon)$, $0 < \epsilon < \frac{1}{4}$.

The case $d(BW_n)/2 < r^2 \leq d(BW_n)(3/4 - \epsilon)$ is not explicitly proved in [GP17], but it is similar to Lemma 2.5 of the paper. We state it below and prove it via Algorithm 5.3, which implements the first splitting strategy (but without the second splitting strategy).

Lemma 5.5. *The list size of BW_n lattices is bounded as*

$$L(n,r) = O(n^{\log_2 4\lfloor\frac{3}{4\epsilon}\rfloor}) \tag{5.41}$$

if $r \leq d(BW_n)(\frac{3}{4} - \epsilon)$, $0 < \epsilon < \frac{1}{4}$.

Proof. The maximum number of elements outputted by Algorithm 5.3 without the removing step (for the recursive calls with $\delta > 1/2$) is an upper bound on the list size $l(n, \delta)$. Hence, a non-recursive bound is given by Equation (4.22), which is the maximum number of elements in the output list of Algorithm 4.5 (the non-recursive version of Algorithm 5.3). To lighten the notations, we write $l(\delta)$ for $l(n/2, \delta)$. Unwinding (4.22) and using Theorem 4.1 for $l(2/3\delta)$ yields

$$\begin{aligned} l(n,\delta) \leq & 2\big[l(\frac{2}{3}\delta)l(\delta) + l(\delta)l(\frac{2}{3}\delta)\big] = 4l(\frac{2}{3}\delta)l(\delta) = 4\lfloor\frac{3}{4\epsilon}\rfloor l(\delta), \\ = & \left(4\lfloor\frac{3}{4\epsilon}\rfloor\right)^{\log_2 n} \cdot l(\mathbb{Z}_2, \delta), \\ = & O(n^{\log_2 4\lfloor\frac{3}{4\epsilon}\rfloor}). \end{aligned} \tag{5.42}$$

□

The proof highlights that the removing step should be kept for $\delta \leq 1/2$ because we rely on the bounds given by Theorem 4.1 for this situation.

Equation (5.40) shows that the list size of BW lattices is of the form $n^{O(\log \frac{1}{\epsilon})}$ and thus polynomial in the lattice dimension for any radius bounded away from the minimum distance. Combining (5.39) with (5.40), the list decoder complexity becomes $n^{O(\log \frac{1}{\epsilon})}$ for any $r < d(BW_n)(1-\epsilon)$, $\epsilon > 0$. This result is of theoretical interest: It proves that there exists a polynomial time decoding algorithm (in the dimension) for any radius bounded away from the minimum distance.

First, we show that the constant in (5.40) can be improved from 16 down to 8, as stated by the following lemma, proved in Appendix 5.3.4.

Lemma 5.6 (Improved constant)**.** *The list size of the BW_n lattices is bounded as:*

- *$L(n, r) = O(n^{2\log_2 12})$ if $r = \frac{3}{4}d(BW_n)$.*

And

$$L(n, r) = O(n^{8\log_2 \frac{1}{\epsilon}}), \tag{5.43}$$

if $r \leq d(BW_n)(1-\epsilon)$, $0 < \epsilon \leq \frac{1}{4}$.

While this lemma enables to improve the bound on the complexity of the algorithm of [GP17], the quadratic dependence in the list size remains a drawback: As explained at the end of Section 4.3.3, finding an algorithm with quasi-linear dependence in the list size is stated as an open problem in [GP17].

In the following, we demonstrate that if we use our decoding paradigm with the first splitting strategy (i.e. Algorithm 5.3), rather than the automorphism group of BW_n for list decoding, we get complexity linear in the list size. This enables to both improve the list-decoding complexity and get a practical quasi-optimal decoding algorithm on the Gaussian channel up to $n = 128$.

We compute below the complexity of our algorithm for $\delta < 9/16$. The complexity analysis for larger δ (which is the proof of Theorem 5.7) is provided in Appendix 5.3.6. The main idea of the analysis consists in combining the splitting strategies as recommended in Section 5.1.2 and analysing the induced complexity.

If $\delta < \frac{3}{8}$ then $\frac{2}{3}\delta < \frac{1}{4}$ and we have $l(\delta) = O(1)$, $l(\frac{2}{3}\delta) = 1$. Moreover, $\mathfrak{C}(\frac{2}{3}\delta) \leq \mathfrak{C}(\frac{1}{4}) = O(n^2)$ (Theorem 5.6). The baseline equation is (4.23), which becomes

$$\mathfrak{C}(n,\delta) = 4\mathfrak{C}(\delta) + l(\delta)O(n^2) = l(\delta)O(n^2 \log n) = \widetilde{O}(n^2). \tag{5.44}$$

If $\frac{3}{8} \leq \delta < \frac{9}{16}$ and $\mathfrak{C}(\frac{2}{3}\delta) \leq \mathfrak{C}(\frac{3}{8}) = O(n^2 \log n)$. Equation (4.23) becomes

$$\begin{aligned} \mathfrak{C}(n,\delta) = & l(\delta)O(n^2 \log n) \sum_{i=0}^{\log_2 n} \left(\frac{2l(\frac{2}{3}\delta) + 2}{4}\right)^i, \\ = & l(\delta)O(n^{1+\log_2[1+l(\frac{2}{3}\delta)]} \log n), \\ = & l(\delta)\tilde{O}(n^{1+\log_2[1+l(\frac{2}{3}\delta)]}), \end{aligned} \tag{5.45}$$

which is $\tilde{O}(n^{1+\log_2 3})$ if $\delta < 1/2$.

Note that for these cases ($\delta < 1/2$) the decoder of [GP17] is more efficient: Indeed, Theorem 4.1 shows that when $\delta < 1/2$, then $l(n,\delta) = O(1)$ and the decoding complexity, given by (5.39), is $O(n^2)$. Nevertheless, the following theorem shows that our decoder is better for larger values of δ and, as we shall see in the next subsection, is useful even when $\delta < 1/2$ for quasi-optimal decoding on the Gaussian channel.

Dimension n	16	32	64	128	256
Dist. to Polt. (dB) sphere bound	4.05	3.2	2.5	1.9	1.4
Dist. to Polt. (dB) MLD	4.5	3.7	3.1	2.3	?

Table 5.1: Sphere lower bound on the best performance achievable by any lattice Λ for $P_e^{\Lambda}(opt, n, \sigma^2) = 10^{-5}$ and MLD performance of BW_n for $P_e(opt) = 10^{-5}$.

Theorem 5.7. *Let $n = 2^{t+1}$, $y \in \mathbb{R}^n$. The set $BW_n \cap B_\delta(y)$ can be computed in worst-case time:*

- $O(n^2)$ *if* $\delta < \frac{1}{2}$ *(algorithm of [GP17]).*
- $l(\delta)O(n^{2+\log_2\lceil\frac{l(\frac{2}{3}\delta)+1}{2}\rceil}) \approx O(n^{1+\log_2 4\lceil\lfloor\frac{3}{4\epsilon}\rfloor^2\rceil})$ *if* $\delta = \frac{3}{4} - \epsilon$, $0 < \epsilon$.
- $O(n^{1+\log_2 432})$ *if* $\delta = \frac{3}{4}$.
- $l(\delta)O(n^2) = O(n^{8\log_2\frac{1}{\epsilon}+2})$ *if* $\delta = 1 - \epsilon$, $0 < \epsilon < \frac{1}{4}$.

Decoding on the Gaussian channel

We apply the analysis presented in Section 5.1.3 to the case $k = 2$ to establish the smallest list-decoding radius δ required for quasi-optimal decoding. The first element needed is the MLD performance $P_e^{BW_n}(opt, n, \sigma^2)$ of BW_n. As mentioned earlier it is not known for $n > 32$. Nevertheless, $P_e^{\Lambda}(opt, n, \sigma^2)$ can be lower-bounded for *any* lattice Λ in n dimensions using the sphere lower bound (see Section 3.3.2). Table 5.1 provides the sphere lower bound on the best performance achievable for $P_e^{\Lambda}(opt, n, \sigma^2) = 10^{-5}$. With (5.12), we can compute the smallest δ, for the corresponding values of σ^2, such that $P(x \notin \mathcal{T}) = P(||w||^2 > r) \lessapprox 10^{-5}$ with regular list decoding. This value is denoted δ^*. Using δ^* yields quasi-optimal decoding performance, regardless of the MLD performance of BW_n. The values of δ^* as a function of n are presented in Figure 5.6. The corresponding (worst-case) decoding complexity is obtained with Theorem 5.7. It is super-quadratic for all $n \geq 16$.

Running the simulations (with δ^* found at the sphere bound) enables to estimate the MLD performance of BW_n lattices. The results are presented[4] in Table 5.1, and are at ≈ 0.5 dB of the sphere bound[5]. In Figure 5.6 the corresponding values of δ^* (still with regular list decoding) are depicted by the diamonds.

As explained in Section 5.1.3, the error probability of the list-decoding algorithm without the removing step can be estimated with Equation (5.16) (without the splitting strategy) or (5.21) (with the first splitting strategy). Hence, we can also compute the smallest δ such that with this algorithm $P(x \notin \mathcal{T}^{\delta}) \lessapprox 10^{-5}$ (at the MLD performance). We shall also denote this value by δ^*. However, the decoding complexity should be updated to take into account the fact that there is no removing step in the algorithm even when $\delta < 1/2$.

We consider the modified decoding algorithm with the first splitting strategy (Algorithm 5.3 without the removing step). To mitigate the complexity and simplify the analysis, whenever $\delta \leq 1/4$ we shall use the BDD presented in Algorithm 5.1. Hence, in (5.21), $U_n(\delta \leq \frac{1}{4}, \Delta) = P_e(BDD, n, \Delta)$. The error probability $P_e(BDD, n, \Delta)$ is shown in Figure 5.7. Note that due to the rounding operation in $\mathbb{Z}^2$, this BDD has better performance than if a sphere of relative radius $\delta = 1/4$ (the initial condition in (5.17)) were used: MLD decoding in $\mathbb{Z}^2$ is performed. In the literature, the performance of BDDs is often estimated via the "effective error coefficient" [FV96] [SA06]. Nevertheless, it it not always accurate, especially in high dimensions. We therefore rely on the Monte Carlo simulations presented in Figure 5.7 for $P_e(BDD, n, \Delta)$. The estimated δ^* with this decoder, shown in Figure 5.6, are significantly smaller than the ones obtained with the regular list decoder. In particular, $\delta^* < 3/8$ for $n \leq 64$ and $\delta^* < 1/2$ for $n = 128$.

We now study the complexity of this latter algorithm. We shall use the notation $l'(n, \delta, y)$ to denote the number of elements returned by the algorithm without the removing step.

[4] These estimations were not performed with the regular list decoder, but with the algorithm presented in the rest of the section.

[5] We have not yet investigated the case $n = 256$.

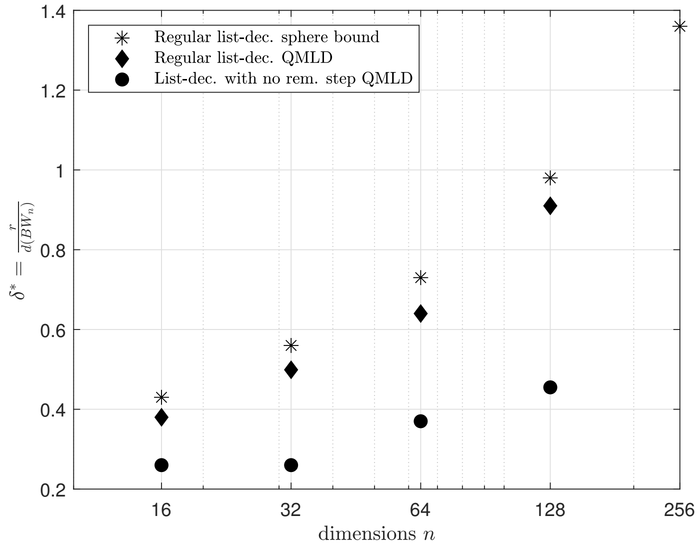


Figure 5.6: Values of the list-decoding relative radius δ^*, for BW_n, such that $P(x \notin \mathcal{T}) \approx 10^{-5}$.

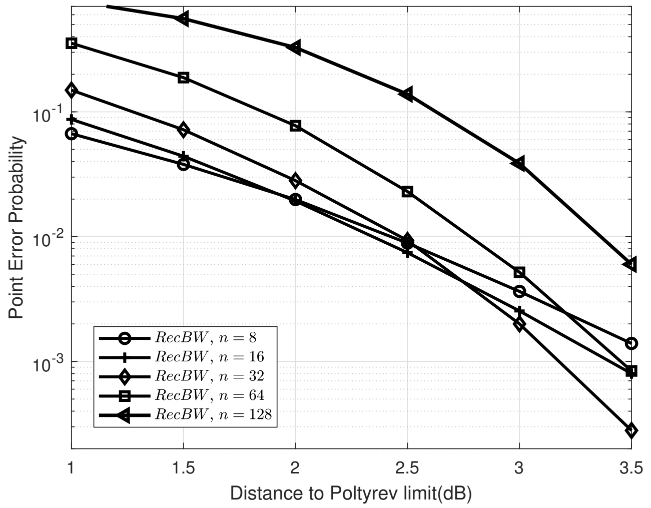


Figure 5.7: Performance of the recursive BDD for the Barnes-Wall lattices on the Gaussian channel.

If $\delta \leq 3/8$ Equation (5.42) becomes

$$\begin{aligned} l'(n, \delta) \leq & 2\big[l'(\frac{1}{4})l'(\delta) + l'(\delta)l'(\frac{1}{4})\big] = 4l'(\delta) \\ =& 4^{\log_2 n} \cdot l(\mathbb{Z}_2, \delta) = O(n^2). \end{aligned} \tag{5.46}$$

However, considering the average complexity, and taking into account the removed duplicates (see Step 12 in Algorithm 5.3), one has

$$\begin{aligned} E_y[l'(n, \delta, y)] \leq & 2\big[l'(\frac{1}{4})E_y[l'(\delta)] + E_y[l'(\delta)]l'(\frac{1}{4}) - l'(\frac{1}{4})l'(\frac{1}{4})\big] - E_y^c. \\ =& 4E_y[l'(\delta)] - 2 - E_y^c, \end{aligned} \tag{5.47}$$

where E_y^c denotes the average number of common elements in the lists obtained with the $(u, u+v)$ recursion and $(u+v, u)$ recursion. We observed experimentally that for $\delta \leq 3/8$, $E_y[l'(n, \delta, y)]$ is close to 1. This observation is not taken into account in the next theorem, which bounds the average list size and the average complexity. It is however in the interpretation following the theorem.

Theorem 5.8. *Let $E_y[l'(n, \delta, y)]$ be the average list size of Algorithm 5.3 without the removing step. Let η denote $E_y[l'(n, 3/8, y)]$. $E_y[l'(n, \delta, y)]$ is bounded from above as*

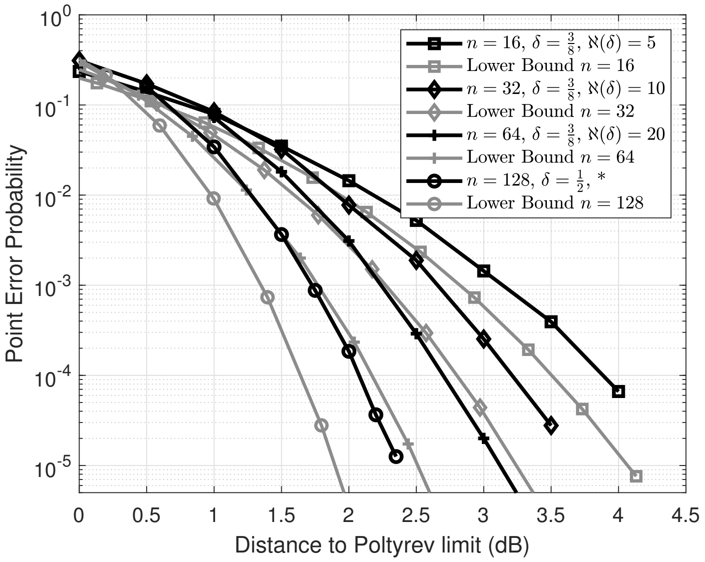


Figure 5.8: Modified Algorithm 5.3 for the BW lattices up to $n = 128$ and the sphere lower bounds. *For $n = 128$, $\aleph(\delta) = 1000$ and $\aleph(2/3\delta) = 4$.

- $E_y[l'(n, \delta, y)] = O(n^{2+\log_2 \eta \cdot})$ *if* $3/8 < \delta \leq 9/16$.

The average complexity is bounded from above as:

- $E_y[\mathfrak{C}(n, \delta)] = \eta \widetilde{O}(n^2)$ *if* $\delta \leq 3/8$.
- $E_y[\mathfrak{C}(n, \delta)] = E_y[l'(\delta, y)] \widetilde{O}(n^{1+\log_2[1+\eta]})$ *if* $3/8 < \delta \leq 9/16$.

See Appendix 5.3.7 for the proof.
As a result, based on the observation that η is close to 1, the average complexity is estimated as:

- $E_y[\mathfrak{C}(n, \delta)] = \widetilde{O}(n^2)$ if $\delta \leq 3/8$.
- $E_y[\mathfrak{C}(n, \delta)] = \widetilde{O}(n^4)$ if $3/8 < \delta \leq 9/16$.

Since $\delta^* < 3/8$ for $n < 64$ and $\delta^* < 1/2$ for $n = 128$ for quasi-MLD, we conclude that the decoding complexity is quadratic for $n \leq 64$ and quartic for $n = 128$.

For a practical implementation, we can bound the maximum number of points kept at each recursive step: I.e. instead of being removed, Step 11 of Algorithm 5.3 is modified such that the $\aleph(\delta)$ best candidates are kept at each recursive step. The size of the list $\aleph(\delta)$, for a given δ, is a parameter to be fine tuned: For n =16, 32, 64, we set $\aleph(\delta)$ =5, 10, 20, respectively. For $n = 128$, $\aleph(\delta) = 1000$ (<< than our bound in $O(n^2)$ on $E_y[l'(\delta, y)]$) and $\aleph(2/3\delta) = 4$ yields quasi-MLD performance. Figure 5.8 depicts the simulation results for BW lattices up to $n = 128$.

Probabilistic improvement.
In the complexity formula (5.38) for the BDD, given Gaussian noise, p_r is computed with (5.12). p_e is more difficult to compute but is typically very small, even with large noise variance, and decreases with the dimension as the relative volume of the packing spheres decreases. We therefore make the approximation that it is equal to 0. Figure 5.9 presents the complexity exponent of (5.38) as a function of the dimension. It is quasi-linear. We made the simulations with the noise variance at the maximum value (equivalent to the Poltyrev limit). When running this BDD we indeed observed that it has quasi-linear complexity.

Can we apply the same idea to list decoding? Unfortunately, Lemma 5.8 (in appendix) does not hold if the decoding radius $r > d(BW_{2n})/4$. The probability that a point x is not in the outputted list must therefore be updated:

$$P(x \notin \mathcal{T}) \leq P(u_1 \notin \mathcal{T}_1)^2 + 2P(v_2 \notin \mathcal{V}_2) + P'. \tag{5.48}$$

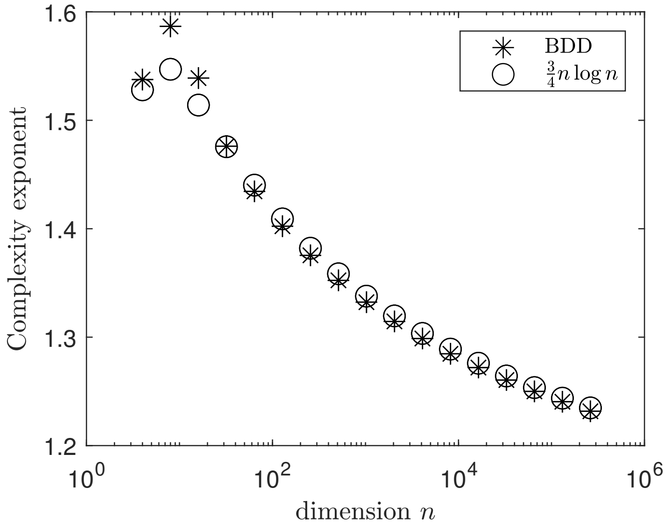


Figure 5.9: Numerical computation of (5.38) with $p_e = 0$, $\sigma^2 = \sigma^2_{max}$, and where we set $O(n) = n$.

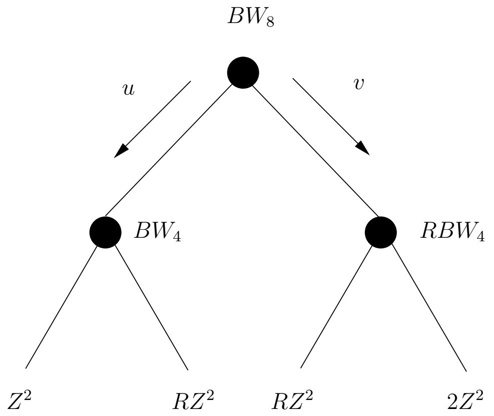


Figure 5.10: Illustration of the alternative single-sided $(u, u + v)$ decoder with BW_8. The edges on the right of a node represent a v-direction and the ones on the left the u-direction. This algorithm is a depth-first search on a tree where the v-directions are first explored.

where P' represents the probability of the two error events in the proof of Lemma 5.8. We leave this analysis as future work, but this approach may yield quasi-linear time quasi-MLD decoding for $n \leq 64$.

Comparison with the "single-sided" $(u, u + v)$ decoder.
As mentionned in the previous subsection on the BDD, our paradigm can be seen as a double-sided $(u, u + v)$ decoder, to be opposed with the single-sided $(u, u + v)$ decoder. A slightly different version of the single-sided decoder was presented in [SB95] [DS06] in the scope of Reed-Muller codes (and also in [TV15] with polar codes): Instead of first decoding u and then v, the order is reversed: v is first decoded and then u follows[6](see also Section 2.3.2). The latter version can be easily translated in our framework and thus adapted to BW lattices.

Let $y = x + w$, with $x = (u, u + v) \in BW_n$ and w the Gaussian noise. The decoder consists in first decoding (or list decoding) $y_2 - y_1$ in $RBW_{\frac{n}{2}}$ as $\hat{v}$ and then decoding (or list decoding) $y_1 + y_2 - \hat{v}$ in $2BW_{\frac{n}{2}}$. This principle is applied recursively down to $BW_2 = \mathbb{Z}^2$. An illustration is provided in Figure 5.10 for $n = 8$.
Let $\Delta(BW_n) = \text{vol}(BW_n)^{\frac{2}{n}}/(2\pi e \sigma^2)$ be the VNR. On the one hand, the VNR decreases in the v-direction: We have $y_2 - y_1 = v + w_1 + w_2$, where $w_1 + w_2 \sim \mathcal{N}(0, 2\sigma^2)$. The VNR for decoding in $RBW_{\frac{n}{2}}$ is

[6] We show below that the two versions are in fact equivalent.

$$\Delta(RBW_{\frac{n}{2}}) = \frac{\mathrm{vol}(RBW_{\frac{n}{2}})^{\frac{2}{n}}}{2\pi e 2\sigma^2} = \frac{\sqrt{2}\,\mathrm{vol}(BW_n)^{\frac{2}{n}}}{2 \cdot 2\pi e \sigma^2} = \frac{\Delta(BW_n)}{\sqrt{2}}. \tag{5.49}$$

On the other hand, the VNR increases in the u-direction: Given that v is correctly decoded, we have $y_1 + y_2 - \hat{v} = 2u + w_1 + w_2$, where again $w_1 + w_2 \sim \mathcal{N}(0, 2\sigma^2)$. The VNR for decoding in $2BW_{\frac{n}{2}}$ is

$$\Delta(2BW_{\frac{n}{2}}) = \frac{4\,\mathrm{vol}(BW_{\frac{n}{2}})^{\frac{2}{n}}}{2\pi e 2\sigma^2} = \frac{4\frac{\mathrm{vol}(BW_n)^{\frac{2}{n}}}{\sqrt{2}}}{2 \cdot 2\pi e \sigma^2} = \sqrt{2}\Delta(BW_n). \tag{5.50}$$

As a result, the error probability can be computed with the following recursion:

$$U_n(\delta, \Delta) = U_{\frac{n}{2}}(\delta, \frac{\Delta}{\sqrt{2}}) + U_{\frac{n}{2}}(\delta, \sqrt{2}\Delta)(1 - U_{\frac{n}{2}}(\delta, \frac{\Delta}{\sqrt{2}})), \tag{5.51}$$

where the initial condition $U_2(\delta, \Delta)$ is for instance given by (5.17) if regular list decoding in $\mathbb{Z}^2$ is performed. Note that the error probability is the same if the "standard" single-sided $(u, u+v)$ decoder (first decode u and then v) is considered. Hence, the two versions are equivalent.

(5.51) can be compared with the error probability of the double-sided $(u, u+v)$ decoder (without the splitting strategy) given by (5.21):

$$U_n(\delta, \Delta) = U_{\frac{n}{2}}(\delta, \frac{\Delta}{\sqrt{2}})^2 + 2U_{\frac{n}{2}}(\delta, \sqrt{2}\Delta)(1 - U_{\frac{n}{2}}(\delta, \frac{\Delta}{\sqrt{2}})). \tag{5.52}$$

We see that the main advantage of the double-sided decoder lies in the squaring of $U_{\frac{n}{2}}(\delta, \Delta/\sqrt{2})$, but which comes at a cost of a greater complexity given a fixed δ. The term $U_{\frac{n}{2}}(\delta, \Delta/\sqrt{2})$ is a severe bottleneck of (5.51).

Nevertheless, the error probability of the single-sided decoder can be improved with the same "trick" as in [DS06]: Subcodes resulting from "frozen" bits in the v-directions can be considered. In our case, this translates in studying sublattices of BW_n with frozen cosets.

Consider the example of Figure 5.10. The right-most leave is the error prone one. Therefore, one could freeze one coset of $2\mathbb{Z}^2/2R\mathbb{Z}^2$ and thus decode in $2R\mathbb{Z}^2$ instead of $2\mathbb{Z}^2$. As a result, the inital condition in this leave becomes $U_2(\delta, 2\Delta)$ instead of $U_2(\delta, \Delta)$. However, the volume of the resulting sublattice Λ of BW_8 is increased: $\mathrm{vol}(\Lambda) = \mathrm{vol}(BW_8) \cdot |2\mathbb{Z}^2/2R\mathbb{Z}^2| = 2\,\mathrm{vol}(BW_8)$ and we have $\Delta(\Lambda) = 2^{\frac{1}{4}}\Delta(BW_8)$. Hence, one should compare $U_n(\delta, \Delta)$ and $U_n(\delta, \frac{\Delta}{2^{\frac{1}{4}}})$ (given by (5.51)) where the inital condition for the right-most leave is replaced by $U_2(\delta, 2\Delta)$ in the latter case.

The study of this algorithm is left for future work.

5.2.2 Parity lattices with $k = n^{\frac{1}{\log \log n}}$

Let us choose $k = n^{\frac{1}{\log \log n}}$. Since $n = c \cdot k^t$, it implies that

$$t = \frac{\log \log n \cdot \log \frac{n}{c}}{\log n} \sim \log \log n. \tag{5.53}$$

We can assume that t is an integer since we consider asymptotic dimensions in this section.
We obtain a family of lattices with a fundamental coding gain growing as $\gamma = O(\log n)$ (using Lemma 5.1). First, we provide bounds on the list size of these lattices.

Lemma 5.7. *The list size of the parity lattices L_n, with $k = n^{\frac{1}{\log \log n}}$, is bounded as*

- $L(L_n, r) = \widetilde{O}(n^3)$ *if* $r \leq d(L_n)(\frac{3}{4} - \epsilon)$.
- $L(L_n, r) = \widetilde{O}(n^6)$ *if* $r = \frac{3}{4}d(L_n)$.
- $L(L_n, r) = \min\{\widetilde{O}(n^{3\log_2 \frac{1}{\epsilon}}), \widetilde{O}(n^{3\log\log n})\}$ *if* $r \leq d(L_n)(1-\epsilon)$, $0 < \epsilon < 1/4$.

Notice that for $3/4 < r < 1$, one can choose between the polynomial bound in $3\log_2 \frac{1}{\epsilon}$ or the super-polynomial bound in $3 \log \log n$ but with a reduced dependency in ϵ.

Then, the list-decoding complexity of these lattices is provided by the following theorem.

Theorem 5.9. *The set $L_n \cap B_\delta(y)$, where $\log k = \frac{\log n}{\log \log n}$, can be computed in worst-case time:*

- $O(n \log n)$ *if* $\delta < \frac{1}{4}$.

- $O(n^2 \log n)$ *if* $1/4 < \delta < \frac{3}{9}$.
- $O(n^2 \log n^{1+log_2 3})$ *if* $\delta < \frac{1}{2}$.
- $l(\delta)\widetilde{O}(n^3) = \widetilde{O}(n^6)$ *if* $\delta = \frac{3}{4} - \epsilon$.
- $\widetilde{O}(n^9)$ *if* $\delta = \frac{3}{4}$.
- $l(\delta)\widetilde{O}(n^2) = \widetilde{O}(n^{3\log_2 \frac{1}{\epsilon}+2})$ *if* $\delta = 1 - \epsilon$, $\epsilon < 1/4$.

The proofs of Lemma 5.7 and Theorem 5.9 are available in Appendix 5.3.5 and 5.3.6, respectively. Figure 5.4 shows how this decoding complexity compare with the decoding complexity of BW lattices. In the proofs and as explained in Section 5.1.2 (see (5.7)), we clearly see that the complexity depends on the number of times the second splitting strategy is used: E.g. $x_1 - 1$ times induces a list-decoding complexity of $\approx n^{x_1}$ (for $\delta < 3/4$). Consequently, reducing this $x_1 - 1$ coefficient (in a similar manner to the probabilistic BDD of BW lattices?) would enable to approach the linear complexity.

These results can be confronted with [DP19], where the authors present the first polynomial-time BDD near Minkowski's bound ($\gamma = O(n/\log n)$), but where the polynomial seems to be of higher order.

5.3 Appendix

5.3.1 The Merge Sort Algorithm

Let $l^{k,n} = (x_1^n, x_2^n ..., x_k^n)$ be a list of k elements x of dimension n (assume for the sake of simplicity that k is a power of 2). This list can be split into two lists of equal size $l_1^{k/2,n}$ and $l_2^{k/2,n}$ and we write $l^{k,n} = (l_1^{k/2,n}, l_2^{k/2,n})$.
Then, we define the function $Merge$ as a function that takes two sorted lists of k elements x of dimension n as input (as well as k and n) and returns a unique sorted list of the $2k$ elements. There exists several variants of this function, but the complexity is always $O(n \cdot k)$.

Algorithm 5.6 Merge Sort Algorithm

```
Function: $MS(l^{k,n}, k, n)$
Input: $l^{k,n} = (l_1^{k/2,n}, l_2^{k/2,n})$, $k \geq 1$, $n \geq 1$.
if $k = 1$ then
    Return $l^{k,n}$.
else
    Return $Merge(MS(l_1^{k/2,n}, \frac{k}{2}, n), M(l_2^{k/2,n}, \frac{k}{2}, n), k, n)$
end if
```

Let $\mathfrak{C}(k,n)$ be the complexity of the MS function (Algorithm 6). The complexity of this algorithm is $\mathfrak{C}(k,n) = 2\mathfrak{C}(k/2,n) + O(k \cdot n) = O(k \log k \cdot n)$

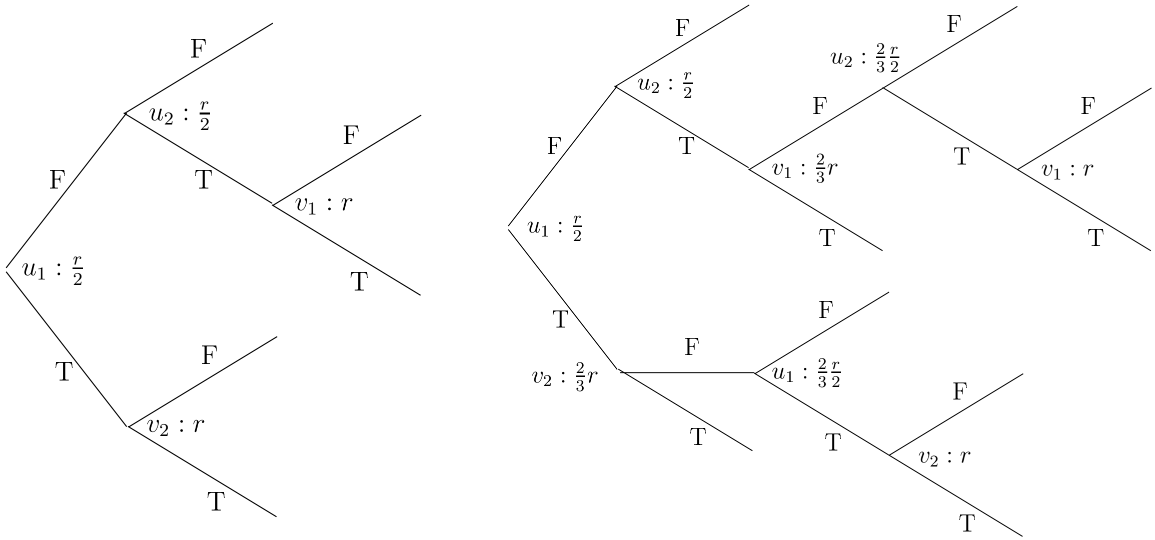
5.3.2 Proof of Theorem 5.3

If Step 14 is removed at the last recursive iteration of Algorithm 5.2 the sent point $x = (x_1, x_2, ..., x_k)$ is not in the outputted list if

- $x_i \notin \mathcal{T}_i$ for at least two lists $\mathcal{T}_i$ (at Step 4 of Algorithm 5.2),
- or if $x_1, ..., x_{j \neq i}, ..., x_k \in \mathcal{T}_1, ..., \mathcal{T}_j, ..., \mathcal{T}_k$, and $x_i - (-\sum_{j \neq i} x_j) \notin \mathcal{V}_i$ (for at least one i).

Let the noise $w = (w_1, ..., w_i, ..., w_k)$. Due to the i.i.d property of the noise, we have $P(||w_1||^2 > \frac{r}{2}) = P(||w_i||^2 > \frac{r}{2})$ for all $1 \leq i \leq k$. As a result, $P(x \notin \mathcal{T})$ becomes

$$\begin{aligned} P(x \notin \mathcal{T}) \leq & \binom{k}{2} P(||w_i||^2 > \frac{r}{2})^2 + kP(||w_i||^2 > r)P(||w_i||^2 < \frac{r}{2})^{k-1}, \\ = & \binom{k}{2} F(\frac{n}{2}, \frac{r}{2}, \sigma^2)^2 + kF(\frac{n}{2}, r, \sigma^2)F(\frac{n}{2}, \frac{r}{2}, \sigma^2)^{k-1}. \end{aligned} \tag{5.54}$$



(a) Without the splitting strategy

(b) With the first splitting strategy (with two cases)

Figure 5.11: Probabilistic tree illustrating the possible events for the case $k = 2$. The labels on the edges represent the result of the question $u_i \in \mathcal{T}_i^{\frac{r}{2}}$? (Or $v_i \in \mathcal{V}_i^r$?), where u_i is the preceding node. The terminating edges labeled with F (False) represent an error event ($x \notin \mathcal{T}$).

Example 5.2. *Figure 5.11a depicts a probabilistic tree representing the possible events for the case*[7] $k = 2$*, where the sent point is* $x = (u_1, u_1 + v_2) = (u_2 + v_1, u_2)$*,* $u_i \in L_n$*,* $v_i \in \theta L_n$*.*

More generally, we have

$$P(x \notin \mathcal{T}) \leq \binom{k}{2} P(x_j \notin \mathcal{T}_j)^2 + kP(x_i - (-\sum_{j \neq i} x_j) \notin \mathcal{V}_i)(1 - P(x_j \notin \mathcal{T}_j))^{k-1}. \tag{5.55}$$

This idea can be recursively applied if we remove Step 11 at each recursion. Let $U_n(r, \sigma^2)$ denote an upper-bound of $P(x \notin \mathcal{T})$. To lighten the notations, we write $U_{\frac{n}{k}}(r)$ for $U_{\frac{n}{k}}(r, \sigma^2)$. We have a recursion of the form

$$U_n(r, \sigma^2) = \binom{k}{2} U_{\frac{n}{k}}(\frac{r}{2})^2 + k \cdot U_{\frac{n}{k}}(r)(1 - U_{\frac{n}{k}}(\frac{r}{2}))^{k-1}, \tag{5.56}$$

where we set $U_n(r, \sigma^2) = 1$ if the right-hand term is greater than 1.

Remember that $\mathrm{vol}(L_{\frac{n}{k}})^{\frac{2}{n/k}} = \mathrm{vol}(L_n)^{\frac{2}{n}}/2^{\frac{1}{k}}$ (see (5.3)) and that $d(\Gamma(V, \beta, k)_{\mathcal{P}}) = d(V) = 2d(T)$. Hence, if we express the recursion as a function of the VNR $\Delta = \frac{\mathrm{vol}(L_{\frac{n}{k}})^{\frac{2}{n/k}}}{2\pi e \sigma^2}$ and the relative radius δ, we get:

$$U_n(\delta, \Delta) = \binom{k}{2} U_{\frac{n}{k}}(\delta, \frac{\Delta}{2^{\frac{1}{k}}})^2 + kU_{\frac{n}{k}}(\delta, 2^{\frac{k-1}{k}}\Delta)(1 - U_{\frac{n}{k}}(\delta, \frac{\Delta}{2^{\frac{1}{k}}}))^{k-1}. \tag{5.57}$$

With the first splitting strategy (but not the second splitting strategy) the error probability is bounded from above as

$$\begin{aligned} P(n, \sigma^2, x \notin \mathcal{T}) \leq \binom{k}{2} P(x_i \notin \mathcal{T}^{\delta})^2 + k\Big[P(x_j \notin \mathcal{T}^{\frac{2}{3}\delta},\ x_j \in \mathcal{T}^{\delta})^{k-1} P(x_i - (-\sum_{j \neq i} x_j) \notin \mathcal{V}_i^{2/3\delta}) \\ + P(x_i - (-\sum_{j \neq i} x_j) \notin \mathcal{V}_i^{\delta})) P(x_j \in \mathcal{T}^{\frac{2}{3}\delta})^{k-1}\Big], \end{aligned} \tag{5.58}$$

[7] For the case $k = 2$ we use the notation $x = (u_1, u_1 + v_2)$, instead of $x = (t_1, t_1 + v_2)$, to match the notation of the Plotkin construction. See Section 5.2.1.

where

$$P(x_j \notin \mathcal{T}^{\frac{2}{3}\delta},\ x_j \in \mathcal{T}^{\delta}) = (1 - \frac{P(x_j \in \mathcal{T}^{\frac{2}{3}\delta})}{P(x_j \in \mathcal{T}^{\delta})})P(x_j \in \mathcal{T}^{\delta}), \\ = P(x_j \in \mathcal{T}^{\delta}) - P(x_j \in \mathcal{T}^{\frac{2}{3}\delta}). \tag{5.59}$$

Example 5.3. *Figure 5.11b depicts a probabilistic tree representing the possible events, for the case $k = 2$, when the first splitting strategy is used.*

5.3.3 Proof of Lemma 5.3

Assuming that $d(y, x) < r = 1/4 \cdot d(BW_{2n})$, the algorithm is re-organized as follows:

- Decode y_1 as u_1. If $||w_1|| < \frac{r}{2}$, then y_1 is correctly decoded as u_1 and so is $y_2 - u_1$ as v_2.
- Else, if $||w_1|| \geq \frac{r}{2}$, y_1 is not correctly decoded as u_1 and:
  - If $d(u_1, y_1) > \frac{r}{2}$, then use the $(u_2 + v_1, u_2)$ construction for decoding.
  - Else $d(u_1, y_1) < \frac{r}{2}$ (y_1 is close to a different point) and
    - If $d(v_2, y_2 - u_1) > r$, this means that u_1 was not correctly decoded. Use the $(u_2 + v_1, u_2)$ construction for decoding.
    - Else $d(v_2, y_2 - u_1) < r$. Check if $d(\hat{x}, y) < r$, where $\hat{x} = (u_1, u_1 + v_2)$:
      - If not, use the $(u_2 + v_1, u_2)$ construction for decoding.
      - This last case, $d(\hat{x}, y) < r$, is not possible as shown by the following lemma.

Lemma 5.8. *Let $x = (u, u + v)$ and $\hat{x} = (u_1, u_1 + v_2)$, $x, \hat{x} \in BW_{2n}$ and $w = (w_1, w_2) \in \mathbb{R}^{2n}$. Let $y = x + w$, where $||w_1||^2 > d(BW_n)/4$ and $||w|| < d(BW_{2n})/4$.*
If $d(u_1, y_1) < d(BW_n)/4$ then $d(v, y_2 - u_1) > d(BW_n)/4$ and $d(\hat{x}, y) > d(BW_{2n})/4$.

Proof. Let $x = (u, u+v) = (v' + m, v'' + m)$ and the point decoded $\hat{x} = (u_1, u_1 + v_2) = (v_2' + m_1, v_2'' + m_1)$, where $v, v', v'', v_2, v_2', v_2'' \in RBW_n$ and $m, m_1 \in [BW_n/RBW_n]$. There are two cases to consider.

