CONTRIBUTIONS TO THE CONSTRUCTION AND LOW-COMPLEXITY DECODING OF ERROR CORRECTING CODES BASED ON OPTIMIZATION ALGORITHMS

Résumé : Les codes à Logique Majoritaire (MLGD) représentent une classe spéciale des codes correcteurs d'erreurs, satisfaisant les prérequis techniques de la couche physique des systèmes de communication sans fil modernes et futurs, ainsi que les systèmes de stockage d'information. Étant donné que la latence de codage/décodage et la complexité s'avèrent être les propriétés les plus intéressantes de ces codes, en plus d'une faible mémoire de stockage requise, nous mettons l'accent sur les avantages pertinents de ces codes, en plus de leurs algorithmes de décodage appropriés qui fournissent des performances compétitives par rapport à d'autres schémas de codage canal. Un autre avantage des codes MLGD s'agit de leurs techniques de construction, qui sont généralement basées sur des notions mathématiques liées aux designs combinatoires, l'algèbre des corps finis ainsi que la géométrie finie, ce qui en conséquence garantit plusieurs propriétés mathématiques et structurelles de ces codes, qui sont héritées des objets mathématiques dont la construction est basée.

Nous proposons une étude compréhensive, une analyse et classification des différentes techniques de construction associées à diverses familles de codes MLGD, ainsi qu'une étude extensive sur les algorithmes de décodage itératif capables d'exploiter la propriété d'orthogonalité qui caractérise ces codes. Nous soulignons nos contributions dans le cadre de la thématique de la construction combinatoire et le décodage itératif des codes à logique majoritaire décodables en une seule étape (OSMLD), et nous adressons également quelques applications de ces codes, notamment dans le standard des systèmes mobiles cellulaires de la 5ème génération (5G New Radio Standard), ainsi que dans les supports de stockage futurs à base de la molécule d'ADN synthétique. Nous faisons appel à des notions solides basées sur des revues de littérature des différents travaux de recherche réalisés auparavant, de la modélisation mathématique ainsi que des techniques d'optimisation locale et globale afin d'adresser différents défis et problématiques dans ces thématiques.

Plus précisément, nous proposons une classification générale des codes MLGD basés sur les designs combinatoires. Nous proposons également un nouvel algorithme de construction des codes OSMLD binaires et non-binaires à base des Algorithmes Génétiques. Dans la partie dédiée au décodage, nous proposons un nouvel algorithme de décodage des codes OSMLD basé sur le Gradient Descendant. En plus, nous introduisons une interprétation universelle des différents algorithmes de décodage à logique majoritaire en tant que processus d'optimisation de fonctions objectives dérivables. Nous adressons l'exploitation des techniques d'optimisation locales par contraintes afin de mettre en œuvre un nouvel algorithme de décodage pour les codes linéaires en bloc à base de la méthode des multiplicateurs du Lagrangien Augmenté. Pour la partie applications, nous proposons d'établir une comparaison des schémas de codage canal proposés et des codes LDPC standardisés pour la 5G NR eMBB. Nous présentons également un nouveau schéma de codage canal à base des codes DSC ternaires pour les systèmes de stockage à base de la molécule d'ADN synthétique.

Mots clés : Codes Correcteurs d'Erreurs, Décodage Itératif, Codes à Logique Majoritaire, Designs Combinatoires, Géométrie Finie, Algorithmes d'Optimisation, Complexité, 5G New Radio, Systèmes de Stockage de Données à base de l'ADN.

Abstract: Majority-Logic Decodable (MLGD) codes represent a special class of error correcting codes that meets the main requirements of the physical layer design of modern and future wireless communication and data storage systems. While the encoding and decoding latency and computational complexity are the most interesting features of these codes, in addition to low storage requirements, we emphasize the major key benefits of these codes as well as their appropriate iterative decoding algorithms that provide competitive performance compared to other channel coding schemes. Another advantage of MLGD codes is their construction techniques, usually derived from combinatorial designs, finite fields algebra and finite geometries, providing guaranteed mathematical structural properties of these codes inheriting from their respective construction techniques.

We propose a comprehensive study, analysis and classification of different construction techniques of various families of MLGD codes, as well as an extensive study on iterative decoding algorithms suitable for exploiting the orthogonality property of these codes. We highlight our contributions on the combinatorial construction and iterative decoding topics of One-Step Majority-Logic Decodable (OSMLD) codes, in addition to some featured applications of a set of these codes in the next generation 5G New Radio standards and future data storage systems based on the synthetic DNA molecule. We use a solid background related to extensive literature reviews, mathematical modeling, global and local optimization algorithms in order to address different open challenges within these topics.

More precisely, we propose first a general classification of MLGD codes based on Combinatorial Designs. Furthermore, a new construction algorithm of binary and non-binary OSMLD codes based on Genetic Algorithms is proposed. In the decoding topic, we propose a new Gradient-Descent based Majority-Logic Decoding algorithm for decoding OSMLD codes. Moreover, a new unified interpretation of various majority-logic decoders as a maximization process of derivable objective functions is introduced. Local constrained optimization techniques are exploited for devising a new decoding algorithm for linear codes based on the Augmented Lagrangian Method of Multipliers. For addressing different applications, we propose to establish a comparison of the proposed channel coding schemes with the standardized LDPC codes for the 5G NR eMBB use case. In addition, a new forward error correction scheme based on ternary Difference-Set codes is proposed for DNA data storage systems.

Keywords: Error Correcting Codes, Iterative Decoding, Majority-Logic Decodable Codes, Combinatorial Designs, Finite Geometry, Optimization Algorithms, Complexity, 5G New Radio, DNA Data Storage Systems.

Anouar YATRIBI

Contributions to the Construction and Low-Complexity Decoding of Error Correcting Codes based on Optimization Algorithms

Année : 2021 N° thèse : 210/ST2I

Année : 2021 **Thèse N° : 210/ST2I**

جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat

École Nationale Supérieure d'Informatique et d'Analyse des Systèmes
Centre d'Études Doctorales en Sciences des Technologies de l'Information et de l'Ingénieur

THÈSE DE DOCTORAT

CONTRIBUTIONS TO THE CONSTRUCTION AND LOW-COMPLEXITY DECODING OF ERROR CORRECTING CODES BASED ON OPTIMIZATION ALGORITHMS

Présentée par

Anouar YATRIBI

Le 19/05/2021

Formation doctorale : Informatique
Structure de recherche : Équipe Information, Communication and Embedded Systems (ICES)

JURY

Professeur Mohamed ESSAAIDI PES, ENSIAS, Université Mohammed V de Rabat	**Président**
Professeur Mostafa BELKASMI PES, ENSIAS, Université Mohammed V de Rabat	**Directeur de thèse**
Professeur Fouad AYOUB PH, CRMEF, Kénitra	**Co-Encadrant de thèse**
Professeur Mohammed BENATTOU PES, Faculté des Sciences, Université Ibn Tofail, Kénitra	**Rapporteur**
Professeur Mustapha BENJILLALI PES, INPT, Rabat	**Rapporteur**
Professeur Abdellatif KOBBANE PES, ENSIAS, Université Mohammed V de Rabat	**Rapporteur**
Professeur Abderrazak FARCHANE PH, Faculté Polydisciplinaire, Université Sultan Moulay Slimane, Beni Mellal	**Examinateur**
Professeur Samir MBARKI PES, Faculté des Sciences, Université Ibn Tofail, Kénitra	**Examinateur**

Acknowledgements

The realization of this thesis was not possible without my Professor and Supervisor Dr. Mostafa Belkasmi. Thank you very much for your high quality supervision, your continuous support, encouragement and help which made the realization of our ideas possible. Thank you for allowing me to participate in many organizing activities in international conferences, and for trusting in my skills and qualifications. I would like also to emphasize the benefits of the working sessions and reunions that you were organizing regularly along my thesis journey, in which an interesting flow of information and presentations helped me as well as all the team members in achieving our goals. Thank you for all your valuable time and your human qualities, I am very grateful. I could not have asked for a better supervision. Thank you Mr. Mostafa Belkasmi.