- If u and u_1 are in the same coset of RBW_n, i.e. $m = m_1$ (but $u \neq u_1$), then $||w_1||^2 > d(RBW_n)/4$.
- If u_1 and u are not in the same coset of RBW_n, i.e. $m \neq m_1$, then $y_2 - u_1 = v'' + m - v_2' - m_1 + w_2$. Since $m \neq m_1$, $y_2 - u_1$ is decoded in $RBW_n + m - m_1$ instead of RBW_n. Hence, even if $v'' = v_2''$, $d(v'' + m, v_2'' + m_1) \geq d(BW_n)/4$. As a result, $d(y, \hat{x}) > 2d(BW_n)/4 = d(BW_{2n})/4$.

□

Figure 5.5 depicts a probabilistic tree summarizing these possible behaviors of the algorithm with the re-ordered recursive calls. With the explanations in the legend of Figure 5.5, we deduce that the complexity is given by

$$\mathfrak{C}(n, r) < \mathfrak{C}(\frac{n}{2}, \frac{r}{2})[\mathbf{2}(p_{\frac{r}{2}} + [(1 - p_{\frac{r}{2}})(1 - p_e)]^2) + \\ \mathbf{3}(1 - p_{\frac{r}{2}})(1 - p_e)(p_{\frac{r}{2}} + (1 - p_{\frac{r}{2}})p_e) + \mathbf{4}(1 - p_{\frac{r}{2}})p_e] + O(n). \tag{5.60}$$

5.3.4 Proof of Lemma 5.6

The same proof technique as the one used to prove Theorem 1.2 in [GP17] is considered, but we optimize the parameters.
We use the (first) splitting strategy according to the recommendations made in Section 5.1.2 (for the case $k = 2$). To lighten the notations, we write $l(\delta)$ for $l(n/2, \delta)$.

• If $\delta = \frac{3}{4}$:
We use Equation (4.34), where we set $a_1 = \delta - a_2/2$. We get

$$l(n, \delta) \leq 2\big[l(\delta)l(\delta - a_1/2) + l(a_1)l(a_1) + l(2[\delta - a_1])l(\delta)\big].$$

Let $a_1 = 1/2 + \epsilon'$, then $2[\delta - a_1] = 1/2 - 2\epsilon'$ and $\delta - a_1/2 = 1/2 - \epsilon'/2$. We choose ϵ' to minimize $l(a_1)$: I.e. such that (see (5.42)) $\lfloor \frac{3}{4[3/4-(1/2+\epsilon)]} \rfloor = \lfloor \frac{3}{4[3/4-1/2]} \rfloor = 12$, e.g. $\epsilon' = \frac{1}{32}$. Then,

$$\begin{aligned} l(n,\delta) &\leq 2l(\delta)\big[l(\frac{31}{64}) + l(\frac{7}{16})\big] + n^{2\log_2 12}, \\ &= 2l(\delta)[16+8] + 2^{2\log_2 12} = 44l(\delta) + n^{2\log_2 12}, \\ &= n^{2\log_2 12} \sum_{i=0}^{\log_2 n} \left(\frac{44}{2^{2\log_2 12}}\right)^i, \\ &= O(n^{2\log_2 12}). \end{aligned}$$

- If $\delta = 1 - \epsilon$, $0 < \epsilon < \frac{1}{4}$:

Our "trick" to improve the constant (besides the above improvement) is to introduce the parameter ϵ'. Equation (4.34), with $k = 2$, is an upper bound on the list size of BW_n lattices $l(n,\delta)$. We set $a_1 = 1 - (2-\epsilon')\epsilon$ and $a_2 = 1/2 - \epsilon$. We want to show that $l(n, 1-\epsilon) \leq c \cdot \left(\frac{1}{\epsilon}\right)^{x\log_2 n}$, where c is a constant and x is a parameter to be determined. The claim is true for $n = 0$. We proceed by induction on n: We assume that it is true for n/k and show that it holds for n.

$$\begin{aligned} l(n,\delta) \leq& 2\big[l(\delta)l(\frac{1}{2} - \frac{\epsilon'}{2}\epsilon) + l(1-(2-\epsilon')\epsilon)l(\frac{3}{4} - \epsilon/2) \\ &+ l(\frac{1}{2} - \epsilon)l(\delta)\big], \\ \leq& 2\big[c_1 l(\delta) + l(1-(2-\epsilon')\epsilon)l(\frac{3}{4}) + c_2 l(\delta)\big], \\ \leq& \frac{1}{\epsilon}^{x(\log_2 n - 1)} \left(4c_1 c + 2c\left[\frac{12^2}{(2-\epsilon')^x}\right]^{\log_2 n - 1}\right). \end{aligned}$$

We choose ϵ' such that $\frac{12^2}{(2-\epsilon')^x} < 1$ for the smallest (integer) value of x possible, e.g. $\epsilon' = \frac{1}{8}$ and $x = 8$. Moreover, we have $c_1 = \frac{1}{\epsilon'\epsilon} = \frac{8}{\epsilon}$. Hence, $4c_1 + 2 = \frac{32}{\epsilon} + 2 \leq \left(\frac{1}{\epsilon}\right)^8$. The equation becomes

$$\leq c(4c_1 + 2) \cdot \left(\frac{1}{\epsilon}\right)^{8(\log_2 n - 1)} \leq c \cdot \left(\frac{1}{\epsilon}\right)^{8\log_2 n}.$$

Note: In [GP17], $l(3/4) = O(n^{2\log_2 24})$ and $\epsilon' = 1/2$, which yields $x = 16$.

5.3.5 Proof of Lemma 5.7

We use the splitting strategy according to the recommandations made in Section 5.1.2. To lighten the notations, we write $l(\delta)$ for $l(n/k, \delta)$.

- If $\delta = \frac{3}{4} - \epsilon$, $0 < \epsilon < \frac{1}{4}$:

Then $l(\frac{2}{3}\delta = \frac{1}{2} - \frac{2}{3}\epsilon) = \frac{3}{4\epsilon}$ and $l(\frac{\delta}{2}) = 4$. Equation (4.32) becomes:

$$\begin{aligned} l(n,\delta) &\leq 2k^3 l(\delta)l(\frac{2}{3}\delta)l(\frac{\delta}{2}) = \frac{3\cdot 8}{4\epsilon}k^3 l(\delta) = \frac{6}{\epsilon}k^3 l(\delta), \\ &= \left(\frac{6}{\epsilon}n^{3/\log\log n}\right)^{\log\log n - \log_k c} \cdot l(L_c, \delta), \\ &= O(n^3 \log n^{\log\frac{6}{\epsilon}}). \end{aligned}$$

- If $\delta = \frac{3}{4}$:

We use Equation (4.35) (to avoid a term of the form $l(n, 3/4) \cdot \Omega(n)$), where we set $a_1 = \delta - a_2/2$.

$$l(n,\delta) \leq c_1 k^3 \big[l(\delta)l(\delta - a_1/2) + l(a_1)l(\delta - a_2/2) + l(a_2)l(\delta)\big].$$

Let $a_1 = 1/2 + \epsilon$. Then $2[\delta - a_1] = 1/2 - 2\epsilon$ and $\delta - a_1/2 = 1/2 - \epsilon/2$. Let c_1 be a positive constant.

$$\begin{aligned} l(n,\delta) &\leq k^3\big[l(\frac{3}{4})l(\frac{1}{2} - \epsilon/2) + l(\frac{1}{2} + \epsilon)l(\frac{1}{2} + \epsilon) + l(\frac{1}{2} - 2\epsilon)l(\frac{3}{4})\big], \\ &= c_1 k^3 l(\frac{3}{4}) + \widetilde{O}(n^6), \\ &= \widetilde{O}(n^6). \end{aligned}$$

• If $\delta = 1 - \epsilon$, $0 < \epsilon < \frac{1}{4}$:
We use Equation (4.35) with $a_1 = 1 - (2-\epsilon')\epsilon$ and $a_2 = 1/2 - \epsilon$. We want to show that $l(n, 1-\epsilon) = O(\left(\frac{1}{\epsilon}\right)^{x \log_2 n})$, where $x > 1$ is a parameter to be determined. The claim is true for $n = 0$. We proceed by induction on n: We assume that it is true for n/k and show that it holds for n.

$$\begin{aligned}
l(n,\delta) \leq& k^3\left[l(\delta)l(\frac{1}{2} - \frac{\epsilon'}{2}\epsilon) + l(1-(2-\epsilon')\epsilon)l(\frac{3}{4} - \epsilon/2) + l(\frac{1}{2} - \epsilon)l(\delta)\right],\\
\leq& k^3\left[c_1 l(\delta) + l(1-(2-\epsilon')\epsilon)l(3/4 - \epsilon/2) + c_2 l(\delta)\right],\\
=& k^3 O(\left(\frac{1}{\epsilon}\right)^{x(\log_2 n - \frac{\log_2 n}{\log\log n})})(1 + \frac{n^3 \log n^{\log\frac{6}{\epsilon}}}{(2-\epsilon')^{x(\log_2 n - \frac{\log_2 n}{\log\log n})}}),\\
\leq& k^3 O(\left(\frac{1}{\epsilon}\right)^{x(\log_2 n - \frac{\log_2 n}{\log\log n})}) \cdot \left(1 + (2-\epsilon')^{x\frac{\log_2 n}{\log\log n}} \left(\frac{2^3 \log n^{\frac{\log\frac{6}{\epsilon}}{\log_2 n}}}{(2-\epsilon')^x}\right)^{\log_2 n}\right).
\end{aligned}$$

We choose $x = 3 + \epsilon''$ and $\epsilon' > 0$ small enough such that the term under the power $\log_2 n$ is $O(1)$. We get

$$\begin{aligned}
l(n,\delta) \leq& k^3 O\left(\left(\frac{1}{\epsilon}\right)^{x(\log_2 n - \frac{\log_2 n}{\log\log n})}\right)(1 + 2^{x\frac{\log_2 n}{\log\log n}}),\\
\leq& O\left(\left(\frac{1}{\epsilon}\right)^{x\log_2 n}\right) \cdot \frac{2^{3\log_2 n/\log\log n}\left(1 + 2^{x\frac{\log_2 n}{\log\log n}}\right)}{O(\left(\frac{1}{\epsilon}\right)^{x\frac{\log_2 n}{\log\log n}})},\\
\leq& O\left(\left(\frac{1}{\epsilon}\right)^{x\log_2 n}\right).
\end{aligned}$$

If ϵ'' is chosen smaller than $\frac{1}{n}$, the above equation becomes:

$$l(n,\delta) = O\left(\left(\frac{1}{\epsilon}\right)^{3\log_2 n + \frac{\log_2 n}{n}}\right) = O\left(\left(\frac{1}{\epsilon}\right)^{3\log_2 n}\right).$$

Alternatively, we can prove the following super-polynomial (in n) bound, but with a reduced dependency in ϵ. Let c_1, c_2 be positive constants. We have

$$\begin{aligned}
l(n,\delta) \leq& 2k^3\left[l(\delta)l(\frac{1}{2} - \frac{\epsilon'}{2}\epsilon) + l(1-(2-\epsilon')\epsilon)l(\frac{3}{4} - \epsilon/2) + l(\frac{1}{2} - \epsilon)l(\delta)\right],\\
\leq& 2k^3\left[c_1 l(\delta) + l(1-(2-\epsilon')\epsilon)l(\frac{3}{4}) + c_2 l(\delta)\right],\\
=& O(n^{3\log\log n} \log n^{\log\frac{6}{\epsilon}}) = \widetilde{O}(n^{3\log\log n}).
\end{aligned}$$

5.3.6 Proof of Theorem 5.7

We use the splitting strategies according to the recommandations made in Section 5.1.2. The second splitting strategy is never used for $k = 2$.
To lighten the notations, we write $l(\delta)$ for $l(n/k, \delta)$ and $\mathfrak{C}(\delta)$ for $\mathfrak{C}(n/k, \delta)$.

• If $\delta < \frac{3}{8}$:
Then, $\frac{2}{3}\delta < \frac{1}{4}$. We have $l(\delta) = O(1)$, $l(\frac{2}{3}\delta) = l(\frac{\delta}{2}) = l(\frac{\delta}{3}) = 1$, and $\mathfrak{C}(\frac{2}{3}\delta) = \frac{1}{2k}O(n^{1+1/log_2 k})$.

- If $k = 2$, the decoder of [GP17], whose complexity is given by (5.39), yields $O(n^2)$.

- If $k = n^{1/\log\log n}$, we use the first splitting strategy with two sub-cases and once the second splitting strategy. The baseline equation is (4.26), which becomes

$$\begin{aligned}
\mathfrak{C}(n,\delta) =& k\mathfrak{C}(\delta) + (k^2 - k)\left[l(\delta)O(n\log n) + \mathfrak{C}(\delta)\right],\\
=& k^2\mathfrak{C}(\delta) + l(\delta)O(n\log n) = l(\delta)O(n\log n)\sum_{i=0}^{\log\log n} k^i,\\
=& l(\delta)O(n^2\log n) = O(n^2\log n).
\end{aligned}$$

- If $\frac{3}{8} \leq \delta < \frac{1}{2}$:
Then, $\frac{\delta}{2} < 1/4$. We have $l(\frac{2}{3}\delta) = 2$, $l(\frac{\delta}{2}) = l(\frac{\delta}{3}) = 1$.

  - If $k = 2$, the decoder of [GP17], whose complexity is given by (5.39), yields $O(n^2)$.
  - If $k = n^{1/\log\log n}$, we use the first splitting strategy with two sub-cases and once the second splitting strategy. The baseline equation is (4.26), which becomes

$$\begin{aligned}\mathfrak{C}(n,\delta) =& k\mathfrak{C}(\delta) + O(n^2 \log n) + (k^2 - k)\left[l(\delta)O(n^2 \log n) + l(\frac{2}{3}\delta)\mathfrak{C}(\delta)\right],\\ =&[(k^2-k)l(\frac{2}{3}\delta) + k]\mathfrak{C}(\delta) + l(\delta)O(n^2 \log n),\\ =&l(\delta)O(n^2\log n)\cdot \sum_{i=0}^{\log\log n}\left(\frac{k^2 l(\frac{2}{3}\delta) - kl(\frac{2}{3}\delta) + k}{k^2}\right)^i,\\ =&l(\delta)O(n^2(\log n)^{1+\log 3}).\end{aligned}$$

- If $\delta = \frac{3}{4} - \epsilon$, $0 < \epsilon \leq 1/4$:
Then, $\frac{2}{3}\delta < \frac{1}{2}$, $\frac{\delta}{2} < \frac{3}{8}$, $\frac{\delta}{3} < \frac{1}{4}$. We have $l(\frac{2}{3}\delta) = l(\frac{\delta}{2}) = O(1)$, $l(\frac{\delta}{3}) = 1$.
We use the first splitting strategy with two sub-cases.

  - If $k = 2$, the baseline equation is (4.23), which becomes

$$\begin{aligned}\mathfrak{C}(n,\delta) =& [2l(\frac{2}{3}\delta) + 2]\mathfrak{C}(\delta) + l(\delta)O(n^2),\\ =&l(\delta)O(n^2)\cdot\sum_{i=0}^{\log_2 n}\left(\frac{2l(\frac{2}{3}\delta)+2}{4}\right)^i,\\ =&l(\delta)O(n^{2+\log_2[\frac{l(\frac{2}{3}\delta)+1}{2}]}) = l(\delta)O(n^{1+\log_2[\lfloor\frac{3}{4\epsilon}\rfloor+1]}).\end{aligned}$$

    If $\delta = \frac{1}{2}$, $l(\delta) \leq 2n$ and $\epsilon = \frac{1}{4}$. Then

$$\mathfrak{C}(n,\delta) = O(n^4).$$

    If $\delta > \frac{1}{2}$, we have $l(\delta) = O(n^{\log_2 4\lfloor\frac{3}{4\epsilon}\rfloor})$. Then,

$$\mathfrak{C}(n,\delta) = O(n^{1+\log_2 4\lfloor\frac{3}{4\epsilon}\rfloor^2}),$$

    where we assumed that $\frac{3}{4\epsilon} >> 1$.

  - If $k = n^{1/\log\log n}$, we use twice the second splitting strategy. The baseline equation is (4.33), which becomes:

$$\begin{aligned}\mathfrak{C}(n,\delta) \leq& l(\delta)l(\frac{\delta}{2})O(n^2\log n^{1+\log 3}) + k^3 l(\frac{2}{3}\delta)\mathfrak{C}(\delta),\\ =&l(\delta)O(n^2\log n^{1+\log 3})\sum_{i=0}^{\log\log n}\left(kl(\frac{2}{3}\delta)\right)^i,\\ =&l(\delta)O(n^3(\log n)^{1+\log 3\lfloor\frac{3}{4\epsilon}\rfloor}).\end{aligned}$$

    If $\delta = \frac{1}{2}$, $l(\delta) \leq 2n$. Then

$$\mathfrak{C}(n,\delta) = O(n^4\log n^4).$$

    If $\delta > \frac{1}{2}$, we have $l(\delta) = \widetilde{O}(n^3)$. The complexity is

$$\mathfrak{C}(n,\delta) = \widetilde{O}(n^6)$$

- If $\delta = \frac{3}{4}$:
Then, $\frac{2}{3}\delta = \frac{1}{2}$, $\frac{\delta}{2} = \frac{3}{8}$. We have $l(\frac{2}{3}\delta) = 2n$, .
We use the first splitting strategy with three sub-cases (to avoid a term of the form $\Omega(n)\cdot\mathfrak{C}(\delta)$).

Let $a_1 = 1/2 - \epsilon_2'$ and $a_2 = 1/2 - \epsilon_1'$. In both cases, $\mathfrak{C}(n, \delta)$ is upper bounded by (see e.g. Equation (4.36) and (4.37))

$$\begin{aligned}\mathfrak{C}(n,\delta) \leq& x\big[l(\delta)\mathfrak{C}(a_1) + l(2(\delta - a_1))\mathfrak{C}(\delta - \frac{a_2}{2}) + l(a_2)\mathfrak{C}(\delta)],\\ =& x\big[l(\delta)\mathfrak{C}(\frac{1}{2} - \epsilon_2') + l(\frac{1}{2} + 2\epsilon_2')\mathfrak{C}(\frac{1}{2} + \frac{\epsilon_1'}{2}) + l(\frac{1}{2} - \epsilon_1')\mathfrak{C}(\delta)],\end{aligned}$$

where $x = k$ if $k = 2$ and $x = k^3$ if $k = n^{1/\log\log n}$.

- If $k = 2$:
  We choose ϵ_1' small enough such that $\mathfrak{C}(1/2 + \frac{\epsilon_1'}{2}) = \mathfrak{C}(\frac{1}{2}) = O(n^{1+\log_2 4\lfloor\frac{3}{4\frac{1}{4}}\rfloor^2})$ and ϵ_2' such that $l(\frac{1}{2} + 2\epsilon_2') = l(\frac{1}{2}) = n^{\log_2 4\lfloor\frac{3}{4\frac{1}{4}}\rfloor}$. The complexity is

$$\begin{aligned}\mathfrak{C}(n,\delta) \leq& l(\delta)O(n^2) + l(\frac{1}{2})\mathfrak{C}(\frac{1}{2}) + c\mathfrak{C}(\delta),\\ =& O(n^{2+2\log_2 12}) + O(n^{1+\log_2(12\cdot 36)}) + c\mathfrak{C}(\delta),\\ =& O(n^{1+\log_2 432}).\end{aligned}$$

- If $k = n^{1/\log\log n}$:
  The complexity is

$$\begin{aligned}\mathfrak{C}(n,\delta) =& \widetilde{O}(n^{6+2}) + \widetilde{O}(n^{3+6}) + ck^3\mathfrak{C}(\delta),\\ =& \widetilde{O}(n^9).\end{aligned}$$

• If $\delta = 1 - \epsilon$, $0 < \epsilon < \frac{1}{4}$:
We use the first splitting strategy with three sub-cases. In both cases, $\mathfrak{C}(n, \delta)$ is upper bounded by (see Equation (4.36) and (4.37))

$$\mathfrak{C}(n,\delta) \leq x\big[l(\delta)\mathfrak{C}(\frac{3}{4} - \epsilon') + l(\frac{1}{2} + \epsilon' - \epsilon)\mathfrak{C}(\frac{3}{4} - \epsilon/2) + l(1/2 - \epsilon)\mathfrak{C}(\delta)\big],$$

where $x = k$ if $k = 2$ and $x = k^3$ if $k = n^{1/\log\log n}$. We choose $\epsilon' = \frac{1}{4} + \epsilon''$, where $\epsilon'' \leq \epsilon$. The bound becomes

$$\mathfrak{C}(n,\delta) \leq 2x\big[l(\delta)\mathfrak{C}(\frac{1}{2} - \epsilon') + l(\frac{3}{4} - (\epsilon - \epsilon'))\mathfrak{C}(\frac{3}{4} - \epsilon/2) + l(1/2 - \epsilon)\mathfrak{C}(\delta)\big]$$

- If $k = 2$:

$$\mathfrak{C}(n,\delta) = l(\delta)O(n^2) = O(n^{8\log_2\frac{1}{\epsilon}+2}).$$

- If $k = n^{1/\log\log n}$:

$$\mathfrak{C}(n,\delta) = l(\delta)\widetilde{O}(n^2) = \widetilde{O}(n^{3\log_2\frac{1}{\epsilon}+2}).$$

5.3.7 Proof of Theorem 5.8

The result on the complexity is obtained by adapting (5.44), (5.45), and the complexity formulas in Theorem 5.7.

- If $3/8 < \delta \leq 9/16$:
  We use the fact $E_y[l'(\frac{n}{2}, \frac{3}{8}, y)] \geq E_y[l'(\frac{n}{4}, \frac{3}{8}, y)] \geq ....$

$$\begin{aligned}E_y[l'(n,\delta,y)] \leq& 2\big[E_y[l'(\frac{3}{8}, y)]l(\delta) + l(\delta)E_y[l(\frac{3}{8}, y)]],\\ \leq& 4E_y[l'(\frac{3}{8}, y)]l(\delta) = O(n^{\log_2(4E_y[l'(\frac{3}{8},y)])}), \qquad (5.61)\\ =& O(n^{2+\log_2 E_y[l'(\frac{3}{8},y)]}).\end{aligned}$$

Chapter 6

Study of some famous group codes

6.1 Turyn's construction of the Leech lattice and the Nebe lattice

The story of Turyn's construction starts in 1967, when Turyn constructed the Golay code from versions of the extended Hamming code [AMT67] [MS77, Chap. 18, sec 7.4]. According to Nebe [Neb10], it has then been remarked independently in [Tit80], [LM82], and [Que84] that there is an analogous construction of Λ_{24} based on E_8. Turyn's construction re-appeared in [CS86] under the form of 4096 cosets of the lattice $(E_8)^3$. Finally, it was rediscovered in the scope of the "cubing construction" in [For88b], i.e. as $\Gamma(V, \alpha, \beta, 3)$.

Among the three groups S, T, V used in the construction $\Gamma(V, \alpha, \beta, 3)$, let us take V as $2S$. To build the Leech lattice, we have $S, T \cong E_8$ and to build the Nebe lattice we have $S, T \cong \Lambda_{24}$. Moreover, to obtain theses two lattices via the $k-$ing construction, the coset representatives α should be chosen such that $d(\Gamma(V, \alpha, \beta, 3)) > 3d(S)$. We already established via (4.8) that choosing α is equivalent to choosing T^*. In the next section, we explain how to get T^* via lattice polarisation.

6.1.1 The polarisation of lattices

The groups $S, T, T^*, V = 2S$ we are considering are lattices of rank n. In the scope of the polarisation of lattices, T^* is a rotation of T by an angle of 2θ. Therefore, it is denoted $T_{2\theta}$.

Definition 6.1. *Given a lattice S, we call $(T, T_{2\theta})$ a polarisation of S [Neb12] if*

$$\begin{aligned} &S \cong T \cong T_{2\theta}, \\ &S = T_{2\theta} + T, \quad \textit{and} \quad T_{2\theta} \bigcap T = 2S. \end{aligned} \tag{6.1}$$

Let G_S be a generator matrix of S. Finding a polarisation of the lattice S (if it exists) is equivalent to finding a scaling-rotation matrix R, $R \cdot R^T = 2I$, with

$$G_T = G_S \cdot R \text{ and } G_{T_{2\theta}} = G_S \cdot R^T,$$

such that the basis vectors g_T^i and $g_{T_{2\theta}}^i$, $1 \leq i \leq n$, are versions of the vectors g_S^i scaled by a factor of $\sqrt{2}$ and rotated by an angle of $\pm\theta = \arctan\sqrt{7}$. Indeed, consider two vectors g_T^i and $g_{T_{2\theta}}^i$ of the same size and having an angle of 2θ. Summing these two vectors yields a vector g_S^i having half the size of g_T^i, as illustrated by Figure 6.1:

$$||g_S^i||^2 = ||g_T^i + g_{T_{2\theta}}^i||^2 = 0.5 \times ||g_T^i||^2, \tag{6.2}$$

since $\cos(2\theta) = -3/4$. One would thus get $G_S = G_T + G_{T_{2\theta}}$. Such a rotation matrix can be found via a $\mathbb{Z}[\lambda]$-structure of S; Let $G_S^{\mathbb{C}}$ be a (complex) generator matrix of S over the ring of integers $\mathbb{Z}[\lambda]$, $\lambda = \sqrt{2}e^{i\theta} = \frac{1+i\sqrt{7}}{2}$. Multiplying $G_S^{\mathbb{C}}$ by λ yields a new matrix whose rows are new vectors belonging to the lattice, scaled by $\sqrt{2}$, and having the desired angle with the basis vectors. Hence, $G_T^{\mathbb{C}}$ can be obtained as $\lambda G_S^{\mathbb{C}}$ and $G_{T_{2\theta}}^{\mathbb{C}}$ as $\psi G_S^{\mathbb{C}}$, where $\psi = \bar{\lambda}$ is the conjugate of λ. Therefore, if we let G_S be the real generator matrix obtained from $G_S^{\mathbb{C}}$ (via (3.3)), the real rotation matrix R for polarisation is $R(n, \lambda) = I_{n/2} \otimes R(2, \lambda)$.

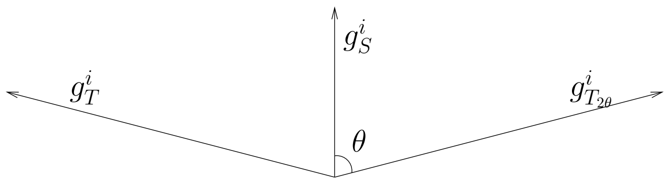


Figure 6.1: Illustration of $g^i_T + g^i_{T_{2\theta}} = g^i_S$.

6.1.2 The lattice $\Gamma(2S, T_{2\theta}, T, 3)$

Given three lattices S, T and $T_{2\theta}$, respecting properties (7.22), a new lattice is obtained from Turyn's construction as

$$\begin{aligned}\Gamma(2S, T_{2\theta}, T, 3) =&\{(a = m' + n'_1,\ b = m' + n'_2,\ c = m' + n'_3),\\ &m' \in T_{2\theta}, n'_1, n'_2, n'_3 \in T, n'_1 + n'_2 + n'_3 \in 2S\},\\ =&\{(a = v_1 + m + n_1,\ b = v_2 + m + n_2, c = v_3 + m - n_1 - n_2),\\ &v_1, v_2, v_3 \in 2S,\ m \in \alpha,\ n_1, n_2 \in \beta\}.\end{aligned} \tag{6.3}$$

A generator matrix of $\Gamma(2S, T_{2\theta}, T, 3)$ is

$$G_{\Gamma(2S,T_{2\theta},T,3)} = \begin{bmatrix} G_{(3,1)} \otimes G_{T_{2\theta}} \\ G_{(3,2)} \otimes G_T \end{bmatrix} \tag{6.4}$$

where $G_{(3,1)}$ and $G_{(3,2)}$ are generator matrices for the $(3, 1)$ binary repetition code and the $(3, 2)$ binary single parity-check code, respectively. Obviously, $\mathbb{F}_2$ is naturally embedded into $\mathbb{Z}$ for the two binary codes. From (6.4), a generator matrix of $\Gamma(2S, T_{2\theta}, T, 3)$ over $\mathbb{Z}[\lambda]$ can be expressed as

$$G^{\mathbb{C}}_{\Gamma(2S,T_{2\theta},T,3)} = \underbrace{\begin{bmatrix} \lambda & \lambda & \lambda \\ \psi & \psi & 0 \\ 0 & \psi & \psi \end{bmatrix}}_{=Pb} \otimes G^{\mathbb{C}}_S, \tag{6.5}$$

where $G^{\mathbb{C}}_S$ is a generator matrix of S over $\mathbb{Z}[\lambda]$.

6.1.3 Construction of the Leech lattice and Nebe lattice

Theorem 6.1. *Let $S \cong E_8$ and T,$T_{2\theta}$ be two lattices respecting properties (7.22). Then, $\Gamma(2S, T_{2\theta}, T, 3)$ is the Leech lattice with fundamental coding gain equal to 4 [Tit80] [LM82] [Que84].*

The following proof is not new, but it enables to make a clear link between the k-ing construction and Λ_{24} using our notations.

Proof. We let E_8 be scaled such that $d(E_8) = 2$ and $\text{vol}(E_8) = 1$. This version of the Gosset lattice is even. Then, $S = \frac{1}{\sqrt{2}}E_8$ has $d(S) = 1$, $\text{vol}(S) = 2^{-4}$ and $\text{vol}(T) = \text{vol}(T_{2\theta}) = 1$, $d(T) = d(T_{2\theta}) = 2$. Also, $|\alpha| = |\beta| = 2^4$ from (3.21).
Firstly, using Theorem 4.2, we have $d(\Gamma(2S, T_{2\theta}, T, 3)) \geq 3$. Then, assume that $a = m' + n'_1$ and $b = m' + n'_2$ (with the notations of (6.3)) have both odd squared norms. This is equivalent to having the scalar products $\langle m', n'_1\rangle = \frac{\nu}{2}$ and $\langle m', n'_2\rangle = \frac{\nu'}{2}$, where ν and ν' are integers. Therefore, $\langle m', n'_1 + n'_2\rangle$ is integer and $c = m' + n'_1 + n'_2$ has an even squared norm. We just proved that $\Gamma(2S, T_{2\theta}, T, 3)$ is even. This implies that $d(\Gamma(2S, T_{2\theta}, T, 3)) = 4$.
The last step aims at proving that $\Gamma(2S, T_{2\theta}, T, 3)$ has a unit volume. (6.3) shows that $\Gamma(2S, T_{2\theta}, T, 3)$ is obtained as the union of $|\alpha||\beta|^2 = 2^{12}$ cosets of $(2S)^3$. Hence, $\text{vol}(\Gamma(2S, T_{2\theta}, T, 3)) = \text{vol}((2S)^3)/2^{12} = 1$. Finally, Λ_{24} is the unique lattice in dimension 24 with fundamental coding gain equal to 4. □

Note: From this polarisation perspective, the construction of the Leech lattice from Forney [For88b] might appear as a surprising result; indeed, he also uses the construction $\Gamma(2S, T_{2\theta}, T, 3)$ but the lattices T and $T_{2\theta}$ are obtained via an unexpected manner. It is discussed in Appendix 6.5.1.

Lemma 6.1. *Let $S \cong \Lambda_{24}$ and T,$T_{2\theta}$ be two lattices respecting properties (7.22). Then, the lattice $\Gamma(2S, T_{2\theta}, T, 3)$ has a fundamental coding gain equal to 6 or 8 [Gri10].*

The proof of Lemma 6.1 is similar to that of Theorem 6.1 (see Appendix 6.5.2). In this case, the polarisation does not ensure $\Gamma(2S, T_{2\theta}, T, 3) > 3d(S) = 6$. Additional work to choose $T_{2\theta}$ is needed.
In Appendix 6.5.3, we discuss the vectors of squared norm 6 in $\Gamma(2S, T_{2\theta}, T, 3)$ and explain how to efficiently count them.

Definition 6.2. *Let S be the $\mathbb{Z}[\lambda]$-structure Λ_{24} with automorphism group $SL_2(25)$ (see Appendix 6.5.3). Set $T_{2\theta} = \lambda S$ and $T = \psi S$. The Nebe lattice is defined as $\mathscr{N}_{72} = \Gamma(2S, T_{2\theta}, T, 3)$.*

Theorem 6.2. *$\mathscr{N}_{72}$ has a fundamental coding gain equal to* 8 *[Neb12].*

Proof. Check that $\Gamma(2S, T_{2\theta}, T, 3)$ has no vectors of squared norm 6 by computing all possibilities of (6.38), as explained in Appendix 6.5.3. □

6.2 Decoders for Leech and Nebe lattices

Given Λ_{24} constructed as $\Gamma(2S, T_{2\theta}, T, 3)$, in [CS86] a point in $\mathbb{R}^{24}$ is decoded in all 2^{12} cosets of $(2S)^3$ and the best candidate is kept (see (ii) below). $\mathscr{N}_{72}$ is the union of 2^{36} cosets of $(2S)^3$ ($S \cong \Lambda_{24}$). As a result, all decoders based on a search in all cosets of $(2S)^3$, as the one proposed by Conway and Forney for Λ_{24}, are intractable for $\mathscr{N}_{72}$. Our decoders are also based on coset decomposition of the lattice. However, as we shall see in the sequel, it is not necessary to investigate all cosets to get efficient decoders.
In this section, we first review the literature on decoding algorithms for Λ_{24} and $\mathscr{N}_{72}$. While the decoding of Λ_{24} has been extensively studied, the literature on decoders for $\mathscr{N}_{72}$ is not as rich: Only [Mey13] studied this aspect, but the proposed decoder is highly suboptimal.
Then, we describe two sets of decoding algorithms for lattices constructed as $\Gamma(2S, T_{2\theta}, T, 3)$. These algorithms are obtained as direct applications of Algorithms 4.2 and 4.3 (and their list-decoding versions) presented in Section 4.3.

6.2.1 Existing decoding algorithms for Λ_{24} and $\mathscr{N}_{72}$

History of the decoders of Λ_{24}

Λ_{24} appeared under many different forms in the literature (which may be equivalent to Turyn's construction). Among others, Λ_{24} can be obtained as (i) 8192 cosets of $4D_{24}$, (ii) 4096 cosets of $(\sqrt{2}E_8)^3$, (iii) 2 cosets of the half-Leech lattice H_{24}, where H_{24} is constructed by applying Construction B on the Golay code C_{24}, and (iv) 4 cosets of the quarter-Leech lattice, where quarter-Leech lattice is also built with Construction B but applied on a subcode of C_{24}. Finally, one of the simplest constructions is due to [BSC95], where the Leech lattice is obtained via Construction A applied on the quaternary Golay code.

The history of MLD algorithms for Λ_{24} starts with [CS84], where Conway and Sloane used (i) to compute the second moment of the Voronoi region of Λ_{24}. The first efficient decoder was presented in [CS86] by the same authors using construction (ii). Two years later, Forney reduced the complexity of the decoder by exploiting the same construction (ii), which he rediscovered in the scope of the "cubing construction", with a 256-state trellis diagram representation [For88b] (see Section 4.3.1 for a presentation of trellis). A year later, it was further improved in [LL89] and [BSS89] thanks to (iii) combined with an efficient decoder of C_{24}. Finally, (iv) along with the Hexacode is used to build the fastest ever known MLD decoder by Vardy and Be'ery [VB93].

To further reduce the complexity, (suboptimal) BDD were also investigated based on the same constructions: e.g. [For89a] with (iii) and [ABV$^+$94] [Var95] [FV96] with (iv). In these papers, it is shown that these BDD do not change the error exponent (i.e. the effective minimum distance is not diminished) but increase the "equivalent error coefficient". The extra loss is roughly 0.1 dB on the Gaussian channel compared to the optimal performance.
As we shall see in the sequel, our decoding paradigm applied to the Leech lattice is more complex than the state-of-the-art decoders of Vardy [Var95] [FV96] which requires only ≈ 300 real operations. But again, this latter decoder is specific to the Leech lattice whereas our decoder is more universal as it can be used, among others, to decode the Nebe lattice and the Barnes-Wall lattices.

	$S \cong \frac{1}{\sqrt{2}}\Lambda_{24}$	$T_{2\theta}, T$	$2S$	$\mathscr{N}_{72}$
d	2	4	8	8
ρ^2	$\frac{1}{2}$	1	2	2
R^2	1	2	4	> 4

Table 6.1: Parameters of $\mathscr{N}_{72}$ with a normalized volume equal to 1.

	$S \cong \frac{1}{\sqrt{2}}E_8$	$T_{2\theta}, T$	$2S$	Λ_{24}
d	1	2	4	4
ρ^2	$\frac{1}{4}$	$\frac{1}{2}$	1	1
R^2	$\frac{1}{2}$	1	2	2

Table 6.2: Parameters of Λ_{24} with a normalized volume equal to 1.

The decoder of the Nebe lattice in [Mey13]

First, notice that we can multiply (on the left) the matrix Pb given in (6.5) by a unimodular matrix to get the following new matrix Pb':

$$Pb' = \begin{bmatrix} 1 & 1 & \lambda \\ 0 & \psi & \psi \\ 0 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ \psi & -\lambda & 0 \end{bmatrix} \cdot Pb. \tag{6.6}$$

Similarly to (6.5), $Pb' \otimes G_S^{\mathbb{C}}$, $S \cong \Lambda_{24}$, is a basis for the Nebe lattice which induces the following structure:

$$\mathscr{N}_{72} = \{(a, b, c) \in \mathbb{C}^{36} : a \in S, b - a \in T, c - (b - a) - \lambda a \in 2S\}, \tag{6.7}$$

where (6.7) is derived from the columns of Pb'. A successive-cancellation-like algorithm can thus be considered: given $y = (y_1, y_2, y_3)$ in $\mathbb{C}^{36}$, y_1 is first decoded in S as t_1, $y_2 - t_1$ is then decoded in T as t_2, and $y_3 - t_2 - \lambda t_1$ is decoded in $2S$ as t_3. In [Mey13], this successive-cancellation algorithm is proposed, with several candidates for t_1 which are obtained via sphere decoding with a given radius r. Among all resulting approximations, the closest to y is kept. It is proved in [Mey13], that the lattice point $\hat{x}$ outputted by the algorithm using a decoding radius $r = R(S)$, the covering radius of S, has an approximation factor $||y - \hat{x}|| \leq \sqrt{7}||y - x_{opt}||$. Additionally, this algorithm is guaranteed to output the closest point $x_{opt} \in \Gamma(2S, T_{2\theta}, T, 3)$ to y if $d(y, x_{opt}) \leq R(S)$, where $R(S)$ is unfortunately smaller by a factor $\sqrt{2}$ than the packing radius $\rho(\mathscr{N}_{72})$, c.f. Tables 6.1 (we also provide Table 6.2 with the parameters of Λ_{24} for comparison).

6.2.2 New BDDs and list decoders for $\Gamma(2S, T_{2\theta}, T, 3)$

We present two decoding strategies: The first one, based Algorithm 4.2, involves $|\alpha|$ decoding of the underlying parity lattice. The second one, based on Algorithm 4.3, involves decoding the component lattices S with a larger radius.

First decoder

We first adapt Algorithm 4.2 to $\Gamma(2S, T_{2\theta}, T, 3)$ by choosing the decoders for T and $V = 2S$, at Steps 3 and 5, as BDDs. We name it Algorithm 4.2'.

Theorem 6.3. *Let $\Gamma(2S, T_{2\theta}, T, 3)$ and y be respectively a lattice and a point in $\mathbb{R}^{3n}$.*
If $d(y, \Gamma(2S, T_{2\theta}, T, 3)) < \rho^2(\Gamma(2S, T_{2\theta}, T, 3))$, then Algorithm 4.2' outputs the closest lattice point $x \in \Gamma(2S, T_{2\theta}, T, 3)$ to y in time

$$\mathfrak{C}_{A.4.2'} = 6|\alpha|\mathfrak{C}_{BDD}^S. \tag{6.8}$$

Proof. We first show that x is the closest lattice point to y. Assume that, at Steps 1-2, m corresponds to the coset of the closest lattice point to y. Then, the result follows from Theorem 4.1 since Algorithm 4.2' is a special case of Algorithm 4.4 used α times.
Regarding the complexity, we use Equation (4.17) with $k = 3$ and where $\mathfrak{C}_{BDD}^S = \mathfrak{C}_{BDD}^{2S}$. □

It is insightful to compare Algorithm 4.2' to trellis decoding. The complexity is reduced from $\approx |\alpha||\beta|^2\mathfrak{C}^S_{CVP}$ to $\approx |\alpha|\mathfrak{C}^S_{BDD}$ (but where trellis decoding is optimal unlike Algorithm 4.2').

Algorithm 4.5' is the list-decoding version of Algorithm 4.2': It consists in repeating $|\alpha|$ times (once for each coset of $\Gamma(2S,\beta,3)_{\mathcal{P}}$) Algorithm 4.5, with k=3, using the first splitting strategy with two sub-cases and once the second splitting strategy. The complexity is obtained by multiplying (4.26) by $|\alpha|$.