I would also like to thank my co-supervisor Pr. Fouad Ayoub, for his support and advises during all my thesis journey. Thank you for all the moments that we have shared together, for your human qualities, for your continuous supervising that helped me to make this work better, and for sharing with me many tools and papers that helped me to achieve my objectives. I am grateful for all your devoted efforts Mr. Fouad Ayoub.

I would like to acknowledge Pr. M. Essaidi for accepting to be a president of the honorable jury, as well as the reporters Pr. M. Benattou, Pr. M. Benjillali, Pr. A. Kobbane, and the examiners Pr. A. Farchane and Pr. S. Mbarki for providing their valuable remarks and feedback on this thesis.

I would also like to thank particularly the Professor Dr. Robert G. Maunder, from the University of Southampton, UK, for his collaboration, support and his continual interaction with me in order to give help and advises in my works, and for his valuable efforts for his reviewing and explanations that helped me a lot to understand the 5G NR LDPC scheme. I am very grateful for this support Mr. Rob Maunder.

Thanks to all my friends, and all my colleagues from the ICES team, and in particular my friend and colleague Zakaria M'rabet, who shared with me many unforgettable moments during all this journey, as well as during our Master degree. Thanks also to my friend and colleague Othmane El Mouaatamid, for our collaboration in many works.

Most of all, thanks to my lovely family, my dear father for his continuous support since my childhood, and for his rigorous scientific background and qualities along with his advises that helped me to achieve my goals. My lovely mother who supported me since my childhood, to whom I particularly dedicate this thesis, my lovely sister, and my dear brother, as well as my cousin. Your love and support makes everything seem easy for me.

Abstract

Majority-Logic Decodable (MLGD) codes represent a special class of error correcting codes that meets the main requirements of the physical layer design of modern and future wireless communication and data storage systems. While the encoding and decoding latency and computational complexity are the most interesting features of these codes, in addition to low storage requirements, we emphasize the major key benefits of these codes as well as their appropriate iterative decoding algorithms that provide competitive performance compared to other channel coding schemes. Another advantage of MLGD codes is their construction techniques, usually derived from combinatorial designs, finite fields algebra and finite geometries, providing guaranteed mathematical structural properties of these codes inheriting from their respective construction techniques.

We propose a comprehensive study, analysis and classification of different construction techniques of various families of MLGD codes, as well as an extensive study on iterative decoding algorithms suitable for exploiting the orthogonality property of these codes. We highlight our contributions on the combinatorial construction and iterative decoding topics of One-Step Majority-Logic Decodable (OSMLD) codes, in addition to some featured applications of a set of these codes in the next generation 5G New Radio standards and future data storage systems based on the synthetic DNA molecule. We use a solid background related to extensive literature reviews, mathematical modeling, global and local optimization algorithms in order to address different open challenges within these topics.

More precisely, we propose first a general classification of MLGD codes based on Combinatorial Designs. Furthermore, a new construction algorithm of binary and non-binary OSMLD codes based on Genetic Algorithms is proposed. In the decoding topic, we propose a new Gradient-Descent based Majority-Logic Decoding algorithm for decoding OSMLD codes. Moreover, a new unified interpretation of various majority-logic decoders as a maximization process of derivable objective functions is introduced. Local constrained optimization techniques are exploited for devising a new decoding algorithm for linear codes based on the Augmented Lagrangian Method of Multipliers. For addressing different applications, we propose to establish a comparison of the proposed channel coding schemes with the standardized LDPC codes for the 5G NR eMBB use case. In addition, a new forward error correction scheme based on ternary Difference-Set codes is proposed for DNA data storage systems.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Forward error correction is an essential key to the success of wireless communication and data storage systems. After 70 years from the pioneering work of Shannon on information and coding theory, extensive research studies and advancements are continuously driven towards achieving reliable communications over different transmission channels, while data reliability and reasonable hardware complexity remain the main motivating challenges for meeting the current industrial requirements. Low-Density Parity-Check (LDPC) codes have proven their ability to asymptotically achieve near capacity performance for very long block lengths. LDPC codes with structural properties have shown to be close to meet these requirements, especially when these codes have the orthogonality property in their dual structure used in the decoding process. Such codes of interest meeting those requirements are the Majority-Logic Decodable (MLGD) codes, which represent a special class of error correcting codes that provide good performance at the cost of very reasonable complexity, in both the encoding and the decoding processes. These codes are generally constructed from techniques derived from finite fields, finite geometries and combinatorial mathematics. Our interest in this thesis is related to the construction of different families of MLGD codes, as well as the investigation on devising suitable decoding algorithms satisfying a reasonable trade-off between performance and complexity.

1.1 Overview on Forward Error Correction

Error correction codes (ECC) have become a key enabler of reliable data transmissions in wireless communication and data storage systems. They are deployed in many current standards of telecommunications in order to guarantee error detection and correction. Addition-

ally, error correcting codes are playing an increasing key role in the analysis and the design of modern cryptosystems. Among various classes of codes, cyclic error correcting codes represent the most interesting family of codes, due to the simplicity of their encoding and decoding implementations that have shown to be the most suitable for modern applications. Encoding cyclic codes can be realized in the hardware using a simple linear feedback shift register. Moreover, decoding cyclic codes requires lower computational complexity than non cyclic codes. Powerful cyclic codes are derived from pseudo-random constructions. Also, cyclic algebraic codes that are derived from combinatorial and geometric constructions have shown to provide attractive performance with many guaranteed structural properties that inherit from their construction mathematical objects.

The fundamental performance limits of forward error correction were determined by Shannon in his 1948 paper [153]. Shannon proved that, by employing forward error correction, arbitrarily reliable communication is possible through channels which corrupt the data transmitted over them only if information is transmitted at a rate less than the capacity of the channel. Furthermore, if the rate of the transmitted data is less than the channel capacity, Shannon's theorem states that an error correction code and decoder must exist for which all the introduced errors during transmission can be corrected, providing a completely reliable communication over that channel.

However, the theorem of Shannon was not proved implicitly, leading the research communities to two main open challenges until now: devising powerful codes capable of achieving the theoretical limit along with suitable decoding algorithms for these codes, able to achieve the Shannon limit for various propagation channels.