Theorem 6.4. *Let $\Gamma(2S,T_{2\theta},T,3)$ and y be respectively a lattice and a point in $\mathbb{R}^{3n}$. Algorithm 4.5' outputs the set $\Gamma(2S,T_{2\theta},T,3)\cap B_\delta(y)$ in worst-case time*

$$\mathfrak{C}_{A.4.5'}(\delta) =|\alpha|\big[3\mathfrak{C}_{T\cap B_\delta(y)} + 6l(T,\delta)l(T,\frac{\delta}{2})\mathfrak{C}_{V\cap B_{\frac{2}{3}\delta}(y)} + l(T,\frac{2}{3}\delta)l(T,\frac{\delta}{3})\mathfrak{C}_{V\cap B_\delta(y)}\big]. \tag{6.9}$$

Proof. The proof to show that all points in the set $\Gamma(2S,T_{2\theta},T,3)\cap B_r(y)$ are outputted by the algorithm is the same as the one of Theorem 4.3. □

Corollary 6.1. *Let $\Lambda_{24}=\Gamma(2S,T_{2\theta},T,3)$ (constructed as in Lemma 6.1). Algorithm 4.5' with a decoding radius $r=d(T)=d(E_8)$, i.e. $\delta=d(E_8)/d(\Lambda_{24})=1/2$, solves the CVP for Λ_{24} with worst-case complexity*

$$\begin{aligned}\mathfrak{C}_{A.4.5'}(\delta=\frac{1}{2}) =&|\alpha|\big[3\mathfrak{C}_{T\cap B_\delta(y)} + 6[2n2\mathfrak{C}_{E_8\cap B_{\frac{1}{3}}(y)} + 3\mathfrak{C}_{E_8\cap B_{\frac{1}{2}}(y)}]\big],\\ \lessapprox&2^4\cdot 6\cdot 2\cdot 8\cdot 2\cdot \mathfrak{C}_{E_8\cap B_{\frac{1}{2}}(y)} \approx 2^{11}\mathfrak{C}_{E_8\cap B_{\frac{1}{2}}(y)}.\end{aligned} \tag{6.10}$$

Proof. If $S\cong E_8$, $d(T)=R^2(\Gamma(2S,T_{2\theta},T,3))$ (the covering radius). □

To the best of our knowledge, the covering radius of $\mathscr{N}_{72}$ appears nowhere in the literature. However, Gabriele Nebe showed in a private communication that it is greater than $\sqrt{2}\rho(\mathscr{N}_{72})$. The proof is available in Appendix 6.5.4. As a result, Algorithm 4.5' with $\delta=1/2$ is not optimal for $\mathscr{N}_{72}$.

Note that the complexity can be slightly reduced if one wants only the closest lattice point and not the entire list: list decoding in V can be replaced by MLD decoding in V.

Second decoder

Algorithm 6.1 implements Algorithm 4.3 as follows:

- Step 1 of Algorithm 4.3 involves list decoding in S with $r=\rho^2(\Gamma(2S,T_{2\theta},T,3))/2$. Consequently, Steps 3-7 of Algorithm 4.3 are performed for any of the combinations of candidates in two lists.
- Regarding Step 2 of Algorithm 4.3, t_i is decomposed as $t_i=m'_i+n'_i$ instead of $t_i=v_i+m_i+n_i$. To this end, we assume that one has a generator matrix G_S of $S=T_{2\theta}+T$, such that $G_S=G_{T_{2\theta}}+G_T$, where $G_{T_{2\theta}}$ and G_T are the generator matrices[1] of $T_{2\theta}$ and T respectively.
- The condition $m_{j_1\neq i}=m_{j_2\neq i}$ of Algorithm 4.3 at step 4 is computed by checking if $m'_{j_1\neq i}-m'_{j_2\neq i}\in V=2S$.
- Step 5 of Algorithm 4.3 becomes $p'_i=m'_{j_1}-n'_{j_1}-n'_{j_2}$.

Let $\rho^2=\rho^2(\Gamma(2S,T_{2\theta},T,3))$. We have $\frac{\rho^2}{2}\leq d(S)/2$, but for the lattices of interest, Λ_{24} and $\mathcal{N}_{72}$, $\frac{\rho^2}{2}=d(S)/2$. Consequently, $\mathfrak{C}_{S\cap B_{\rho^2/2}(y)}=\mathfrak{C}_{S\cap B_{1/2}(y)}$: The relative radius for decoding in S is 1/2. Using Theorem 4.1, we get[2] $L(S,d(S)/2)\leq 2n$ but we have no better bound. The complexity is

$$\begin{aligned}\mathfrak{C}_{A.6.1}(\delta=\frac{1}{4}) \leq&3\mathfrak{C}_{S\cap B_{\frac{1}{2}}(y)} + 3l(S,\frac{1}{2})^2\mathfrak{C}^{2S}_{BDD},\\ \leq&3\mathfrak{C}_{S\cap B_{\frac{1}{2}}(y)} + 12n^2\mathfrak{C}^{2S}_{BDD}.\end{aligned} \tag{6.11}$$

As a result, even for BDD the splitting strategies should be considered.

Algorithm 6.2 (and its subroutine given in Algorithm 6.3) is a modified version of Algorithm 6.1 to use the splitting strategies. It is almost identical to Algorithm 4.5 (and its subroutine given by Algorithm 4.7) except that the sets $S\cap B$ are computed instead of $T\cap B$. The complexity is obtained via (4.26).

[1] Note that the construction presented for Λ_{24} and $\mathscr{N}_{72}$ yields generator matrices satisfying this property.

[2] When $S\cong\Lambda_{24}$, it is an equality and it is conjectured in [Mey13] that this maximum is attained only if y_i is located in a deep hole.

Algorithm 6.1 Second decoder for $\Gamma(2S, T_{2\theta}, T, 3)$ **without the splitting strategies**

Input: $y = (y_1, y_2, y_3) \in \mathbb{R}^{3n}$, $G_S = G_{T_{2\theta}} + G_T$, $0 \leq \delta$.

```
Compute the sets $\mathcal{T}_1 = S \cap B_{2\delta}(y_1)$, $\mathcal{T}_2 = S \cap B_{2\delta}(y_2)$, $\mathcal{T}_3 = S \cap B_{2\delta}(y_3)$.
for $1 \leq i \leq k = 3$ do
  Set $j_1 < j_2, j_1, j_2 \in \{1, 2, 3\} \setminus \{i\}$.
  for each $(t_{j_1}, t_{j_2}) \in \mathcal{T}_{j_1} \times \mathcal{T}_{j_2}$ do
    Compute $z_{j_1} = t_{j_1} G_S^{-1}$, $z_{j_2} = t_{j_2} G_S^{-1}$.
    Compute $m'_{j_1} = z_{j_1} G_{T_{2\theta}}$, $m'_{j_2} = z_{j_2} G_{T_{2\theta}}$, and $n'_{j_1} = z_{j_1} G_T$, $n'_{j_2} = z_{j_2} G_T$.
    if $m'_{j_1} - m'_{j_2} \in 2S$ then
      Compute $p'_i = m'_{j_1} - n'_{j_1} - n'_{j_2}$.
      Compute the sets $\mathcal{V}_i = V \cap B_{\delta_2}(y_i - p'_i)$
      for $v_i \in \mathcal{V}_i$ do
        Add $(t_1, ..., t_{i-1}, v_i + p'_i, t_{i+1}, ..., t_k)$ to the list $\mathcal{T}$.
      end for
    end if
  end for
end for
Return $\mathcal{T}$.
```

Algorithm 6.2 Second decoder for $\Gamma(2S, T_{2\theta}, T, 3)$ **with the splitting strategy** (first splitting strategy with two sub-cases)

Input: $y = (y_1, y_2, y_3) \in \mathbb{R}^{3n}$, $G_S = G_{T_{2\theta}} + G_T$, $0 \leq \delta$.

```
for $\eta \in \{2\delta, \frac{2}{3}2\delta, \frac{2\delta}{2}^*, \frac{2\delta}{3}^*\}$ do
  Set $\mathcal{T}_1^\eta, \mathcal{T}_2^\eta, \mathcal{T}_3^\eta$ as global variables.
  Compute the sets $\mathcal{T}_1^\eta = S \cap B_\eta(y_1)$,
  $\mathcal{T}_2^\eta = S \cap B_\eta(y_2), \mathcal{T}_3^\eta = S \cap B_\eta(y_3)$.
end for
for $1 \leq i \leq k$ do
  $\mathcal{T}_1 \leftarrow SubR(y_1, y_2, ..., y_k, \delta, 2/3\delta, i)$.
  $\mathcal{T}_2 \leftarrow SubR(y_1, y_2, ..., y_k, 2/3\delta, \delta, i)$.
end for
Return $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2\}$.
*The sets $\mathcal{T}_i^{\frac{\delta}{2}}$ and $\mathcal{T}_i^{\frac{\delta}{3}}$ are computed only if used by the Subroutine.
```

Algorithm 6.3 Subroutine of Algorithm 6.2 (with once the second splitting strategy)

Function $SubR(y_1, y_2, ..., y_k, \delta_1, \delta_2, i)$
Input: $y = (y_1, y_2, ..., y_k) \in \mathbb{R}^{kn}$, $1 \leq t$, $0 \leq \delta_1, \delta_2$, $1 \leq i \leq k$.

```
for $1 \leq l \neq i \leq k$ do
  Set $j = \{1, 2, 3\} \setminus \{i, l\}$.
  for each $(t_l, t_j) \in \mathcal{T}_l^{\delta_1} \times \mathcal{T}_j^{\delta_1/2}$ do
    Compute $z_l = t_l G_S^{-1}$, $z_j = t_j G_S^{-1}$.
    Compute $m'_l = z_l G_{T_{2\theta}}$, $m'_j = z_j G_{T_{2\theta}}$,
    and $n'_l = z_l G_T$, $n'_j = z_j G_T$.
    if $m'_j - m'_l \in 2S$ then
      Compute $p'_i = m'_l - n'_l - n'_j$.
      Compute the sets $\mathcal{V}_i = V \cap B_{\delta_2}(y_i - p'_i)$
      for $v_i \in \mathcal{V}_i$ do
        Add $(t_1, ..., t_{i-1}, v_i + p'_i, t_{i+1}, ..., t_k)$ to the list $\mathcal{T}$.
      end for
    end if
  end for
end for
Return $\mathcal{T}$.
```

Theorem 6.5. *Let $\Gamma(2S, T_{2\theta}, T, 3)$ and y be respectively a lattice and a point in $\mathbb{R}^{3n}$. Algorithm 6.2 outputs the set $\Gamma(2S, T_{2\theta}, T, 3) \cap B_\delta(y)$ in worst-case time*

$$\mathfrak{C}_{A.6.2}(\delta) = 3\mathfrak{C}_{S\cap B_{2\delta}(y)} + 6[l(S,2\delta)l(S,\delta)\mathfrak{C}_{V\cap B_{\frac{2}{3}\delta}(y)} + l(S,\frac{4}{3}\delta)l(S,\frac{2\delta}{3})\mathfrak{C}_{V\cap B_\delta(y)}]. \tag{6.12}$$

For instance, for BDD ($\delta = \frac{1}{4}$) we now get a complexity

$$\begin{aligned}\mathfrak{C}_{A.6.2}(\delta = \frac{1}{4}) =&3\mathfrak{C}_{S\cap B_{\frac{1}{2}}(y)} + 6l(S,\frac{1}{3})\mathfrak{C}_{V\cap B_{\frac{1}{4}}(y)} + 12n\mathfrak{C}_{V\cap B_{\frac{1}{6}}(y)},\\ \leq&3\mathfrak{C}_{S\cap B_{\frac{1}{2}}(y)} + [18+12n]\mathfrak{C}^V_{BDD},\end{aligned} \tag{6.13}$$

i.e. a complexity linear in n whereas it was quadratic within (6.11).

Which list decoder should we use among the two presented? The choice between the two strategies depends if it is more expensive to pay $|\alpha|$ or $\mathfrak{C}_{T\cap B_{2\delta}(y)}$ and $l(S,2\delta)l(S,\delta)$.

6.2.3 Bounding the list size of Λ_{24} and $\mathcal{N}_{72}$

We bound the list size of Λ_{24} and $\mathcal{N}_{72}$ via the maximum number of elements in the list $\mathcal{T}$ returned by Algorithm 4.5' (without the removing step) or Algorithm 6.2, similarly to what we did with the parity lattices. The bounds are obtained via (4.25) combined with the following lemma.

Lemma 6.2. *Let $\Lambda = \Lambda' + [\Lambda/\Lambda']$ and $y \in \mathbb{R}^n$. Let $x_i \in\ [\Lambda/\Lambda']$.*

$$|B_r(y)\cap\Lambda| = \sum_{i=1}^{|\Lambda/\Lambda'|} |B_r(y - x_i)\cap\Lambda'|, \tag{6.14}$$

$$\leq |\Lambda/\Lambda'| \cdot \max_i |B_r(y - x_i)\cap\Lambda'|. \tag{6.15}$$

Lemma 6.3. *The list size of Λ_{24} and $\mathcal{N}_{72}$ are respectively bounded as*

$$\begin{aligned}l(\Lambda_{24},\delta) \leq& \min\{2^4\cdot 6\cdot l(E_8,\delta)l(E_8,\frac{2}{3}\delta)\big[l(E_8,\frac{\delta}{2}) + l(E_8,\frac{\delta}{3})\big],\\ &6[l(E_8,2\delta)l(E_8,\delta)l(E_8,\frac{2}{3}\delta) + l(E_8,\frac{4}{3}\delta)l(E_8,\frac{2\delta}{3})l(E_8,\delta)]\},\end{aligned} \tag{6.16}$$

$$\begin{aligned}l(\mathcal{N}_{72},\delta) \leq& \min\{2^{12}\cdot 6\cdot l(\Lambda_{24},\delta)l(\Lambda_{24},\frac{2}{3}\delta)\big[l(\Lambda_{24},\frac{\delta}{2}) + l(\Lambda_{24},\frac{\delta}{3})\big],\\ &6[l(\Lambda_{24},2\delta)l(\Lambda_{24},\delta)l(\Lambda_{24},\frac{2}{3}\delta) + l(\Lambda_{24},\frac{4}{3}\delta)l(\Lambda_{24},\frac{2\delta}{3})l(\Lambda_{24},\delta)]\}.\end{aligned} \tag{6.17}$$

The first splitting strategy with three cases could also be used to obtain better bounds for large δ (see the section on the parity lattices).

6.2.4 Decoding Λ_{24} and $\mathcal{N}_{72}$ on the Gaussian channel

The analysis is similar to the one performed for *BW* lattices in Section 5.2.1. We will therefore be brief on the explanations.

The sphere lower bound for $P_e^\Lambda(opt, n, \sigma^2) = 10^{-4}$ in dimension 72 yields a distance to Poltyrev limit of 2.1 dB. The MLD performance of Λ_{24} for this error probability is 3.3 dB. Regarding the relative radius to ensure $P(x \notin \mathcal{T}^\delta) = P(||w||^2 > r) \lessapprox 10^{-4}$ with regular list decoding, we find with Equation (5.12) a value $\delta^* \approx 0.57$ for $\mathcal{N}_{72}$ and $\delta^* \approx 0.41$ for Λ_{24}.

An important observation when computing the performance of the modified list decoders on the Gaussian channel is the following. Let $T \in \mathbb{R}^n$. For Λ_{24} and $\mathcal{N}_{72}$ constructed as $\Gamma(2S, T_{2\theta}, T, 3)$, we have (see e.g. the proof of Theorem 6.1)

$$\mathrm{vol}(\Gamma(2S, T_{2\theta}, T, 3))^{\frac{2}{3n}} = \mathrm{vol}(T)^{\frac{2}{n}}, \tag{6.18}$$

whereas for the parity lattices, we have $\mathrm{vol}(L_{kn})^{\frac{2}{3n}} = 2^{\frac{1}{k}}\,\mathrm{vol}(T)^{\frac{2}{n}}$. This means that the equivalent VNR Δ is the same when decoding in $\Gamma(2S, T_{2\theta}, T, 3)$ and in T. This will be taken into account in the formulas below to estimate δ^*.

Decoding Λ_{24}.
Note that $E_8 = BW_8$ (used as S to construct Λ_{24}) can be decoded via the algorithm presented in Section 5.2.1.

With the first list decoder (the list-decoding version of Algorithm 4.2' without the splitting strategy), (5.16) becomes (see the proof of Theorem 5.3)

$$U_{24}(\delta,\Delta) = \min\{3U_8(\delta,\Delta)^2 + 3U_8(\delta,2\Delta)(1-U_8(\delta,\Delta))^2, 1\}. \tag{6.19}$$

Assume that $\delta^* \leq \frac{1}{4}$ with this algorithm. If this holds, $T, V \cong E_8$ are decoded with Algorithm 5.1 (the recursive BDD). Hence, $U_8(\frac{1}{4},\Delta) = P_e(BDD,\Delta)$ is given by the curve $n = 8$ in Figure 5.7. With (6.19), for $\Delta = 3.3$ dB we find $U_{24}(\delta = \delta^*, \Delta) \leq 10^{-4}$, which confirms that $\delta^* < 1/4$. As a result, we can use Algorithm 4.2' for quasi-MLD decoding of Λ_{24}. The complexity of Algorithm 4.2' is

$$\begin{aligned}\mathfrak{C}^{\Lambda_{24}}_{QMLD} = \mathfrak{C}_{A.4.2'}(\Lambda_{24}, \delta = \delta^*) =& 2^4(3\mathfrak{C}_{A.5.1}(E_8) + 3\mathfrak{C}_{A.5.1}(RE_8)),\\ =& 96\mathfrak{C}_{A.5.1}(E_8).\end{aligned} \tag{6.20}$$

Alternatively, if we use the second list decoder without the splitting strategy (i.e. Algorithm 6.1), (5.16) becomes (since $\text{vol}(S)^{\frac{2}{n}} = 1/2\,\text{vol}(T)^{\frac{2}{n}}$)

$$U_{24}(\delta,\Delta) = \min\{3U_8(2\delta,\frac{\Delta}{2})^2 + 3U_8(\delta,2\Delta)(1-U_8(\delta,\frac{\Delta}{2}))^2, 1\}, \tag{6.21}$$

where $U_8(\delta,\Delta)$ is now computed with (5.21) since E_8 is list decoded as in Section 5.2.1 (i.e. with Algorithm 5.3 without the removing step). With this strategy, we find $\delta^* \approx 1/4$. The complexity is (similarly to (6.11))

$$\mathfrak{C}^{\Lambda_{24}}_{QMLD} = \mathfrak{C}_{A.6.1}(\Lambda_{24}, \delta = \delta^*) = 3\mathfrak{C}(E_8, 2\delta) + 3l'(E_8, 2\delta)^2\mathfrak{C}(E_8,\delta). \tag{6.22}$$

Decoding $\mathscr{N}_{72}$.
Regarding $\mathscr{N}_{72}$, with the first decoder we have (similarly to (6.19))

$$U_{72}(\delta,\Delta) = \min\{3U_{24}(\delta,\Delta)^2 + 3U_{24}(\delta,2\Delta)(1-U_{24}(\delta,\Delta))^2, 1\}. \tag{6.23}$$

Consider a MLD decoder for Λ_{24} such that $U_{24}(\delta,\Delta) = P_e^{\Lambda_{24}}(opt,\Delta)$. Then, when $\Delta > 1$, $U_{72}(\delta,\Delta) \approx 3(P_e^{\Lambda_{24}}(opt,\Delta))^2$. The performance of this decoder is shown by the curve $U_{72}(\Delta)$ on Figure 5.1 in Example 5.1. Unlike for the parity lattices, the curve for $P_e^{\Lambda_{24}}(opt,\Delta)$ should not be shifted to the left before squaring, as expalined in Example 5.1. We easily see that this decoder is powerful enough to get quasi-MLD performance for $\mathscr{N}_{72}$. The complexity is then

$$\begin{aligned}\mathfrak{C}^{\mathscr{N}_{72}}_{QMLD} =& 2^{12}\cdot[3\mathfrak{C}^{\Lambda_{24}}_{MLD} + 3\mathfrak{C}^{\Lambda_{24}}_{MLD}],\\ =& 2^{12}\cdot 6\cdot \mathfrak{C}^{\Lambda_{24}}_{MLD}.\end{aligned} \tag{6.24}$$

If we use the second list decoder without the splitting strategy (i.e. Algorithm 6.1), (5.16) becomes

$$U_{72}(\delta,\Delta) = \min\{3U_{24}(2\delta,\frac{\Delta}{2})^2 + 3U_{24}(\delta,2\Delta)(1-U_{24}(\delta,\frac{\Delta}{2}))^2, 1\}, \tag{6.25}$$

where $U_{24}(\delta,\Delta)$ is computed with (6.19). We find $\delta^* \approx 0.22$. The complexity is

$$\mathfrak{C}^{\mathscr{N}_{72}}_{QMLD} = \mathfrak{C}_{A.6.1}(\mathscr{N}_{72}, \delta = \delta^*) = 3\mathfrak{C}(\Lambda_{24}, 2\delta) + 3l'(\Lambda_{24}, 2\delta)^2\mathfrak{C}(\Lambda_{24},\delta). \tag{6.26}$$

Finally, if we use Algorithm 6.2, the complexity is

$$\begin{aligned}\mathfrak{C}_{A.6.2}(\mathscr{N}_{72},\delta) =& 3\mathfrak{C}(\Lambda_{24},2\delta) + 6(l'(\Lambda_{24},2\delta)l'(\Lambda_{24},\delta)\mathfrak{C}(\Lambda_{24},2/3\delta) +\\ & l'(\Lambda_{24},\frac{4}{3}\delta)l'(\Lambda_{24},\frac{2}{3}\delta)\mathfrak{C}(\Lambda_{24},\delta)).\end{aligned} \tag{6.27}$$

Nevertheless, computing the optimal δ^* for quasi-MLD in this latter case is more involved and left for future work.

The curve of quasi-optimal performance of $\mathscr{N}_{72}$ on the Gaussian channel is depicted in Figure 6.2. The figure also shows the performance of $L_{3\cdot 24}$ discussed in the next section. The performance of $\mathscr{N}_{72}$ is at a distance of 2.6 dB only from Poltyrev limit at around 10^{-5} of error per point.

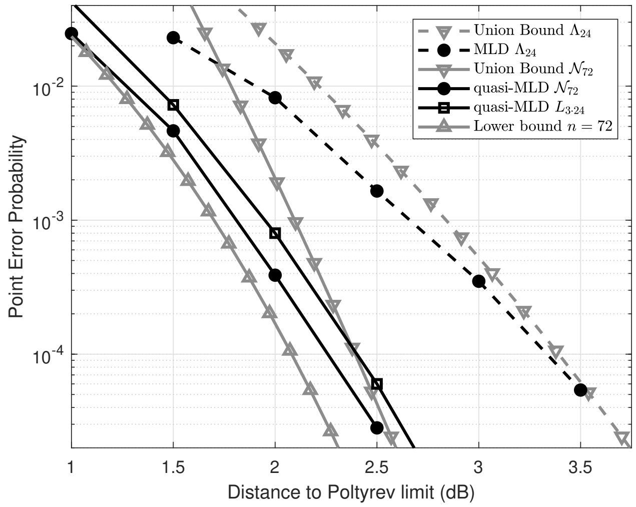


Figure 6.2: Performance of $\mathscr{N}_{72}$ and $L_{3\cdot 24}$ on the Gaussian channel. The union bound is computed from the two first lattice shells of $\mathscr{N}_{72}$. The curves for Λ_{24} are also provided for comparison.

6.2.5 The 3-parity-Leech lattice in dimension 72

Looking at the complexity given by Equations (6.8) and (6.9), it is tempting to consider the parity lattice $\Gamma(2S, \beta, 3)_{\mathcal{P}}$ instead of $\mathscr{N}_{72}$. Indeed, the worst-case complexity of Algorithms 4.2' and 4.5' would be reduced by a factor of $|\alpha| = 2^{12}$. This lattice $L_{3\cdot 24} = \Gamma(\lambda\Lambda_{24}, [\Lambda_{24}/\lambda\Lambda_{24}], 3)_{\mathcal{P}}$ is the 3-parity-Leech lattice considered in Example 5.1. $L_{3\cdot 24}$ has the same minimum distance as $\mathscr{N}_{72}$ and a volume $\text{vol}(L_{3\cdot 24}) = \text{vol}(\mathscr{N}_{72}) \times |\alpha|$ (using (3.21)). Its fundamental coding gain is:

$$\gamma(L_{3\cdot 24}) = \gamma(\Gamma(2S, T_{2\theta}, T, 3)) \times \frac{1}{(2^{12})^{\frac{2}{72}}} \approx 6.35. \tag{6.28}$$

Lemma 6.4. *The kissing number of* $L_{3\cdot 24}$ *is 28,894,320.*

The proof is provided in Appendix 6.5.5. The kissing number is about $2^{7.75}$ smaller than the kissing number of $\mathscr{N}_{72}$ (which is $6,218,175,600$). As a result, one can state the following regarding the relative performance of these two lattices on the Gaussian channel: 1 dB is lost by the parity-Leech lattice due to a smaller γ, but using the rule of thumb that 0.1 dB is lost each time the kissing number is doubled [FU98], there is also an improvement of 0.8 dB. Overall, we expect the performance of these two lattices to be only 0.2 dB apart but where the decoding complexity of the 3-parity-Leech lattice is significantly reduced. The quasi-MLD performance is shown in Figure 6.2 and it is indeed at 0.2 dB from the one of $\mathscr{N}_{72}$.

Consider Algorithm 4.4 for decoding. (5.16) yields

$$U_{3\cdot 24}(\delta, \Delta) = 3U_{24}(\delta, \frac{\Delta}{2^{\frac{1}{3}}})^2 + 3U_{24}(\delta, 2^{\frac{2}{3}}\Delta)(1 - U_{24}(\frac{1}{4}, \frac{\Delta}{2^{\frac{1}{3}}}))^2. \tag{6.29}$$

A MLD decoder for Λ_{24} as subroutine is not powerful enough to get quasi-MLD performance (see the curve for $U_{3\cdot 24}$ on Figure 5.1). We can for instance consider a sphere decoder computing $\Lambda_{24} \cap B_{\delta\cdot d(\Lambda_{24})}(y)$. Then $U_{24}(\delta, \Delta) = F(24, \delta \cdot d(\Lambda_{24}), \sigma^2)$ and δ^* should be chosen such that $3 \cdot F(24, \delta^* \cdot d(\Lambda_{24}), \sigma^2)^2 \approx 1/2 \cdot P_e^{L_{3\cdot 24}}(opt, \sigma^2)$. We find $\delta^* \approx 25/64$. With Theorem 4.1 we get that $l(\Lambda_{24}, \delta^*) = 4$. The (worst-case) complexity of Algorithm 4.4, given by (4.1), becomes

$$\begin{aligned} \mathfrak{C}_{QMLD}^{L_{3\cdot 24}} &= 3\mathfrak{C}_{\Lambda_{24} \cap B_{\delta^*\cdot d(\Lambda_{24})}(y)} + 3l(\Lambda_{24}, \delta^*)^2 \mathfrak{C}_{\Lambda_{24} \cap B_{\delta^*\cdot d(\Lambda_{24})}(y)}, \\ &= 51 \cdot \mathfrak{C}_{\Lambda_{24} \cap B_{\delta^*\cdot d(\Lambda_{24})}(y)}. \end{aligned} \tag{6.30}$$

6.3 Construction and decoding of codes

6.3.1 Reed-Muller codes as single parity-check group

Another single parity-check group respecting the property $d(V) = 2d(T)$ is the well-known Reed-Muller codes, defined by the squaring construction as:

$$\begin{aligned} RM(r,m) =&\{(v_1' + m, v_2' + m), v_1', v_2' \in RM(r-1, m-1), \\ & m \in [RM(r, m-1)/RM(r-1, m-1)]\}, \end{aligned} \tag{6.31}$$

where $r \leq m$ and with initial condition:

$$RM(a, 0) = \mathbb{F}_2,\ RM(-a-1, 0) = \{0\}, a \in \mathbb{Z}^+.$$

In our framework the Reed-Muller codes are:

$$\begin{aligned} RM(r,m) =&\Gamma(V, \beta, 2)_{\mathcal{R}}, V = RM(r-1, m-1), \\ & \beta = [RM(r, m-1)/RM(r-1, m-1)]. \end{aligned} \tag{6.32}$$

Algorithm 4.1 can be adapted to Reed-Muller codes to yield the recursive Algorithm 6.4 (the analog of Algorithm 5.1), with complexity $O(n^2)$.

Algorithm 6.4 Recursive BDD of $RM(r,m)$ (where $2n = 2^m$)

```
Function $RecRM(y, r, m)$
Input: $y \in \mathbb{R}^{2^m}, r \geq 0, m \geq 0$
if $m = 0$ then
    if $r \geq 0$ then
        if $y > 0$ then $x_{opt} \leftarrow 1$ else $x_{opt} \leftarrow 0$ end if   // Decoding in $\mathbb{F}_2$
    else
        $x_{opt} \leftarrow 0$   // Decoding in $\{0\}$
    end if
else
    $u_1 \leftarrow RecRM(y_1, r, m-1)$, $u_2 \leftarrow RecRM(y_2, r, m-1)$
    $v_2 \leftarrow RecRM((y_2 - u_1), r-1, m-1)$. Store $\hat{x} \leftarrow (u_1, v_2 + u_1)$.
    $v_1 \leftarrow RecRM((y_1 - u_2), r-1, m-1)$. Store $\hat{x}' \leftarrow (u_2 + v_1, u_2)$.
    $x_{opt} = \underset{x \in \{\hat{x}, \hat{x}'\}}{\text{argmin}} ||y - x||$
end if
Return $x_{opt}$
```

6.3.2 Codes obtained via the k-ing construction

In this subsection, the groups S, T, V involved in the construction $\Gamma(V, \alpha, \beta, n)$ are finite, i.e. we consider codes.

The distance metric $d(\cdot, \cdot)$ that we use with codes is the Hamming distance. Again, for $k = 3$, choosing codes such that $d(V) = 2d(T) = 4d(S)$ might yield a minimum distance of $\Gamma(V, \alpha, \beta, 3)$ equal to $3d(S)$ (see Theorem 4.2 in Section 4.2). We also need the concept of polarisation for codes.

The polarisation of codes

Let S, T, and V be codes. The analog of (7.22) for codes instead of lattices is a code T^* satisfying

$$T \cong T^*, \quad \text{and} \quad S = T + T^*, \quad \text{and} \quad T \bigcap T^* = V. \tag{6.33}$$

In this scope, $T \cong T^*$ means that T and T^* are different versions of the same code. The main difference with (7.22) is that S and T are not required to be the same code.

Such a pair (T, T^*) can be obtained via tools of the theory of cyclic codes, namely *idempotents* [MS77, Chap. 8, Sec. 3]. Indeed, idempotents enable to easily compute the sum and intersection of two codes.

Let C be a cyclic code, i.e. an ideal in the ring of polynomial over the field $\mathbb{F}_2$ modulo $x^n + 1$. The codewords of C are all multiples of a generator polynomial $g(x)$. Therefore, we say that C is generated by $g(x)$ and we use the notation $C = \langle g(x) \rangle$. An idempotent $E(x)$ is a polynomial having the property:

$$E(x) = E(x)^2. \tag{6.34}$$

It is shown in [MS77, Chap. 8, Sec. 3] that any cyclic code contains a unique idempotent such that $C = \langle E(x) \rangle$. Consider two codes $C = \langle E(x) \rangle$ and $C' = \langle E'(x) \rangle$. The sum and the intersection of these two codes can be derived thanks to the properties of idempotents:

$$\begin{aligned} C + C' &= \langle E(x) + E'(x) + E(x)E'(x) \rangle. \\ C \cap C' &= \langle E(x)E'(x) \rangle. \end{aligned} \tag{6.35}$$

The proof of (6.35) is straightforward (see problem (10) in [MS77, Chap. 8, Sec. 3]). Moreover, an idempotent is said to be *primitive* in the ring $\mathbb{F}_2[x]/x^n + 1$ if it is the idempotent of an ideal that does not contain any smaller ideal (a minimal ideal, also called an irreducible code). Any two primitive idempotents $\theta_i(x)$ and $\theta_j(x)$ satisfy: $\theta_i(x) \cdot \theta_j(x) = 0$. Finally, the dual of C is generated by the reciprocal $1 + E(x)$.
Note: this connection between idempotents and the polarisation is believed to be new.

Decoders for some binary codes

Let $T = (8, 4, 4)$ be a extended binary Hamming code and $T^* = (8, 4, 4)^*$ a different version, by coordinate permutation in $\mathbb{F}_2^8$, of this same code such that $T + T^*$ is the $(8, 7, 2)$ single parity-check code and $T \cap T^*$ is the $(8, 1, 8)$ repetition code. T and T^* are obtained as follows: It is well-known that the simplex code $\mathcal{S}_m = (n = 2^m - 1, m, 2^{m-1})$ is generated by any primitive idempotent $\theta_s(x)$, where s is prime with n. A version of the dual of the simplex code is the Hamming code, which has therefore $1 + \theta_s(x)$ as idempotent. If $m = 3$, the two possibilities are $\theta_1(x)$ and $\theta_3(x)$. $\langle 1 + \theta_1(x) \rangle$ as well as $\langle 1 + \theta_3(x) \rangle$ are both the $(7, 4, 3)$ Hamming codes. Hence, if we choose T and T^* as $\langle 1 + \theta_1(x) \rangle$ and $\langle 1 + \theta_3(x) \rangle$ both extended by adding an overall parity-check bit we get the desired result.
The k-ing construction for the two following codes was previously presented in [DB04], but without the use of idempotents.

Theorem 6.6. *Take $V = T \cap T^*$, $\alpha = |(T + T^*)/T|$, and $\beta = |T/(T \cap T^*)|$. $\Gamma(V, \alpha, \beta, 3)$ is the binary Golay code of length 24 and $\Gamma(V, \alpha, \beta, 5)$ is a $(40, 20, 8)$ type II extremal code [DB04].*

Proof. Similarly to the proof of Theorem 6.1, one needs to show that the minimum distance of $\Gamma(V, \alpha, \beta, k)$ is a multiple of 4. Combining this property with Theorem 4.2, i.e. $d(\Gamma(V, \alpha, \beta, k)) \geq 6$, yields the result. See [DB04] for more details. □

Similarly to Λ_{24}, one can use Algorithm 4.2 since $|\alpha| = 2^3$ is small . This yields an efficient BDD algorithm for these two codes, with complexity:

$$\mathfrak{C}_{A.4.2} = 2^3 \cdot (k \cdot \mathfrak{C}^T_{BDD} + k \cdot \mathfrak{C}^V_{BDD}), \tag{6.36}$$

where $k = 3$ for the Golay code and $k = 5$ for the second code of length 40.
Now, let $T = (24, 12, 8)$ be the binary Golay code and $T^* = (24, 12, 8)^*$ a different version of this same code such that $T + T^*$ is the $(24, 23, 2)$ single parity-check code and $T \cap T^*$ is the $(24, 1, 24)$ repetition code.

Theorem 6.7. *Take $V = T \cap T_*$, $\alpha = |(T + T^*)/T|$, and $\beta = |T/(T \cap T^*)|$. $\Gamma(V, \alpha, \beta, 3)$ is a [72,36,12] code [DB04].*

Again, Algorithm 4.2 can be considered for BDD of this code, where the decoding in T needs to be done with a radius equal to only 3 since $d(\Gamma(V, \alpha, \beta, 3))/2 < 2d(T)/2$. The complexity is

$$\mathfrak{C}_{A.4.2} = 2^{11} \cdot (3\mathfrak{C}^T_{BDD} + 3\mathfrak{C}^V_{BDD}).$$

With these settings, Algorithm 4.2 is similar to the one introduced in [DBNS08, Sec. 5] since the first step in both algorithms, i.e. trying all the cosets $m' \in \alpha$, is the same (but no BDD property is mentioned in [DBNS08]). However, since $|\alpha|$ is large in this case, the decoding complexity may be lower with Algorithm 4.3.

6.4 Addtional numerical results

6.4.1 Lattice decoding benchmark

We compare the performance of lattices and decoders shown in the previous sections to existing schemes in the literature at $P_e = 10^{-5}$. For fair comparison at different dimensions, we let P_e be either the

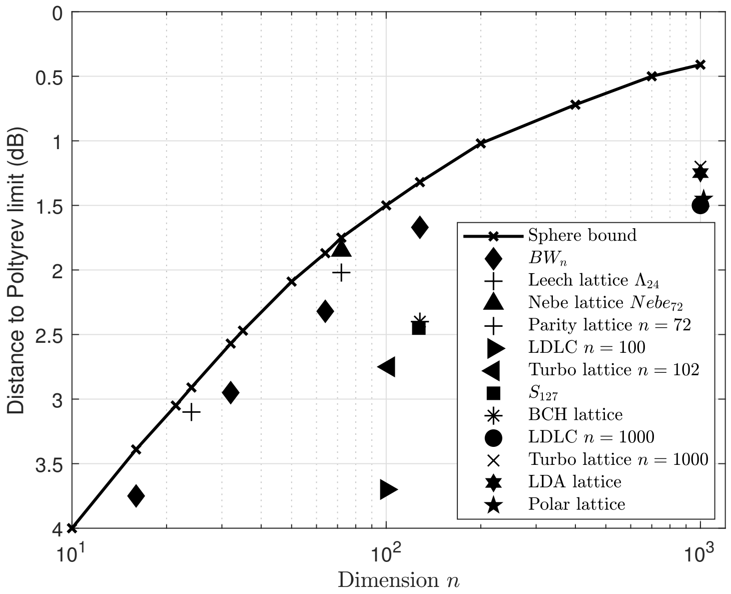


Figure 6.3: Performance of different lattices for normalized error probability $P_e = 10^{-5}$.

symbol-error probability or the normalized error probability, which is equal to the point-error probability divided by the dimension (as done in e.g. [TVZ99]).

First, several constructions have been proposed for block-lengths around $n = 100$ in the literature. In [MKO18] a two-level construction based on BCH codes with $n = 128$ achieves this error probability at 2.4 dB. The decoding involves an OSD of order 4 with 1505883 candidates. In [AV00] the multilevel (non-lattice packing) $\mathcal{S}_{127}$ ($n = 127$) has similar performance but with much lower decoding complexity via generalized minimum distance decoding. In [SSP11] a turbo lattice with $n = 102$ and in [SFS08] a LDLC with $n = 100$ achieve the error probability with iterative methods at respectively 2.75 dB and 3.7 dB (unsurprisingly, these two schemes are efficient for larger block-lengths). All these schemes are outperformed by BW_{64}, the 3-parity-Leech lattice, and $\mathscr{N}_{72}$, where $P_e = 10^{-5}$ is reached at respectively 2.3 dB, 2.02 dB and 1.85 dB. Moreover, BW_{128} has $P_e = 10^{-5}$ at 1.7 dB, which is similar to many schemes with block-length $n = 1000$ such as the LDLC (1.5 dB) [SFS08], the turbo lattice (1.2 dB) [SSP11], the polar lattice with $n = 1024$ (1.45 dB) [YLW13], and the LDA lattice (1.27 dB) [dPBZB12]. This benchmark is summarized on Figure 6.3.

6.4.2 Finite constellation

Since the decoding complexity of BW_{64} is only quadratic, this lattice is a good candidate to design finite constellations in dimension 64. We uncover the performance of a Voronoi constellation [CS83] [For89b] based on the partition $BW_{64}/2^{\eta}BW_{64}$ via Monte Carlo simulation, where η is the desired rate in bits per channel use (bpcu): i.e. both the coding lattice and the shaping lattice are based on BW_{64}. It follows that the encoding complexity is the same as the decoding complexity. Figure 6.4 exhibits the performance of our scheme for $\eta = 4$ bpcu. In our simulation, the errors are counted on the uncoded symbols. Consequently, the error probability also includes potential errors due to incomplete encoding, which seem to be negligible compared to decoding errors. Again, we plotted the best possible performance of *any* lattice-based constellation in dimension 64 (obtained from [TVZ99]). The scheme performs within 0.7 dB of the bound.

6.5 Appendix

6.5.1 The polarisation of Forney in [For88b]

Let us consider a $\mathbb{Z}[i]$-structure of E_8, and let G_S be a real generator matrix obtained from this structure (e.g. via (3.3)). Forney applies the rotation $R(8, \phi)$ on G_S to obtain the generator matrix $G_{T_{2\theta}}$ of $T_{2\theta}$

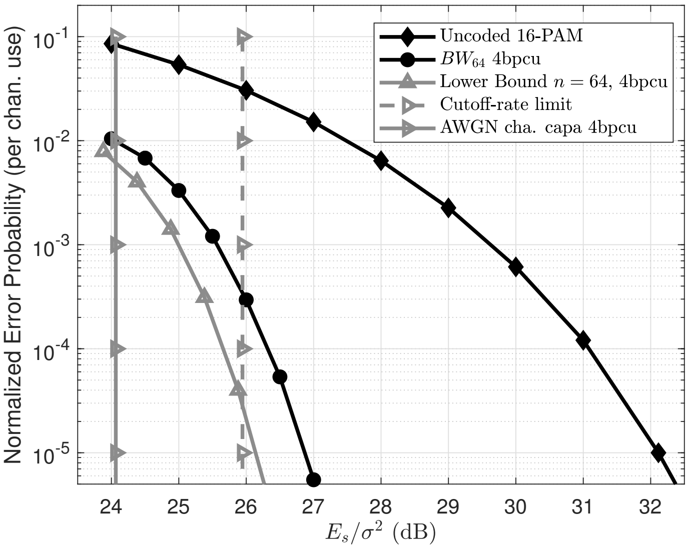


Figure 6.4: Performance of a Voronoi constellation based on the partition $BW_{64}/2^4BW_{64}$ where Algorithm 5.3', with $\delta = 3/8$ and $\aleph(\delta) = 20$, is used for encoding and decoding. The cutoff-rate limit is 1.7+0.179 dB right to Shannon limit (coding + shaping loss for $n = 64$) [FU98].

($T_{2\theta}$ is called RE_8 in [For88b]). I.e. the lattice S is scaled by a factor of $\sqrt{2}$ but rotated by an angle of $\pi/4 \neq \theta = \text{arc tan } \sqrt{7}$. The lattice $T_{2\theta}$ (R^*E_8 in [For88b]) is then constructed via the union of 8 cosets of $2S$, which is called the small miracle octad generators (SMOG). Forney does not explain how he found the SMOG.

It is possible to find a rotation matrix $B \in Aut(RE_8)$ such that $G_{RE_8} \cdot B$ is a also generator matrix of RE_8 but where the vectors have an angle of θ with the corresponding basis vectors in G_S. It follows that the "standard" polarisation rotation operator is $R = G_S^{-1} \cdot G_{RE_8} \cdot B$ and a "standard" generator matrix for $T_{2\theta}$ is $G_{T_{2\theta}} = G_S \cdot R^T$. Finally, one can verify that the SMOG is composed of coset leaders of $2S$ in the lattice generated by $G_{T_{2\theta}}$.