After the Shannon's theorem, many decades have witnessed the benefits of error correction codes with algebraic structure to be appropriate for many decoding algorithms, exhibiting performances very close to the fundamental limits. Specifically, codes based on graphs have shown a particular interest from many research and industrial communities due to their hardware implementations satisfying many requirements of modern communication systems. For instance, LDPC codes represent the most interesting family that meets the fundamental requirements of many telecommunication industries, as these codes are very suitable for message-passing decoding algorithms that achieve robust performances, in addition to the low storage requirements and complexity of these codes provided by sparse parity-check matrices used for the decoding process.

LDPC codes were first discovered in 1962 by Gallager [48], right after the discovery of Golay codes [50] and Hamming codes [58]. Golay and Hamming codes have shown to

have interesting mathematical properties, used by many communities to understand how it is possible to design efficient error correcting codes. In contrast, LDPC codes provide many more important features and a design flexibility, leading the researchers to continuously investigate the design of powerful LDPC codes. In 1962, the hardware implementation of LDPC codes was impossible because of the unavailability of electronic techniques as well as the semi-conductor industry which was not yet sufficiently developed at that time. Consequently, before the Gallager discovery, in 1953, the first majority-logic decodable (MLGD) codes were discovered by Reed [133], well known as Reed-Muller codes. Also the first nontrivial majority-logic decoding method was introduced by the same author. After these works, in the same decade of the discovery of LDPC codes, algebraic Difference-Set codes were discovered independently by Rudolph in 1964 [145] and Weldon in 1966 [185]. Majority-logic decoding and threshold decoding were extensively studied by Massey [113] who proposed the first unified formulation of majority-logic decoding, as well as Kolesnik and Mironchikov [84]. Furthermore, the weighted one-step majority-logic decoding was introduced by Rudolph [142, 143]. These works were mainly motivated by the simplicity of the decoding process of these codes, requiring a simple majority-logic vote than can be realized by using only logical gates, and also motivated by the guaranteed properties that these codes have.

Many decades ago until the 1990s, many research efforts have been devoted for the construction of LDPC codes suitable for meeting the requirements of modern communication and data storage systems. Both random and algebraic techniques were investigated to devise a large database of LDPC codes. Convolutional codes have gained the attention of industrial communities in the 1990s due to their flexibility in selecting various information block lengths, a property that was very useful for its use in 2^{nd} generation (GSM) and the 3^{rd} generation of mobile networks (UMTS, HSPA, HSDPA). However, for achieving robust performance and achieving higher data rates and throughput, Turbo-codes were introduced by Berrou, Glavieux and Thitimajshima in 1993 [13], proving their capacity-approaching to the fundamental limits promised by Shannon. Turbo-codes are based on a parallel concatenation of elementary convolutional codes, separated by an interleaver. Decoding Turbo-codes requires a small number of iterations at a cost of a relatively high complexity in implementations. Besides that, Turbo-codes were adopted for the 4G Long Term Evolution (4G LTE) mobile networks as a standardized coding scheme, along with Hybrid Automatic Repeat Request (HARQ) protocols for the combination between forward error correction and retransmission, which led for the first time in high reliable transmissions of data over the air interface.

The introduction of Turbo-codes was a radical starter toward the re-evaluation of graph-based forward error correction, and how effective graph-based decoding algorithms must be devised. The pioneering works of Tanner in 1981 [165] on graph decoding of LDPC codes was reconsidered seriously with a large interest from many research communities. Particularly, it was shown that LDPC codes can be graphically represented by a Tanner Graph, and that powerful LDPC codes suitable for iterative decoding are those who have a small number of short cycles, ideally the shortest cycle must be greater than 4 (i.e. a girth[1] ≥ 6), and it is known that these LDPC codes that have this property are One-Step Majority-Logic Decodable (OSMLD) codes. Thereafter, the late 1990s was a remarkable period where LDPC codes were rediscovered by Mackay in 1996 [109], where their capacity-approaching performance was proved experimentally. Then, Richardson *et al.* have proved in 2001 the capacity-approaching of irregular LDPC codes [137], especially for asymptotically very large block lengths. Also, in the same year, finite geometry LDPC codes were rediscovered in [86], and the authors have shown that their performance are better compared to their randomly-generated counterparts, achieving near capacity performance. Directly after this contribution, MLGD codes have gained considerable interests from various research communities. Moreover, in 2000, it was shown in [106] that MLGD codes outperform many families of block codes when decoded with the Belief Propagation Sum-Product (BP-SP) algorithm.

In order to meet the requirements of modern communication systems, where the flexibility in block lengths and code rates is needed, a new concept of protograph (or multi-edge) LDPC codes was introduced independently in [166] and [138]. The main concept behind this new approach was the introduction of a small graph (protograph) used as a prototype that encapsulates the macroscopic properties of a large number of desired LDPC codes supporting a wide range of blocklengths and code rates. This graph is also called Base Graph. After that, many other publications were dealing with the construction of robust base graphs for generating protograph LDPC codes [33, 34, 104, 135, 166]. Note that in the 5G New Radio (5G NR) mobile networks, particularly in Enhanced Mobile Broadband (eMBB) scenarios, this class of codes was finally adopted by the 3GPP for data channels, where two different base graphs (BG1 and BG2) were adopted, providing a high flexibility in contrast to previous generations of mobile communication systems.

[1]The girth of a Tanner graph of a LDPC code is defined by the length of the shortest cycle contained in it.

1.2 Problem Statement and Motivation

Actually, achieving reliable communications over various propagation channels is a key requirement for modern standards of wireless communications. In contrast to previous generations, the optimization of the trade-off between performance and complexity is a main challenge to meet the requirements of these modern wireless communication systems. Additionally, decades of advancements in coding theory have shown that performance near the Shannon capacity can be achieved by LDPC codes when decoded with the BP-SP algorithm, and especially for very high information and code lengths. However, recently, many communication standards require transmissions of short to moderate block lengths, with a high reliability and low-complexity. Moreover, these energy-constrained systems require low storage requirements, due to their limited hardware memory.

LDPC codes with the row-column (RC) constraint in their associated Tanner graphs have shown to be the most suitable codes for modern communication systems, providing performances near the Shannon capacity. These codes are considered OSMLD codes, and can be decoded using simplest majority-logic decoding algorithms compared to the BP-SP algorithm that requires a relatively higher computational complexity. The construction of different MLGD codes is essentially derived from combinatorial and geometric constructions, which use deep notions of finite fields and combinatorial mathematics.

In addition, regular MLGD codes that have a cyclic structure represent the most appropriate coding schemes for energy-constrained communication systems and data storage systems, especially for short blocklengths. Also, these codes have shown to exhibit very low error floors given at a Bit Error Rate (BER) of 10^{-15}, which is moreover very useful for systems that operate in high Signal to Noise Ratios (SNRs). Consequently, the comprehensive understanding of different constructions of these codes may lead to a profound investigation on how is it possible to design powerful MLGD codes suitable for low-complexity systems.