To conclude, the lattices T and $T_{2\theta}$ chosen by Forney form a polarisation of S, but the generator matrices chosen are unconventional.

6.5.2 Proof of Lemma 6.1

Proof. We let Λ_{24} be scaled such that $d(\Lambda_{24}) = 4$ and $\text{vol}(\Lambda_{24}) = 1$. Then, $S = \frac{1}{\sqrt{2}}\Lambda_{24}$ has $d(S) = 2$, $\text{vol}(S) = 2^{-12}$ and $\text{vol}(T) = \text{vol}(T_{2\theta}) = 1$, $d(T) = d(T_{2\theta}) = 4$. Also, $|\alpha| = |\beta| = 2^{12}$ from (3.21).

Using Theorem 4.2, we have $d(\Gamma(2S, T_{2\theta}, T, 3)) \geq 6$. Then, (6.3) shows that $\Gamma(2S, T_{2\theta}, T, 3)$ is obtained as the union of $|\alpha||\beta|^2 = 2^{36}$ cosets of $(2S)^3$. Hence, $\text{vol}(\Gamma(2S, T_{2\theta}, T, 3)) = \text{vol}((2S)^3)/2^{36} = 1$. The lattice has a fundamental coding gain of $d/\text{vol}^{2/72} \geq 6$. □

6.5.3 The good polarisation of the Leech lattice

Equation (4.13) is re-written below:

$$\begin{aligned} &\Gamma(V, \alpha, \beta, k) = \{(p_1, p_2, ..., p_k), \\ &p_1 = v_1 + m + n_1, p_2 = v_2 + m + n_2, ..., \\ &p_k = v_k + m - (n_1 + ... + n_{k-1})), v_i \in V, m \in \alpha, n_i \in \beta\}. \end{aligned} \tag{6.37}$$

Let $d(V), 2d(T) > kd(S)$ and consider an element $x \in \Gamma(V, \alpha, \beta, k)$. Assume that $v_i = 0$, $\forall i$, and that m and the $n_i's$ are chosen so that the first $k-1$ components of x have $d(0, p_i) = d(S)$. Observe that given m and the $n_i's$, there is no degree of freedom on p_k and one can wonder whether it is possible to have $d(0, p_k) = d(S)$. If not, we have succeeded to get $d(\Gamma(V, \alpha, \beta, k)) > kd(S)$.

Vectors of squared norm 6 in $\Gamma(2S, T_{2\theta}, T, 3)$

Let us consider a version of $\Gamma(2S, T_{2\theta}, T, 3)$ with a unit volume such that the squared minimum distance is either 6 or 8. We investigate the vectors of squared norm equal to 6 in $\Gamma(2S, T_{2\theta}, T, 3)$.
Let $x = (a, b, c) \in \Gamma(2S, T_{2\theta}, T, 3)$. Obviously, the vectors of squared norm 6 in $\Gamma(2S, T_{2\theta}, T, 3)$ exist if we can find a point x such that $||a|| = ||b|| = ||c|| = d(S)$. Assume that $||a|| = ||b|| = d(S)$, can we find c such that $||c|| = d(S)$? With (6.37), we deduce that this is equivalent to asking: Can we find m, n_1, n_2 such that

$$\begin{aligned} ||m + n_1 \mod 2S||^2 &= 2, \\ ||m + n_2 \mod 2S||^2 &= 2, \\ ||m + n_1 + n_2 \mod 2S||^2 &= 2? \end{aligned} \tag{6.38}$$

The computational cost of running through all possibilities is $\approx |\alpha||\beta|^2 = 2^{36}$. This can be however reduced as follows. Let us choose the coset representatives p of $S/2S$ of minimum squared norm; i.e. 0,2,3,4. The kissing number of Λ_{24} being $2 \times 24 \times (2^{12} - 1)$, the number of coset representatives of p of squared norm 2 is limited to $24 \times (2^{12} - 1)$. Since $m + n \mod 2S$ yields one of these coset representatives p of minimum squared norm, for a given m, there can be only $\frac{24\times(2^{12}-1)}{2^{12}-1}$ vectors n such that $||m + n \mod 2S||^2$ is of squared norm 2. Hence, the computational cost of the search can be reduced to $\approx |\alpha| \times 24^2$, which is tractable via computer search.

Similarly, the squared-norm-6 vectors in $\Gamma(2S, T_{2\theta}, T, 3)$ can also be characterized as follows. Consider a point $(a, b, c) \in \Gamma(2S, T_{2\theta}, T, 3)$ of squared norm 6 in $\Gamma(2S, T_{2\theta}, T, 3)$; $a = m + n_1 \neq b = m + n_2 \neq c = m + n_1 + n_2$, with $||a|| = ||b|| = ||c|| = d(S)$. As stated above, for a given m, there can be only 24 distinct pairs $m + n$ of minimum squared norm. Obviously, a, b, c should be three distinct elements among one of the $|\alpha|$ possible sets of 24 distinct pairs.
Since $a+b+c \mod 2S \in \alpha$, having vectors of squared norm 6 in $\Gamma(2S, T_{2\theta}, T, 3)$ is equivalent to asking if we can find three distinct minimum squared norm elements in $m + \beta$ such that their sum is in α. Hence, the bad vectors of the polarisation $(T, T_{2\theta})$ of S, as defined in [NP13], are:

$$B((T, T_{2\theta})) = \bigcup_{m\in\alpha} \{a + b + c \in \alpha : a \neq b \neq c,\ ||a|| = ||b|| = ||c|| = d(S), a, b, c \in m + \beta\}.$$

The good polarisation

Let $(T, T_{2\theta})$ and $(T', T'_{2\theta})$, $T \neq T'$ and $T_{2\theta} \neq T'_{2\theta}$, be two distinct polarisations of a lattice S. These two polarisations are said to be equivalent if they are in the same orbit under the action of $Aut(S)$. Indeed, two distinct polarisations in the same orbit have the same set of bad vectors $B((T, T_{2\theta}))$. If $S \cong E_8$, $Aut(S)$ is transitive on the set of polarisation matrices $(T, T_{2\theta})$. However, Nebe proved in [NP13] that there are 16 orbits if $S \cong \Lambda_{24}$. Moreover, polarisations from only one of these orbits give $B((T, T_{2\theta})) = \varnothing$. Therefore, only these polarisations enable to build a extremal lattice of dimension 72. Polarisations from this orbit are called good polarisations.
A good polarisation can be obtained via one of the nine $\mathbb{Z}[\lambda]$-structures of the Leech lattice ($\lambda = \frac{1+\sqrt{-7}}{2}$). Each $\mathbb{Z}[\lambda]$-structures is characterized by its automorphism group on $\mathbb{Z}[\lambda]$ (see Table 1 in [Neb12]). The good polarisation, found by Nebe [Neb12], is obtained via the $\mathbb{Z}[\lambda]$-structure of Λ_{24} with the automorphism group called $\mathrm{SL}_2(25)$.

6.5.4 A proof that $R(\mathscr{N}_{72}) > \sqrt{2}\rho(\mathscr{N}_{72})$

Lemma 6.5. *$R(\mathscr{N}_{72}) > \sqrt{2}\rho(\mathscr{N}_{72})$.*

The proof of this lemma is due to Gabriele Nebe (private communication).

Proof. Let $\mathscr{N}_{72}$ be scaled such that $\rho(\mathscr{N}_{72}) = \sqrt{2}$. The proof is done by contradiction. Assume that $R(\mathscr{N}_{72}) = \sqrt{2}\rho(\mathscr{N}_{72}) = 2$. Then, for any point $1/2v \in 1/2\mathscr{N}_{72}$, there is a point $x \in \mathscr{N}_{72}$ with $||x - 1/2v|| \leq 2$. Squaring leads to $||2x - v||^2 \leq 16$. So each of the 2^{72} cosets of $2\mathscr{N}_{72}$ in $\mathscr{N}_{72}$ has to contain a point $w = 2x - v$ of squared norm smaller or equal to 16.

Now $\mathscr{N}_{72}$ has exactly 107502190683149087281 pairs $\pm w$ of squared norm ≤ 16 (obtained from the theta series of $\mathscr{N}_{72}$). This number is smaller than $|\mathscr{N}_{72}/2\mathscr{N}_{72}|$. Hence the covering radius of $\mathscr{N}_{72}$ is strictly larger than 2. □

6.5.5 Proof of Lemma 6.4

Proof. The proof is similar to that of Theorem 3.3 in [Neb12]. The vectors of squared norm in $\Gamma(V, T, 3)_{\mathcal{P}}$ have only the following possible forms.

1. $(a, 0, 0), a \in V$ and $||a||^2 = 8$. The number of such vectors (counting the combinations) is $196560 \cdot 3$ vectors. I.e. the minimal vectors in $(V)^3$.

2. $(n_1, n_2, 0)$, $n_1, n_2 \in T, n_1 + n_2 \in V$ and $||n_1||^2 = ||n_2||^2 = 4$. The number of such vectors (counting the combinations) is $196560 \cdot 48 \cdot 3$. There are 196560 possibilities for n_1. Given n_1 how many choices are they for n_2? This is equivalent to asking the number of squared norm 8 vectors in the coset $m + V$, which are therefore congruent mod V. It is well-known (see Theorem 2 in [CS99, Chap.12]) that this number is 48, 24 mutually orthogonal pairs of vectors (one can check that $|T/V| = 2^{12} \cdot 48 = 196560$, the number of minimal vectors of Λ_{24}). Hence, there are 48 choices for n_2. Finally, the factor 3 comes from the combinations.

□

Part

Lattices and neural networks with and without learning

Chapter 7

Neural network approaches to point lattices decoding

In 2012 Alex Krizhevsky and his team presented a revolutionary deep neural network in the ImageNet Large Scale Visual Recognition Challenge [KSH12]. The network largely outperformed all the competitors. This event triggered not only a revolution in the field of computer vision but has also affected many different engineering fields, including the field of digital communications.

In our specific area of interest, the physical layer, countless studies have been published since 2016. For instance, reference papers such as [OH17] gathered more than 800 citations in less than three years. However, most of these papers present simulation results: e.g. a decoding problem is set and different neural network architectures are heuristically considered. Learning via usual gradient-descent-like techniques is performed and the results are presented.

Our approach is different: we try to characterize the complexity of the decoding probem that should be solved by the neural network.

Neural network learning is about two key aspects: first, finding a function class $\Phi = \{f\}$ that contains a function "close enough" to a target function f^*. Second, finding a learning algorithm for the class Φ. Naturally, the less "complex" the target function f^*, the easier the problem. We argue that understanding this function f^* encountered in the scope of the decoding problem is of interest to find new efficient solutions.

Indeed, the first attempts to perform decoding operations with "raw" neural networks (i.e. without using the underlying graph structures of existing sub-optimal algorithms, as done in [NBB16]) were unsuccessful. For instance, an exponential number of neurons in the network is needed in [GCHtB17] to achieve satisfactory performance when decoding small length polar codes. We made the same observation when we tried to decode dense lattices typically used for channel coding [CBCB18b]. So far, it was not clear whether such a behavior is due to either an unadapted learning algorithm or a consequence of the complexity of the function to learn. However, unlike for channel decoding (i.e. dense lattice decoding), neural networks can sometimes be successfully trained in the scope of MIMO detection [SDW17] [CBCB18b].

In this chapter, the problem of lattice decoding is investigated. Lattices are well-suited to understand these observed differences as they can be used both for channel coding and to model MIMO channels.

We embrace a feed-forward neural network perspective. These neural networks are aggregation of perceptrons and compute a composition of the functions executed by each perceptron. For instance, if the activation functions are rectified linear unit (ReLU), each perceptron computes a piecewise affine function. Consequently, all functions in the function class Φ of this feed-forward neural network are CPWL.

We shall see that, under some assumptions, the lattice decoding problem is equivalent to computing a CPWL. The target f^* is thus a CPWL. The complexity of f^* can be assessed, for instance, by counting its number of affine pieces.

It has been shown that the minimum size of shallow neural networks, such that Φ contains a given CPWL function f^*, directly depends on the number of affine pieces of f^* whereas deep neural networks can "fold" the function and thus benefit of an exponential complexity reduction [MPCB14]. On the one hand, it is critical to determine the number of affine pieces in f^* to figure out if shallow neural networks can solve the decoding problem. On the other hand, when this is not the case, we can investigate if there exist preproccessing techniques to reduce the number of pieces in the CPWL function. We shall see that these preprocessing techniques are sequential and thus involve using a deep neural network.

Due to the nature of feed-forward neural networks, our approach is mainly geometric and combinatorial. It is restricted to low and moderate dimensions. Again, our main contribution is not to present new decoding algorithms but to provide a better understanding of the decoding/detection problem from a neural network perspective.

7.1 Preliminaries

Geometry. First, given a lattice Λ, the set $\mathcal{T}_f(x)$, for $x \in \Lambda$, denotes the set of lattice points having a common Voronoi facet with x.

Let $\overline{\mathcal{P}}(\mathcal{B})$ be the topological closure of $\mathcal{P}(\mathcal{B})$ and $\overset{\circ}{\mathcal{P}}(\mathcal{B})$ the interior of $\mathcal{P}(\mathcal{B})$. A k-dimensional element of $\overline{\mathcal{P}}(\mathcal{B}) \setminus \overset{\circ}{\mathcal{P}}(\mathcal{B})$ is referred to as k-face of $\mathcal{P}(\mathcal{B})$. There are 2^n 0-faces, called corners or vertices. This set of corners is denoted $\mathcal{C}_{\mathcal{P}(\mathcal{B})}$. The subset of $\mathcal{C}_{\mathcal{P}(\mathcal{B})}$ obtained with $z_i = 1$ is $\mathcal{C}^1_{i,\mathcal{P}(\mathcal{B})}$ and $\mathcal{C}^0_{i,\mathcal{P}(\mathcal{B})}$ for $z_i = 0$. To lighten the notations, we shall sometimes use $\mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ and $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$.

The remaining k-faces of $\mathcal{P}(\mathcal{B})$, $k > 0$, are parallelotopes. For instance, a $(n-1)$-face of $\mathcal{P}(\mathcal{B})$, say $\mathcal{F}_i$, is itself a parallelotope of dimension $n-1$ defined by $n-1$ vectors of $\mathcal{B}$. Throughout this part of the thesis, the term facet refers to a $n-1$-face.

Let v_j denote the vector orthogonal to the hyperplane

$$\{y \in \mathbb{R}^n : \ y \cdot v_j - p_j = 0\}. \tag{7.1}$$

A polytope (or convex polyhedron) is defined as the intersection of a finite number of half-spaces (as in e.g. [Cox73])

$$P_o = \{x \in \mathbb{R}^n : \ x \cdot A \leq b, \ A \in \mathbb{R}^{n \times m}, \ b \in \mathbb{R}^m\}, \tag{7.2}$$

where the columns of the matrix A are m vectors v_j.
Since a parallelotope is a polytope, it can be alternatively defined from its bounding hyperplanes. Note that the vectors orthogonal to the facets of $\mathcal{P}(\mathcal{B})$ are basis vectors of the dual lattice. Hence, a second useful definition for $\mathcal{P}(\mathcal{B})$ is obtained through the basis of the dual lattice:

$$\mathcal{P}(\mathcal{B}) = \{x \in \mathbb{R}^n : \ x \cdot G^{-1} \geq 0 \ , \quad x \cdot G^{-1} \leq 1, G \in \mathbb{R}^{n \times n}\}, \tag{7.3}$$

where each column vector of G^{-1} is orthogonal to two facets of $\mathcal{P}(\mathcal{B})$ and $(G^{-1})^T$ is a basis for the dual lattice of Λ.

We say that a function $g : \mathbb{R}^{n-1} \to \mathbb{R}$ is CPWL if there exists a finite set of polytopes covering $\mathbb{R}^{n-1}$, and g is affine over each polytope. The number of pieces of g is the number of distinct polytopes partitioning its domain.

$\vee$ and $\wedge$ denote respectively the maximum and the minimum operator. We define a convex (resp. concave) CPWL function formed by a set of affine functions related by the operator $\vee$ (resp. $\wedge$). If $\{g_k\}$ is a set of K affine functions, the function $f = g_1 \vee ... \vee g_K$ is CPWL and convex.

Neural networks. Given n scalar inputs $y_1, ..., y_n$ a perceptron performs the operation $\sigma(\sum_i w_i \cdot y_i)$. The parameters w_i are called the weights or edges of the perceptron and $\sigma(\cdot)$ is the activation function. The activation function $\sigma(x) = \max(0, x)$ is called ReLU. A perceptron can alternatively be called a neuron.
Given the input $y_1, ..., y_n$, a feed-forward neural network simply performs the operation [GBC16]:

$$\hat{z} = \sigma_d(...\sigma_2(\sigma_1(y \cdot G_1 + b_1) \cdot G_2 + b_2) \cdot ... \cdot G_d + b_d), \tag{7.4}$$

where:

- d is the number of layer of the neural network.

- Each layer of size m_i is composed of m_i neurons. The weights of each neuron of a layer are stored in the m_i columns of the matrix G_i. The vector b_i represents m_i biases.

- The activation functions σ_i are applied componentwise.

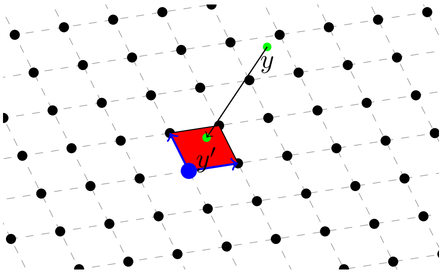


Figure 7.1: The point $y \in \mathcal{P}(\mathcal{B}) + x$ is translated in the fundamental parallelotope (in red on the figure) to get the point $y' \in \mathcal{P}(\mathcal{B})$. The blue arrows represent a basis $\mathcal{B}$ of the lattice.

7.2 From the CVP in $\mathbb{R}^n$ to the CVP in $\mathcal{P}(\mathcal{B})$.

It is well-known in lattice theory that $\mathbb{R}^n$ can be partionned as $\mathbb{R}^n = \bigcup_{x \in \Lambda} (\mathcal{P}(\mathcal{B}) + x)$. The parallelotope to which a point $y_0 \in \mathbb{R}^n$ belongs is:

$$y_0 \in \mathcal{P}(\mathcal{B}) + x, \tag{7.5}$$

with

$$x = \lfloor y_0 G^{-1} \rfloor \cdot G, \tag{7.6}$$

where the floor function $\lfloor \cdot \rfloor$ is applied componentwise. This floor function should not be confused with the round function $\lfloor \cdot \rceil$. Hence, a translation of y_0 by $-x$ results in a point y located in the fundamental parallelotope $\mathcal{P}(\mathcal{B})$. An instance of this operation is illustrated on Figure 7.1. As a result, a point y_0 to decode can be processed as follows:

- Step 0: a noisy lattice point $y_0 = x + w$ is observed, where $x \in \Lambda$ and $w \in \mathbb{R}^n$ is an additive noise.
- Step 1: compute $t = \lfloor y_0 \cdot G^{-1} \rfloor$ and get $y = y_0 - t \cdot G$ which now belongs to $\mathcal{P}(\mathcal{B})$.
- Step 2: find $\hat{z}$, where $\hat{x} = \hat{z} \cdot G$ is the closest lattice point to y.
- Step 3: the closest point to y_0 is $\hat{x}_0 = \hat{x} + t \cdot G$.

Since Step 1 and Step 3 have negligible complexity, an equivalent problem to the CVP in $\mathbb{R}^n$ is the CVP in $\mathcal{P}(\mathcal{B})$, which can simply be stated as follows.

Problem 7.1. *Given a point $y \in \mathcal{P}(\mathcal{B})$, find the closest lattice point $\hat{x} = \hat{z} \cdot G$.*

Remark 7.1. *Consider a point $y = x + w$, where $w = \epsilon_1 g_1 + ... + \epsilon_n g_n$, $x \in \Lambda$, $0 \leq \epsilon_1, ..., \epsilon_n < 1$, $g_1, ..., g_n \in \mathcal{B}$. Obviously, $y \in x + \mathcal{P}(\mathcal{B})$. The well-known Zero-Forcing (ZF) decoding algorithm computes*

$$\hat{z} = \lfloor y \cdot G^{-1} \rceil = \lfloor y_0 \cdot G^{-1} \rceil + xG^{-1}. \tag{7.7}$$

In other words, it simply replaces each ϵ_i by the closest integer, i.e. 0 or 1. The solution provided by this algorithm is one of the corners of the parallelotope $x + \mathcal{P}(\mathcal{B})$.

Remark 7.2. *From a complexity theory view point, Problem 7.1 is NP-hard. Indeed, since the above Steps 0,1, and 3 are of polynomial complexity, Problem 7.1 can be polynomially reduced to the CVP, which is known to be NP-hard (as stated in the previous section).*

7.3 Voronoi-reduced lattice basis

7.3.1 Voronoi- and quasi-Voronoi-reduced basis

The natural question arising from Problem 7.1 is the following: Is the closest lattice point to any point $y \in \mathcal{P}(\mathcal{B})$ one of the corners of $\mathcal{P}(\mathcal{B})$? Unfortunately, as illustrated on Figure 7.2, this is not always the case. Consequently, we introduce a new type of basis reduction.

Definition 7.1. *Let $\mathcal{B}$ be the $\mathbb{Z}$-basis of a rank-n lattice Λ in $\mathbb{R}^n$. $\mathcal{B}$ is said Voronoi-reduced if, for any point $y \in \mathcal{P}(\mathcal{B})$, the closest lattice point $\hat{x}$ to y is one of the 2^n corners of $\mathcal{P}(\mathcal{B})$, i.e. $\hat{x} = \hat{z}G$ where $\hat{z} \in \{0,1\}^n$.*

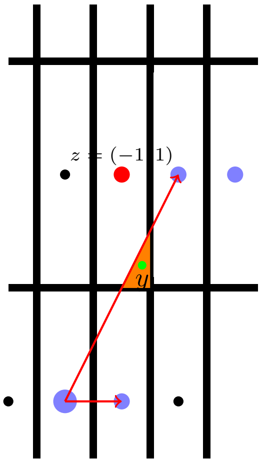


Figure 7.2: The red arrows represent the basis vectors. The orange area in $\mathcal{P}(\mathcal{B})$ belongs to the Voronoi region of the point $x = z \cdot G$, where $z = (-1, 1)$ (in red on the figure). Since this lattice point is not a corner of $\mathcal{P}(\mathcal{B})$, any point in this orange area, such as y, is not decoded to one of the corner of $\mathcal{P}(\mathcal{B})$ (the four blue points on the figure).

We will use the abbreviation *VR basis* to refer to a Voronoi-reduced basis. Figure 7.3 shows the hexagonal lattice A_2, its Voronoi cells, and the fundamental parallelotope of the basis $\mathcal{B}_1 = \{v_1, v_2\}$, where $v_1 = (1, 0)$ corresponds to $z = (1, 0)$ and $v_2 = (\frac{1}{2}, \frac{\sqrt{3}}{2})$ corresponds to $z = (0, 1)$. The basis $\mathcal{B}_1$ is Voronoi-reduced because

$$\mathcal{P}(\mathcal{B}_1) \subset \mathcal{V}(0) \cup \mathcal{V}(v_1) \cup \mathcal{V}(v_2) \cup \mathcal{V}(v_1 + v_2). \tag{7.8}$$

Lattice basis reduction is an important field in Number Theory. In general, a lattice basis is said to be of good quality when the basis vectors are relatively short and close to being orthogonal. We cite three famous types of reduction to get a good basis: Minkowski-reduced basis, Korkin-Zolotarev-reduced (or Hermite-reduced) basis, and LLL-reduced basis for Lenstra-Lenstra-Lovász [MG02] [Coh96]. A basis is said to be LLL-reduced if it has been processed by the LLL algorithm. This algorithm, given an input basis of a lattice, outputs a new basis in polynomial time where the new basis respects some criteria, see e.g. [Coh96]. The LLL-reduction is widely used in practice to improve the quality of a basis. The basis $\mathcal{B}_1$ in Figure 7.3 is Minkowski-, KZ-, and Voronoi-reduced.

Note that this new notion ensures that the closest lattice point $\hat{x}$ to any point $y \in \mathcal{P}(\mathcal{B})$ is obtained with a vector $\hat{z}$ having only binary values (where $\hat{x} = \hat{z} \cdot G$). As a result, it enables to use a decoder with only binary outputs to optimally solve the CVP in $\mathcal{P}(\mathcal{B})$.

Unfortunately, not all lattices admit a VR basis. Nevertheless, as we shall see in the sequel, some famous dense lattices listed in [CS99] admit a VR basis. Also, in some cases the LLL-reduction leads to a *quasi*-VR basis. Indeed, the strong constraint defining a VR basis can be relaxed as follows.

Definition 7.2. *Let $\mathcal{C}(\mathcal{B})$ be the set of the 2^n corners of $\mathcal{P}(\mathcal{B})$. Let O be the subset of $\mathcal{P}(\mathcal{B})$ that is covered by Voronoi cells of points not belonging to $\mathcal{C}(\mathcal{B})$, namely*

$$O = \mathcal{P}(\mathcal{B}) \setminus \left(\mathcal{P}(\mathcal{B}) \bigcap \left(\bigcup_{x \in \mathcal{C}(\mathcal{B})} V(x) \right) \right). \tag{7.9}$$

The basis $\mathcal{B}$ is said quasi-Voronoi-reduced if $\text{vol}(O) \ll \text{vol}(\Lambda)$.

Let $d^2_{OC}(\mathcal{B}) = \min_{x \in O, x' \in \mathcal{C}(\mathcal{B})} \|x - x'\|^2$ be the minimum squared Euclidean distance between O and $\mathcal{C}(\mathcal{B})$. The sphere packing structure associated to Λ guarantees that $d^2_{OC} \geq \rho^2$. Let $P_e(\mathcal{B})$ be the probability of error for a decoder where the closest corner of $\mathcal{P}(\mathcal{B})$ to y is decoded. In other words, the space of solution for this decoder is restricted to $\mathcal{C}_{\mathcal{P}(\mathcal{B})}$. The following lemma tells us that a quasi-Voronoi-reduced basis exhibits quasi-optimal performance on a Gaussian channel at high signal-to-noise ratio. In practice, the quasi-optimal performance is also observed at moderate values of signal-to-noise ratio.

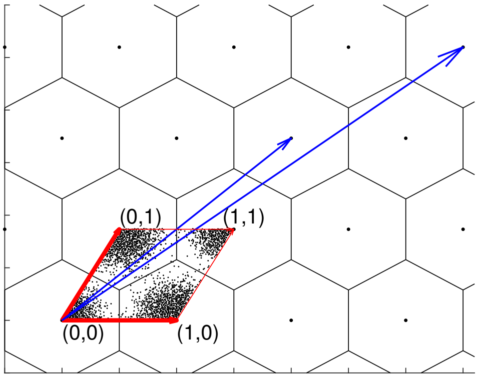


Figure 7.3: Voronoi-reduced basis $\mathcal{B}_1$ for A_2 (in red) and a non-reduced basis $\mathcal{B}_2$ (in blue). $\mathcal{P}(\mathcal{B}_1)$ is partitioned into 4 parts included in the Voronoi cells of its corners. $\mathcal{P}(\mathcal{B}_2)$ has 10 parts involving 10 Voronoi cells. The small black dots in $\mathcal{P}(\mathcal{B})$ represent Gaussian distributed points in $\mathbb{R}^2$ that have been aliased in $\mathcal{P}(\mathcal{B})$.

Lemma 7.1. *The error probability on the Gaussian channel when decoding a lattice Λ in $\mathcal{P}(\mathcal{B})$ can be bounded from above as*

$$\begin{aligned} P_e(\mathcal{B}) \leq P_e(ub) + & \\ \frac{\text{vol}(O)}{\det(\Lambda)} \cdot (e\Delta)^{n/2} \cdot \exp(-\frac{\pi e \Delta \gamma}{4} \cdot \frac{d_{OC}^2}{\rho^2}), & \end{aligned} \tag{7.10}$$

for Δ large enough and where $P_e(ub)$ is defined by (3.43).

Proof. If $\mathcal{B}$ is Voronoi-reduced and the decoder works inside $\mathcal{P}(\mathcal{B})$ to find the nearest corner, then the performance is given by $P_e(opt)$.
If $\mathcal{B}$ is quasi-Voronoi-reduced and the decoder only decides a lattice point from $\mathcal{C}(\mathcal{B})$, then an error shall occur each time y falls in O. We get

$$\begin{aligned} P_e(\mathcal{B}) &\leq P_e(opt) + P_e(O), \\ &\leq P_e(ub) + P_e(O). \end{aligned} \tag{7.11}$$

where

$$\begin{aligned} P_e(O) &= \int \cdots \int_O \frac{1}{\sqrt{2\pi\sigma^2}^n} \exp(-\frac{\|x\|^2}{2\sigma^2})\, dx_1 \ldots dx_n \\ &\leq \frac{1}{\sqrt{2\pi\sigma^2}^n} \exp(-\frac{d_{OC}^2}{2\sigma^2}) \;\, \text{vol}(O) \\ &= \frac{\text{vol}(O)}{\det(\Lambda)} \cdot (e\Delta)^{n/2} \cdot \exp(-\frac{\pi e \Delta \gamma}{4} \cdot \frac{d_{OC}^2}{\rho^2}). \end{aligned}$$

This completes the proof. □

7.3.2 Some examples

Structured lattices

We first state the following three theorems on the existence of VR bases for some famous lattices. The proofs are provided in Appendix 7.6.1.

Consider a basis for the lattice A_n with all vectors from the first lattice shell. Also, the angle between any two basis vectors is $\pi/3$. Let J_n denote the $n \times n$ all-ones matrix and I_n the identity matrix. The

Gram matrix is

$$\Gamma_{A_n} = G \cdot G^T = J_n + I_n = \begin{pmatrix} 2 & 1 & 1 & ... & 1 \\ 1 & 2 & 1 & ... & 1 \\ 1 & 1 & 2 & ... & 1 \\ . & . & . & ... & . \\ 1 & 1 & 1 & ... & 2 \end{pmatrix}. \tag{7.12}$$

Theorem 7.1. *A lattice basis of A_n defined by the Gram matrix* (7.12) *is Voronoi-reduced.*

Consider the following Gram matrix of E_8.

$$\Gamma_{E_8} = \begin{pmatrix} 4 & 2 & 0 & 2 & 2 & 2 & 2 & 2 \\ 2 & 4 & 2 & 0 & 2 & 2 & 2 & 2 \\ 0 & 2 & 4 & 0 & 2 & 2 & 0 & 0 \\ 2 & 0 & 0 & 4 & 2 & 2 & 0 & 0 \\ 2 & 2 & 2 & 2 & 4 & 2 & 2 & 0 \\ 2 & 2 & 2 & 2 & 2 & 4 & 0 & 2 \\ 2 & 2 & 0 & 0 & 2 & 0 & 4 & 0 \\ 2 & 2 & 0 & 0 & 0 & 2 & 0 & 4 \end{pmatrix}. \tag{7.13}$$

Theorem 7.2. *A lattice basis of E_8 defined by the Gram matrix* (7.13) *is Voronoi-reduced with respect to $\mathring{\mathcal{P}}(\mathcal{B})$.*

Theorem 7.3. *There exists no Voronoi-reduced basis for Λ_{24}.*

Unfortunately, for most lattices such theorems can not be proved. However, quasi-Voronoi-reduced bases can sometimes be obtained. For instance, the following Gram matrix corresponds to a quasi-Voronoi-reduced basis of E_6:

$$\Gamma_{E_6} = \begin{pmatrix} 3 & \frac{3}{2} & 0 & 0 & \frac{3}{2} & \frac{3}{2} \\ \frac{3}{2} & 3 & 0 & 0 & \frac{3}{2} & \frac{3}{2} \\ 0 & 0 & 3 & \frac{3}{2} & \frac{3}{2} & \frac{3}{2} \\ 0 & 0 & \frac{3}{2} & 3 & \frac{3}{2} & \frac{3}{2} \\ \frac{3}{2} & \frac{3}{2} & \frac{3}{2} & \frac{3}{2} & 3 & \frac{3}{2} \\ \frac{3}{2} & \frac{3}{2} & \frac{3}{2} & \frac{3}{2} & \frac{3}{2} & 3 \end{pmatrix}, \tag{7.14}$$

with $\frac{d^2_{OC}}{\rho^2} = 1.60$ (2dB of gain) and $\frac{\text{vol}(O)}{\det(\Lambda)} = 2.47 \times 10^{-3}$. The ratio of $P_e(ub)$ by the second term of the right-hand side of (7.10) is about 10^{-4} at $\Delta = 1$ then vanishes further for increasing Δ.

Obviously, the quasi-VR property is good enough to allow the application of a decoder working with $\mathcal{C}(\mathcal{B})$.If an optimal decoder is required, e.g. in specific applications such as lattice shaping and cryptography, the user should let the decoder manage extra points outside $\mathcal{C}(\mathcal{B})$. For example, the disconnected region O (see (7.9)) for E_6 defined by Γ_{E_6} includes extra points where $z_i \in \{-1, 0, 1, +2\}$ instead of $\{0, 1\}$ as for $\mathcal{C}(\mathcal{B})$.

Unstructured MIMO lattices

We investigate the VR properties of typical random MIMO lattices where the lattice is generated by a matrix G with i.i.d. $\mathcal{N}(0,1)$ components. The basis obtained via this random process is in general of poor quality. As a mentionned in the previous subsection, the standard and cheap process to obtained a basis of better quality is to apply the LLL algorithm. As a result, we are interested in the following question: Is a LLL-reduced random MIMO lattice quasi-Voronoi-reduced?

In the previous subsection, we highlighted that two quantities enable to characterize the loss in the error probability on the Gaussian channel ($P_e(O)$, see Equation (7.11)) due to non-VR parts of $\mathcal{P}(\mathcal{B})$: Vol(O) and $d_{OC}(\mathcal{B})$. Unfortunately, for a given basis, these quantities are in general difficult to compute because it requires sampling in a n-dimensional space. Nevertheless, one can directly estimate the term $P_e(O)$, without computing these two quantities, via Monte Carlo simulations: Noisy points y are generated, aliased in $\mathcal{P}(\mathcal{B})$ and decoded with an optimal algorithm. If the decoded point is not a corner of $\mathcal{P}(\mathcal{B})$, i.e. $\hat{z} \notin \{0,1\}^n$, we declare an error. Comparing the resulting performance with the one obtained with the optimal algorithm enables to assess the term $P_e(O)$ and observe the loss in the error probability on the Gaussian channel caused by the non-VR parts of $\mathcal{P}(\mathcal{B})$.

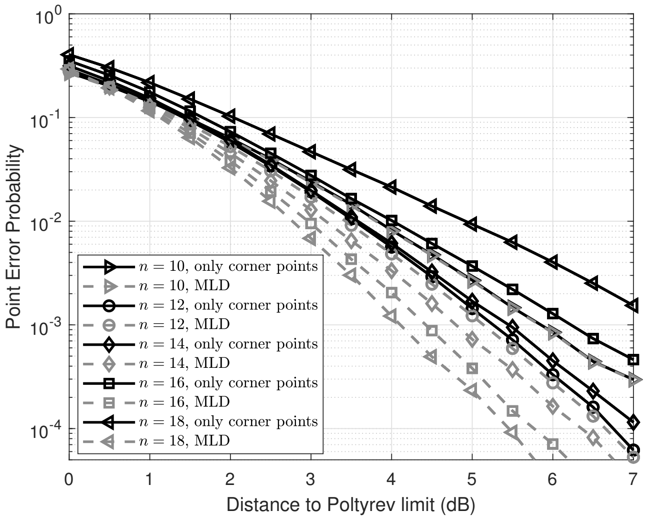


Figure 7.4: Assessment of the performance loss, on the Gaussian channel, due to non-VR parts of $\mathcal{P}(\mathcal{B})$ for LLL-reduced random MIMO lattice; the random generator matrix G is generated with i.i.d. $\mathcal{N}(0,1)$ components and the LLL-reduced basis is used for decoding. For each point, we average the performance obtained with 1000 random generator matrices G.

The simulation results are depicted on Figure 7.4. Up to dimension $n = 12$, considering only the corners of $\mathcal{P}(\mathcal{B})$ yields no significant loss in performance. We can conclude that, on average for the considered model, a LLL-reduced basis for $n \leq 12$ is quasi-VR. However, for larger dimensions, the loss increases and becomes significant.

In summary, the VR approximation can be made for a LLL-reduced random MIMO lattice up to dimension 12 (e.g. in the scope of the MIMO channel, up to 6 antennas).

7.4 Finding the closest corner of $\mathcal{P}(\mathcal{B})$ for decoding

Thanks to the previous section, we know that the CVP in $\mathcal{P}(\mathcal{B})$, with a VR basis, can be optimaly solved with an algorithm having only binary outputs. In this section, we introduce the notion of decision boundary for decoding. It enables to find, componentwise, the closest corner of $\mathcal{P}(\mathcal{B})$ to any point $y \in \mathcal{P}(\mathcal{B})$. This process exactly solves the CVP if the basis is VR. This descrimination can be implemented with the hyperplane logical decoder (HLD). It can also be applied to lattices admitting only a quasi-VR basis to yield quasi-MLD performance in presence of additive white Gaussian noise. The complexity of the HLD depends on the number of affine pieces in the decision boundary, which is exponential in the dimension. More generally, we shall see that this exponential number of pieces induces shallow neural networks of exponential size.

7.4.1 The decision boundary

We show how to decode one component of the vector $\hat{z}$. Without loss of generality, if not specified, the integer coordinate to be decoded for the rest of this section is $\hat{z}_1$. The process presented in this section should be repeated for each z_i, $1 \leq i \leq n$ to recover all the components of $\hat{z}$. Given a lattice with a VR basis, exactly half of the corners of $\mathcal{P}(\mathcal{B})$ are obtained with $z_1 = 1$ and the other half with $z_1 = 0$. Therefore, one can partition $\mathcal{P}(\mathcal{B})$ in two regions, where each region is:

$$\mathcal{R}_{\mathcal{C}^i_{\mathcal{P}(\mathcal{B})}} = \bigcup_{x \in \mathcal{C}^i_{\mathcal{P}(\mathcal{B})}} \mathcal{V}(x) \cap \mathcal{P}(\mathcal{B}), \tag{7.15}$$

with $i = 1$ or 0. The intersections between $\mathcal{R}_{\mathcal{C}^1_{\mathcal{P}(\mathcal{B})}}$ and $\mathcal{R}_{\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}}$ define a boundary. This boundary splitting $\mathcal{P}(\mathcal{B})$ into two regions $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ and $\mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$, is the union of some of the Voronoi facets of the corners of $\mathcal{P}(\mathcal{B})$.

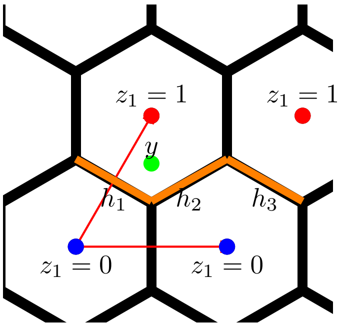


Figure 7.5: The hexagonal lattice A_2 with a VR basis. The two upper corners of $\mathcal{P}(\mathcal{B})$ (in red) are obtained with $z_1 = 1$ and the two other ones with $z_1 = 0$ (in blue). The decision boundary is illustrated in orange. Given the green point y, $\hat{z}_1$ should be decoded to 1 because y is above the decision boundary.

Each facet can be defined by an affine function over a compact subset of $\mathbb{R}^{n-1}$, and the boundary is locally described by one of these functions.

Obviously, the position of a point to decode with respect to this boundary determines whether $\hat{z}_1$ should be decoded to 1 or 0. For this reason, we call this boundary the decision boundary. Moreover, the hyperplanes involved in the decision boundary are called boundary hyperplanes. An instance of a decision boundary is illustrated on Figure 7.5.

7.4.2 Decoding via a Boolean equation

Let $\mathcal{B}$ be VR basis. The CVP in $\mathcal{P}(\mathcal{B})$ is solved componentwise, by comparing the position of y with the Voronoi facets partitioning $\mathcal{P}(\mathcal{B})$. This can be expressed in the form of a Boolean equation, where the binary (Boolean) variables are the positions with respect to the facets (on one side or another). Therefore, the one should compute the position of y relative to the decision boundary via a Boolean equation to guess whether $\hat{z}_1 = 0$ or $\hat{z}_1 = 1$.

Consider the orthogonal vectors to the hyperplanes containing the Voronoi facet of a point $x \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ and a point from $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$. These vectors are denoted by v_j as in (7.1). A Boolean variable $u_j(y)$ is obtained as:

$$u_j(y) = \text{Heav}(y \cdot v_j - p_j) \in \{0, 1\}, \tag{7.16}$$

where Heav$(\cdot)$ stands for the Heaviside function. Since $\mathcal{V}(x) = \mathcal{V}(0) + x$, orthogonal vectors v_j to all facets partitioning $\mathcal{P}(\mathcal{B})$ are determined from the facets of $\mathcal{V}(0)$.

Example 7.1. *Let $\hat{z} = (\hat{z}_1, \hat{z}_2)$ and $y \in \mathcal{P}(\mathcal{B})$ the point to be decoded. Given the red basis on Figure 7.5, the first component $\hat{z}_1$ is 1 (true) if y is above hyperplanes h_1 and h_2 simultaneously or above h_3. Let $u_1(y), u_2(y)$, and $u_3(y)$ be Boolean variables, the state of which depends on the location of y with respect to the hyperplanes h_1, h_2, and h_3, respectively. We get the Boolean equation $\hat{z}_1 = u_1(y) \cdot u_2(y) + u_3(y)$, where $+$ is a logical OR and $\cdot$ stands for a logical AND.*

Given a lattice $\Lambda \subset \mathbb{R}^n$ of rank n, Algorithm 7.1 enables to find the Boolean equation of a coordinate $\hat{z}_i$. It also finds the equation of each hyperplane needed to get the value of the Boolean variables involved in the equation. This algorithm can be seen as a “training” step to “learn” the structure of the lattice. It is a brute-force search that may quickly become too complex as the dimension increases. However, we shall see in Section 7.4.4 and 7.4.5 that these Boolean equations can be analysed without this algorithm, via a study of the basis. Note that the decoding complexity does not depend on the complexity of this search algorithm.

7.4.3 The HLD

The HLD is a brute-force algorithm to compute the Boolean equation provided by Algorithm 7.1. The HLD can be executed via the three steps summarized in Algorithm 7.2.

Algorithm 7.1 Brute-force search to find the Boolean equation of a coordinate $\hat{z}_i$ for a lattice Λ

```
Select the $2^{n-1}$ corners of $\mathcal{P}(\mathcal{B})$ where $z_i = 1$ and all relevant Voronoi vectors of $\Lambda$.
for each of the $2^{n-1}$ corners where $z_i = 1$ do
  for each relevant Voronoi vector of $\Lambda$ do
    Move in the direction of the selected relevant Voronoi vector by half its norm + $\epsilon$ ($\epsilon$ being a small number).
    if The resulting point is outside $\mathcal{P}(\mathcal{B})$. then
      Do nothing. //There is no decision boundary hyperplane in this direction.
    else
      Find the closest lattice point $x' = z'G$ (e.g. by sphere decoding [AEVZ02]).
      if $z'_i = 1$ then
        Do nothing. //There is no decision boundary hyperplane in this direction.
      else
        Store the decision boundary orthogonal to this direction. //$z'_i = 0$
      end if
    end if
  end for
  for each decision boundary hyperplane found (at this corner) do
    Associate and store a Boolean variable to this hyperplane (corresponding to the position of the point to be decoded with respect to the hyperplane).
  end for
  The Boolean equation of $\hat{z}_i$ contains a Boolean AND of these variables.
end for
The equation is the Boolean OR of the $2^{n-1}$ AND coming from all corners.
```

Algorithm 7.2 HLD

```
Compute the inner product of $y$ with the vectors orthogonal to the decision boundary hyperplanes.
Apply the Heaviside function on the resulting quantities to get its relative positions under the form of Boolean variables.
Compute the logical equations associated to each coordinate.
```

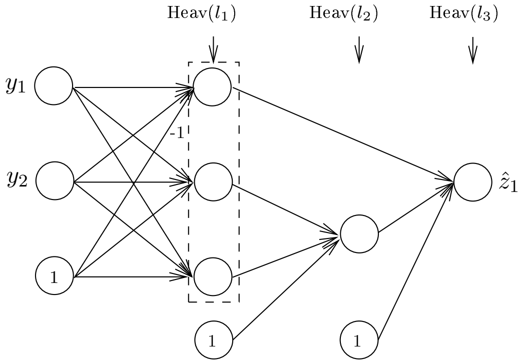


Figure 7.6: Neural network performing HLD decoding on the first symbol $\hat{z}_1$ of a point in $\mathcal{P}(\mathcal{B})$ for the lattice A_2 (see Example 7.1). Heav($\cdot$) stands for Heaviside($\cdot$). The first part of the network computes the position of y with respect to the boundary hyperplanes to get the variables $u_j(y)$. The second part (two last layers) compute the Boolean ANDs and Boolean ORs of the decoding Boolean equation.

Implementation of the HLD

Since Steps 1-2 are simply linear combinations followed by activation functions, these operations can be written as:

$$l_1 = \sigma(y \cdot G_1 + b_1), \tag{7.17}$$

where σ is the Heaviside function, G_1 a matrix having the vectors v_j as columns, and b_1 a vector of biases containing the p_j. Equation (7.17) describes the operation performed by a layer of a neural network (see (7.4)) . The layer l_1 is a vector containing the Boolean variables $u_j(y)$.

Let l_{i-1} be a vector of Boolean variables. It is well known that both Boolean AND and Boolean OR can be expressed as:

$$l_i = \sigma(l_{i-1} \cdot G_i + b_i),$$

where G_i a matrix composed of 0 and 1, and b_i a vector of biases. Therefore, the mathematical expression of the HLD is:

$$z_1 = \sigma(\sigma(\sigma(y \cdot G_1 + b_1) \cdot G_2 + b_2) \cdot G_3 + b_3). \tag{7.18}$$

Equation (7.18) is exactly the definition of a feed-forward neural network (see (7.4)) with three layers. Figure 7.6 illustrates the topology of the neural network obtained when applying the HLD to the lattice A_2.

7.4.4 The decision boundary as a piecewise affine function

In order to better understand the decision boundary, we characterize it as a function rather than a Boolean equation. We shall see in the sequel that it is sometimes possible to efficiently compute this function and thus reduce the decoding complexity.

Let $\{e_i\}_{i=1}^n$ be the canonical orthonormal basis of the vector space $\mathbb{R}^n$. For $y \in \mathbb{R}^n$, the i-th coordinate is $y_i = y \cdot e_i$. Denote $\tilde{y} = (y_2, \ldots, y_n) \in \mathbb{R}^{n-1}$ and let $\mathcal{H} = \{h_j\}$ be the set of affine functions involved in the decision boundary. The affine boundary function $h_j : \mathbb{R}^{n-1} \to \mathbb{R}$ is

$$h_j(\tilde{y}) = y_1 = \left(p_j - \sum_{k \neq 1} y_k v_j^k\right) / v_j^1, \tag{7.19}$$

where v_j^k is the kth component of vector v_j. For the sake of simplicity, in the sequel h_j shall denote the function defined in (7.19) or its associated hyperplane depending on the context.

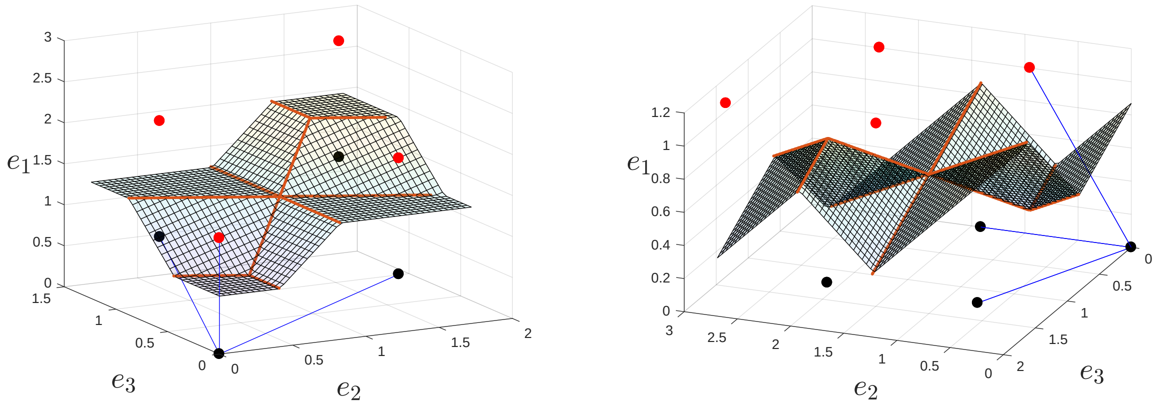
Theorem 7.4. *Consider a lattice defined by a VR basis $\mathcal{B} = \{g_i\}_{i=1}^n$. Let $\mathcal{H} = \{h_j\}$ be the set of affine functions involved in the decision boundary. Assume that $g_1^1 > 0$. Suppose also that $x_1 > \lambda_1$ (in the basis $\{e_i\}_{i=1}^n$), $\forall x \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ and $\forall \lambda \in \mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$. Then, the decision boundary is given by a CPWL function $f : \mathbb{R}^{n-1} \to \mathbb{R}$, expressed as*

$$f(\tilde{y}) = \wedge_{m=1}^M \{\vee_{k=1}^{l_m} h_{m,k}(\tilde{y})\}, \tag{7.20}$$

where $h_{m,k} \in \mathcal{H}$, $1 \leq l_m < \tau_f$, and $1 \leq M \leq 2^{n-1}$.

The proof is provided in Appendix 7.6.2. In the previous theorem, the orientation of the axes relative to $\mathcal{B}$ does not require $\{g_i\}_{i=2}^{n}$ to be orthogonal to e_1. This is however the case for the next corollary, which involves a specific rotation satisfying the assumption of the previous theorem. Indeed, with the following orientation, any point in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ is in the hyperplane $\{y \in \mathbb{R}^n : \ y \cdot e_1 = 0\}$ and has its first coordinate equal to 0, and $g_1^1 > 0$ (if it is negative, simply multiply the basis vectors by -1).

Corollary 7.1. *Consider a lattice defined by a VR basis $\mathcal{B} = \{g_i\}_{i=1}^{n}$. Suppose that the $n-1$ points $\mathcal{B} \backslash \{g_1\}$ belong to the hyperplane $\{y \in \mathbb{R}^n : \ y \cdot e_1 = 0\}$. Then, the decision boundary is given by a CPWL function as in (7.20).*



(a) The orientation of the basis satisfies the assumptions of Theorem 7.4 but not the ones of Corollary 7.1.

(b) The orientation of the basis satisfies the assumptions of Theorem 7.4 and Corollary 7.1.

Figure 7.7: CPWL decision boundary function for A_3. The basis vectors are represented by the blue lines. The corner points in $\mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ are in red and the corner points in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ in black.

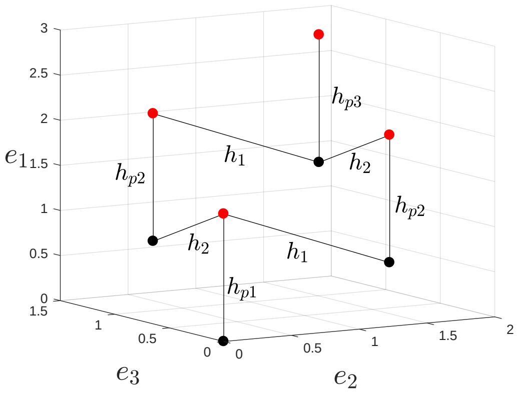


Figure 7.8: "Neighbor figure" of $\mathcal{C}_{\mathcal{P}(\mathcal{B})}$ for A_3. Each edge connects a point $x \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ to an element of $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$. Consequently, each edge is orthogonal to a local affine function of the decision boundary function f. The edges are labeled with the names of the corresponding affine functions. Theorem 7.5 and its proof show that each set $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ generates a convex part of the decision boundary function f with $|\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}|$ pieces. E.g. on the figure there are one set with $|\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}| = 3$, two with $|\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}| = 2$, and one with $|\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}| = 1$, thus f has 8 pieces.

Example 7.2. *Consider the lattice A_3 defined by the Gram matrix* (7.12). *To better illustrate the symmetries we rotate the basis*[1] *to have g_1 collinear with e_1. Theorem 7.4 ensures that the decision*

[1] Note that the orientation of the basis does not satisfy the assumptions of Corollary 7.1.

boundary is a function. The function is illustrated on Figure 7.7a. On Figure 7.8 each edge is orthogonal to a local affine function of f. The edges are labeled with the name of the corresponding affine function. The $[\cdot]$ groups all the set of convex pieces of f that includes the same h_{p_j}.

The equation of the function is (we omit the $\tilde{y}$ in the formula to lighten the notations):

$$f = \Big[h_{p_1} \vee h_1 \vee h_2\Big] \wedge \Big[\,(h_{p_2} \vee h_1) \wedge (h_{p_2} \vee h_2)\,\Big] \wedge \Big[h_{p_3}\Big],$$

where h_{p_1}, h_{p_2} and h_{p_3} are hyperplanes orthogonal to g_1 (the p index stands for plateau). Functions for higher dimensions (i.e. $A_n, n \geq 3$) are available in Appendix 7.6.3. On Figure 7.7b, the same function is illustrated with the rotation of Corollary 7.1.

The notion of decision boundary function can be generalized to non-VR basis under the assumptions of the following definition. A surface in $\mathbb{R}^n$ defined by a function g of $n-1$ arguments is written as $\mathrm{Surf}(g) = \{y = (g(\tilde{y}), \tilde{y}) \in \mathbb{R}^n, \tilde{y} \in \mathbb{R}^{n-1}\}$.

Definition 7.3. *Let $\mathcal{B}$ be a is quasi-Voronoi-reduced basis of Λ. Assume that $\mathcal{B}$ and $\{e_i\}_{i=1}^n$ have the same orientation as in Corollary 7.1. The basis is called semi-Voronoi-reduced (SVR) if there exists at least two points $x_1, x_2 \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ such that $Surf(\vee_{k=1}^{\ell_1} g_{1,k}) \bigcap Surf(\vee_{k=1}^{\ell_2} g_{2,k}) \neq \varnothing$, where $\ell_1, \ell_2 \geq 1$, $g_{1,k}$ are the facets between x_1 and all points in $\mathcal{T}_f(x_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$, and $g_{2,k}$ are the facets between x_2 and all points in $\mathcal{T}_f(x_2) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$.*

The above definition of a SVR basis imposes that the boundaries around two points of $\mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$, defined by the two convex functions $\vee_{k=1}^{\ell_m} h_{m,k}$, $m = 1, 2$, have a non-empty intersection. Consequently, the min operator $\wedge$ leads to a boundary function as in (7.20).

Corollary 7.2. *$\mathcal{P}(\mathcal{B})$ for a SVR basis $\mathcal{B}$ admits a decision boundary defined by a CPWL function as in (7.20).*

From now on, the default orientation of the basis with respect to the canonical axes of $\mathbb{R}^n$ is assumed to be the one of Corollary 7.1. We call f the decision boundary function. The domain of f (its input space) is $\mathcal{D}(\mathcal{B}) \subset \mathbb{R}^{n-1}$. The domain $\mathcal{D}(\mathcal{B})$ is the projection of $\mathcal{P}(\mathcal{B})$ on the hyperplane $\{e_i\}_{i=2}^n$. It is a bounded polyhedron that can be partitioned into convex regions which we call linear regions. For any $\tilde{y}$ in one of these regions, f is described by a unique local affine function h_j. The number of those regions is equal to the number of affine pieces of f.

7.4.5 Complexity analysis: the number of affine pieces of the decision boundary

An efficient neural lattice decoder should have a reasonable size, i.e. a resonable number of neurons. Obviously, the size of the neural network implementing the HLD (such as the one of Figure 7.6) depends on the number of affine pieces in the decision boundary function. It is thus of high interest to characterize the number of pieces in the decision boundary as a function of the dimension. Unfortunately, it is not possible to treat all lattices in a unique framework. Therefore, we investigate this aspect for some well-known lattices.

The lattice A_n

We count the number of affine pieces of the decision boundary function f obtained for z_1 with the basis defined by the Gram matrix (7.12).

Theorem 7.5. *Consider an A_n-lattice basis defined by the Gram matrix (7.12). Let o_i denote the number of sets $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$, $x \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$, where $|\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}| = i$. The decision boundary function f has a number of affine pieces equal to*

$$\sum_{i=1}^{n} i \cdot o_i, \tag{7.21}$$

with $o_i = \binom{n-1}{n-i}$.

Proof. For any given point $x \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$, each element in the set $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ generates a Voronoi facet of the Voronoi region of x. Since any Voronoi region is convex, the $|\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}| = i$ facets are convex.

Consequently, the set $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ generates a convex part of the decision boundary function with i pieces.

We now count the number of sets $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ with cardinality i. It is obvious that $\forall x \in \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$: $x+g_1 \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$. We walk in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ and for each of the 2^{n-1} points $x \in \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ we investigate the cardinality of the set $\mathcal{T}_f(x+ \ g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$. This is achieved via the following property of the basis.

$$\begin{aligned}&\forall x \in \ \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}, \ x' \in A_n \backslash \{g_j, 0\}, \ 2 \leq j \leq n : \\ &x+g_j \in \mathcal{T}_f(x+g_1), \ x+x' \notin \mathcal{T}_f(x+g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}.\end{aligned} \tag{7.22}$$

Starting from the lattice point 0, the set $\mathcal{T}_f(0+g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ is composed of 0 and the $n-1$ other basis vectors. Then, for all g_{j_1}, $2 \leq j_1 \leq n$, the sets $\mathcal{T}_f(g_{j_1}+g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ are obtained by adding any of the $n-2$ remaining basis vectors to g_{j_1}. Indeed, if we add g_{j_1} to g_{j_1}, the resulting point is outside of $\mathcal{P}(\mathcal{B})$. Hence, the cardinality of these sets is $n-1$ and there are $\binom{n-1}{1}$ ways to choose g_{j_1}: any basis vectors except g_1. Similarly, for $g_{j_1}+g_{j_2}$, $j_1 \neq j_2$, the cardinality of the sets $\mathcal{T}_f(g_{j_1}+g_{j_2}+g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ is $n-2$ and there are $\binom{n-1}{2}$ ways to choose $g_{j_1}+g_{j_2}$. More generally, there are $\binom{n-1}{k}$ sets $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ of cardinality $n-k$.

Summing over $k = n-i = 0 \ldots n-1$ gives the announced result. □

Theorem 7.5 implies that the HLD, applied on A_n, induces a neural network (having the form given by (7.18)) of exponential size. Indeed, remember that the first layer of the neural network implementing the HLD performs projections on the orthogonal vectors to each affine pieces.

Neverthless, one can wonder whether a neural network with a different architecture can compute the decision boundary more efficiently. We first address another category of shallow neural networks: ReLU neural networks with two layers. Deep neural networks shall be discussed later in the chaper. Note that in this case we do not consider a single function computed by the neural network, like the HLD, but any function that can be computed by this class of neural network.

Theorem 7.6. *A ReLU neural network with two layers needs at least*

$$\sum_{i=2}^{n} (i-1) \times \binom{n-1}{n-i} \tag{7.23}$$

neurons for optimal decoding of the lattice A_n.

The proof is provided in Appendix 7.6.4. Consequently, this class of neural networks is not efficient. However, we shall see in the sequel that deep neural networks are better suited.

Other dense lattices

Similar proof techniques can be used to compute the number of pieces obtain with some bases of other dense lattices such as $D_n, n \geq 2$, and E_n, $6 \leq n \leq 8$.

Consider the Gram matrix of D_n given by (7.24). All basis vectors have the same length but we have either $\pi/3$ or $\pi/2$ angles between the basis vectors. This basis is not VR but SVR. It is defined by the following Gram matrix.

$$\Gamma_{D_n} = \begin{pmatrix} 2 & 0 & 1 & ... & 1 \\ 0 & 2 & 1 & ... & 1 \\ 1 & 1 & 2 & ... & 1 \\ . & . & . & ... & . \\ 1 & 1 & 1 & ... & 2 \end{pmatrix}. \tag{7.24}$$

Theorem 7.7. *Consider a D_n-lattice basis defined by the Gram matrix* (7.24). *Let o_i denote the number of sets $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$, $x \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$, where:*

- $|\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}| = \underbrace{[1+(n-2-i)]}_{(l_i)}$, *and*

- $|\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}| = \underbrace{\left[\underbrace{1+2(n-2-i)}_{(1)} + \underbrace{\binom{n-2-i}{2}}_{(2)}\right]}_{(ll_i)}$.

The decision boundary function f has a number of affine pieces equal to

$$\sum_{i=0}^{n-2} ((l_i) + (ll_i)) \times o_i - 1, \tag{7.25}$$

with $o_i = \binom{n-2}{i}$.

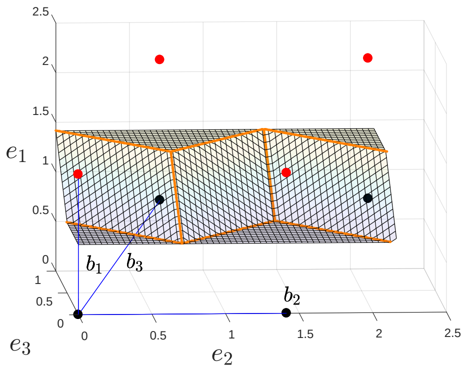


Figure 7.9: CPWL boundary function for D_3. The basis is rotated to better illustrate the symmetry: g_1 is collinear with e_1.

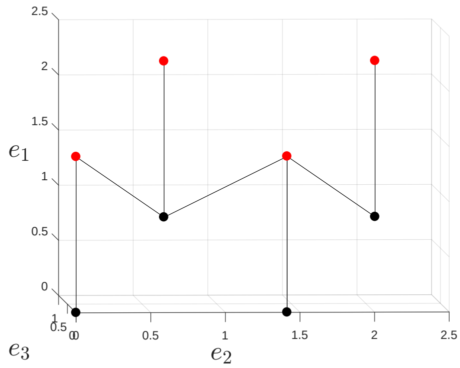


Figure 7.10: "Neighbor figure" of $\mathcal{C}_{\mathcal{P}(\mathcal{B})}$ for D_3. Each edge connects a point $x \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ to an element of $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$.

We presents the two different "neighborhood patterns" encountered with this basis of D_n (this gives (l_i) and (ll_i)). In the proof available in Appendix 7.6.5, we then count the number of simplices (i.e. (o_i)) in each of these two categories.

The decision boundary function for D_3 is illustrated on Figure 7.9. We investigate the different "neighborhood patterns" by studying Figure 7.10: I.e. we are looking for the different ways to find the neighbors of $x \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ in $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$, depending on the form of x. In the sequel, (l_i), (ll_i), and (1), (2) refer to Equation (7.25) and $\sum_j g_j$ denotes any sum of points in the set $\{0, g_j\}_{j=3}^n$, where g_2 is the basis vector orthogonal to g_1. We recall that adding g_1 to any point $x \in \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ leads to a point in $\mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$.

(l_i) This pattern is the same as the (only) one encountered for A_n with the basis given by Equation (7.12). We first consider any point in $\mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ of the form $\sum_j g_j + g_1$. Its neighbors in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ are $\sum_j g_j$ and any $\sum_j g_j + g_i$, where g_i is any basis vector having an angle of $\pi/3$ with g_1 such that $\sum_j g_j + g_i$ is not outside $\mathcal{P}(\mathcal{B})$. Hence, $|\mathcal{T}_f(\sum_{j=1}^i g_j + g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}| = 1 + n - 2 - i$. E.g. for $n = 3$, the closest neighbors of $0 + g_1$ in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ are 0 and g_3. g_2 is perpendicular to g_1 and is not a closest neighbor of g_1.

(ll_i) The second pattern is obtained with any point of the form $\sum_j g_j + g_2 + g_1$ and its neighbors in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$. $\sum_j g_j + g_2$ and any $\sum_j g_j + g_2 + g_i$, $\sum_j g_j + g_k$ are neighbors of this point in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$, where g_i, g_k are any basis vectors having an angle of $\pi/3$ with g_1 such that (respectively) $\sum_j g_j + g_2 + g_i$, $\sum_j g_j + g_k$ are not outside $\mathcal{P}(\mathcal{B})$. This terms generate the (1) in the formula. E.g. for $n = 3$, the closest neighbors of $0 + g_2 + g_1$ in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ are g_2, $g_2 + g_3$, and g_3. Moreover, for $n = 3$ one "neighborhood case" is not happening: from $n = 4$, the points $g_i + g_j \in \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$, $3 \leq i < j \leq n$, are also closest neighbors of $g_2 + g_1$. This explains the binomial coefficient (2). Hence, $|\mathcal{T}_f(\sum_{j=1}^i g_j + g_2 + g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}| = 1 + 2(n - 2 - i) + \binom{n-2-i}{2}$.

Finally, we investigate E_n, $6 \leq n \leq 8$. The basis we consider is almost identical to the basis of D_n given by (7.24), except one main difference: there are two basis vectors orthogonal to g_1 instead of one. This basis is not VR but SVR. It is defined by the following Gram matrix.

$$\Gamma_{E_n} = \begin{pmatrix} 2 & 0 & 0 & 1 & ... & 1 \\ 0 & 2 & 1 & 1 & ... & 1 \\ 0 & 1 & 2 & 1 & ... & 1 \\ 1 & 1 & 1 & 2 & ... & 1 \\ . & . & . & . & ... & . \\ 1 & 1 & 1 & 1 & ... & 2 \end{pmatrix}. \tag{7.26}$$

Theorem 7.8. *Consider an E_n-lattice basis, $6 \leq n \leq 8$, defined by the Gram matrix (7.24). The decision boundary function f has a number of affine pieces equal to*

$$\sum_{i=0}^{n-3} \Big(\underbrace{[1+(n-3-i)]}_{(l_i)} + \underbrace{2\left[1+2(n-3-i)+\binom{n-3-i}{2}\right]}_{(ll_i)} + \underbrace{\left[\underbrace{1+3(n-3-i)}_{(1)} + \underbrace{3\binom{n-3-i}{2}}_{(2)} + \underbrace{\binom{n-3-i}{3}}_{(3)}\right]}_{(lll_i)} \Big) \underbrace{\binom{n-3}{n-i}}_{(o_i)} - 3. \tag{7.27}$$

We first highlight the similarities with the function of D_n defined by (7.24). As with D_n, we have case (l_i). Case (ll_i) of D_n is also present but obtained twice because of the two orthogonal vectors. The terms $n-2-i$ in (l_i) and (ll_i) of Equation (7.25) are replaced by $n-3-i$ also because of the additional orthogonal vector.

Then, there is a new pattern (lll_i): Any point of the form $\sum_j g_j + g_3 + g_2 + g_1$ and its neighbors in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$, where $\sum_j g_j$ represents any sum of points in the set $\{0, g_j\}_{j=4}^n$. For instance, the closest neighbors in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ of $g_3 + g_2 + g_1 \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ are the following points, which we can sort in three groups as on Equation (7.27): (1) $g_2 + g_j$, $g_3 + g_j$, $g_2 + g_3 + g_j$, (2) $g_j + g_k$, $g_2 + g_j + g_k$, $g_3 + g_j + g_k$, (3) $g_j + g_i + g_k$, $4 \leq i < j < k \leq n$. The formal proof is available in Appendix 7.6.7.

7.5 Complexity reduction

In this section, we first show that a technique called the folding strategy enables to compute the decision boundary function at a reduced (polynomial) complexity. The folding strategy can be seen as a preprocessing step to simplify the function to compute. The implementation of this technique involves a deep neural network. As a result, the exponential complexity of the HLD is reduced to a polynomial complexity by moving from a shallow neural network to a deep neural network. The folding strategy and its implementation is first presented for the lattice A_n. We then show that folding is also possible for D_n and E_n.

In the second part of the section, we argue that, on the Gaussian channel, the problem to be solved by neural networks is easier for MIMO lattices than for dense lattices: In low to moderate dimensions, many pieces of the decision boundary function can be neglected for quasi-optimal decoding. Assuming that usual training techniques naturally neglect the useless pieces, this explains why neural networks of reasonable size are more efficient with MIMO lattices than with dense lattices.

7.5.1 Folding strategy

The algorithm

Obviously, at a location $\tilde{y}$, we do not want to compute all affine pieces in (7.20), whose number is for instance given by (7.21) for A_n. To reduce the complexity of this evaluation, the idea is to exploit the symmetries of f by "folding" the function and mapping distinct regions of the input domain to the same location. If folding is applied sequentially, i.e. fold a region that has already been folded, the gain becomes exponential. The notion of folding the input space in the context of neural networks was introduced in [SM13] and [MPCB14]. We first present the folding procedure for the lattice A_n and explain how this translate into a deep neural networks. We then show that this strategy can also be applied to the other dense lattices studied in Section 7.4.5.

Folding of A_n

The input space $\mathcal{D}(\mathcal{B})$ is defined as in Section 7.4.4. Given the basis orientation as in Corollary 7.1, the projection of g_j on $\mathcal{D}(\mathcal{B})$ is g_j itself, for $j \geq 2$. We also denote the bisector hyperplane between two vectors g_j, g_k by $BH(g_j, g_k)$ and its normal vector is taken to be $v_{j,k} = g_j - g_k$. Let $\tilde{y} \in \mathcal{D}(\mathcal{B})$ and let $\tilde{v}_{j,k}$ be a vector with the $n-1$ last coordinates of $v_{j,k}$. First, we define the function $F_{j,k}$, where $2 \leq j < k \leq n$, which performs the following reflection. Compute $\tilde{y} \cdot \tilde{v}_{j,k}$. If the scalar product is non-positive, replace $\tilde{y}$ by its mirror image with respect to $BH(g_j, g_k)$. Since $2 \leq j < k \leq n$, there are $\binom{n-1}{2} = (n-1)(n-2)/2$ functions $F_{j,k}$. The function F_{A_n} performes sequentially these $O(n^2)$ reflections:

$$F_{A_n} = F_{2,2} \circ F_{2,3} \circ F_{3,3} \circ ... \circ F_{n,n}, \tag{7.28}$$

and

$$F_{A_n} : \mathcal{D}(\mathcal{B}) \to \mathcal{D}(\mathcal{B})'. \tag{7.29}$$

Theorem 7.9. *Let us consider the lattice A_n defined by the Gram matrix (7.12). We have (i) $\mathcal{D}(\mathcal{B})' \subset \mathcal{D}(\mathcal{B})$, (ii) for all $\tilde{y} \in \mathcal{D}(\mathcal{B})$, $f(\tilde{y}) = f(F_{A_n}(\tilde{y}))$ and (iii) f has exactly*

$$2n-1 \tag{7.30}$$

pieces on $\mathcal{D}(\mathcal{B})'$.

Equation (7.30) is to be compared with (7.21).

Example 7.3. *This example follows Example 7.2. The function f for A_3 restricted to $\mathcal{D}(\mathcal{B})'$ (i.e. the function to evaluate after folding), say $f_{\mathcal{D}(\mathcal{B})'}$, is*

$$f_{\mathcal{D}(\mathcal{B})'} = \Big[h_{p1} \vee h_1\Big] \wedge \Big[h_{p2} \vee h_2\Big] \wedge \Big[h_{p3}\Big]. \tag{7.31}$$

The general expression of $f^n_{\mathcal{D}(\mathcal{B})'}$ for any dimension n is

$$f^n_{\mathcal{D}(\mathcal{B})'} = \Big[h_{p_1} \vee h_1\Big] \wedge \Big[h_{p_2} \vee h_2\Big] \wedge ... \wedge \Big[h_{p_{n-1}} \vee h_{n-1}\Big] \wedge \Big[h_{p_n}\Big].$$

Proof. To prove (i) we use the fact that $BH(g_j, g_k)$, $2 \leq j < k \leq n$, is orthogonal to $\mathcal{D}(\mathcal{B})$, then the image of $\tilde{y}$ via the folding F is in $\mathcal{D}(\mathcal{B})$.

(ii) is the direct result of the symmetries in the A_n basis where the n vectors have the same length and the angle between any two basis vectors is $\pi/3$. A reflection with respect $BH(g_j, g_k)$ switches g_j and g_k in the hyperplane containing $\mathcal{D}(\mathcal{B})$ and orthogonal to e_1. Switching g_j and g_k does not change the decision boundary because of the basis symmetry, hence f is unchanged.

Now, for (iii), how many pieces are left after all reflections? Similarly to the proof of Theorem 7.5, we walk in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ and for a given point $x \in \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ we count the number of elements of $\mathcal{T}_f(x+b_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ (via Equation (7.22)) that are on the proper side of all bisector hyperplanes. Starting with $\mathcal{T}_f(x+b_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$, only 0 and g_2 are on the proper side: any other point g_j, $j \geq 3$, is on the other side of the the bisector hyperplanes $BH(g_2, g_j)$. Hence, the lattice point g_1, which had n neighbors in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ before folding, only has 2 now. f has only two pieces around g_1 instead of n. Then, from g_2 one can add g_3 but no other for the same reason. The point $g_2 + g_1$ has only 2 neighbors in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ on the proper side. The pattern replicates until the last corner reaching $g_1 + g_2 + \ldots + g_n$ which has only one neighbor. So we get $2(n-1)+1$ pieces. □

From folding to a deep ReLU neural network

For sake of simplicity and without loss of generality, in addition to the standard ReLU activation function $\text{ReLU}(a) = \max(0, a)$, we also allow the function $\max(0, -a)$ and the identity as activation functions in the neural network.

To implement a reflection $F_{j,k}$, one can use the following strategy.

- Step 1: rotate the axes to have the ith axis e_i perpendicular to the reflection hyperplane and shift the point (i.e. the ith coordinate) to have the reflection hyperplane at the origin.
- Step 2: take the absolute value of the ith coordinate.
- Step 3: do the inverse operation of step 1.

Now consider the ReLU neural network[2] illustrated in Figure 7.11. The edges between the input layer and the hidden layer represent the rotation matrix, where the ith column is repeated twice, and p is a bias applied on the ith coordinate. Within the dashed square, the absolute value of the ith coordinate is computed and shifted by $-p$. Finally, the edges between the hidden layer and the output layer represent the inverse rotation matrix. This ReLU neural network computes a reflection $F_{j,k}$. We call it a reflection block. Note tha the width of a reflexion block is $O(n)$.

The function F_{A_n} can be implemented by a simple concatenation of reflection blocks. This leads to a very deep and narrow neural network of depth $\mathcal{O}(n^2)$ (the number of functions $F_{j,k}$) and width $\mathcal{O}(n)$ (the width of a reflection block).

[2] This neural network uses both ReLU and linear activation functions. It can still be considered as a ReLU neural network as a linear activation function can be implemented with ReLU neurons.

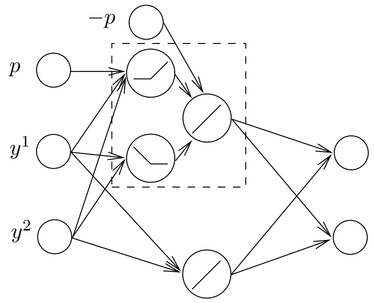


Figure 7.11: Reflection ReLU neural network (called reflection block). The edges between the input neurons and the dashed square performes the rotation (Step 1). The activation functions in the dashed square, $\max(0, a)$, $\max(0, -a)$, and a, implement the absolute value opearation (Step 2). The rest of the network does the inverse rotation (Step 3).

Regarding the $2n-1$ remaining pieces after folding, we have two options (in both cases, the number of operations involved is negligible compared to the previous folding operations). To directly discriminate the point with respect to f, we implement the HLD on these remaining pieces with two additional hidden layers (as in Figure 7.6): project y_{folded} on the $2n-1$ hyperplanes (see Theorem 7.9), with one layer of width $2n+1$, and compute the associated Boolean equation with an additional hidden layer. If needed, we can evaluate $f(\tilde{y})$ via $\mathcal{O}(\log(n))$ additional hidden layers. First, compute the $n-1$ 2-$\vee$ via two layers of size $\mathcal{O}(n)$ containing several "max ReLU neural networks" (see e.g. Figure 3 in [ABMM18]). Then, compute the n-$\wedge$ via $\mathcal{O}(\log(n))$ layers.

Consequently, f can be computed by a ReLU network of depth $\mathcal{O}(n^2)$ and width $\mathcal{O}(n)$.

Folding of other dense lattices

We now present the folding procedure for other lattices.

First, we consider D_n defined by the Gram matrix (7.24). F_{D_n} is defined as F_{A_n} except that we keep only the $F_{j,k}$ for $j,k \geq 3$. Moreover, the g_i are now the basis vectors of D_n instead of A_n, where g_2 is the basis vector orthogonal to g_1.

There are $\binom{n-2}{2} = (n-2)(n-3)/2$ functions $F_{j,k}$ and the function F_{D_n} performs sequentially the $O(n^2)$ reflections.

Theorem 7.10. *Let us consider the lattice D_n defined by the Gram matrix* (7.24). *We have (i) for all $\tilde{y} \in \mathcal{D}(\mathcal{B})$, $f(\tilde{y}) = f(F_{D_n}(\tilde{y}))$ and (ii) f has exactly*

$$6n - 12 \tag{7.32}$$

pieces on $\mathcal{D}'(\mathcal{B})$.

Equation (7.32) is to be compared with (7.25).

Sketch of proof. To count the number of pieces of f, defined on $\mathcal{D}'(\mathcal{B})$, we need to enumerate the cases where both $x \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ and $x' \in \mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ are on the non-negative side of all reflection hyperplanes. Among the points in $\mathcal{C}_{\mathcal{P}(\mathcal{B})}$ only the points

1. $x_1 = g_3 + ... + g_{i-1} + g_i$ and $x_1 + g_1$,
2. $x_2 = g_3 + ... + g_{i-1} + g_i + g_2$ and $x_2 + g_1$,

$i \leq n$, are on the non-negative side of all reflection hyperplanes. It is then easily seen that the number of pieces of f, defined on $\mathcal{D}'(\mathcal{B})$, is given by equation (7.25) reduced as follows. The three terms $(n-2-i)$ (i.e. $2(n-2-i)$ counts for two), the term $\binom{n-2-i}{2}$, and the term $\binom{n-2}{i}$ become 1 at each step i, for all $0 \leq i \leq n-3$ (except $\binom{n-2-i}{2}$ which is equal to 0 for $i = n-3$). Hence, (7.25) becomes $(n-3) \times (2+4) + (2+3) + 1$, which gives the announced result.

Consequently, f can be computed by a ReLU network of depth $\mathcal{O}(n^2)$ and width $\mathcal{O}(n)$ (i.e. the same size as the one for A_n).

Second, we show how to fold the function for E_n. F_{E_n} is defined as F_{A_n} except that, for the functions $F_{j,k}$, $4 \leq j < k \leq n$ and $j = 2, k = 3$ instead of $2 \leq j < k \leq n$, where g_2, g_3 are the basis vectors orthogonal to g_1. There are $\binom{n-3}{2}+1 = (n-3)(n-4)/2+1$ functions $F_{j,k}$ and the function F_{E_n} performs sequentially the $O(n^2)$ reflexions.

Dimension n	10	12	14	16
Average number of points	59	109	201	361

Table 7.1: Average number of points in a sphere of squared radius $2 \cdot d^2(\Lambda)$ centered at the origin for random MIMO lattices Λ .

Theorem 7.11. *Let us consider the lattice E_n, $6 \leq n \leq 8$, defined by the Gram matrix* (7.7). *We have (i) for all $\tilde{y} \in \mathcal{D}(\mathcal{B})$, $f(\tilde{y}) \;=\; f(F_{E_n}(\tilde{y}))$ and (ii) f has exactly*

$$12n - 40 \tag{7.33}$$

pieces on $\mathcal{D}'(\mathcal{B})$.

Equation (7.33) is to be compared with (7.27). Consequently, f can be computed by a ReLU network of depth $\mathcal{O}(n^2)$ and width $\mathcal{O}(n)$.

7.5.2 Neglecting many affine pieces in the decision boundary

In the previous section, we showed that complexity reduction can be achieved for some structured lattices by exploiting their symmetries. What about unstructured lattices? We consider the problem of decoding on the Gaussian channel. The goal is to obtain quasi-MLD performance.

Empirical observations

In [CBCB18b], we performed several simulations with dense lattices (e.g. E_8) and MIMO lattices (such as the ones considered in [SDW17]), which are typically not dense in low to moderate dimensions. We tried to minimize the number of parameters in a standard fully-connected feed-forward sigmoid neural network [GBC16] while maintaining quasi-MLD performance. The training was performed with usual gradient-descent-like techniques [GBC16]. The network considered is shallow, similarly to the HLD, as it contains only three hidden layers. Let W be the number of parameters in the neural networks (i.e. the number of edges). To be competitive, W should be smaller than 2^n. For E_8 we obtained a complexity ratio $\frac{\log_2 W}{n} = 2.0$ whereas for the MIMO lattice the ratio is only $\frac{\log_2 W}{n} = 0.78$.

Similarly, we also compared the decoding complexity of MIMO lattices and dense lattices (BW_{16} in this case) in [CBCB18a], with a different network architectures (but still having the form of a feed-forward neural network). The conclusion was the same: while it is possible to get a reasonable complexity for MIMO lattices, it is much more challenging for dense lattices.

See Chapter 9 for more details on these simulations.

Explanation

We explained in the first part of the paper that all pieces of the decision boundary function are facets of Voronoi regions. As a result, the (optimal) HLD needs to consider all Voronoi relevant vectors, which is equal to $\tau_f = 2^{n+1}-2$ for random lattices. However, as stated below Equation (3.43) in the preliminaries, only the first lattice shells need to be considered for quasi-MLD performance on the Gaussian channel.

Consequently, we performed simulations to know how many Voronoi facets contribute to the quasi-MLD error probability for random MIMO lattices generated by a matrix G with random i.i.d $\mathcal{N}(0,1)$ components. We numerically generated 200000 random MIMO lattices Λ and computed the average number of lattice points in a sphere of squared radius $2 \cdot d^2(\Lambda)$ centered at the origin. The results are reported in Table 7.1. Figure 7.12 also provide the distribution for $n = 14$. For comparison, the number of points in such a sphere is 25201 for the dense Coxeter-Todd lattice in dimension 12 and 588481 for the dense Barnes-Wall lattice in dimension 16 [CS99, Chap. 4]. Note however that while the numbers shown in Table 7.1 are relatively low, the increase seems to be exponential: The number of lattice points in the sphere almost doubles when adding two dimensions. Note that estimates of these numbers could also be obtained with the Gaussian Heuristic (see Equation (3.23)). From this perspective, it is not surprising to get an exponential increase.