The decoding of current LDPC codes uses the belief-propagation sum-product (BP-SP) algorithm, which requires relatively higher computational complexity, because of the mathematical calculations involved in the decoding process, that may be relatively high-cost for some particular communication systems. Majority-logic decoding algorithms represent a highly likable candidate for achieving reliable communications at the cost of low computational complexities. Furthermore, those codes with a cyclic structure are capable of providing lower complexities and low latency communications.

For instance, the latest standard of mobile wireless communication systems is the 5G NR

standard, which was partially completed and finished in June 2018. In contrast to previous generations of mobile networks, the 5G NR is a combination of many technologies and various use cases and seeks to connect the entire ecosystem by connecting and redefining various industries, ranging from mobile communications, immersive applications, vehicular communications to the internet of things (IoT) and industrial automation. The major expected applications of the 5G systems include: enhanced Mobile Broadband (eMBB), dedicated for ultra high capacity and high throughput applications, including mobile phone communications, high resolution (>4K) video streaming, mobile gaming and immersive extended reality applications (VR, AR), etc. Another use case includes Ultra Reliable Low-Latency Communications (URLLC), dedicated for mission critical machine to machine communications, vehicular communications (V2V, V2P, V2I, and in general cV2X) and remote surgeries in the e-Health industry, as well as many other use cases in various industries. The third expected application of the 5G NR is massive Machine-Type Communications (mMTC), dedicated for the IoT industry, smart homes, smart cities, smart farming, industrial IoT and other related machine to machine communications requiring short blocklengths and limited memory storage requirements.

The URLLC and mMTC scenarios in the 5G NR, and more especially the mMTC use case, require transmissions of short data blocks with a high reliability and very low complexity and latency. The key parameter to meet these requirements is the use of an optimal trade-off between performance and complexity providing reliable communications for these systems.

The main challenge and motivations behind this thesis is to emphasize the benefits of MLGD codes in these use cases of modern communication systems in the 5G NR standard, and to show that MLGD codes represent a promising candidate for meeting these industrial requirements. Also, it was shown in the literature that for short and moderate blocklengths, the irregularity is no longer advantageous for achieving good performance, while regular OSMLD codes are the most promising candidates for meeting these performance requirements. This motivates us in this thesis to investigate different constructions of MLGD codes and to establish an extensive state of art on the construction techniques, in order to enrich the code database of our research team. Also, our motivations are extended to review the existing decoding algorithms for these codes, that are already proposed in the literature, towards the proposition of new decoding algorithms that compete the previous works, as well as in order to result in a unified view and understanding of majority-logic decoding algorithms for improving the current state of art.

1.3 Thesis Aim, Objectives and Organization

The topic of this thesis is the investigation of the construction, iterative decoding and some applications of majority-logic decodable codes. These codes represent a particular class of error correcting codes, with guaranteed structural properties, providing the orthogonality in their corresponding dual structures. Based on the orthogonality property that these codes inherit from their associated construction methods, MLGD codes can be decoded using majority-logic decoding algorithms, requiring a relatively lower computational complexity.

In fact we are interested to emphasize the attractive advantages of MLGD codes for their eventual use in many standards of wireless communication systems, where both the reliability and low-complexity are a key requirement. Using majority-logic decoding, low decoding latency can be easily achieved, providing interesting features for modern and future wireless communication and data storage systems. For instance, as mentioned before, the mMTC use case in 5G mobile networks is a particular scenario where MLGD codes can compete well the existing channel coding techniques. Actually, investigations and studies on theses scenarios are still in progress in most wireless communication industries, and the standard remains incomplete, waiting for further technical contributions from various research communities over the world.

Consequently, we develop in this thesis different background and state of art on the combinatorial and algebraic construction of MLGD codes with various code lengths and rates. The main challenge is to obtain a large database for these codes with guaranteed mathematical properties, in order to evaluate their performances over various realistic scenarios, and to emphasize their benefits for the semi-conductor and telecommunications industries that require low-complexity and high performances. Besides wireless communication and data storage systems, another benefit of these codes consists of their proposition in new low-complexity and post-quantum cryptosystems in the IoT industry as well as for vehicular communications in the URLLC 5G NR use cases, as the security is actually a key challenge behind the commercialization and the roll-out of the complete standalone 5G NR standard.

In order to exploit the performances of the constructed MLGD codes, we are especially interested in establishing an extensive state of art on decoding MLGD codes, and devising new decoding techniques suitable to result in robust performances, while maintaining a reasonable computational complexity, for meeting the future requirements of wireless communication and data storage systems.

A brief introduction of the required background on coding and information theory, and the

historical evolution of the launched projects and research contributions on forward error correction, as well as the essential notions of cyclic codes were given above in 1.1. The remainder of this thesis is divided into three main parts. The first part, including Chapters 2 and 3, is dedicated to the combinatorial construction of various MLGD codes, where balanced incomplete block designs and other related combinatorial designs and algebraic techniques are used to derive various classes of MLGD codes. In the second part, which includes Chapters 4 and 5, we will investigate and analyze the iterative decoding of two distinct classes of MLGD codes, namely OSMLD codes and TSMLD codes, respectively. In the last part of this thesis, namely Chapter 6, we focus our interest to some applications of OSMLD codes in the 5G NR and synthetic DNA data storage systems.

More specifically the contents of the chapters of this thesis are organized as follows:

In Chapter 2, we address our interest to the classification of different families of MLGD codes, using backgrounds and mathematical techniques derived from combinatorial theory, finite fields, algebra and finite geometries. We will introduce the notions of balanced incomplete block designs and difference-sets, and we will explore various families of difference-sets, as well as their generalization to difference-families. Additionally, we will briefly present a literature review on finite geometry MLGD codes and their construction methods. This extensive literature review will result in a classification of different MLGD codes, with various structures, i.e. cyclic, quasi-cyclic and non cyclic MLGD codes.

In Chapter 3, we will focus our interest to the construction of cyclic OSMLD codes, derived from advanced notions of cyclotomic cosets and parity-check idempotents. We will present the mathematical constraints on how is it possible to construct feasible parity-check idempotents for generating cyclic OSMLD codes. We will also give a correspondence between the design of cyclic OSMLD codes and difference-sets and Golomb rulers. A literature review on the existing construction techniques already proposed previously is established, where the exhaustive search based construction is entirely reviewed, before establishing an analysis on the search space and investigating how it is possible to simplify the construction to obtain diverse code lengths (short, moderate and long) and code rates. Then we will introduce our proposed construction algorithms based on Genetic Algorithms, which aims to further simplify the construction complexity and to easily result in new cyclic OSMLD codes with attractive properties.