This means that the number of Voronoi facets significantly contributing to the error probability is much smaller for random MIMO lattices compared to dense lattices in these dimensions. As a result,

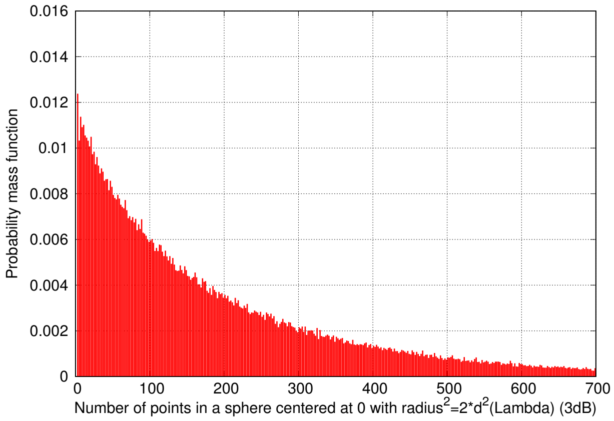


Figure 7.12: Distribution of the number of lattice points in a sphere of squared radius $2 \cdot d^2(\Lambda)$. The random lattices in dimension $n = 14$ are generated by a matrix G with random i.i.d. $\mathcal{N}(0, 1)$ componens. We numerically generated 200000 random lattices in dimensions $n = 14$ to compute the distribution.

the number of hyperplanes that should be taken into account for quasi-MLD is much smaller for random MIMO lattices. In other words, the function to compute for quasi-optimal decoding is “simpler”: A piecewise linear boundary with a relatively low amount of affine pieces can achieve quasi-MLD for random MIMO lattices.

We argue that regular learning techniques with Gaussian distributed training data, at moderate SNR, naturally selects these Voronoi facets contributing to the error probability. This explains why, for quasi-optimal decoding in low to moderate dimensions, shallow neural networks can achieve satisfactory performance at reasonable complexity, whereas it is much more challenging for dense lattices.

Alternatively, a reduced complexity decoding could also be performed via a modified HLD, where the brute-force search (Algorithm 7.1) should consider only the Voronoi facets contributing to the quasi-MLD error probability.

7.6 Appendix

7.6.1 Proofs of Section 7.3.2

Proof of Theorem 7.1

We need to show that none of $y \in \mathcal{V}(x)$, $x \in \Lambda \backslash \mathcal{C}_{\mathcal{P}(\mathcal{B})}$, crosses a facet of $\overline{\mathcal{P}}(\mathcal{B})$. In this scope, we first find the closest point to a facet of $\overline{\mathcal{P}}(\mathcal{B})$ and show that its Voronoi region do not cross $\overline{\mathcal{P}}(\mathcal{B})$. It is sufficient to prove the result for one facet of $\overline{\mathcal{P}}(\mathcal{B})$ as the landscape is the same for all of them.

Let $H_{\mathcal{F}_1}$ denote the hyperplane defined by $\mathcal{B} \backslash g_1$ where the facet $\mathcal{F}_1$ of $\overline{\mathcal{P}}(\mathcal{B})$ lies. While g_1 is in $\overline{\mathcal{P}}(\mathcal{B})$ it is clear that $-g_1$ is not in $\overline{\mathcal{P}}(\mathcal{B})$. Adding to $-g_1$ any linear combination of the $n-1$ vectors generating $\mathcal{F}_1$ is equivalent to moving in a hyperplane, say H_{P_1}, parallel to $\mathcal{F}_1$ and it does not change the distance from $H_{\mathcal{F}_1}$. Additionally, any integer multiplication of $-g_1$ results in a point which is further from the hyperplane (except by ±1 of course). Note however that the orthogonal projection of $-g_1$ onto $H_{\mathcal{F}_1}$ is not in $\mathcal{F}_1$. The only lattice point in H_{P_1} having this property is obtained by adding all g_j, $2 \leq j \leq n$, to $-g_1$, i.e. it is the point $-g_1 + \sum_{j=2}^{n} g_j$.

This closest point to $\overline{\mathcal{P}}(\mathcal{B})$, along with the points $\mathcal{B} \backslash g_1$, form a simplex. The centroid of this simplex is a hole of the lattice (but it is not a deep hole of A_n for $n \geq 3$). It is located at a distance of $\alpha/(n+1)$, $\alpha > 0$, to the center of any facet of the simplex and thus to $\mathcal{F}_1$ and $\overline{\mathcal{P}}(\mathcal{B})$.

Proof of Theorem 7.2

In this appendix, we prove Lemma 7.2. One can check that any generator matrix G obtained from the following Gram matrix generates E_8 and satisfies the assumption of Lemma 7.2. Consequently, it proves Theorem 7.2.

$$\Gamma_{E_8} = \begin{pmatrix} 4 & 2 & 0 & 2 & 2 & 2 & 2 & 2 \\ 2 & 4 & 2 & 0 & 2 & 2 & 2 & 2 \\ 0 & 2 & 4 & 0 & 2 & 2 & 0 & 0 \\ 2 & 0 & 0 & 4 & 2 & 2 & 0 & 0 \\ 2 & 2 & 2 & 2 & 4 & 2 & 2 & 0 \\ 2 & 2 & 2 & 2 & 2 & 4 & 0 & 2 \\ 2 & 2 & 0 & 0 & 2 & 0 & 4 & 0 \\ 2 & 2 & 0 & 0 & 0 & 2 & 0 & 4 \end{pmatrix}. \tag{7.34}$$

Lemma 7.2. *Let G be a generator matrix of E_8, where the set of basis vectors $\mathcal{B}$ are all from the first lattice shell. Let $\mathring{\mathcal{P}}(\mathcal{B})$ be the interior of the fundamental parallelotope of E_8. If $(G^{-1})^T$ is a generator matrix of E_8 with basis vectors from the first shell, then the G basis is Voronoi-reduced with respect to $\mathring{\mathcal{P}}$.*

To prove Lemma 7.2, we need the next lemma.

Lemma 7.3. *Let G be a generator matrix of a lattice Λ, where the rows of G form a basis $\mathcal{B}$ of Λ with lattice points from the first shell. Let $H_{\mathcal{F}_i}$ denote the hyperplane defined by $\mathcal{B}\backslash g_i$ where the facet $\mathcal{F}_i$ of $\overline{\mathcal{P}}(\mathcal{B})$ lies. If $(G^{-1})^T$ generates Λ^* with lattice points from the first shell of this dual lattice, then the minimum distance between any $H_{\mathcal{F}_i}$ and a lattice point in $\Lambda\backslash\overline{\mathcal{P}}(\mathcal{B})$ is*

$$d(\Lambda\backslash\overline{\mathcal{P}}(\mathcal{B})), H_{\mathcal{F}_i}) = \frac{d(\Lambda)}{\sqrt{\gamma(\Lambda^*)} \times \sqrt{\gamma(\Lambda)}}. \tag{7.35}$$

Proof. We derive the minimum distance between a lattice point outside of $\overline{\mathcal{P}}(\mathcal{B})$, $x \in \Lambda\backslash\overline{\mathcal{P}}(\mathcal{B})$, and $H_{\mathcal{F}_i}$. This involves two steps: First, we find one of the closest lattice point by showing that any other lattice point is at the same distance or further and then we compute the distance between this point and $H_{\mathcal{F}_i}$. In the following, u_i is the basis vector of the dual lattice Λ^* orthogonal to $\mathcal{F}_i$ and g_i the only basis vector of Λ where $u_i \cdot g_i \neq 0$, $g_i \in \mathcal{B}$.

As explained in the proof for A_n, while g_i is in $\overline{\mathcal{P}}(\mathcal{B})$ it is clear that $-g_i$ is not in $\overline{\mathcal{P}}(\mathcal{B})$. Adding any linear combination of the $n-1$ vectors generating the facet is equivalent to moving in a hyperplane parallel to $H_{\mathcal{F}_i}$. It does not change the distance from $H_{\mathcal{F}_i}$. Additionally, any integer multiplication of $-g_i$ results in a point which is further from the facet (except by ±1 of course). Therefore, $-g_i$ is one of the closest lattice points in $\Lambda\backslash\overline{\mathcal{P}}(\mathcal{B})$ from $H_{\mathcal{F}_i}$.

How far is this point from $\overline{\mathcal{P}}(\mathcal{B})$? This distance is obtained by projecting $-g_i$ on u_i, the vector orthogonal to $\mathcal{F}_i$

$$d(\Lambda\backslash\overline{\mathcal{P}}(\mathcal{B}), H_{\mathcal{F}_i}) = \frac{|g_i \cdot u_i|}{||u_i)||}. \tag{7.36}$$

First, the term $g_i \cdot u_i = 1$ since $G \cdot G^{-1} = I$. Second, from the Hermite constant of the dual lattice Λ^*, and using $\det G \ \cdot \ \det G^{-1} = 1$, we get:

$$d(\Lambda^*) = \frac{\sqrt{\gamma(\Lambda^*)}}{|\det\ G|^{1/n}}. \tag{7.37}$$

Since all vectors of Λ^* are from the first shell (i.e. their norm is $d(\Lambda^*)$, assumption of the lemma), (7.36) becomes

$$d(\Lambda\backslash\mathcal{P}(\mathcal{B}), H_{\mathcal{F}_i}) = \frac{1}{d(\Lambda^*)} = \frac{|\det\ G|^{1/n}}{\sqrt{\gamma(\Lambda^*)}}. \tag{7.38}$$

The result follow by expressing $\det G$ as a function of $\gamma(\Lambda)$ and $d(\Lambda)$.

□

We are now ready to prove Lemma 7.2.

Proof (of Lemma 7.2). g_i, u_i, and $H_{\mathcal{F}_i}$ are defined as in the previous proof. We apply (7.35) to E_8. Since this lattice is self-dual, $\gamma(E_8^*) = \gamma(E_8) = 2$ and (7.35) becomes

$$d(E_8\backslash\overline{\mathcal{P}}(\mathcal{B}), H_{\mathcal{F}_i}) = \frac{d(E_8)}{2} = \rho(E_8),$$

As a result, the closest lattice point outside of $\overline{\mathcal{P}}(\mathcal{B})$ is at a distance equal to the packing radius. Since the covering radius is larger than the packing radius, the basis is VR only if the Voronoi region of the closest points have a specific orientation relatively to the parallelotope.

The rest of the proof consists in showing that $H_{\mathcal{F}_i}$ is a reflection hyperplane for $-g_i$. Indeed, this would mean that there is a lattice point of E_8 on the other side of $H_{\mathcal{F}_i}$, located at a distance $\rho(E_8)$ from $H_{\mathcal{F}_i}$. It follows that this lattice point is at a distance $d(E_8)$ from $-g_i$ and is one of its closest neighbor. Hence, one of the facet of its Voronoi region lies in the hyperplane perpendicular to the vector joining the points, at a distance $\rho(E_8)$ from the two lattice points. Consequently, this facet and $H_{\mathcal{F}_i}$ lie in the same hyperplane. Finally, the fact that a Voronoi region is a convex set implies that the basis is VR.

To finish the proof, we show that $H_{\mathcal{F}_i}$ is indeed a reflection hyperplane for $-g_i$. The reflection of a point with respect to the hyperplane perpendicular to u_i (i.e. $H_{\mathcal{F}_i}$) is expressed as

$$s_{u_i}(-g_i) = -g_i + 2 \cdot \frac{u_i \cdot g_i}{||u_i||^2} \cdot u_i.$$

We have to show that this point belongs to E_8. The dual of the dual of a lattice is the original lattice. Hence, if the scalar product between $s_{u_i}(-g_i)$ and all the vectors of the basis of E_8^* is an integer, it means that this point belongs to E_8.

$$s_{u_i}(-g_i) \cdot u_j = -g_i \cdot u_j + 2 \cdot \frac{u_i \cdot g_i}{||u_i||^2} \cdot u_i \cdot u_j.$$

We analyse the terms of this equation: $g_i \cdot u_j \in \mathbb{Z}$ since they belong to dual lattices. We already know that $u_i \cdot g_i = 1$. Also $u_i \cdot u_j \in \mathbb{Z}$ as E_8^* is an integral lattice. With Equation (7.37), we get that $\frac{2}{||u_i||^2} = 1$. We conclude that $s_{u_i}(-g_i) \cdot u_j \in \mathbb{Z}$. □

Proof of Theorem 7.3

Proof. Λ_{24} is self-dual with $\gamma(\Lambda_{24}) = 2$ and $d(\Lambda_{24}) = 2$. Asumme that we have two generator matrices G and G^{-1} satisfying the assumption of Lemma 7.3. Equation (7.35) gives

$$d(\Lambda_{24} \backslash \mathcal{P}(\mathcal{B}), H_{\mathcal{F}_i}) = \frac{d(\Lambda_{24})}{4} = \frac{\rho(\Lambda_{24})}{2}. \tag{7.39}$$

This distance is clearly smaller than the packing radius of Λ_{24}.

Moreover, Equation (7.36) shows that if G^{-1} contains a point which is not from the first shell, $\min_i d(\Lambda \backslash \overline{\mathcal{P}}(\mathcal{B}), H_{\mathcal{F}_i})$ becomes smaller has $\max_i ||u_i||$ is greater. Hence, (7.39) is an upper bound on $d(\Lambda_{24} \backslash \mathcal{P}(\mathcal{B}), H_{\mathcal{F}_i})$. □

7.6.2 Proof of Theorem 7.4

All Voronoi facets of f associated to a same point of $\mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ form a polytope. The variables within a AND condition of the HLD discriminate a point with respect to the boundary hyperplanes where these facets lie: The condition is true if the point is on the proper side of all these facets. For a given point $y \in \mathcal{P}(\mathcal{B})$, we write a AND condition m as $\text{Heav}(yA_m + q_m) > 0$, where $A_m \in \mathbb{R}^{n \times l_m}$, $q_m \in \mathbb{R}^{l_m}$. Does this convex polyhedron lead to a convex CPWL function?

Consider Equation (7.16). The direction of any v_j is chosen so that the Boolean variable is true for the point in $\mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ whose Voronoi facet is in the corresponding boundary hyperplane. Obviously, there is a boundary hyperplane, which we name ψ, between the lattice point $0 \in \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ and $g_1 \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$. This is also true for any $x \in \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ and $x + g_1 \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$. Now, assume that one of the vector v_j has its first coordinate v_j^1 negative. It implies that for a given location $\tilde{y}$, if one increases y_1 the term $y \cdot v_j^T - p_j$ decreases and eventually becomes negative if it was positive. Note that the Voronoi facet corresponding to this v_j is necessarily above ψ, with respect to the first axis e_1, as the Voronoi cell is convex. It means that there exists $\tilde{y}$ where one can do as follows. For a given y_1 small enough, y is in the decoding region $z_1 = 0$. If one increases this value, y will cross ψ and be in the decoding region $z_1 = 1$. If one keeps increasing the value of y_1, y eventually crosses the second hyperplane and is back in the region $z_1 = 0$. In this case f has two different values at the location $\tilde{y}$ and it is not a function. If no v_j^1 is negative, this situation is not possible. All v_j^1 are positive if and only if all $x \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ have their first coordinates x_1 larger than the first coordinates of all $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$. Hence, the convex polytope leads to a function if and only if this condition is respected. If this is the case, we can write $\text{Heav}(yA_m + q) > 0 \Leftrightarrow \wedge_{k=1}^{l_m} y \cdot a_{m,k} + q_{m,k} > 0$,

$a_{m,k}, q_{m,k} \in \{v_j, p_j\}$. We want $y_1 > h_{m,k}(\tilde{y})$, for all $1 \leq k \leq l_m$, which is achieved if y_1 is greater than the maximum of all values. The maximum value at a location $\tilde{y}$ is the active piece in this convex region and we get $y_1 = \vee_{k=1}^{l_m} h_{m,k}(\tilde{y})$.

A Voronoi facet of a neighboring Voronoi cell is concave with the facets of the other Voronoi cell it intersects. The region of f formed by Voronoi facets belonging to distinct points in $\mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ form concave regions that are linked by a OR condition in the HLD. The condition is true if y is in the Voronoi region of at least one point of $\mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$: $\vee_{m=1}^{M}\{\wedge_{k=1}^{l_m} y \cdot a_{m,k} + q_{m,k}\} > 0$. We get $f(\tilde{y}) = \wedge_{m=1}^{M}\{\vee_{k=1}^{l_m} h_{m,k}(\tilde{y})\}$.

Finally, l_m is strictly inferior to τ_f because all Voronoi facets lying in the affine function of a convex part of f are facets of the same corner point. Regarding the bound on M, the number of logical OR term is upper bounded by half of the number of corner of $\mathcal{P}(\mathcal{B})$ which is equal to 2^{n-1}.

7.6.3 First order terms of the decision boundary function before folding for A_n

The equations of the boundary function for A_n are the following.

$$f^{n=2} = \Big[h_{p1} \vee h_1\Big] \wedge \Big[h_{p2}\Big].$$

$$f^{n=3} = \Big[h_{p1} \vee h_1 \vee h_2\Big] \;\wedge \Big[(h_{p2} \vee h_1) \wedge (h_{p2} \vee h_2)\Big] \;\wedge \Big[h_{p3}\Big].$$

$$\begin{aligned} f^{n=4} = &\Big[h_{p1} \vee h_1 \vee h_2 \vee h_3\Big] \wedge \Big[(h_{p2} \vee h_1 \vee h_2) \wedge (h_{p2} \vee h_2 \vee h_3) \wedge (h_{p2} \vee h_1 \vee h_3)\Big] \wedge \\ &\Big[(h_{p3} \vee h_1) \wedge (h_{p3} \vee h_2) \wedge (h_{p3} \vee h_3)\Big] \wedge \Big[h_{p4}\Big]. \end{aligned}$$

7.6.4 Proof of Theorem 7.6

A ReLU neural network with n inputs and W_1 neurons in the hidden layer can compute a CPWL function with at most $\sum_{i=0}^{n} \binom{W_1}{i}$ pieces [PMB13]. This is easily understood by noticing that the non-differentiable part of $\max(0, a)$ is a $n-2$-dimensional hyperplane that separates two linear regions. If one sums W_1 functions $\max(0, d_i \cdot y)$, where d_i, $1 \leq i \leq w_1$, is a random vector, one gets W_1 of such $n-2$-hyperplanes. The result is obtained by counting the number of linear regions that can be generated by these W_1 hyperplanes.

The proof of the theorem consists in finding a lower bound on the number of such $n-2$-hyperplanes (or more accurately the $n-2$-faces located in $n-2$-hyperplanes) partitioning $\mathcal{D}(\mathcal{B})$. This number is a lower-bound on the number of linear regions. Note that these $n-2$-faces are the projections in $\mathcal{D}(\mathcal{B})$ of the $n-2$-dimensional intersections of the affine pieces of f.

We show that many intersections between two affine pieces linked by a $\vee$ operator (i.e. an intersection of affine pieces within a convex region of f) are located in distinct $n-2$-hyperplanes. To prove it, consider all sets $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ of the form $\{x, x+g_1, x+g_j\}$, $x \in \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$, $x+g_j \in \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$. The part of decision boundary function f generated by any of these sets has 2 pieces and their intersection is a $n-2$-hyperplane. Consider the set $\{0, g_1, g_2\}$. Any other set is obtained by a composition of reflections and translations from this set. For two $n-2$-hyperplanes associated to different sets to be the same, the second set should be obtained from the first one by a translation along a vector orthogonal to the 2-face defined by the points of this first set. However, the allowed translations are only in the direction of a basis vector. None of them is orthogonal to one of these sets.

Finally, note that any set $\{x \cup (\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})})\}$ where $|\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}| = i$, encountered in the proof of Theorem 7.5, can be decomposed into $i-1$ of such sets (i.e. of the form $\{x, x-g_1, x-g_1+g_j\}$). Hence, from the proof of Theorem 7.5, we get that the number of this category of sets, and thus a lower bound on the number of $n-2$-hyperplanes, is $\sum_{k=0}^{n-1}(n-1-k)\binom{n-1}{k}$. Summing over $k = n-i = 0 \ldots n-1$ gives the announced result.

7.6.5 Proof of Theorem 7.7

We count the number of sets $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ with cardinality i. We walk in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ and for each of the 2^{n-1} points $x \in \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ we investigate the cardinality of the set $\mathcal{T}_f(x+g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$. In this scope, the points in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ can be sorted into two categories: (l_i) and (ll_i). In the sequel, $\sum_j b_j$ denotes any sum of points in the set $\{0, b_j\}_{j=3}^n$. These two categories and their properties (see also the explanations below Theorem 7.7), are:

$$\begin{aligned}(l_i)\ \forall\ x = \sum_j g_j \in\ \mathcal{C}^0_{\mathcal{P}(\mathcal{B})},\ x' \in D_n \backslash \{g_k, 0\},\ 3 \leq k \leq n :\\ x + g_k \in \mathcal{T}_f(x+g_1),\ x + x' \notin \mathcal{T}_f(x+g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}.\end{aligned} \tag{7.40}$$

$$\begin{aligned}&(ll_i)\ \forall\ x = \sum_j g_j + g_2 \in\ \mathcal{C}^0_{\mathcal{P}(\mathcal{B})},\\ &\quad x' \in D_n \backslash \{g_i, -g_2+g_i, -g_2+g_i+g_k, 0\},\ 3 \leq i < k \leq n :\\ &\quad (1)\ (a)\ x + g_i \in \mathcal{T}_f(x+g_1),\ (b)\ x - g_2 + g_i \in \mathcal{T}_f(x+g_1),\\ &\quad (2)\ x - g_2 + g_i + g_k \in \mathcal{T}_f(x+g_1),\\ &\quad (3)\ x + x' \notin \mathcal{T}_f(x+g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}.\end{aligned} \tag{7.41}$$

We count the number of sets $\mathcal{T}_f(x)\ \cap\ \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ with cardinality i per category.

(l_i) is like A_n. Starting from the lattice point 0, the set $\mathcal{T}_f(0+g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ is composed of 0 and the $n-2$ other basis vectors (i.e. without g_2 because it is perpendicular to g_1). Then, for all g_{j_1}, $3 \leq j_1 \leq n$, the sets $\mathcal{T}_f(g_{j_1} + g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ are obtained by adding any of the $n-3$ remaining basis vectors to g_{j_1} (i.e. not g_1, g_2, or g_{j_1}). Indeed, if we add again g_{j_1}, the resulting point is outside $\mathcal{P}(\mathcal{B})$ and should not be considered. Hence, the cardinality of these sets is $n-2$ and there are $\binom{n-2}{1}$ ways to choose g_{j_1}: any basis vectors except g_1 and g_2. Similarly, for $g_{j_1} + g_{j_2}$, $j_1 \neq j_2$, the cardinality of the sets $\mathcal{T}_f(g_{j_1} + g_{j_2} + g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ is $n-3$ and there are $\binom{n-2}{2}$ ways to choose $g_{j_1} + g_{j_2}$. More generally, there are $\binom{n-2}{i}$ sets $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ of cardinality $n-1-i$.

(ll_i) To begin with, we are looking for the neighbors of $g_2 + g_1$. First (i.e. property (1)), we have the following $1 + 2 \times (n-2)$ points in $\mathcal{T}_f(g_2+g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$: g_2, any $g_j + g_2$, $3 \leq j \leq n$, and any g_j, $3 \leq j \leq n$. Second (i.e. property (2)), the $\binom{n-2}{2}$ points $g_j + g_k$, $3 \leq j < k \leq n$, are also neighbors of $g_2 + g_1$. Hence, $g_2 + g_1$ has $1 + 2 \times (n-2) + \binom{n-2}{2}$ neighbors in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$. Then, the points $g_1 + g_2 + g_{j_1}$, $3 \leq j_1 \leq n$, have $1 + 2 \times (n-2-1) + \binom{n-2-1}{2}$ neighbors of this kind, using the same arguments, and there are $\binom{n-2}{1}$ ways to chose g_{j_1}. In general, there are $\binom{n-2}{i}$ sets of cardinality $1 + 2 \times (n-2-i) + \binom{n-2-i}{2}$.

To summarize, each set replicates $\sum_i \binom{n-2}{i}$ times, where for each i we have both (l_i) sets of cardinality $1 + (n-2-i)$ and (ll_i) sets of cardinality $1 + 2 \times (n-2-i) + \binom{n-2-i}{2}$. As a result, the total number of pieces of f is obtained as

$$\sum_{i=0}^{n-2} \left(\underbrace{[1 + (n-2-i)]}_{(l_i)} + \underbrace{\left[\underbrace{1 + 2(n-2-i)}_{(1)} + \underbrace{\binom{n-2-i}{2}}_{(2)} \right]}_{(ll_i)} \right) \times \underbrace{\binom{n-2}{i}}_{(o_i)} - 1, \tag{7.42}$$

where the -1 comes from the fact that for $i = n-2$, the piece generated by (l_i) and the piece generated by (ll_i) are the same. Indeed, the bisector hyperplane of x, $x + g_1$ and the bisector hyperplane of $x + g_2$, $x + g_2 + g_1$ are the same since g_2 and g_1 are perpendicular.

7.6.6 Proof of Theorem 7.10

Lemma 7.4. *Among the elements of $\mathcal{C}_{\mathcal{P}(\mathcal{B})}$, only the points of the form*

1. $x_1 = g_3 + ... + g_{i-1} + g_i$ *and* $x_1 + g_1$,
2. $x_2 = g_3 + ... + g_{i-1} + g_i + g_2$ *and* $x_2 + g_1$,

$i \leq n$, are on the non-negative side of all $BH(g_j, g_k)$, $3 \leq j < k \leq\ n$.

Proof. In the sequel, $\sum_i g_i$ denotes any sum of points in the set $\{0, g_i\}_{i=3}^n$. For 1), consider a point of the form $g_3 + ... + g_{j-1} + g_{j+1} + ... + g_{i-1} + g_i$, $j+1 < i-1 \leq n-1$. This point is on the negative side of all $BH(g_j, g_k)$, $j < k \leq i$. More generally, any point $\sum_i g_i$, where $\sum_i g_i$ includes in the sum g_k but not g_j, $j < k \leq n$, is on the negative side of $BH(g_j, g_k)$. Hence, the only points in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ that are on the non-negative side of all hyperplanes have the form $g_3 + ... + g_{i-1} + g_i$, $i \leq n$.

Moreover, if $x \in \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ is on the negative side of one of the hyperplanes $BH(g_j, g_k)$, $3 \leq j < k \leq n$, so is $x + g_1$ since g_1 is in all $BH(g_j, g_k)$.

2) is proved with the same arguments. □

Proof. See the proof of Lemma 7.4. □

Proof. (of Theorem 7.10) (i) The folding via $BH(g_j, g_k)$, $3 \leq j < k \leq n$, switches g_j and g_k in the hyperplane containing $\mathcal{D}(\mathcal{B})$, which is orthogonal to e_1. Switching g_j and g_k does not change the decision boundary because of the basis symmetry, hence f is unchanged.

Now, for (ii), how many pieces are left after all reflections? To count the number of pieces of f, defined on $\mathcal{D}'(\mathcal{B})$, we need to enumerate the cases where both $x \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ and $x' \in \mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ are on the non-negative side of all reflection hyperplanes.

Firstly, we investigate the effect of the folding operation on the term $\sum_{i=0}^{n-2}[1 + (n-2-i)] \times \binom{n-2}{i}$ in Equation (7.42). Remember that it is obtained via (l_i) (i.e. Equation (7.40)). Due to the reflections, among the points in $\mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ of the form $\sum_j g_j + g_1$ only $x = g_3 + g_4 + ... + g_{i-1} + g_i + g_1$, $j \leq n$, is on the non-negative side of all reflection hyperplanes (see result 1. of Lemma 7.4). Similarly, among the elements in $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$, only $x - g_1$ and $x - g_1 + g_{i+1}$ (instead of $x - g_1 + g_k$, $3 \leq k \leq n$) are on the non-negative side of all reflection hyperplanes. Hence, at each step i, the term $[1 + (n-2-i)]$ becomes 2 (except for $i = n-2$ where it is 1). Therefore, the folding operation reduced the term $\sum_{i=0}^{n-2}[1 + (n-2-i)] \times \binom{n-2}{i}$ to $(n-2) \times 2 + 1$.

Secondly, we investigate the reduction of the term $\sum_{i=0}^{n-2}\left[1 + 2(n-2-i) + \binom{n-2-i}{2}\right] \times \binom{n-2}{i}$ obtained via (ll_i) (i.e. Equation 7.41). The following results are obtained via item 2. of Lemma 7.4. Among the points denoted by $\sum_j g_j + g_2 + g_1 \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ only $x = g_3 + g_4 + ... + g_{i-1} + g_i + g_2 + g_1$ is on the proper side of all reflection hyperplanes. Among the neighbors of any of these points, of the form $(ll_i) - (2)$, only $x + g_{i+1} + g_{i+2}$ is on the proper side of all hyperplanes. Additionally, among the neighbors of the form $(ll_i) - (1)$ and $(ll_i) - (b)$, i.e. $x + g_k$ or $x - g_2 + g_k$, $3 \leq k \leq n$, g_k can only be g_{i+1}. Therefore, the folding operation reduces the term $\sum_{i=0}^{n-2}[1 + 2(n-2-i) + \binom{n-2-i}{2}] \times \binom{n-2}{i}$ to $(n-3) \times 4 + 3 + 1$.

□

7.6.7 Proof of Theorem 7.8

Proof. We count the number of sets $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ with cardinality i. We walk in $\mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ and for each of the 2^{n-1} points $x \in \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ we investigate the cardinality of the set $\mathcal{T}_f(x + g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$. In this scope, we group the lattice points $x \in \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ in three categories. The numbering of these categories matches the one given in the sketch of proof (see also Equation 7.47 below). $\sum_j g_j$ denotes any sum of points in the set $\{0, g_j\}_{j=4}^n$.

$$\begin{aligned}&(l_i)\ \forall\ x = \sum_j g_j \in\ \mathcal{C}^0_{\mathcal{P}(\mathcal{B})},\ x' \in D_n \backslash \{g_j, 0\},\ 4 \leq k \leq n:\\ &\quad x + g_k \in \mathcal{T}_f(x + g_1),\ x + x' \notin \mathcal{T}_f(x + g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}.\end{aligned} \tag{7.43}$$

$$\begin{aligned}&(ll_i) - A\ \forall\ x = \sum_j g_j + g_2 \in\ \mathcal{C}^0_{\mathcal{P}(\mathcal{B})},\\ &\quad x' \in D_n \backslash \{g_i, -g_2 + g_i, -g_2 + g_i + g_k, 0\},\ 4 \leq i < k \leq n:\\ &\quad (1)\ x + g_i \in \mathcal{T}_f(x + g_1),\ x - g_2 + g_i \in \mathcal{T}_f(x + g_1),\\ &\quad (2)\ x - g_2 + g_i + g_k \in \mathcal{T}_f(x + g_1),\\ &\quad (3)\ x + x' \notin \mathcal{T}_f(x + g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}.\end{aligned} \tag{7.44}$$

$$\begin{aligned}&(ll_i) - B\ \forall\ x = \sum_j g_j + g_3 \in\ \mathcal{C}^0_{\mathcal{P}(\mathcal{B})},\\ &\quad x' \in D_n \backslash \{g_i, -g_3 + g_i, -g_3 + g_i + g_k, 0\},\ 4 \leq i < k \leq n:\\ &\quad (1)\ x + g_i \in \mathcal{T}_f(x + g_1),\ x - g_3 + g_i \in \mathcal{T}_f(x + g_1),\\ &\quad (2)\ x - g_3 + g_i + g_k \in \mathcal{T}_f(x + g_1),\\ &\quad (3)\ x + x' \notin \mathcal{T}_f(x + g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}.\end{aligned} \tag{7.45}$$

$$
\begin{aligned}
&(lll_i)\ \forall\ x = \sum_j g_j + g_2 + g_3 \in\ \mathcal{C}^0_{\mathcal{P}(\mathcal{B})},\\
&\qquad x' \in D_n \backslash \{g_i, g_i + g_k, g_i + g_k + g_l, 0\},\ 4 \leq i < k < l \leq n :\\
&\qquad (1)\ x - g_2 + g_k \in \mathcal{T}_f(x + g_1),\ x - g_3 + g_k \in \mathcal{T}_f(x + g_1),\\
&\qquad x + g_k \in \mathcal{T}_f(x + g_1),\\
&\qquad (2)\ x - g_3 - g_2 + g_i + g_k \in \mathcal{T}_f(x + g_1),\\
&\qquad x - g_2 + g_i + g_k \in \mathcal{T}_f(x + g_1),\ x - g_3 + g_i + g_k \in \mathcal{T}_f(x + g_1),\\
&\qquad (3)\ x + g_i + g_k + g_l \in \mathcal{T}_f(x + g_1),\\
&\qquad (4)\ x + x' \notin \mathcal{T}_f(x + g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}.
\end{aligned}
\tag{7.46}
$$

We count the number of i-simplices per category.

(l_i) is like A_n. Starting from the lattice point 0, the set $\mathcal{T}_f(0 + g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ is composed of 0 and the $n-3$ other basis vectors (i.e. without g_2 and g_3 because they are perpendicular to g_1). Then, for all g_{j_1}, $4 \leq j_1 \leq n$, the sets $\mathcal{T}_f(g_{j_1} + g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ are obtained by adding any of the $n-4$ remaining basis vectors to g_{j_1} (i.e. not g_1, g_2, g_3 or g_{j_1}). Hence, the cardinality of these sets is $n-3$ and there are $\binom{n-3}{1}$ ways to choose g_{j_1}: any basis vectors except g_1, g_2, and g_3. Similarly, for $g_{j_1} + g_{j_2}$, $j_1 \neq j_2$, the cardinality of the sets $\mathcal{T}_f(g_{j_1} + g_{j_2} + g_1) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ is $n-4$ and there are $\binom{n-3}{2}$ ways to choose $g_{j_1} + g_{j_2}$. More generally, there are $\binom{n-3}{i}$ sets $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ of cardinality $n-2-i$.

(ll_i) is like the basis of D_n (see (ll_i) in the proof in Appendix 7.6.5), repeated twice because we now have two basis vectors orthogonal to g_1 instead of one. Hence, we get that there are $\binom{n-3}{i}$ sets of cardinality $2 \times \left(1 + 2(n-3-i) + \binom{n-3-i}{2}\right)$.

(lll_i) is the new category. We investigate the neighbors of a given point $x = \sum_j g_j + g_3 + g_2 + g_1$. First (1), any $\sum_j g_j + g_3 + g_2$ is in $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$. Any $\sum_j g_j + g_2 + g_k$, $\sum_j g_j + g_3 + g_k$, and $\sum_j g_j + g_3 + g_2 + g_k$, where $4 \leq k \leq n$ and $k \notin \{j\}$ are also in $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$. Hence, there are $3 \times (n-3-i)$ of such neighbors, where $i = |\{j\}|$ (in $\sum_j g_j$). Then, (2) any $\sum_j g_j + g_i + g_k$, $\sum_j g_j + g_2 + g_i + g_k$, and $\sum_j g_j + g_3 + g_i + g_k$, where $4 \leq i < k \leq n$ and $i, k \notin \{j\}$, are in $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$. There are $3 \times \binom{n-3-i}{2}$ possibilities, where $i = |\{j\}|$. Finally (3), any $\sum_j g_j + g_i + g_k + g_l$, $4 \leq i < k < l \leq n$ and $i, k, l \notin \{j\}$ are in $\mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$. There are $\binom{n-3-i}{3}$ of them, where $i = |\{j\}|$.

To summarize, each set replicates $\sum_i \binom{n-3}{i}$ times, where for each i we have (l_i) sets of cardinality $1 + n - 3 - i$, (ll_i) $2 \times \left(1 + 2(n-3-i) + \binom{n-3-i}{2}\right)$, and (lll_i) $1 + 3 \times (n-3-i) + 3 \times \binom{n-3-i}{2} + \binom{n-3-i}{3}$. As a result, the total number of pieces of f is obtained as

$$
\sum_{i=0}^{n-3} \Bigg(\underbrace{[1 + (n-3-i)]}_{(l_i)} + 2 \underbrace{\left[1 + 2(n-3-i) + \binom{n-3-i}{2}\right]}_{(ll_i)} + \tag{7.47}
$$

$$
\underbrace{\left[\underbrace{1 + 3(n-3-i)}_{(1)} + \underbrace{3\binom{n-3-i}{2}}_{(2)} + \underbrace{\binom{n-3-i}{3}}_{(3)}\right]}_{(lll_i)} \Bigg) \times \underbrace{\binom{n-3}{n-i}}_{(o_i)} - 3, \tag{7.48}
$$

where the -3 comes from the fact that for $i = n-3$, the four pieces generated by (l_i), (ll_i), and (lll_i) are the same. Indeed, the bisector hyperplane of x, $x + g_1$, is the same as the one of $x + g_2$, $x + g_2 + g_1$, of $x + g_3$, $x + g_3 + g_1$, and of $x + g_2 + g_3$, $x + g_2 + g_3 + g_1$, since both g_2 and g_3 are perpendicular to g_1. □

7.6.8 Proof of Theorem 7.11

Lemma 7.5. *Among the elements of $\mathcal{C}_{\mathcal{P}(\mathcal{B})}$, only the points of the form*

1. $x_1 = g_4 + ... + g_{i-1} + g_i$ *and* $x_1 + g_1$,

2. $x_2 = g_4 + ... + g_{i-1} + g_i + g_2$ *and* $x_2 + g_1$,

3. $x_3 = g_4 + ... + g_{i-1} + g_i + g_2 + g_3$ *and* $x_3 + g_1$,

$i \leq n$, are on the non-negative side of all $BH(g_j, g_k)$, $4 \leq j < k \leq\ n$.

Proof. (of Theorem 7.11) (i) The folding via $BH(g_j, g_k)$, $4 \leq j < k \leq n$ and $j = 2, k = 3$, switches g_j and g_k in the hyperplane containing $\mathcal{D}(\mathcal{B})$, which is orthogonal to e_1. Switching g_j and g_k does not change the decision boundary because of the basis symmetry, hence f is unchanged.

Now, for (ii), how many pieces are left after all reflections? To count the number of pieces of f, defined on $\mathcal{D}'(\mathcal{B})$, we need to enumerate the cases where both $x \in \mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$ and $x' \in \mathcal{T}_f(x) \cap \mathcal{C}^0_{\mathcal{P}(\mathcal{B})}$ are on the non-negative side of all reflection hyperplanes.

Firsly, we investigate the effect of the folding operation on the term $\sum_{i=0}^{n-3}[1 + n - 3 - i] \times \binom{n-3}{i}$ in Equation (7.47). Remember that it is obtained via (l_i) (i.e. Equation (7.43)). Due to result 1 of Lemma 7.5 and similarly to the corresponding term in the proof of Theorem 7.10, this term reduces to $(n-3) \times 2 + 1$.

Secondly, we investigate the reduction of the term $2\left[1 + 2(n-3-i) + \binom{n-3-i}{2}\right] \times \binom{n-3}{i}$, obtained via (ll_i) (i.e. Equation (7.44)). The following results are obtained via item 2 of Lemma 7.5. $\binom{n-3}{i}$ reduces to 1 at each step i because in $\mathcal{C}^1_{\mathcal{P}(\mathcal{B})}$, only the points $x = g_2 + g_3 + g_{i-1} + g_i + g_1$ are on the non-negative side of all hyperplanes, $i \leq n$. Then, since any $\sum_j g_j + g_3 + g_1$ is on the negative side of the hyperplane $BH(g_2, g_3)$, $(ll_i) - (B)$ generates no piece in f (defined to $\mathcal{D}'(\mathcal{B})$). $(ll_i) - (A)$ is the same case as the case (ll_i) in the proof of Theorem 7.10. Hence, the term reduces to $(n-3) \times (4) + 3 + 1$.

Finally, what happens to the term $\left[1 + 3(n-3-i) + 3\binom{n-3-i}{2} + \binom{n-3-i}{3}\right]\binom{n-3}{n-i}$, obtained via (lll_i) (i.e. Equation (7.45))? The following results are obtained via item 3 of Lemma 7.5. As usual, $\binom{n-3}{n-i}$ reduces to 1 at each step i. Then, $3(n-3-i)$, due to $(lll_i) - (1)$, becomes 2×1 at each step i because any $x - g_2 + g_k$ (in $(lll_i) - (1)$), $k \leq 4 \leq n$, is on the negative side of $BH(g_2, g_3)$. For $x - g_3 + g_k$ and $x + g_k$, only one valid choice of g_k remains at each step i, as explained in the proof of Theorem 7.10. Regarding the term $3\binom{n-3-i}{2}$, due to $(lll_i) - (2)$, any point $x - g_2 + g_i + g_k$ (in $(lll_i) - (2)$) is on the negative side of $BH(g_2, g_3)$ and at each step i there is only one valid way to chose g_j and g_k for both $x - g_3 - g_2 + g_j + g_k$ and $x - g_3 + g_j + g_k$. Eventually, for the last term due to $(lll_i) - (3)$ only one valid choice remain at each step i. Therefore, the term due to (lll_i) is reduced to to $(n-4) \times 6 + 5 + 3 + 1$. □

Chapter 8

A lattice-based approach to the expressivity of deep ReLU neural networks

In this chapter, we show how lattice theory can be used to establish new results in the field of deep learning. More specifically, we focus on the *expressive power of deep neural networks*. Typically, the goal of this line of research is to show that there exist functions that can be well approximated by a deep neural network with a polynomial number of parameters whereas an exponential number of parameters is required for any shallow neural network. We therefore speak of separation theorems.

There already exist several separation theorems between shallow and deep neural networks in the literature, which depend on various parameters. Many results like [MPCB14], [Tel16], [ABMM18] utilize functions that are based on one dimensional approaches. In particular, a separation theorem is obtained by Telgarsky [Tel16] via a one dimensional triangle wave function (i.e. similar to the decision boundary for A_2, illustrated on Figure 7.5). The theorem does not depend on the dimension but on a parameter k characterizing the number of periods in the triangle wave function. Additional details on the work of [Tel16] and a survey on recent results on this topic can be found in the Appendices 8.2.2 and 8.2.1, respectively.

The CPWL boundary function f, obtained from A_n in the previous chapter, is a n-dimensional generalization of the triangle wave function used by [Tel16]. We argue that this function enlighten the missing dimensional dependency in the bound of [Tel16] and thus enables to prove a dimension dependent separation theorem: We show that there exists a function f that can be computed by a deep neural network of polynomial size where any function g computed by a shallow neural network of polynomial size induces an approximation error growing exponentially with the dimension. The novelty of our result lies in the fact that the approximation error grows exponentially with the dimension. To the best of our knowledge, this is the first separation theorem depending on the space dimension.

8.1 The advantage of depth over width

The result presented in this chapter, Theorem 8.1, is partially based on the previous chapter: On the one hand, Theorem 7.6 shows that any ReLU neural network with only one hidden layer needs an exponential number of neurons, $\Omega(2^n)$, to exactly compute the CPWL boundary function obtained for A_n. On the other hand, we have shown that if one is allowed quadric depth, then there exist neural networks of polynomial size to compute the decision boundary functions. This is an instance of the superiority of depth over width in neural network architectures. However, we did not quantify the approximation error. A deeper analysis of the problem yields the followig theorem.

Theorem 8.1 (The dimension dependent separation theorem)**.** *There exists a function $f : \mathbb{R}^{n-1} \rightarrow \mathbb{R}$, computed by a standard ReLU neural network in $\mathcal{O}(n^2)$ layers and $\mathcal{O}(n^3)$ neurons, where any function g computed by a ReLU neural network with $\leq n$ layers and $\leq 2^{n-1}$ neurons induces a L_1 approximation error, over the domain of the function, $|| \ f \ - \ g \ ||_1 \ = \ \Omega(2^{(n-1)^3-n\log_2(n)})$.*

The proof of Theorem 8.1 can be found in Appendix 8.2.3. The proof is based on a periodic extension in $\mathbb{R}^n$ of the decision boundary of A_n (i.e. the function not restricted to $\mathcal{P}(\mathcal{B})$ but to a larger compact set).

8.2 Appendix

8.2.1 Some results on the expressive power of deep neural networks

The ultimate goal of research on the expressive power of deep neural networks is to find a large function class that can only be addressed via deep neural networks and no other ways, including shallow networks and "conventional approaches" (i.e. not deep neural networks). Results of research works in this field are usually either capacity bounds (i.e. what can do a deep neural network) or separation bounds. These bounds can depend on (i) the approximation error, (ii) the dimension of the input as well as (iii) the width and (iv) the depth of the neural network.

Unfortunately, results on larger function class tend to be looser as the bounds have to hold for the worst-case scenario. Moreover, one of the (empirically observed) strength of neural networks compared to other techniques is their ability to efficiently approximate a given function. Therefore, stronger theorems can be obtained for specific functions but are less representative.

Consequently, papers in the literature can be sorted based on the "size" of the function class addressed and whether or not the results depend on (i),(ii),(iii), and (iv). The present work addresses a small function class (even though it may be a starting point to study algebraic functions), (i), (ii), (iii) and (iv). The following list is not exhaustive and does not include older results related to the field of circuit complexity.

[ES16] proved a separation theorem including (i), (ii), and (iii) for a large class of function, namely "radial" functions. Nevertheless, this separation holds only for two-layer and three-layer neural networks, thus (iv) is missing. Also, note that [Dan17] found a simpler proof of this result and [SS17] extended this separation result between two-layer and three-layer network to a larger class of function including the Euclidean unit ball.

[MPCB14] achieved the best capacity theorem for deep ReLU neural networks including (ii), (iii), and (iv). Similarly to our work, this is achieved via a small function class. These functions can be computed via "conventional methods" as they are based on a periodic one dimensional function.

[Tel16] proved a separation theorem between shallow and deep networks (this separation theorem was improved by [ABMM18] by re-using the same ideas) including (i), (iii) and (iv). Since this theorem is based on a one dimensional triangle wave function (see Appendix 8.2.2), (ii) is missing (a multi-dimensional function is considered but the bound does not depend on (ii)).

[ABMM18] achieved a multi-dimensional construction with an exponential number of linear regions requiring only a polynomial number of parameters (part (i) of Theorem 3.9 in the paper) but the proof is based on the fact that the high dimensional part of this function can be computed by a conventional method (i.e. the function with w^n pieces considered can be computed via a w 2-max, as shown in the proof of Lemma 3.7).

[RPK$^+$16] showed that any random deep ReLU network achieves an exponential number of linear region depending on (ii),(iii) and (iv). Additionally, via the trajectory length, they observed that most of the random linear regions in trained networks are in fact noise that should be addressed through regularization.

Finally, [PMR$^+$17] and [PV18] are recent results addressing large function class.

8.2.2 The triangle wave function of [Tel16]

Telgarsky considers a one dimensional triangle wave function. The key observation is that adding two (shifted) copies of a triangle wave function increases the number of pieces in an additive manner, while composition acts multiplicatively. Within a neural network, increasing the width of a layer is equivalent to adding functions, while increasing the depth is equivalent to composing functions. Hence, a function computed by a deep network, say $f : \mathbb{R} \to \mathbb{R}$, can have many more oscillations than functions computed by networks with few layers, say $g : \mathbb{R} \to \mathbb{R}$. Roughly speaking, if the activation function in each neuron is a triangle wave function with p pieces, a two-layer w-wide network leads to a triangle wave function of wp pieces while a L-layer network with $\mathcal{O}(1)$-width leads to p^L pieces.

The difference (or "error") between f and a line can be characterized by the triangle areas illustrated on Figure 8.1. Hence, the L^1 error between f and g is then bounded from below after summing the triangle areas above the line (resp. below the line) whenever g is below (resp. above) this same line. Indeed, since g has a number of pieces inferior to f, it can only cross this line a limited number of times compared to f.

This one-dimensional result is then extended to the n-dimensional case in the following manner. A function $p_{\tilde{y}}(y_1) = (y_1, \tilde{y})$ is defined. $\tilde{y}$ can be understood as an offset. The network is then only applied

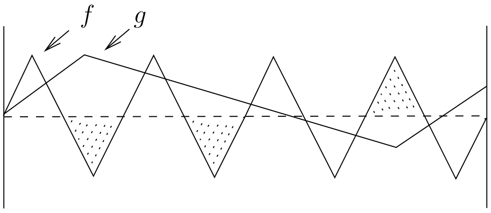


Figure 8.1: Triangle wave function considered by Telgarsky. The doted triangle areas are used to get a lower bound of the error between f and g.

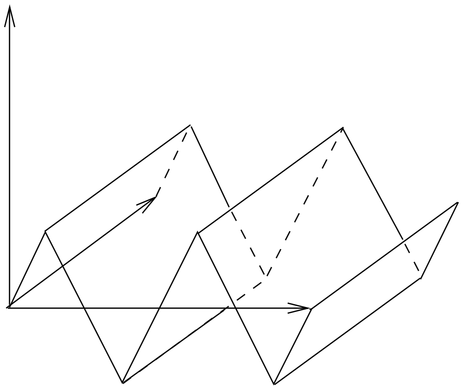


Figure 8.2: Triangle wave function with offset in $\mathbb{R}^3$. The number of pieces is not increased compared to the baseline function in $\mathbb{R}^2$. There is no dimensional dependence.

on y_1 but the error averaged in the cube $[0,1]^n$.

8.2.3 Proof of Theorem 8.1

We use the same notations as in Chapter 7. Consider the basis for the lattice A_n given by $\Gamma_{A_n} = GG^T = J_n + I_n$. Let $\mathcal{N}(f)$ denote the number of pieces of a CPWL function f. Assume that one is only given k pieces to build a function g approximating f, with $k < \mathcal{N}(f)$. What is the minimum possible approximation error?

The decision boundary function f for A_2 is similar to the triangle wave function used to prove the main separation theorem between deep and shallow networks in [Tel16]. We quickly recall the main ideas of the proof in [Tel16] (a more detailed explanation is also available in Appendix 8.2.2). A triangle wave function f with p periods is considered. It has $2p+1$ affine pieces. [Tel16] established a lower bound of the average pointwise disagreement $|f(\tilde{y}) - g(\tilde{y})|$ over a compact set between f and a function g having k pieces where $k < 2p+1$. This is achieved by summing the triangle areas above (resp. below) the dashed black line, illustrated on Figure 8.1, whenever g is below (resp. above) this same line. Indeed, since g has a limited number of pieces, it can only cross this line a limited number of times.

What happens if we consider a similar function in $\mathbb{R}^3$, where we replace triangles by tetrahedra? Such a function, limited to $\mathcal{D}(B)$, is illustrated on Figure 7.7b. It is the decision boundary obtained for A_3 defined by (7.12). The dashed line of Figure 8.1 should now be replaced by the plane $\Phi^{n=3} = \{y \in \mathbb{R}^3 : \ y \cdot e_1 = \frac{1}{2} \times (g_1 \cdot e_1)\}$. Similarly to the triangle wave function, $f^{n=3}$ is oscillating around $\Phi^{n=3}$: all pieces of $f^{n=3}$ cross $\Phi^{n=3}$. Note that the number of pieces is significantly increased compared to a simple extension of the triangle wave function in $\mathbb{R}^3$ (see Figure 8.2). The same pattern is observed for any space dimension n, where the triangles or tetrahedra become n-simplices.

Consider any convex part of f, say $f_m = \vee_{k=1}^{l_m} h_{m,k}$ (see (7.20)). There are 2^{n-1} of such f_m. The polytope

$$\{y \in \mathcal{P}(\mathcal{B}) : y_1 \geq f_m(\tilde{y}), \ y \cdot e_1 \leq \frac{1}{2} \times (g_1 \cdot e_1)\}$$

is a truncated simplex due to the limitation of f to $\mathcal{D}(\mathcal{B})$. For all m, $1 \leq m \leq 2^{n-1}$, these polytopes are the truncated version of a n-simplex. This (non-truncated) simplex is illustrated for $n = 3$ on Figure 8.3. See again Figure 7.7b to see the truncated simplices for $n = 3$.

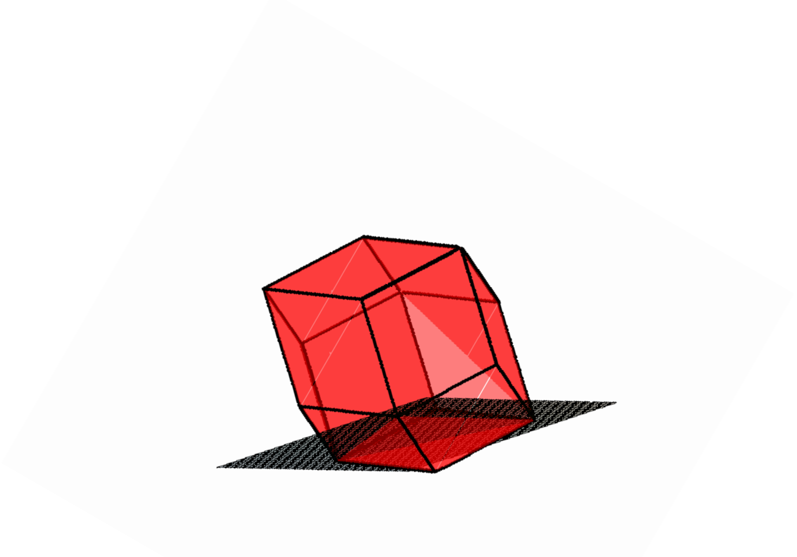


Figure 8.3: The Voronoi cell of A_3 is a rhombic dodecahedron. A subset of the facets of this polytope generates some affine pieces of f. The non-truncated tetrahedron is the part of the dodecahedron below the plane $\Phi^{n=3}$.

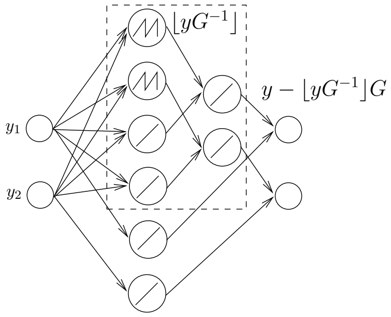


Figure 8.4: Translation block.

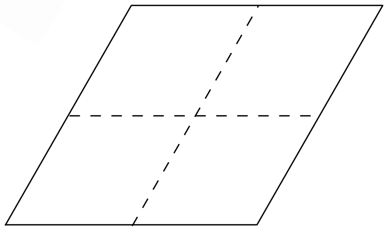


Figure 8.5: Partition of $\mathcal{P}(2\mathcal{B})$ induced by one translation block.

This same function can be extended to $\mathbb{R}^n$ by periodicity; we have same boundary in $\mathcal{P}(\mathcal{B}) + x$ as in $\mathcal{P}(\mathcal{B})$, for any lattice point x. Indeed, $\mathcal{P}(\mathcal{B})$ is a fundamental region of the lattice and one can perform a tessellation of $\mathbb{R}^n$ with $\mathcal{P}(\mathcal{B})$. This translates into extending the boundary function of (7.20) as follows: $f(\tilde{y}) = f(\tilde{y_0})$ where $y = y_0 - x \in \mathcal{P}(\mathcal{B})$. We consider a set $\mathcal{P}(\{g_1, \alpha g_2, \alpha g_3, \dots, \alpha g_n\})$, where $\alpha = 2^M$ and $M \geq 1$ is an integer. The new scaled region has $2^{M(n-1)}$ copies of $\mathcal{P}(\mathcal{B})$. This extended function is defined over the domain $\mathcal{D}(\{g_1, \alpha g_2, \alpha g_3, \dots, \alpha g_n\})$, which is the projection of the scaled region on the hyperplane $\{e_i\}_{i=2}^n$. If we let M grow with n, the exponential increase of the volume yields a total number of pieces superexponential in n. The next proposition shows that this extended function can be efficiently computed by a deep and narrow neural network.

Proposition 8.1. *Consider a VR or SVR basis $\mathcal{B}$ defining any lattice. Let f be the boundary function defined on $\mathcal{D}(\mathcal{B}) = \mathcal{D}(\{g_1, g_2, g_3, \dots, g_n\})$. Consider also its extended decision boundary function defined on the compact set $\mathcal{D}(\{g_1, \alpha g_2, \alpha g_3, \dots, \alpha g_n\})$, where $\alpha = 2^M$. Let L and w be the depth and the width of the neural network evaluating f on $\mathcal{D}(\mathcal{B})$. Then, the extended boundary function has $\Omega(2^{M(n-1)})$ pieces and it can be computed by a ReLU neural network of width* $\max(3(n-1), w)$ *and depth* $3M + L$.

Proof. First, let us define (without loss of generality) the 2-sawtooth ReLU activation function as ReLU'$(u) = u \mod 1, \ \forall \ u \in [0, 2]$. This function allows to divide any interval into two equal subintervals and then translates the point near the origin. For illustration in $\mathbb{R}^2$, as shown in Figures 8.4 and 8.5, (y_1, y_2) is multiplied by G^{-1}, the 2-sawtooth ReLU' is applied twice (on each coordinate), the output is subtracted from the other output to implement the floor operation, and then the result is multiplied again by the generator matrix G. This amounts to partitioning $\mathcal{P}(2\mathcal{B})$ into four equal regions $\{\mathcal{P}(\mathcal{B}), \mathcal{P}(\mathcal{B}) + g_1, \mathcal{P}(\mathcal{B}) + g_2, \mathcal{P}(\mathcal{B}) + g_1 + g_2\}$.

In $\mathbb{R}^n$, the 2-sawtooth ReLU' is used to partition and translate $\mathcal{P}(\alpha\mathcal{B})$, where $\alpha = 2^M$ and $M \geq 1$ is an integer. At step ℓ, $\ell = 1 \dots M$, a translation block similar to the one on Figure 8.4 executes the three operations: multiply by $G^{-1}/2^{M-\ell}$, apply n times a 2-sawtooth ReLU', finally multiply by $2^{M-\ell}G$. $\mathcal{P}(\alpha\mathcal{B})$

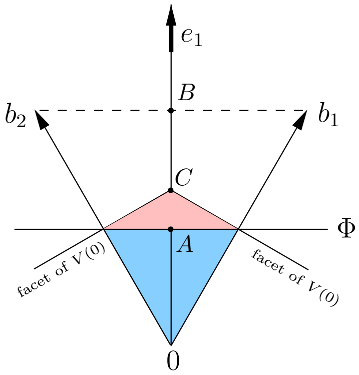


Figure 8.6: Illustration of the non-truncated simplex (in pink on the figure).

has $\alpha^n = 2^{Mn}$ regions equivalent to $\mathcal{P}(\mathcal{B})$. Similarly, if we consider the set $\mathcal{P}(\{g_1, \alpha g_2, \alpha g_3, \dots, \alpha g_n\})$, there are $2^{M(n-1)}$ regions equivalent to $\mathcal{P}(\mathcal{B})$ and the extended decision boundary function defined on the domain $\mathcal{D}(\{g_1, \alpha g_2, \alpha g_3, \dots, \alpha g_n\})$ has $\Omega(2^{M(n-1)})$ pieces.

Hence, the extended boundary function is computed via two neural networks: A first neural network with $3M$ layers based on M translation blocks of maximum width $3(n-1)$ converts $y_0 \in \mathcal{P}(\{g_1, \alpha g_2, \alpha g_3, \dots, \alpha g_n\})$ into $y \in \mathcal{P}(\mathcal{B})$. Subsequently, the second neural network, evaluating f defined on $\mathcal{D}(\mathcal{B})$, takes y as its input. □

The (extended) boundary function f for A_n on $\mathcal{D}(\{g_1, \alpha g_2, \alpha g_3, \dots, \alpha g_n\})$ is used to prove the in-approximability result for shallow networks. Theorem 8.1 is obtained as corollary of Theorem 8.2 with $M = n^2$.

Theorem 8.2. *Consider an A_n-lattice basis defined by the Gram matrix (7.12). Let f be the (extended) decision boundary function, defined on the compact set $\mathcal{D}(\{g_1, \alpha g_2, \alpha g_3, \dots, \alpha g_n\})$, where $\alpha = 2^M$ and $||g_i|| = \sqrt{2}$, $1 \leq i \leq n$. For M large enough, any function g that can be computed by a L-deep, w-wide ReLU neural network where $L\log_2(w) \leq M - n$ has an error*

$$||f - g||_1 = \Omega\left(2^{(n-1)M - n\log_2(n)}\right), \tag{8.1}$$

whereas if $L = 3M + \mathcal{O}(n^2)$ and $w = 3(n-1)$, f can be computed by the network.

Proof. We begin with the first part of the theorem. If the compact set $\mathcal{P}(\{g_1, \alpha g_2, \alpha g_3, \dots, \alpha g_n\})$, where $\alpha = 2^M$, is large enough, we can make the following approximation: There are roughly as many Voronoi cell as parallelotopes $\mathcal{P}(\mathcal{B})$ in $\mathcal{P}(\{g_1, \alpha g_2, \alpha g_3, \dots, \alpha g_n\})$. This implies that the extended decision boundary f "contains" at least one non-truncated simplex for each $\mathcal{P}(\mathcal{B})$. With Proposition 8.1, we get that there are $2^{M(n-1)}$ $\mathcal{P}(\mathcal{B})$ in $\mathcal{P}(\{g_1, \alpha g_2, \alpha g_3, \dots, \alpha g_n\})$.

The proof involves a lower bound on the volume of one non-truncated simplex. We show that the volume of one non-truncated simplex is $\Omega\left(1/n^n\right)$ for an edge length of $\sqrt{2}$. Let V_n be the volume of the non-truncated simplex. This simplex is equivalent to a hyperpyramid obtained by intersecting $V(0)$ with the hyperplane Φ orthogonal to e_1 and located at a shift of $\frac{1}{2}g_1 \cdot e_1$. Figure 8.6 illustrates the volume V_n in pink color in two dimensions (see also Figure 8.3 for the case in three dimensions). The blue color represents the simplex whose vertices are $\{0, \frac{1}{2}g_1, \frac{1}{2}g_2, \dots, \frac{1}{2}g_n\}$.

The volume of the hyperpyramid is $V_n = \frac{S \times h}{n}$, where S is the $n-1$-dimensional volume of this hyperpyramid facet lying on Φ and h is the hyperpyramid height.

We start by determining h. Let O be the point representing the origin in $\mathbb{R}^n$. Denote by C the centroid of the simplex whose vertices are $\{0, g_1, g_2, \dots, g_n\}$. The line OC cuts Φ at the point A and the hyperplane $\{g_i\}_{i=1}^n$ at the point B. Then $h = OC - OA$ becomes

$$h = OC - \frac{1}{2}OB = \frac{n}{n+1}OB - \frac{1}{2}OB = \frac{n-1}{2(n+1)}\sqrt{\frac{n+1}{n}},$$

because $OB = \sqrt{\frac{n+1}{n}}$ is the height of the simplex with edge length $\sqrt{2}$. The area S, i.e. the $n-1$-dimensional volume of the facet lying on Φ, is bounded from below by the area S' of the blue simplex facet lying on Φ. Figure 8.6 shows them equal in $\mathbb{R}^2$, but the facet of the pink simplex will be larger that its blue counterpart for $n \geq 3$. From the formula of a simplex volume, we get

$$S \geq S' = \frac{a^{n-1}}{(n-1)!} \frac{\sqrt{n}}{2^{(n-1)/2}}, \quad a = \frac{1}{2}\|g_1\| = \frac{1}{\sqrt{2}}.$$

Finally, the lower bound of V_n is

$$\begin{aligned} V_n \geq \frac{S' \times h}{n} &= \frac{n(n-1)}{2^n \times (n+1)^{3/2} \times n!} \\ &\sim \frac{1}{\sqrt{2\pi}} \frac{1}{(2n/e)^n}. \end{aligned}$$

Hence, the volume of the non-truncated simplex is $\Omega(1/n^n)$.

Hence, if K is the number of $\mathcal{P}(\mathcal{B})$ in $\mathcal{P}(\{g_1, \alpha g_2, \alpha g_3, \ldots, \alpha g_n\})$, the error between f and g is bounded from below by

$$\int_{\mathcal{D}} |f(\tilde{y}) - g(\tilde{y})| d\tilde{y} = K\Omega\left(1/n^n\right), \tag{8.2}$$

where $K = 2^{M(n-1)} = 2^{M(n-1)-n\log_2(n)} \cdot 2^{n\log_2(n)}$.

Similarly to the strategy of Telgarsky (see Appendix 8.2.2), we can assume that each additional piece in g cancels (at most) the volume of $\mathcal{O}(1)$ simplices in the bound. Moreover, via Theorem 1 of [RPK+16] we know that no L-deep w-wide ReLU network with input in $\mathbb{R}^{n-1}$ can compute more than $\mathcal{O}(2^{(n-1)L\log_2(w)})$ pieces. Consequently, the approximation error is bounded from below by

$$a \times 2^{(n-1)(M-\log_2(n))-\log_2(n)} - b \times 2^{(n-1)L\log_2(w)}, \tag{8.3}$$

where a and b are some constants. As a result, if we choose $M \geq L\log_2(w) + n$, then the approximation error is $\Omega\left(2^{(n-1)M-n\log_2(n)}\right)$.

The second part of the result is a direct consequence of Proposition 8.1, where the part of f on $\mathcal{D}(\mathcal{B})$ is evaluated as follows: we implement the $\mathcal{O}(n^2)$ reflections, that enable to reduce the number of pieces to compute down to a linear number, via a ReLU neural network of depth $\mathcal{O}(n^2)$ and width $\mathcal{O}(n)$. □

Chapter 9

Decoding with Deep Learning

This last chapter reports some simulation results where deep learning is used to decode/detect.

9.1 Learning protocol and training statistics

9.1.1 Brief introduction to deep learning vocabulary

Figure 9.1 displays the usual framework to learn the parameters of a neural network [GBC16]. The weights of the neural network are updated via usual stochastic gradient descent methods (e.g. Adam optimizer). The gradient is computed with the backpropagation algorithm[1]. The optimization function considered is the mean squared error. The label, associated to a given input of the neural network, refers to the data considered for the training: I.e. the neural network minimizes the optimization function when its outputs matche the labels. The batch size refers to the number of samples used to estimate the gradient at each iteration of the training algorithm. Note that on Figure 9.1, MLD labels are used as training statistics, not the data sent.

9.1.2 Training statistics for the Gaussian channel

We consider the case of Gaussian noise. Only a limited amount of studies discuss what training statistics (i.e. the data and labels to train the neural network) should be used for efficient training of a neural-based decoder. In [GCHtB17], they introduce the notion of normalized validation error to investigate which SNR, to generate the training data, is most suited for efficient training. Indeed, in [GCHtB17] the labels are the codewords sent, not the MLD codeword. They empirically observed that a SNR neither too high nor too low is the most efficient. In most papers, authors mix noisy data obtained at different SNRs to perform training, in hope that the network is efficient at all those SNRs. To the best of our knowledge, in all papers on neural networks for decoding, the input message z associated to a noisy received signal y is used as label for the training.

Let us call $\mathcal{C}$ a given constellation/code/lattice that we want to decode with a neural network and x an element of $\mathcal{C}$. Regardless of the noise, the "ideal" label that should be used for a given y is what would have been decoded by the optimal decoder, not the transmitted sequence. Take for instance $\mathcal{C}$ as a simple BPSK[2]. If the noise moves a point (e.g. $+1$) further than the decoding threshold (e.g. -0.2), one should not tell the neural network to try to recover the original point (here $+1$): It should decode the point associated to the decoding region the received y belongs to (here -1).

Remember that the optimal decoder performs the following operation. Given a y (anywhere) in the space of $\mathcal{C}$, it finds the x associated to the decoding region where y is located. Moreover, if we want the network to learn the entire structure of $\mathcal{C}$, the training sample should be composed of points sampled randomly in its space. Equivalently, one could randomly choose elements of $\mathcal{C}$ (with equiprobability) and add uniformly distributed noise.

Nevertheless, as explained in Section 7.5.2, to get quasi-MLD performance on the Gaussian channel the network does not need to learn the entire structure of $\mathcal{C}$ but rather the most relevant decision boundaries around the x. Indeed, some regions near the boundaries are so far from x such that the Gaussian noise almost never sends x in these regions. Therefore, a quasi-MLD network can potentially

[1] The primary use of deep learning libraries is to provide this backpropagation algorithm.

[2] Binary phase-shift keying.

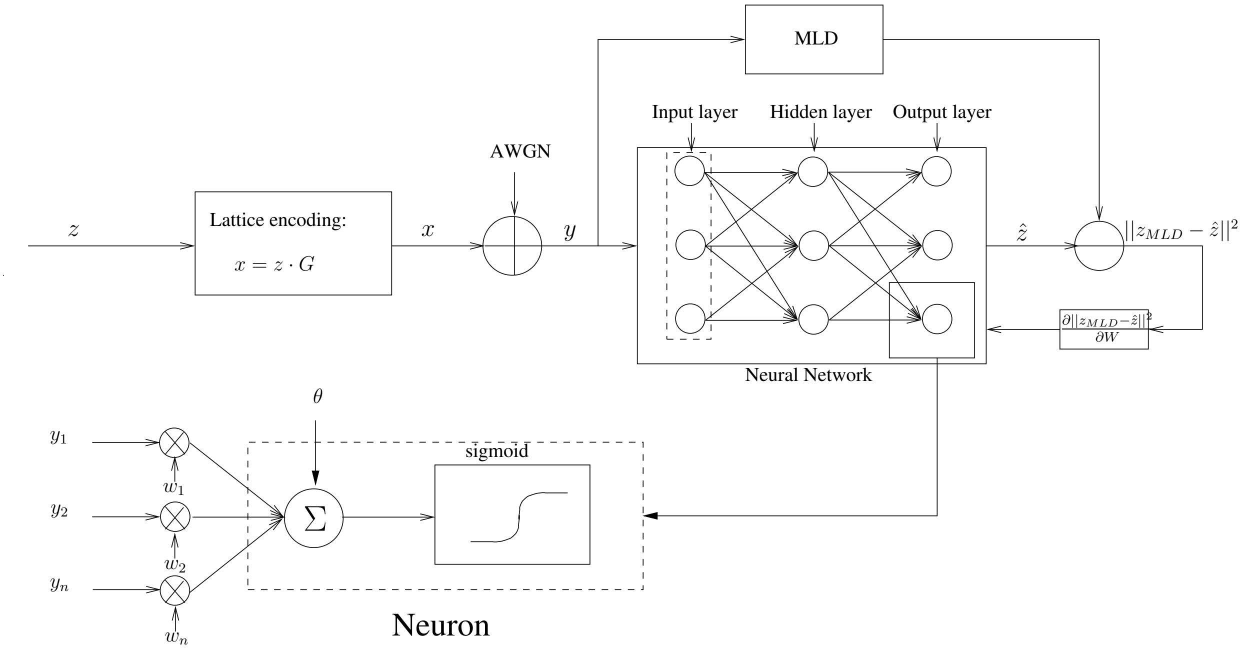


Figure 9.1: Framework to learn the parameters of a neural network. W denote the parameters of the neural network.

make many simplifications compared to a perfect MLD network and thus reduce its complexity. These simplifications can be learned by training the network with Gaussian noise.

Unfortunately, getting MLD label can be very costly (especially compared to using the input message z): any sample should be decoded with the optimal decoder and potentially stored. Hence, if we were to use z as label for the training due to limited resources, what SNR should be used on the Gaussian channel? In light of the above discussion, we would want both to learn the necessary structure of the code to get quasi-MLD performance (i.e. the SNR should not be too high) but the "noise" in the label (i.e. messages that are wrongly labeled with respect to the optimal decoder) should not be too high either. Empirically, we observed that the SNR corresponding to an error probability of 10^{-2} is a good trade-off: only one sample out of 100 is mis-labeled but the SNR is low enough to properly explore $\mathcal{C}$.

9.2 Multilevel MIMO detection with Deep Learning

We address the case of multilevel symbol detection on MIMO channels, as defined in Section 3.4.5, via deep neural networks. There exist many algorithms to perform MIMO detection, whose performance ranges from optimal to highly suboptimal. A first category of decoders includes sphere decoding methods based on lattice points enumeration and radius adaptation. The complexity of sphere decoding is clearly less prohibitive than an exhaustive search and is polynomial in the dimension for small dimensions. Detection based on sphere decoding is quasi-optimal and is very competitive in terms of number of operations for dimensions less than 32. However, the dynamic tree structure of sphere decoding makes it hardware-unfriendly.

In a second category we find linear receivers: the zero-forcing (ZF) detector and the minimum mean squared error (MMSE) detector. Finally, a non-exhaustive list of decoders having performance somewhere between these two categories includes: the decision feedback-equalizer (DFE), the K-best sphere decoder, message passing methods (e.g. belief propagation, approximate message passing, expected propagation) and semidefinite relaxation. While some of these algorithms are near-optimal in specific settings, their performance are largely degraded when these specific conditions are not respected. As a result, the problem of finding hardware-friendly low-complexity methods exhibiting near-optimal performance in most settings remains open. Neural network based implementation could offer new solutions.

MIMO detection with neural networks has already been investigated by several research groups. In [SDW17] [SDW18], the quadratic form of the MIMO channel is used to build the network. In [TXB+18] [LL18] [ITW18] [HWJL20] sub-optimal message passing iterative MIMO decoders are improved with the approach introduced in [HRF14] [NBB16]. The main idea of these studies is to unfold the underlying graph used by an iterative algorithm to get improvement via learning. Simulations show that in most cases learning enhances the performance of the considered algorithm. Nonetheless, these results are almost never compared to optimal detection. It is therefore difficult to assess the real efficiency of such an approach. Additionally, most studies consider binary inputs only. In [SDW18], one-hot encoding is used to address the case of non-binary inputs. Unfortunately, the number of output neurons increases significantly with the spectral efficiency making this solution impractical.

9.2.1 The network structure

Architecture

In [SDW17], the architecture of the network is inspired from the projected gradient descent, where the message at step $i+1$ is estimated as:

$$\hat{z}_{i+1} = \prod\left(\hat{z}_i - \eta \cdot \frac{\partial ||y - z.G||^2}{\partial z}|_{z=\hat{z}_i}\right) = \prod\left(\hat{z}_i - 2\eta \cdot yG^T + \eta \cdot \hat{z}_i GG^T\right), \tag{9.1}$$

where $\prod$ is a projection operator. Our neural network embraces the same paradigm. It takes the form of an iterative algorithm where an estimate of the output is available after each iteration. It is illustrated in Figure 9.2. A generic iteration has two layers, as shown in the figure, where the network structure is derived from the following matrix equations:

$$\begin{aligned} \xi_k &= \sigma_c\left(W^1_{1,k}\hat{z}_k + W^2_{1,k} yG^T + W^3_{1,k}\hat{z}_k GG^T + W^4_{1,k} v_k + \text{bias}_{1,k}\right), \\ \hat{z}_{k+1} &= \sigma_c\left(W_{2,k}\xi_k + \text{bias}_{2,k}\right), \quad v_{k+1} = W_{3,k} z_k + \text{bias}_{3,k}. \end{aligned}$$

In the expression of ξ_k, we can clearly recognize the terms used by the gradient descent, weighted by W^i_1 instead of η (the two other terms are a hidden variable and a bias term commonly used in neural networks). The intuition behind this expression is that the network will learn specific learning rates η for each iteration and each component. The operation performed between the ξ layer and the next layer can be interpreted as the projection operator $\prod$. The activation function used σ_c is described in the next section.

In [SDW17], the matter of how $\hat{z}_0$ should be initialized for the first iteration of the neural network is not discussed. We address and take advantage of this question in the section on the twin-network.

The multilevel activation function

The default approach to address a multi-class problem with neural networks is to use the so-called "one-hot encoding". Namely, if the network should classify data between more than two categories, say M categories, it will have M output neurons where legal combinations of values are only the M combinations with a single neuron equal to 1 and all the others equal to 0. Unfortunately, this approach implies a large amount of output neurons. In the network of Figure 9.2, if each component of the input message z can take M levels, using one-hot encoding means having $n \times M$ output neurons (the neurons labeled z_{k+1} in Figure 9.2) instead of n in the binary case. This implies a greater complexity as well as longer training.

To address this issue we introduce a novel activation function. We adapt the non-linearity in the output neurons to take into account non-binary symbols. Let the standard sigmoid function be $\sigma(t) = 1/(1+e^{-t})$. Our customized sigmoid function shall be defined as a sum of standard sigmoids,

$$\sigma_c(t) = \sum_{i=1}^{M} \sigma(t - \tau_i) + A,$$

where τ_i are sigmoid shifts and A is an overall translation. As an example, for $z \in \{-2,-1,0,1,2\}^n$ ($M=5$), the customized sigmoid is taken to be $\sigma_c(t) = \sigma(t+15)+\sigma(t+5)+\sigma(t-5)+\sigma(t-15)+\sigma(t-25)-2$, as depicted on Figure 9.3.

The twin-network

To further improve our system, we considered the paradigm of random forests [SSBD14]: "divide and conquer". With a random forest, many decision trees are trained on a random subset of the training data with a randomly picked subset of dimensions. One decision tree alone tends to highly overfit. However,

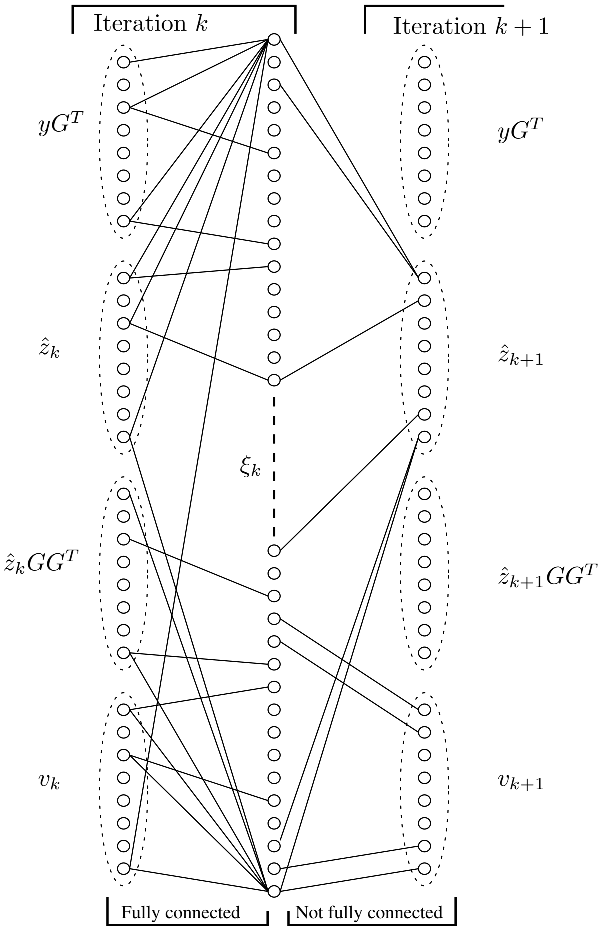


Figure 9.2: Deep neural network architecture, two layers per iteration.

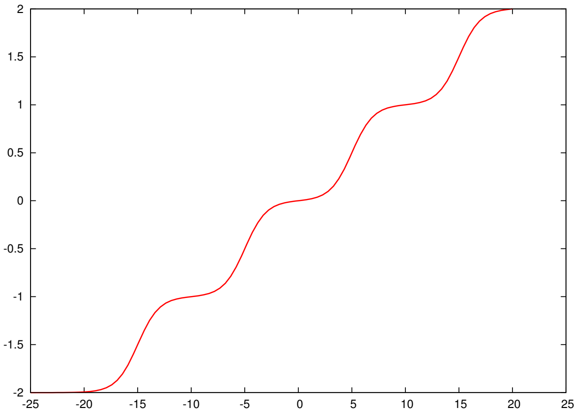


Figure 9.3: $\sigma_c(t)$ for 5-level integer symbols.

the random forest, based on the aggregation of the trees and a majority decision rule, has very good and consistent results. The important idea is to introduce some randomness between the trees. The concept of a random forest is analogous to extreme pruning successfully utilized by the cryptography community for sphere decoding [GNR10] (also mentionned in Section 3.4.3). In this latter paper, they consider decoding trees having low success rate and repeated the operation many times with different bases of the lattice. They observed that complexity decreases much faster than the performance deterioration. Therefore, in case of sub-optimality of the neural network, a solution can be to duplicate the network and introduce randomness instead of increasing the number of parameters in the deep neural network (DNN). An easy way to introduce randomness is to initialize neural networks, constructed as the one of Figure 9.2, with distinct $\hat{z}_0$. An instance of such system is illustrated in Figure 9.4. The first DNN is initialized with a random $\hat{z}_0$, while the second DNN receives an initial $\hat{z}_0$ obtained by ZF.

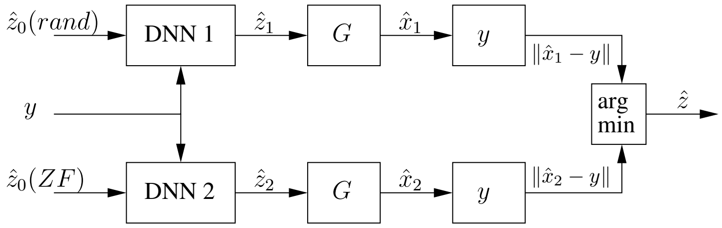


Figure 9.4: Block representation of the neural system.

9.2.2 Simulation results

We present neural networks performance observed under several settings. For each of these settings, the results reported are the best complexity-performance trade-off obtained, i.e. we decreased the neural network size as much as possible while keeping quasi-MLD performance.

For the first set of simulations, depicted in Figure 9.5, the settings are the following. We take $n = 8$ and $M = 5$ levels on each z_i. The MIMO channel is a static channel randomly sampled from an i.i.d. Gaussian matrix. The considered matrix instance has condition number 17 and Hermite constant -4.7dB (as a real lattice), i.e., this is a bad channel realization and an interesting challenge to our DNN. Additionally, we used the multilevel activation function. The training is done with the Adam optimizer and a small batch size ($\approx$ 200). The multilevel MIMO detector used for these simulations has $1.25n$ iterations, ξ is of size $7n$ and v of size n. Hence, the twin-DNN has $2 \times 1.25n \times 42n^2 \approx 100n^3$ parameters (which is about 10 times smaller than 5^8).

We observe that the twin-network DNN performance is close to the MLD performance and clearly outperforms the single DNN (we show only the curve for the randomly initialized single DNN because it matches the one initialized with the ZF point). This means that, under a different initialization, the two single DNNs are almost never wrong at the same time (except for the cases that cannot be recovered by the optimal decoder). Hence, this approach can be beneficial to improve a sub-optimal neural network.

The second set of simulations was performed under the same settings as the one described above, but the batch size is increased to $\approx 3e^4$ to train the network. Moreover, the size of the ξ layer is decreased to $4n$. In Figure 9.6, we show a significant improvement of performance for the single DNN case: Within just three iterations (<< 1.25n) and with a decreased network size, we manage to get quasi-MLD performance. The number of parameters in the network is decreased to $3 \times 24n^2$. We don't believe that the improvement is caused by a larger amount of data used to train the network: Firstly, in the small-batch simulations we let the networks learn for a large enough amount of time. Secondly, the convergence to quasi-MLD performance with a large batch size is very fast. We rather believe that a non-noisy gradient is better suited for efficient learning in our settings.

We compare the performance of multilevel activation functions and one-hot encoding. Note that one-hot encoding associated to the soft-max activation function yields soft outputs. Hence, we modify the network used in Figure 9.6 by replacing each M-level output neuron (i.e. the neurons labeled z_{k+1} in Figure 9.2) by M neurons to get soft outputs. Moreover, we used 10 iterations. The result obtained is depicted in Figure 9.7. We observe that we don't manage to get quasi-optimal performance as in Figure 9.6. Additionally, the training phase of this network took significantly more time than the previous one and required much more fine tuning of hyper-parameters. To summarize, this network is more complex and harder to train.