Chapter 4 is dedicated to the iterative decoding of OSMLD codes. We will give some preliminaries that will be useful along the chapter, before establishing a literature review on the existing majority-logic decoding algorithms. We will define each algorithm in order to pro-

vide a deep understanding of different decoding approaches, before giving a classification of these decoding algorithms. Then we will explore the previous works that use the Gradient-Descent technique for decoding error correcting codes, before introducing the proposed decoding algorithm (GD-MLGD) based on the gradient-descent optimization technique along with its quantized version that aims to further reduce the computational complexity. Performance analysis and comparisons with previous works will be addressed, in terms of the computational complexity, average number of decoding iterations and error rates. Finally an universal view of gradient-descent decoding algorithms is proposed based on the established literature review. Moreover, we explore other variants of the GD technique in order to result in enhanced update rules of the reviewed MLGD algorithms. Additionally, we extend our generalization to constrained optimization techniques, basically based on the Augmented Lagrange Method of Multipliers, and we propose a more generalized model suitable for decoding every linear block code characterized by a parity-check matrix, and not essentially containing orthogonal parity-check equations.

In Chapter 5, we propose a unified view of majority-logic decoding algorithms as an unconstrained minimization/maximization of a first-order derivable objective function with n variables, where n is the code-length. We show that for each MLGD algorithm, it is possible to associate a suitable objective function representing the decoding problem. Our study reveals that each derived objective function can be maximized using the Gradient-Descent (GD) optimization technique. The calculations of the corresponding first-order partial derivatives of these functions are investigated, and have shown to coincide with their corresponding soft-reliabilities. Moreover, we propose enhanced versions of these algorithms based on introducing new update rules derived from other variants of the Gradient-Descent. In addition, we extend our mathematical formulation to constrained optimization models, where we propose a new decoding algorithm, so-called the Augmented Lagrangian Multiplier Method Decoder (ALMM-D), derived from the ALMM constrained optimization technique. Our contribution aims to establish a solid mathematical foundation of the decoding problem, and to facilitate the task of devising a decoding algorithm suitable for decoding LDPC codes, especially those with no short cycles (considered as OSMLD codes) as well as other linear block codes, by adequately choosing an appropriate mathematical model, in which the involved complexity only depends on the arithmetical calculations of the first-order partial derivatives of the incorporated objective function which represents the decoding problem. Furthermore, the extension of these models as well as the employed optimization techniques for solving it depend on the use case, where many compromises between performance, complexity and latency can be

mathematically jointly optimized.

Chapter 6 is dedicated to the proposition of the use of OSMLD codes in two different applications: the 5G NR mMTC and URLLC applications, and DNA data storage systems. In the first part we will first present the benefit of OSMLD codes in the 5G NR as a competitors of other channel coding schemes, where some preliminary notions about the URLLC and mMTC scenarios are introduced, as well as the protograph LDPC codes already standardized for data channels in the 5G eMBB applications. Then we will present a brief literature review on the current advancement in the coding schemes already proposed for the URLLC and mMTC scenarios. This part will be concluded by performance comparisons of the proposed OSMLD codes with the previous propositions in the state of art. The second part of this chapter includes an application of cyclic ternary difference-set codes defined over $GF(3)$ in DNA data storage systems. We will present some preliminaries on the next generation DNA data storage technology. We will introduce DNA channels and DNA modulations in a information theory considerations, then we establish a literature review on the previous DNA forward encoding schemes, before introducing our proposed scheme based a double protection of DNA molecules based on cyclic ternary difference-sets. Performance evaluation of the proposed DNA encoding scheme along with a comparison with previous works will be finally exposed.

General concluding remarks and future research directions are presented in Chapter 7. Finally, Appendix A presents detailed definitions of some combinatorial designs and objects. Appendix B presents a non-exhaustive list of new constructed cyclic binary and non-binary OSMLD codes using the OSMLD-GA algorithm, while Appendix C explains the calculation of the analytical form of the error rate for the WOSMLGD algorithm, followed by a glossary and the bibliography.

1.4 Related publications

Journal papers

- Karim Rkizat, Anouar Yatribi, Mohammed Lahmer, and Mostafa Belkasmi. Iterative threshold decoding of high rates quasi-cyclic OSMLD codes. *International Journal of Advanced Computer Science and Applications*, 7, 2016.
- A. Yatribi, M. Belkasmi and F. Ayoub. Gradient-descent decoding of one-step majority-logic decodable codes. *Elsevier, Physical Communication* (2020), doi: `https://doi.`

org/10.1016/j.phycom.2019.100999.

- Anouar Yatribi, Mostafa Belkasmi and Fouad Ayoub. Unified Models for Majority-Logic Decoding of LDPC Codes using Local Optimization Techniques. *Accepted in the International Journal of Computer Science & Applications (IJCSA),* 2020.

- Anouar Yatribi, Mostafa Belkasmi, Fouad Ayoub and Robert G. Maunder. Deep Learning aided Iterative Threshold Decoding of Two-Step Majority-Logic Decodable Codes. *In preparation for submission to IEEE Transactions on Communications,* 2021.

Book chapters

- Anouar Yatribi, Mostafa Belkasmi, and Fouad Ayoub. An Efficient and Secure Forward Error Correcting Scheme for DNA Data Storage. In *International Conference on Soft Computing and Pattern Recognition*, pp. 226–237. Springer, 2018.

Conference papers

- • Anouar Yatribi, Fouad Ayoub, Ahmed Azouaoui, Mostafa Belkasmi. Comparison of Hybrid Automatic Repeat Request Protocols using Turbo Codes. In *NDENT'13*, 2013.

- Anouar Yatribi, Fouad Ayoub, Zakaria M'rabet, Ahmed Azouaoui, and Mostafa Belkasmi. Hybrid Automatic Repeat Request Protocols: Turbo-Codes against Cyclic Binary Low-Density Parity-Check Codes. In *2014 5th Workshop on Codes, Cryptography and Communication Systems (WCCCS)*, pages 86–91. IEEE, 2014.

- Anas Aboudeine, Fouad Ayoub, Anouar Yatribi, and Mohammed Benattou. Performance analysis of Coding using Distributed Turbo Codes. In 2*015 Third International Workshop on RFID And Adaptive Wireless Sensor Networks (RAWSN)*, pages 78–81. IEEE, 2015.

- Anouar Yatribi, Fouad Ayoub, and Mostafa Belkasmi. Construction of Cyclic One-Step Majority-Logic Decodable Codes using Genetic Algorithms. In *2015 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pages 1–6. IEEE, 2015.

- Anouar Yatribi, Fouad Ayoub, and Mostafa Belkasmi. An efficient FEC/ARQ scheme using Iterative Threshold Decoder. In *JDSIRT'15*, 20:2, 2015.

- Anouar Yatribi, Mostafa Belkasmi, Fouad Ayoub, and Zakaria M'rabet. Non-Binary Cyclic Majority-Logic Decodable Codes: An Algebraic Construction by using Genetic Algorithms. In *2016 International Conference on Advanced Communication Systems and Information Security (ACOSIS)*, pages 1–9. IEEE, 2016.

- Zakaria M'rabet, Fouad Ayoub, Mostafa Belkasmi, Anouar Yatribi, and Alaoui Ismaili Zine El Abidine. Non-Binary Euclidean Geometry Codes: Majority Logic Decoding. In *2016 International Conference on Advanced Communication Systems and Information Security (ACOSIS)*, pages 1–7. IEEE, 2016.

- Souad Labghough, Anouar Yatribi, Fouad Ayoub and Mostafa Belkasmi. CSOC Codes Performance over AWGN Channel. In *NDENT'17*, 2017.

- Otmane El Mouaatamid, Mohamed Lahmer, Mostafa Belkasmi, Zakaria M'rabet, and Anouar Yatribi. One-Step Majority-Logic Decodable Codes derived from Oval designs. In *2017 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pages 1–6. IEEE, 2017.

Chapter 2

Majority-Logic Decodable Codes

2.1 Introduction and Background

Low-density parity-check codes represent a class of linear block codes with implementable decoders, which provide near-capacity performance on a large set of data transmission and data storage channels. LDPC codes were invented by Gallager in his 1960 doctoral dissertation [48] and were mostly ignored during the 35 years that followed. One notable exception is the important work of Tanner in 1981 [165], in which Tanner generalized LDPC codes and introduced a graphical representation of these codes, now called a Tanner graph. The study of LDPC codes was resurrected in the mid 1990s, where it was noticed, independently of Gallager's work, the advantages of linear block codes with sparse (low-density) parity-check matrices.

LDPC codes are generally decoded using message-passing decoding algorithms. The most known decoding algorithms for these codes are the Belief-Propagation Sum-Product algorithm (BP-SPA), as well as it simplification known as belief-propagation Min-Sum (BP-MS). Other variants of the BP-MS algorithm were proposed in the literature in order to improve the performance. Other decoding techniques include the original bit-flipping algorithm [48] and many of its improved variants [20, 54, 72, 77, 86, 90, 99, 100, 103, 121, 122, 125, 159, 172, 184, 187, 188, 198, 199]. Later, due the works of Feldman in his doctoral dissertation in 2003 [44], a new approach for decoding LDPC codes was introduced, based on the use of constrained optimization techniques, namely Linear Programming. These works have motivated a lot of research groups to investigate other advanced constrained and convex optimization techniques for devising appropriate decoding algorithms for LDPC codes [9, 22, 45, 82, 101, 102, 162, 182, 183, 202] in various transmission channels.

We shall consider only binary LDPC codes for the sake of simplicity, although LDPC codes can be generalized to non-binary alphabets. A low-density parity-check code is a linear block code given by the null space of an $m \times n$ sparse (i.e. low density of 1s) parity-check matrix H. A regular LDPC code is a linear block code whose parity-check matrix H has column weight γ and row weight ρ, where $\rho = \gamma(n/m)$ and $\gamma \ll m$. If H is low density, but its row and column weight are not both constant, then the code is an *irregular LDPC code*. For reasons that will become apparent later, almost all LDPC code constructions impose the following additional structural property on H: no two rows (or two columns) have more than one position in common that contains a nonzero element. This property is called the *row–column constraint*, or simply, the RC constraint. When a LDPC code has the RC constraint satisfied, it can be considered as a *One-Step Majority-Logic Decodable (OSMLD) code*.

The descriptor "low density" is unavoidably vague and cannot be precisely quantified, although a density of 0.01 or lower can be called low density (1% or fewer of the entries of H are 1s). The density needs only be sufficiently low to permit effective iterative decoding. This is in fact the key innovation behind the invention of LDPC codes. It is well known that optimum (e.g., maximum-likelihood) decoding of the general linear block codes that is useful for applications is not possible due to the vast complexity involved. The low-density aspect of LDPC codes accommodates iterative decoding, which typically has near-maximum-likelihood performance at error rates of interest for many applications. The construction of LDPC codes usually involves the construction of H, which need not be full rank. In this case, the code rate r for a regular LDPC code is bounded as

$$r \geq 1 - \frac{m}{n} \tag{2.1}$$

with equality when H is full rank.

The most known constructions of powerful LDPC codes with the RC constraint are based on finite fields and finite geometries, including Euclidean and Projective geometries and other sub-geometries. Other constructions include those based on combinatorial designs, especially *Balanced Incomplete Block Designs (BIBDs)*, as will be presented in this chapter. There are also computer-generated LDPC codes, in which the construction is random, and these codes are referred to *random LDPC codes* [66, 167].

In this chapter, the notions of balanced incomplete block designs and difference-sets are introduced in Section 2.2, before reviewing the most known constructions of LDPC codes in Section 2.3, including computer-generated and algebraic LDPC codes. Section 2.4 reviews the constructions of cyclic, quasi cyclic and non structured OSMLD codes derived from various

classes of combinatorial designs. Multi-Step Majority-Logic Decodable (MSMLD) codes derived from Euclidean geometries are introduced in Section 2.5, followed by a classification summary of the correspondence between different families of MLGD codes and combinatorial designs. Finally, a summary of this chapter is presented in Section 2.7.

2.1.1 Definitions and Preliminaries

We shall begin by some background and definitions that will be used throughout the thesis. We will give the definitions that are specific to a particular problem in the section pertaining to that particular problem.

Definition 2.1 (Linear code)**.** Let $\mathbb{F}_q^n$ denotes an n-dimensional vector space over a finite field $\mathbb{F}_q$ of q elements. An $(n,k,d_{min})_q$ linear code $\mathscr{C}$ is a k-subset of $\mathbb{F}_q^n$, with a dimension k. Each vector of $\mathscr{C}$ is a n-dimensional vector of $\mathbb{F}_q^n$ denoted by $c = (c_0, c_1, ..., c_{n-1})$.

Note that there is q^k codewords in a code $\mathscr{C}$ defined over $GF(q)$. A codeword of $\mathscr{C}$ is obtained by linear combinations of other codewords of the same sub-space, and the component-wise sum of all codewords is an all zero vector, called the trivial codeword. Consequently the code $\mathscr{C}$ is said to be *linear*.

Throughout the thesis, if the subscript q is omitted, i.e. $\mathscr{C} = (n,k,d_{min})$, then $q = 2$ and $\mathscr{C}$ is a *binary* code.

Definition 2.2 (Code rate)**.** The code rate r of an $(n,k,d_{min})_q$ linear code $\mathscr{C}$ is defined by the ratio $r = \frac{k}{n}$.

Definition 2.3 (Hamming weight)**.** For a vector $v = (v_0, v_1, ..., v_{n-1}) \in \mathbb{F}_q^n$, the Hamming weight of v, denoted by $wt_H(v)$, is the number of non zero elements in the vector. That is

$$wt_H(v) = |\{v_j \neq 0 \,|\, 0 \leq j < n\}|$$

Definition 2.4 (Support)**.** The support **sup(v)** of a vector $v \in \mathbb{F}_q^n$, denotes a set of coordinates of v for which the value is non zero:

$$\text{sup(v)} = \{j \,|\, v_j \neq 0, 0 \leq j < n\}$$

Definition 2.5 (Hamming distance)**.** Let the vectors $u, v \in \mathbb{F}_q^n$, the Hamming distance of u and v, denoted by $d_H(u,v)$, represents the number of coordinates in which u and v are different,

$$d_H(u,v) = wt_H(u-v) = |\{(u_j - v_j) \neq 0 \,|\, 0 \leq j < n\}|$$

Definition 2.6 (Minimum Hamming distance)**.** The minimum Hamming distance of a code $\mathscr{C}$, denoted by d_{min}, is the smallest value of all Hamming distances between any two distinct codewords of $\mathscr{C}$. Because of the linearity property, d_{min} can also be defined as:

$$\begin{aligned} d_{min} &= \min\left\{d_H(c,c') \,|\, \text{for all } c,c' \in \mathscr{C}\right\} \\ &= \min\left\{wt_H(c) \,|\, \text{for all } c \in \mathscr{C}\right\} \end{aligned}$$

A code of Hamming distance d_{min} is capable of correcting $t = \left[\frac{d_{min}-1}{2}\right]$ symbol errors. The parameter t is the correction capacity of the code $\mathscr{C}$.