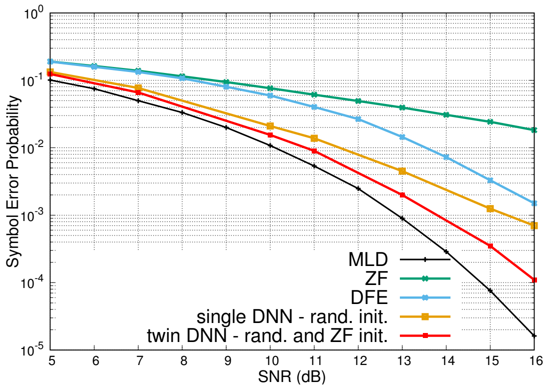


Figure 9.5: First simulations, with small batch size training.

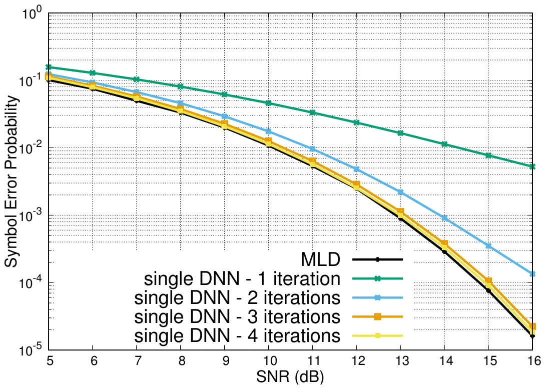


Figure 9.6: Second simulations, with large batch size ($\approx 3e^4$).

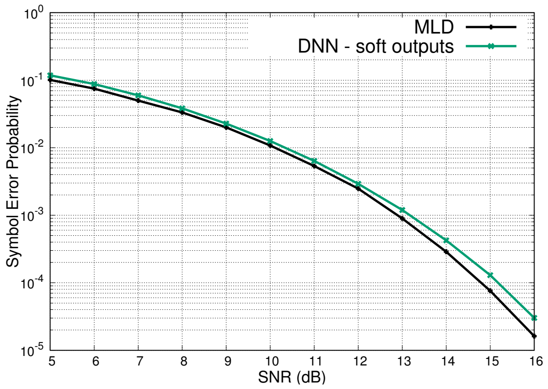


Figure 9.7: DNN with soft outputs.

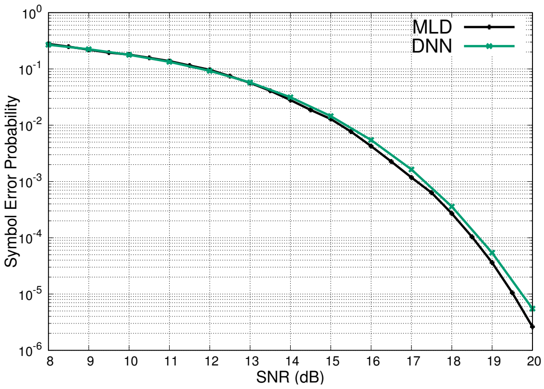


Figure 9.8: DNN for the T55 MIMO channel.

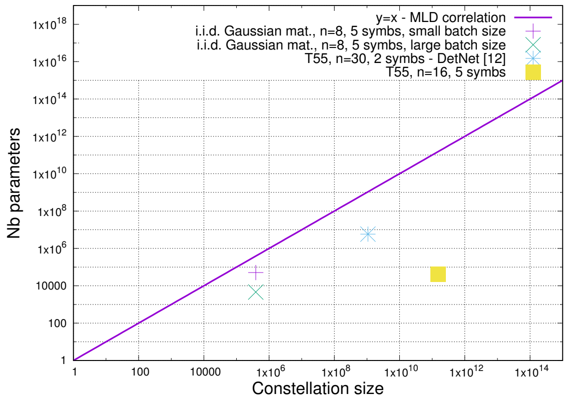


Figure 9.9: Complexity analysis of the considered models.

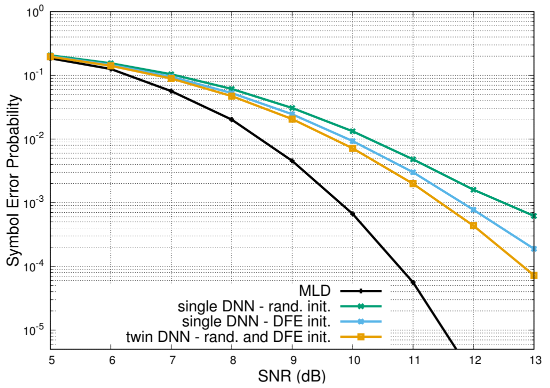


Figure 9.10: The MIMO channel is taken to be the generator of BW_{16}. The number of parameters in the neural networks is significantly increased compared to the (usual) MIMO case.

We also performed a simulation with the $T55$ MIMO channel used in [SDW17]. The associated matrix is ill-conditioned, which makes it challenging for linear detectors but not necessarily for the sphere decoder. We take $n = 16$, $M = 5$ levels, and consider the multilevel activation function on output neurons. We observe in Figure 9.8 that this situation is well handled by our neural network.

The complexity of the different models presented in this section is summarized in Figure 9.9. We plot the number of parameters (number of edges) of the network as a function of the cardinality of the constellation (obtained as M^n). We also show in blue the complexity of the neural network used in [SDW17] for the T55 MIMO matrix. We believe that the number of parameters of this neural network could be diminished without degrading the performance if a larger batch size is considered for the training.

In light of these results, we may conclude that deep learning, with the proposed approach, is competitive for a large range of MIMO channels. However, deep learning in some extremal situations is difficult to set up, namely for specific channels where the function to be approximated is very challenging. For instance, if the MIMO channel is the generator matrix of a dense lattice (e.g. E_8, BW_{16}, Λ_{24} [CS99]), the function to learn is more complex (see Chapter 7). Figure 9.10 shows that even a neural network with a large number of iterations and an increased size for each layer fails to achieve quasi-MLD performance.

9.3 Decoding in $\mathcal{P}(\mathcal{B})$

In this last section, the point y to decode, at the output of a Gaussian channel, is aliased in $\mathcal{P}(\mathcal{B})$ before decoding (see Section 7.2). We use a standard fully-connected feed-forward sigmoid network without any constraint on its architecture. To be competitive, the number of parameters in the neural network should grow slower than 2^n. We consider the lattice E_8 ($n = 8$) and the MIMO lattice $T55$ ($n = 16$) taken from [SDW17]. We try to reduce the size of the neural network as much as possible while maintaining quasi-MLD performance. The size of first hidden layer is taken to be of the same order of magnitude as the lattice kissing number ($\tau(E_8) = 240$ and $\tau(T55) = 30$). For E_8, the network has three hidden layers each with 200 neurons. Its performance is quasi-MLD but this model has $W = 83200$ parameters and is too complex relative to HLD for E_8. The ratio $\frac{\log_2(W)}{n} = 2.0$. For $T55$, quasi-MLD performance is achieved with a neural network made up of three hidden layers with 30-50-50 neurons respectively and $W = 6280$ parameters. The ratio $\frac{\log_2(W)}{n} = 0.78$. As in the previous subsection, the decoding complexity is significantly higher the dense lattice compared to the MIMO lattice.

Now, we introduce a learning model with L1 regularization to simplify the structure of the network. It means that the optimization function includes a term which grows with the magnitude of the weights of the neural network. L1 regularization is a well-known technique to get a sparse neural network, with a lower number of parameters. The idea is to reduce the complexity of the HLD while maintaining quasi-optimal performance. The model considered shall therefore have a constrained architecture: its first hidden layer is fixed and taken from the HLD model. The model shall thus decode the first coordinate z_1 of the message z. This model is initialized with exactly the same architecture and parameters as the HLD and we let the training simplify its architecture (except the first layer) while limiting the performance

degradation. With D_4, the model simplifies the Boolean equation of z_1 to two terms only: the second hidden layer (performing the AND operations, see Section 7.4.3) shrank to two neurons while maintaining quasi-MLD performance.
The study of this approach with other lattices is left for future work.

Part

Conclusions and perspectives

9.4 Thesis conclusions

We introduced a unified framework for building lattices and codes over groups in the second part of the thesis. It relies on a simple parity check, which can be applied recursively. Famous lattices such as the Leech lattice in 24 dimensions, Nebe's extremal lattice in 72 dimensions, and Barnes-Wall lattices are obtained in this framework. A new decoding paradigm is established from this unified group codes construction by taking into account the coset parity constraint. The paradigm leads to new bounded-distance decoders, list decoders, and quasi-optimal decoders on the Gaussian channel in terms of probability of error per lattice point. Quasi-optimal performance for BW_{64}, $\mathscr{N}_{72}$, and BW_{128} are shown to be achievable at reasonable complexity. New parity lattices, such as $L_{3 \cdot 24}$, are also considered. They offer an excellent performance-complexity trade-off.

We have already argued why the "combining paradigm on a tree" was likely to be successful (see Section 2.3.2). We now try to go further and identify fundamental ingredients of our approach. In particular, what are the key elements allowing to efficiently decode on the Gaussian channel with the recursive parity-check construction?

- First, we made the simple observation that, unless the coding gain of the lattice is $\Theta(n)$, BDD is not powerful enough in moderate and large dimensions. A decoding radius larger than half the packing radius should be used.

- When considering list decoding, the explored region can be different from a sphere.

- The parity-check relation between the component codes allows to consider the relative decoding radius. This should be opposed with standard multistage decoding (see e.g. Figure 3.4) of multilevel schemes where each level is decoded with the same radius.

- Nevertheless, deeper levels should be decoded with a relatively stronger decoder (i.e. with a larger relative decoding radius) than upper levels. This may imply using decoders stronger than MLD decoders for the component codes at the deeper levels (see e.g. Section 6.2.5).

- Even though the decoders should be stronger at the deeper levels, the dimensions of the component codes at these deeper levels are smaller, which reduces the complexity.

- The splitting strategies allow to control the number of candidates in the list at each recursive step without sacrificing the performance.

As emphasized several times in the thesis, we believe that many ideas related to this framework should be further investigated:

- We noticed that the MLD performance of the parity lattices on the Gaussian channel behaves similarly to the performance of the proposed modified list decoder (see Theorem 5.4). It would be interesting to better understand this observation. Remember that the two first methods to prove that a family of lattices achieves the Poltyrev limit (Minkowski-Hlawaka based proof and Low-density Construction A lattices), presented in Section 3.3.2, are also based on list decoding. They consist in showing that, on average over the ensemble, there is only one lattice point in the decoding sphere. With the parity lattices, we approach the Poltyrev limit even though there are several lattice points in the decoding sphere and it is not an average performance over an ensemble. As a result, showing the parity lattices achieve/approach the Poltyrev limit requires new proof techniques.

- Given a large k, Theorem 5.1 shows that less recursive steps t are needed to achieve a given density. Moreover, on the Gaussian channel, Theorem 5.4 states that the threshold $\mathcal{T}_t$ is better (i.e. it increases more slowly) and thus so better is the decoder. The numerical simulations of formula (5.16) with $k \approx 10$, reported on Figure 5.2, clearly indicate that the scheme is promising: Impresive decoding performance is shown with t as small as 4. As a result, we recommend a study of the practical performance of parity lattices with $k \approx 10$, similar to the one performed with $k = 2$ in Section 5.2.1. Nevertheless, without the splitting strategy, the decoding complexity is higher for large k (see (4.20)). The study should thus involve the second splitting strategy.

- Moreover, the second splitting strategy should also be studied for the decoding of Λ_{24}, $\mathscr{N}_{72}$, and $L_{3 \cdot 24}$ on the Gaussian channel to further reduce the decoding complexity.

- In Section 6.2.5, we have depicted the remarkable performance (error probability and complexity) of the 3-parity-Leech lattice $L_{3\cdot 24}$. We believe that the lattice $L_{3\cdot 3\cdot 24}$, in dimension 216, has impressive performance and admit a reasonable-complexity decoder.
- Of course, it is tempting to continue the study of BW lattices for $n = 256$. BW_{256} was not decoded due to lack of time, not because we failed.
- We presented a BDD for Reed-Muller codes very similar to the BDD of BW lattices. The list decoders of BW lattices could also be adapted to Reed-Muller codes to yield quasi-MLD decoders.
- The sublattices of BW_n with frozen cosets, combined with the successive-cancellation decoder, (discussed at the end of Section 5.2.1) should be further investigated both on the theoretical side and pratical side.
- We performed preliminary simulations to compute the second order moment of the Nebe lattice $\mathscr{N}_{72}$. It seems to be very close to the sphere bound. However, we have not yet computed the interval of confidence of our numerical estimation.

In summary, we believe that the elegance of the single parity-check construction and its associated decoders are promising for the study of group codes in moderate and large dimensions.

In the third part of the thesis, the decoding problem has been investigated from a neural network pespective. We discussed what can and cannot be done with feed-forward neural networks in light of the complexity of the decoding problem. We have highlighted that feed-forward neural networks should compute a CPWL boundary function to decode. When the number of pieces in the boundary function is too high, the size of the shallow neural networks becomes prohibitive and deeper neural networks should be considered. For dense structured lattices, this number of pieces is high even in moderate dimensions whereas it remains reasonable in low and moderate dimensions for unstructured random lattices. Note that the underlying reasons explaining the failure of shallow neural networks are similar to the ones justifying the cutoff rate (see Section 2.2): the too high number of Voronoi facets to consider.

Regarding the training aspect, learning via gradient-descent techniques allows to achieve satisfactory decoding performance when shallow neural networks are considered along with a target function having a limited number of affine pieces. However, when deeper neural networks are used, even when we know that their function class contains the target function, the training is much more challenging. In particular, even learning simple one dimensional oscillatory functions, such as the triangle wave function illustrated on Figure 8.1, is very difficult whereas they can be easily computed via folding. This can only be worst for high dimensional oscillatory functions such as the boundary decision functions.

This might explain why many researchers now study model-based techniques, where the neural network architectures are established by unfolding known decoding algorithms and where the weights are initialized based on these algorithms [NBB16]. Learning is then used to explore the functions in the function class of the neural network that are not “too far” from the initial point in the optimization space. Nevertheless, the initial point should already be of good quality to get satisfactory performance and learning amounts to fine tuning the algorithm.

To conclude, we believe that current learning techniques are not able to learn efficient decoding paradigm, such as the “combining paradigm on a tree”. As a result, unless an already existing efficient algorithm is embeeded in the neural network architecture, current neural networks cannot learn to operate near the Poltyrev/Shannon limit due to the curse of dimensionality. A new result in this direction would be a real breakthrough.

Bibliography

[ABF$^+$20] M. Albrecht, S. Bai, P-A. Fouque, P. Kirchner, D. Stehlé, and W. Wen. Faster enumeration-based lattice reduction: Root hermite factor $k^{1/(2k)}$ in time $k^{k/8+o(k)}$. *CRYPTO*, 2020.

[ABMM18] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee. Understanding deep neural networks with rectified linear units. *International Conference on Learning Representations*, 2018.

[ABV$^+$94] O. Amrani, Y. Be'ery, A. Vardy, F.-W. Sun, and H. C. A. van Tilborg. The Leech lattice and the Golay code: bounded-distance decoding and multilevel constructions. *IEEE Transactions on Information Theory*, 40(4):1030–1043, Jul. 1994.

[AEVZ02] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger. Closest point search in lattices. *IEEE Transactions on Information Theory*, 48(8):2201–2214, Aug. 2002.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems. *Preliminary version in STOC*, 1996.

[AMT67] E. F. Assmus, H. F. Mattson, and R. J. Turyn. Research to Develop the Algebraic Theory of Codes. *Report AFCRL-67-0365 Air Force Cambridge Research Laboratories*, Jun. 1967.

[Ari09] E. Arikan. Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels. *IEEE Transactions on Information Theory*, 55(7):3051–3073, July 2009.

[Ari16] E. Arikan. On the Origin of Polar Coding. *IEEE Journal on Selected Areas in Communications*, 34(2):209–223, Feb. 2016.

[AV00] D. Agrawal and A. Vardy. Generalized minimum distance decoding in Euclidean space: performance analysis. *IEEE Transactions on Information Theory*, 46(1):60–83, Jan. 2000.

[Ber68] E. Berlekamp. Non-binary BCH decoding. *IEEE Transactions on Information Theory*, 14(2):242, Mar. 1968.

[BGT93] C. Berrou, A. Glavieu, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1. *IEEE International Conference on Communications*, 1993.

[BK98] A. Banihashemi and A. Khandani. On the complexity of decoding lattices using the Korkin-Zolotarev reduced basis. *IEEE Transactions on Information Theory*, 44(1):162–171, 1998.

[BLP$^+$13] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. *STOC*, 2013.

[Bou20] J. J. Boutros. Private communication. 2020.

[BRC60] R. Bose and D. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, 1960.

[BSC95] A. Bonnecaze, P. Sole, and A. R. Calderbank. Quaternary quadratic residue codes and unimodular lattices. *IEEE Transactions on Information Theory*, 41(2):366–377, 1995.

[BSS89] Y. Be'ery, B. Shahar, and J. Snyders. Fast decoding of the Leech lattice. *IEEE Journal on Selected Areas in Communications*, 7(6):959–967, Aug. 1989.

[BW59] E. S. Barnes and G. E. Wall. Some extreme forms defined in terms of abelian groups. *Journal of the Australian Mathematical Society*, 1(1):47–63, Aug. 1959.

[CBCB18a] V. Corlay, J.J. Boutros, P. Ciblat, and L. Brunel. Multilevel MIMO Detection with Deep Learning. *52th Asilomar Conference on Signals, Systems and Computers*, May 2018.

[CBCB18b] V. Corlay, J.J. Boutros, P. Ciblat, and L. Brunel. Neural Lattice Decoders. *6th IEEE Global Conference on Signal and Information Processing, also available at: arXiv preprint arXiv:1703.02930*, Dec. 2018.

[CF07] D. Costello and G. Forney. Channel coding: The road to channel capacity. *Proceedings of the IEEE*, 95(6):1150 – 1177, Jul. 2007.

[CKM$^+$17] H. Cohn, A. Kumar, S. D. Miller, D. Radchenko, and M. Viazovska. The sphere packing problem in dimension 24. *Annals of Mathematics*, 185(3):1017–1033, April 2017.

[CN11] Y. Chen and P. Nguyen. BKZ 2.0: Better Lattice Security Estimates. *ASIACRYPT 2011*, pages 1–20, 2011.

[Coh96] H. Cohen. *A course in computational algebraic number theory.* Springer-Verlag, New York, 3rd edition edition, 1996.

[Cox73] H. Coxeter. *Regular Polytopes.* 3rd edition, 1973.

[CS83] J. H. Conway and N. J. A. Sloane. A fast encoding method for lattice codes and quantizers. *IEEE Transactions on Information Theory*, 29(6):820–824, Nov. 1983.

[CS84] J. H. Conway and N. J. A. Sloane. On the Voronoi Regions of Certain Lattice. *SIAM Journal on Algebraic Discrete Methods*, 5(3):1984, 1984.

[CS86] J. H. Conway and N. J. A. Sloane. Soft decoding techniques for codes and lattices, including the Golay code and the Leech lattice. *IEEE Transactions on Information Theory*, 32(1):41–50, Jan. 1986.

[CS92] J. H. Conway and N. J. A. Sloane. Low-Dimensional Lattices. VI. Voronoi Reduction of Three-Dimensional Lattices. *Proceedings: Mathematical and Physical Sciences by the Royal Society*, pages 55–68, Jan. 1992.

[CS99] J. Conway and N. Sloane. *Sphere packings, lattices and groups.* Springer-Verlag, New York, 3rd edition edition, 1999.

[Dan17] A. Daniely. Depth Separation for Neural Networks. *34th Annual Conference on Learning Theory*, pages 690–696, 2017.

[DB04] A. Desideri-Bracco. Treillis de codes quasi-cycliques. *European Journal of Combinatorics*, 25(18):505–516, May. 2004.

[DBNS08] A. Desideri-Bracco, A-M. Natividad, and P. Solé. On quintic quasi-cyclic codes. *Discrete applied mathematics*, 156(18):3362–3375, Nov. 2008.

[DCB00] M. O. Damen, A. Chkeif, and J.-C. Belfiore. Lattice code decoder for space-time codes. *IEEE Communications Letters*, 4(6):161 – 163, 2000.

[deB75] R. deBuda. The upper error bound of a new near-optimal code. *IEEE Transactions on Information Theory*, 21(4):441–445, 1975.

[Des86] Roger Descombes. *Éléments de théorie des nombres.* PUF Mathématiques, 1986.

[DGC03] M. O. Damen, H. El Gamal, and G. Caire. On maximum-likelihood detection and the search for the closest lattice point. *IEEE Transactions on Information Theory*, 49(10):2389 – 2402, 2003.

[DP19] L. Ducas and C. Pierrot. Polynomial time bounded distance decoding near Minkowski's bound in discrete logarithm lattice. *Designs, Codes and Cryptogry*, 87:1737–1748, Aug. 2019.

[dPBZB12] N. di Pietro, J. J. Boutros, G. Zémor, and L. Brunel. Integer low-density lattices based on construction A. *IEEE Information Theory Workshop*, pages 422–426, Sep. 2012.

[DS06] I. Dumer and K. Shabunov. Soft-decision decoding of Reed-Muller codes: recursive lists. *IEEE Transactions on Information Theory*, 52(3):1260–1266, Mar. 2006.

[dt19] The FPLLL development team. *FPLLL, a lattice reduction library (provided with a Python interface)*, 2019.

[dZB18] N. di Pietro, G. Zémor, and J. J. Boutros. LDA Lattices Without Dithering Achieve Capacity on the Gaussian Channel. *IEEE Transactions on Information Theory*, 64(3):1561–1594, 2018.

[Ebe99] W. Ebeling. *Lattices and Codes.* Springer Spektrum, Wiesbaden, 3rd edition edition, 1999.

[Eli55] P. Elias. Coding for Noisy Channels. *Proceedings of the IRE*, 43:356–356, 1955.

[ES16] R. Eldan and O. Shamir. The power of depth for feedforward neural networks. *29th Annual Conference on Learning Theory*, page 907–940, 2016.

[ETV99] T. Etzion, A. Trachtenberg, and A. Vardy. Which codes have cycle-free Tanner graphs? *IEEE Transactions on Information Theory*, 45(6):2173–2181, 1999.

[EZ04] U. Erez and R. Zamir. Achieving 1/2 log (1+SNR) on the AWGN channel with lattice encoding and decoding. *IEEE Transactions on Information Theory*, 50(10):2293–2314, 2004.

[FL95] M. P. C. Fossorier and S. Lin. Soft-decision decoding of linear block codes based on ordered statistics. *IEEE Transactions on Information Theory*, 41(1379-1396):1379–1396, Sep. 1995.

[For88a] G. D. Forney. Coset codes. I. Introduction and geometrical classification. *IEEE Transactions on Information Theory*, 34(5):1123–1151, 1988.

[For88b] G. D. Forney. Coset codes. II. Binary lattices and related codes. *IEEE Transactions on Information Theory*, 34(5):1152–1187, 1988.

[For89a] G. D. Forney. A bounded-distance decoding algorithm for the Leech lattice, with generalizations. *IEEE Transactions on Information Theory*, 35(4):906–909, Jul. 1989.

[For89b] G. D. Forney. Multidimensional constellations. II. Voronoi constellations. *IEEE Journal on Selected Areas in Communications*, 7(6):941–958, Aug. 1989.

[FP85] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *ACM SIGSAM Bull.*, 44:463–471, April 1985.

[FTS00] G. D. Forney, M. D. Trott, and Sae-Young Chung. Sphere-bound-achieving coset codes and multilevel coset codes. *IEEE Transactions on Information Theory*, 46(3):820–850, 2000.

[FU98] G. D. Forney and G. Ungerboeck. Modulation and coding for linear Gaussian channels. *IEEE Transactions on Information Theory*, 44(6):2384–2415, Oct. 1998.

[FV96] G. D. Forney and A. Vardy. Generalized minimum-distance decoding of Euclidean-space codes and lattices. *IEEE Transactions on Information Theory*, 42(6):1992–2026, Nov. 1996.

[FZ99] A. J. Felstrom and K. S. Zigangirov. Time-varying periodic convolutional codes with low-density parity-check matrix. *IEEE Transactions on Information Theory*, 45(6):2181–2191, Sep. 1999.

[Gal62] R. Gallager. Low-Density Parity-Check Codes. *IRE Transactions on Information Theory*, 8(1):21–28, Jan. 1962.

[Gal68] R. Gallager. *Information Theory and Reliable Communication.* Wiley, New York, 1968.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* The MIT Press, 3rd edition edition, 2016.

[GCHtB17] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink. On deep learning-based channel decoding. *Conference on Information Sciences and Systems*, March 2017.

[GN08] N. Gama and P. Nguyen. Predicting lattice reduction. *EUROCRYPT 2008*, pages 31–51, April 2008.

[GNR10] N. Gama, P. Nguyen, and O. Regev. Lattice enumeration using extreme pruning. *EUROCRYPT 2010*, 2010.

[Gol49] M. Golay. Notes on Digital Coding. *Bell System Technical Journal*, 27, 1949.

[GP17] E. Grigorescu and C. Peikert. List-Decoding Barnes-Wall Lattices. *Computational complexity*, 26:365–392, Jun. 2017.

[Gri10] R. L. Griess. Rank 72 high minimum norm lattices. *Journal of Number Theory*, 130(7):1512–1519, Jul. 2010.

[GS99] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[Ham50] R. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2), 1950.

[Hoc59] A. Hocquenghem. Codes correcteurs d'erreurs. *Chiffres (Paris)*, 2(116):147–156, 1959.

[HRF14] J.R. Hershey, J. Le Roux, and F.Weninger. Deep Unfolding: Model-Based Inspiration of Novel Deep Architectures. *arXiv preprint arXiv:1409.2574*, Nov. 2014.

[HS10] G. Hanrot and D. Stehlé. A complete worst-case analysis of Kannan's shortest lattice vector algorithm. 2010.

[HV05] B. Hassibi and H. Vikalo. On the sphere-decoding algorithm I. Expected complexity. *IEEE Transactions on Signal Processing*, 53(8):2806–2818, 2005.

[HVB13] J. Harshan, E. Viterbo, and J. C. Belfiore. Practical Encoders and Decoders for Euclidean Codes from Barnes-Wall Lattices. *IEEE Transactions on Information Theory*, 61(11):4417–4427, Nov. 2013.

[HWJL20] H. He, C. Weny, S. Jin, and G. Ye Liz. A Model-Driven Deep Learning Network for MIMO Detection. *IEEE Transactions on Signal Processing*, 68:1702–1715, Feb. 2020.

[IH77] H. Imai and S. Hirakawa. A new multilevel coding method using error-correcting codes. *IEEE Transactions on Information Theory*, 23(3):371–377, 1977.

[ITW18] M. Imanishi, S. Takabe, and T. Wadayama. Deep Learning-aided iterative detector for massive overloaded MIMO channels. *arXiv preprint arXiv:1806.10827*, June 2018.

[IZF13] A. Ingber, R. Zamir, and M. Feder. Finite-Dimensional Infinite Constellations. *IEEE Transactions on Information Theory*, 59(3):1630–1656, Mar. 2013.

[JO05] J. Jalden and B. Ottersten. On the complexity of sphere decoding in digital communications. *IEEE Transactions on Signal Processing*, 53(4):1474–1484, 2005.

[Kan83] R. Kannan. Improved algorithms for integer programming and related lattice problems. *STOC*, pages 193–206, April 1983.

[KO14] W. Kositwattanarerk and F. Oggier. Connections between Construction D and related constructions of lattices. *Designs, Codes and Cryptography*, 73:441–455, 2014.

[KRU13] S. Kudekar, T. Richardson, and R. Urbanke. Spatially Coupled Ensembles Universally Achieve Capacity Under Belief Propagation. *IEEE Transactions on Information Theory*, 59(12):7761 – 7813, Dec. 2013.

[KSH12] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 25*, pages 1097–1105, 2012.

[Lee67] J. Leech. Notes on sphere packings. *Canadian Journal of Mathematics*, 19:251–257, 1967.

[Lin11] C. Ling. On the Proximity Factors of Lattice Reduction-Aided Decoding. *IEEE Transactions on Signal Processing*, 59(6):2795–2808, 2011.

[LL89] G. Lang and F. Longstaff. A Leech lattice modem. *IEEE Journal on Selected Areas in Communications*, 7(6):968–973, Aug. 1989.

[LL18] X. Liu and Y. Li. Deep MIMO Detection Based on Belief Propagation. *IEEE Information Theory Workshop (ITW)*, Nov. 2018.

[LLL82] A. Lenstra, H. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[LM82] J. Lepowsky and A. Meurman. An E8 approach to the Leech lattice and the Conway group. *Journal of Algebra*, 77(2):484–504, Aug. 1982.

[Loe97] H.-A. Loeliger. Averaging bounds for lattices and linear codes. *IEEE Transactions on Information Theory*, 43:1767–1773, Nov. 1997.

[LS01] S. Ling and P. Solé. On the algebraic structure of quasi-cyclic codes .I. Finite fields. *IEEE Transactions on Information Theory*, 47(7):2751–2760, Nov. 2001.

[LvdPdW12] T. Laarhoven, J. van de Pol, and B. de Weger. Solving Hard Lattice Problems and the Security of Lattice-Based Cryptosystems. *IACR Cryptol. ePrint Arch.*, 2012:533, 2012.

[Mar03] J. Martinet. *Perfect Lattices in Euclidean Spaces.* Springer, 2003.

[Mas69] J. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15(1):122–127, Jan. 1969.

[Mey13] A. Meyer. On the number of lattice points in a small sphere and a recursive lattice decoding algorithm. *Designs, Codes and Cryptography*, 66:375–390, Jan. 2013.

[MG02] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: A Cryptographic Perspective.* Springer-Verlag, Berlin, 2002.

[MG10] M.Schneider and N. Gama. *Darmstadt SVP Challenges*, 2010.

[MKO18] T. Matsumine, B. M. Kurkoski, and H. Ochiai. Construction D Lattice Decoding and Its Application to BCH Code Lattices. *2018 IEEE Global Communications Conference*, pages 1–6, Dec. 2018.

[MN96] D. MacKay and R. Neal. Near Shannon limit performance of low density parity check codes. *Electronics Letters*, 32(18):1645, Aug. 1996.

[MN08] D. Micciancio and A. Nicolosi. Efficient bounded distance decoders for Barnes-Wall lattices. *IEEE International Symposium on Information Theory*, pages 2484–2488, Jul. 2008.

[MO90] J. Mazo and A. Odlyzko. Lattice points in high dimensional spheres. *Monatsheft Mathematik*, 17:47–61, 1990.

[MPCB14] G. Montùfar, R. Pascanu, K. Cho, and Y. Bengio. On the Number of Linear Regions of Deep Neural Networks. *Advances in neural information processing systems*, pages 2924–2932, 2014.

[MR09] D. Micciancio and O. Regev. Lattice-based Cryptography. *Post-Quantum Cryptography*, pages 147–191, 2009.

[MS77] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes.* North-Holland, Amsterdam, The Netherland, 1977.

[Mul54] D. Muller. Application of Boolean algebra to switching circuit design and to error detection. *Transactions of the I.R.E. Professional Group on Electronic Computers*, 3:6–12, 1954.

[NBB16] E. Nachmani, Y. Be'ery, and D. Burshtein. Learning to decode linear codes using deep learning. *Annual Allerton Conference on Communication, Control, and Computing*, pages 341–346, Sept. 2016.

[Neb10] G. Nebe. A generalisation of Turyn's construction of self-dual codes. *RIMS workshop: Research into vertex operator algebras, finite groups and combinatorics*, pages 51–59, Dec. 2010.

[Neb12] G. Nebe. An even unimodular 72-dimensional lattice of minimum 8. *Journal für die reine und angewandte Mathematik*, 2012(673):237–247, Dec 2012.

[NP13] G. Nebe and R. Parker. On extremal even unimodular 72-dimensional lattices. *Mathematics of Computation*, 83(287):1489–1494, Jul. 2013.

[OH17] T. O'Shea and J. Hoydis. An introduction to deep learning for the physical layer. *IEEE Trans. on Cognitive Communications and Networking*, 3(4):563–575, Dec. 2017.

[Pei09] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. *STOC*, pages 333–342, 2009.

[Plo60] M. Plotkin. Binary codes with specified minimum distance. *IEEE Transactions on Information Theory*, 6(4):445–450, Sep. 1960.

[PMB13] R. Pascanu, G. Montufar, and Y. Bengio. On the number of inference regions of deep feed forward with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*, 2013.

[PMR$^+$17] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao. Why and When Can Deep – but Not Shallow – Networks Avoid the Curse of Dimensionality: a Review. *Center for Brains, Minds and Machines (CBMM) Memo No. 58*, 2017.

[Poh81] M. Pohst. On the computation of lattice vectors of minimal length. *ACM SIGSAM Bull.*, 15:37–44, 1981.

[PPV10] Y. Polyanskiy, V. Poor, and S. Verdu. Channel Coding Rate in the Finite Blocklength Regime. *IEEE Transactions on Information Theory*, 56(5):2307 – 2359, May 2010.

[PS08] J. Proakis and M. Salehi. *Digital Communications.* McGraw Hill, 5th edition edition, 2008.

[PV18] P. Petersen and F. Voigtlaender. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Networks, Elsevier*, 108:296–330, Dec. 2018.

[Que84] H.-G. Quebbemann. A construction of integral lattices. *Mathematika*, 31(1):137–140, Jun. 1984.

[Ree54] I. Reed. A Class of Multiple-Error-Correcting Codes and the Decoding Scheme. 1954.

[Reg05] O. Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *STOC*, 2005.

[Reg09] O. Regev. *Lecture notes*, 2009.

[RPK$^+$16] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein. On the expressive power of deep neural networks. *arXiv preprint arXiv:1606.05336*, June 2016.

[RS60] I. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[SA06] A. J. Salomon and O. Amrani. Encoding and Decoding Binary Product Lattices. *IEEE Transactions on Information Theory*, 52(12):5485–5495, Dec. 2006.

[SB95] G. Schnabl and M. Bossert. Soft-decision decoding of Reed-Muller codes as generalized multiple concatenated codes. *IEEE Transactions on Information Theory*, 41(1):304–308, Jan. 1995.

[Sch87] C.-P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53(2):201–224, 1987.

[SDW17] N. Samuel, T. Diskin, and A. Wiesel. Deep MIMO detection. *arXiv preprint arXiv:1706.01151*, June 2017.

[SDW18] N. Samuel, T. Diskin, and A. Wiesel. Learning to Detect. *arXiv preprint arXiv:1805.0763*, May 2018.

[SE94] C.-P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program*, 66:181–199, 1994.

[SFS08] N. Sommer, M. Feder, and O. Shalvi. Low-Density Lattice Codes. *IEEE Transactions on Information Theory*, 54(4):1561–1585, Apr. 2008.

[Sha48] C. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423, 1948.

[SI17] E. Strey and S. I.Costa. Lattices from Codes over Zq: Generalization of Constructions D, D' and D. *Des. Codes Cryptography*, 85(1):77–95, Oct. 2017.

[SM13] L. Szymanski and B. McCane. Learning in deep architectures with folding transformations. *International Joint Conference on Neural Networks*, pages 1–8, 2013.

[SS17] I. Safran and O. Shamir. Depth-Width Tradeoffs in Approximating Natural Functions with Neural Networks. *34th Annual Conference on Learning Theory*, pages 2979–2987, 2017.

[SSBD14] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, 2014.

[SSP11] A. Sakzad, M. Sadeghi, and D. Panario. Turbo Lattices: Construction and Performance Analysis. *Available: https://arxiv.org/abs/1108.1873*, Aug. 2011.

[Sud97] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, Apr. 1997.

[Tel16] M. Telgarsky. Benefits of depth in neural networks. *29th Annual Conference on Learning Theory*, page 1517–1539, 2016.

[Tho83] T. Thompson. *From error-correcting codes though sphere packings to simple groups.* Mathematical Association of America, 1983.

[Tit80] J. Tits. Four presentations of Leech's lattice. *Finite simple groups, II, Proceedings of a London Mathematical Society Researsch Symposium*, pages 306–307, 1980.

[TV15] I. Tal and A. Vardy. List Decoding of Polar Codes. *IEEE Transactions on Information Theory*, 61(5):2213–2226, Mar. 2015.

[TVZ99] V. Tarokh, A. Vardy, and K. Zegerr. Universal bound on the performance of lattice codes. *2013 IEEE International Symposium on Information Theory*, 45(2):670–681, March 1999.

[TXB$^+$18] X. Tan, W. Xu, Y. Be'ery, Z. Zhang, X. You, and C. Zhang. Improving Massive MIMO Belief Propagation Detector with Deep Neural Network. *arXiv preprint arXiv:1804.01002*, Apr. 2018.

[UR08] R. Urbanke and T. Richardson. *Modern Coding Theory.* Cambridge University Press, 2008.

[Var95] A. Vardy. Even more efficient bounded-distance decoding of the hexacode, the Golay code, and the Leech lattice. *IEEE Transactions on Information Theory*, 41(5):1495–1499, Sep. 1995.

[VB93] A. Vardy and Y. Be'ery. Maximum likelihood decoding of the Leech lattice. *IEEE Transactions on Information Theory*, 39(4):1435–1444, Jul. 1993.

[VB99] E. Viterbo and J. Boutros. A universal lattice code decoder for fading channels. *IEEE Transactions on Information Theory*, 45(5):1639–1642, July 1999.

[Via17] M. Viazovska. The sphere packing problem in dimension 8. *Annals of mathematics*, 185(3):991–1015, 2017.

[Vit67] A. Viterbi. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[WABM16] J. Wonterghem, A. Alloum, J Boutros, and M. Moeneclaey. Performance comparison of short-length error-correcting codes. *Symposium on Communications and Vehicular Technologies*, 2016.

[WFH99] U. Wachsmann, R. Fischer, and J. Huber. Multilevel codes: Theoretical concepts and practical design rules. *IEEE Transactions on Information Theory*, 45:1361–1391, July 1999.

[WH95] U. Wachsmann and J. Huber. Power and bandwidth efficient digital communication using turbo codes in multilevel codes. *Euro. Trans. Telecomm.*, 6:557–567, Sept. 1995.

[Woz57] J. Wozencraft. Sequential Decoding for Reliable Communication. *National IRE Convention Record*, 5(2):11–25, 1957.

[YLW13] Y. Yan, C. Ling, and X. Wu. Polar lattices: Where Arıkan meets Forney. *2013 IEEE International Symposium on Information Theory, Istanbul, Turkey*, pages 1292–1296, Jul. 2013.

[Zam14] R. Zamir. *Lattice Coding for Signals and Networks: A Structured Coding Approach to Quantization, Modulation and Multiuser Information Theory.* Cambridge University Press, 2014.

Titre: Algorithmes de décodage pour les réseaux de points

Mots clés: Réseaux de points; algorithmes de décodage; codes sur les groupes; réseaux de neurones.

Résumé: Cette thèse aborde deux problèmes liés aux réseaux de points, un vieux problème et un nouveau. Tous deux sont des problèmes de décodage de réseaux de points : À savoir, étant donné un point dans l'espace, trouver le point du réseau le plus proche.
Le premier problème est lié au codage de canal en dimensions intermédiaires. Alors que des systèmes efficaces basés sur les réseaux de points existent dans les petites dimensions $n \leq 30$ et les grandes dimensions $n \geq 1000$, ce n'est pas le cas des dimensions intermédiaires. Nous étudions le décodage de réseaux de points intéressants dans ces dimensions intermédiaires. Nous introduisons de nouvelles familles de réseaux de points obtenues en appliquant le contrôle de parité de manière récursive. Ces familles comprennent des réseaux de points célèbres, tels que les réseaux Barnes-Wall, les réseaux Leech et Nebe, ainsi que de nouveaux réseaux de parité. Nous montrons que tous ces réseaux de points peuvent être décodés efficacement avec un nouveau décodeur récursif par liste.
Le deuxième problème concerne les réseaux de neurones. Depuis 2016, d'innombrables articles ont tenté d'utiliser l'apprentissage profond pour résoudre le problème de décodage/détection rencontré dans les communications numériques. Nous proposons d'étudier la complexité du problème que les réseaux de neurones doivent résoudre. Nous introduisons une nouvelle approche du problème de décodage afin de l'adapter aux opérations effectuées par un réseau de neurones. Cela permet de mieux comprendre ce qu'un réseau de neurones peut et ne peut pas faire dans le cadre de ce problème, et d'obtenir des indications concernant la meilleure architecture du réseau de neurones. Des simulations informatiques validant notre analyse sont fournies.

Title: Decoding Algorithms for Lattices

Keywords: Lattice; decoding algorithm; group code; neural network.

Abstract: This thesis discusses two problems related to lattices, an old problem and a new one. Both of them are lattice decoding problems: Namely, given a point in the space, find the closest lattice point.
The first problem is related to channel coding in moderate dimensions. While efficient lattice schemes exist in low dimensions $n \leq 30$ and high dimensions $n \geq 1000$, this is not the case of intermediate dimensions. We investigate the decoding of interesting lattices in these intermediate dimensions. We introduce new families of lattices obtained by recursively applying parity checks. These families include famous lattices, such as Barnes-Wall lattices, the Leech and Nebe lattices, as well as new parity lattices. We show that all these lattices can be efficiently decoded with an original recursive list decoder.
The second problem involves neural networks. Since 2016 countless papers tried to use deep learning to solve the decoding/detection problem encountered in digital comunications. We propose to investigate the complexity of the problem that neural networks should solve. We introduce a new approach to the lattice decoding problem to fit the operations performed by a neural network. This enables to better understand what a neural network can and cannot do in the scope of this problem, and get hints regarding the best architecture of the neural network. Some computer simulations validating our analysis are provided.

Institut Polytechnique de Paris
91120 Palaiseau, France