Definition 2.7 (Generator matrix)**.** An $(n,k,d_{min})_q$ linear code $\mathscr{C}$ has a generator matrix G with dimension $k \times n$, which contains k linearly independent codewords of $\mathscr{C}$. A codeword $c \in \mathscr{C}$ may be obtained by taking any linear row combination of G.

The matrix G can be transformed into a reduced-echelon or systematic form by elementary row operations and if necessary, some column permutations. In systematic form, the first k coordinates of G is an identity matrix. Hence, for an arbitrary information vector $u \in \mathbb{F}_q^k$, $c = uG$ and $c \in \mathscr{C}$ is a systematic codeword where the first k symbols are the information vector u and the remaining $n-k$ coordinates contain the redundant symbols.

Definition 2.8 (Parity-check matrix)**.** The parity-check matrix H of a code $\mathscr{C}$ is a $m \times n$ matrix, such that $m \geq n-k$, and it contains m linearly independent vectors of $\mathbb{F}_q^n$ so that $G.H^T = 0$. Equivalently, this implies $cH^T = 0$ for all $c \in \mathscr{C}$. Each row in the matrix H is called a **parity-check equation**.

Definition 2.9 (Dual code)**.** The dual code of an $(n,k,d_{min})_q$ linear code $\mathscr{C}$, denoted by $\mathscr{C}^\perp$, is an $(n,n-k,d'_{min})$ linear code where $cc^\perp = 0$ for all $c \in \mathscr{C}$ and all $c^\perp \in \mathscr{C}^\perp$.

While any linear row combination of G produces c, any linear row combination of H produces $c^\perp$. The matrices H and G are the generator and parity-check matrices of $\mathscr{C}^\perp$ respectively.

Definition 2.10 (Self-dual and self-orthogonal code)**.** An (n,k) code $\mathscr{C}$ is self-orthogonal if $\mathscr{C} \subset \mathscr{C}^\perp$ and self-dual if $\mathscr{C} = \mathscr{C}^\perp$. The length of a self-dual code is even and $n = 2k$. An $(n, \frac{n}{2})$ code $\mathscr{C}$ is formally self-dual if $\mathscr{C}$ and $\mathscr{C}^\perp$ have the same weight distribution.

Definition 2.11 (Syndrome)**.** Given a vector $z \in \mathbb{F}_q^n$, the vector $s(z) \in \mathbb{F}_q^m$ defined by

$$s(z) = z.H^T$$

is the syndrome of a code whose parity-check matrix is H. If the vector $z \in \mathscr{C}$, then $s(z) = 0$, otherwise at least one coordinate of $s(z)$ has a non zero value.

Definition 2.12 (Hamming weight enumerator polynomial and weight distribution)**.** Given an $(n,k,d_{min})_q$ linear code $\mathscr{C}$, let $A_i = |\{wt_H(c) = i \,|\, \forall c \in \mathscr{C}\}|$, i.e. the number of codewords of Hamming weight i. The Hamming weight enumerator polynomial of $\mathscr{C}$ is given by

$$A_{\mathscr{C}}(z) = \sum_{i=0}^{n} A_i z^i$$

where z is an indeterminate. The distribution of A_i for $0 \leq i \leq n$ is known as the weight distribution of $\mathscr{C}$.

Theorem 2.1 (MacWilliams identity)**.** *Let $\mathscr{C}$ be an (n,k) code over $\mathbb{F}_q$ with weight enumerator $A(x)$, and let $B(x)$ be the weight enumerator of $C^{\perp}$. Then*

$$q^k B(x) = (1+(q-1)x)^n A\left(\frac{1-x}{1+(q-1)x}\right). \tag{2.2}$$

Definition 2.13 (Equivalence and automorphism of codes)**.** Two codes are equivalent if one of the codes can be obtained from the other by permuting the coordinates (in all codewords) and permuting the symbols within one or more coordinate positions.

Two codes are *isomorphic* if they differ by a permutation of the coordinates only. An *automorphism* of a code is any equivalence of the code with itself [27].

Theorem 2.2 (sphere packing or Hamming bound)**.** *A q-ary code $\mathscr{C}$ of length n and minimum distance $d = 2t+1$ satisfies*

$$|\mathscr{C}| \leq \frac{q^n}{\sum_{i=0}^{t} \binom{n}{i} (q-1)^i}. \tag{2.3}$$

A code that achieves the sphere packing bound is a perfect code. Equivalently, a code is perfect if the whole space is covered by the spheres of radius t around all codewords. [27].

Definition 2.14 (Union Bound Probability)**.** For a code with rate $r = \frac{k}{n}$ and a minimum distance d_{min}, the bit error probability P_b for a transmission over a noisy channel with signal to noise ratio $\frac{E_b}{N_0}$ and noise variance $\sigma^2 = \frac{N_0}{2}$ is given by:

$$P_b = \sum_i A_i Q(\sqrt{2 r d_i E_b / N_0}) \tag{2.4}$$

where

$$Q(x) = 0.5erfc(\frac{x}{\sqrt{2}}) \tag{2.5}$$

and

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-t^2} dt \tag{2.6}$$

where A_i is the Hamming weight of a codeword c_i, E_b is the energy per bit, N_0 is the noise spectral density and erfc(.) is the complementary error function given by (1.5).

The error probability P_b can be approximated as the probability of error associated with constellations at the minimum distance d_{min} multiplied by the number of neighbors at this distance, denoted by $K_{d_{min}}$. This approximation is the union bound, which becomes increasingly tight as the SNR increases, and takes the following form:

$$P_b \leq K_{d_{min}} Q\left(\frac{d_{min}}{2\sigma}\right) \tag{2.7}$$

Throughout the thesis, the considered transmission flow is illustrated in Figure 1.1. The source encoder transforms the source output into a binary sequence u of size k so that the source information is compressed and can be reconstructed without ambiguity. Source coding is another topic out of the scope of this thesis. The channel encoder transforms the information sequence u into a codeword $c \in \mathbb{F}_q^n$, where q is alphabet size of the considered code. The channel encoding procedure is necessary for adding $n-k$ redundancy symbols in order to protect the transmitted signal from the channel noise perturbation. Before the transmission over the noisy channel, the encoded sequence is modulated into a waveform which is suitable for transmission (or recording in a storage medium). The modulated waveform is therefore transmitted over the propagation channel (or storage medium) and is corrupted by noise. There are many channel models in the literature, adequate for various wireless communication systems. Each of these channel models is subject to a different noise distribution. The demodulator processes each received signal and produces a discrete (quantized) or continuous (unquantized) output for the channel decoder. The channel decoder is responsible for detecting and correcting the channel errors, and then outputs an estimated binary information sequence $\hat{u}$. The decoding strategy depends on the considered family of error correcting codes and the decoding algorithm, as well as the noise characteristics of the channel. Ideally, the decoder outputs a replica of the information sequence u, although the noise may introduce some decoding errors. Finally, the source decoder transforms the estimated sequence $\hat{u}$ into an estimate of the source output and delivers it to the destination. In this thesis, we focus our major interest to the channel encoder and channel decoder blocks of this transmission chain.

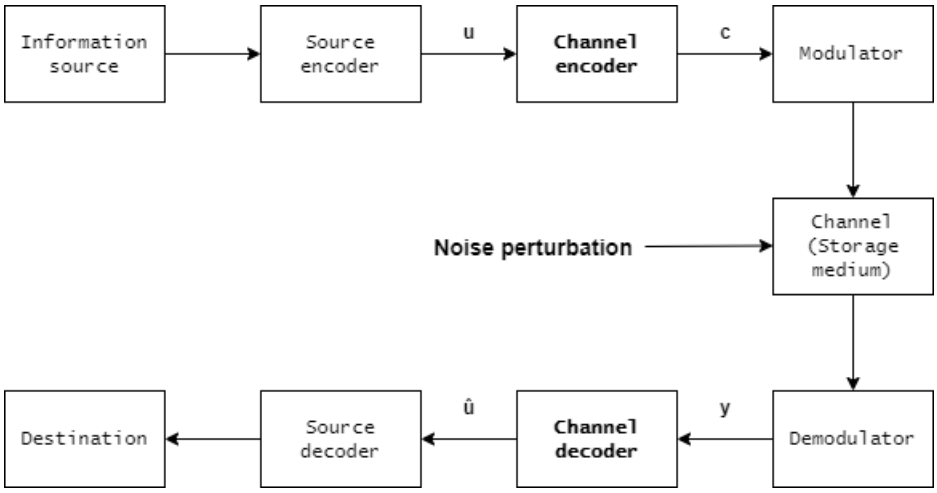

Figure 2.1: Block diagram of a data transmission (or storage system)

2.1.2 Balanced Incomplete Block Designs

As mentioned in the introduction of this chapter, many interesting construction methods of LDPC codes and MLGD codes are derived from special classes of combinatorial designs. The mot important notion of these designs includes Balanced Incomplete Block Designs (BIBDs) as well as other related designs [27, 57]. BIBDs represent a family of combinatorial designs, widely used in statistical and control applications. They represent particular objects with incidence properties that controls the occurrence of each object into blocks. The correspondence between BIBDs and error correcting codes is motivated by the following facts:

- The incidence matrix of a combinatorial design is the equivalent of a parity-check matrix of a regular LDPC code.
- Both are sparse binary matrices with constant row and column weights.
- The design for LDPC codes with Tanner graphs free of 4-cycles (MLGD codes) is analogous to a well studied problem in combinatorics, that of constructing **Steiner 2-designs**.

The next two definitions describe the notions t-designs and BIBDs, followed by other definitions that characterize their existence and properties.

Definition 2.15 (t-(v,k,λ) designs)**.** A t-(v,k,λ) design, is a pair $(\mathscr{X},\mathscr{B})$ where $\mathscr{X}$ is a v-set of points and $\mathscr{B}$ is a collection of k-subsets of $\mathscr{X}$ (blocks) with the property that every t-subset of $\mathscr{X}$ is contained in exactly λ blocks. The parameter λ is the index of the design.

Remark 2.1. When $t = 2$, a 2-(v,k,λ) design, or a (v,k,λ) design in short is a balanced incomplete block design (BIBD).

Definition 2.16 (Balanced Incomplete Block Designs)**.** A Balanced Incomplete Block Design (BIBD) is an arrangement of v distinct objects into b blocks such that each block contains exactly k distinct objects, each object occurs in exactly r different blocks, and every t distinct objects (a_i, a_j) occurs together in exactly λ blocks.

An (v,b,r,k,λ)-BIBD is a collection V and B of objects and blocks with a relation of incidence indicating which objects belong to which blocks. We will only consider the case where $t = 2$, thus the parameter t will be omitted.

Example 2.1. Let $\mathscr{X} = \{1,...,6\}$. Let $\mathscr{B} = \{124,126,134,135,156,235,236,245,346,456\}$. Then $(\mathscr{X},\mathscr{B})$ is a 2-$(6,3,2)$ design.

Theorem 2.3 below was proved by Assmus and Mattson in 1969, and provides an important method for the construction of t-designs as support designs in linear codes with relatively few nonzero weights [27].

Theorem 2.3 (Assmus–Mattson theorem)**.** *Let $\mathscr{C}$ be an (n,k,d_{min}) linear code over $\mathbb{F}_q$ and $\mathscr{C}^\perp$ be the dual $(n,n-k,d^\perp_{min})$ code. Denote by n_0 the largest integer $\leq n$ such that $n_0 - \frac{n_0+q-2}{q-1} < d_{min}$, and define $n_0^\perp$ similarly for the dual code $\mathscr{C}^\perp$. Suppose that for some integer t, $0 < t < d_{min}$, there are at most $d-t$ nonzero weights ω in $\mathscr{C}^\perp$ such that $\omega \leq n-t$. Then, the support design:*

1. *for any weight u, $d \leq u \leq n_0$ in $\mathscr{C}$ is a t-design;*
2. *for any weight ω, $d^\perp_{min} \leq \omega \leq \min\{n-t, n_0^\perp\}$ in $\mathscr{C}^\perp$ is a t-design.*

Remark 2.2. The largest value of t for any known t-design derived from a code via the Assmus–Mattson Theorem is $t = 5$. All such 5-designs come from self-dual codes.

The properties of a BIBD are characterized by its corresponding incidence matrix, along with the incidence properties defined in the next definition.

Definition 2.17. The incidence matrix of a BIBD with parameters v,b,r,k,λ is a $v \times b$ matrix $A = (a_{ij})$, in which $a_{ij} = 1$ when the i^{th} element of V occurs in the j^{th} block of B, and $a_{ij} = 0$ otherwise. The incidence matrix of a BIBD is governed by the following two equalities:

$$bk = vr \tag{2.8}$$

$$r(k-1) = \lambda(v-1) \tag{2.9}$$

where the incidence property 2.8 indicates that each of the b blocks are containing k objects and each of the v objects being contained in r blocks. The property 2.9 means that each object occurs in r blocks, and in each of these is a pair with the $(k-1)$ remaining objects, while on the other hand, a_1 is paired λ times with each of the remaining $(v-1)$ objects.

Given a 2-(v,k,λ)-design, its corresponding incidence matrix A must satisfy the equation stated in Theorem 2.4 below [57].

Theorem 2.4. *Let A be the incidence matrix of the (v,k,λ)-design. Then A is a binary matrix of order v that satisfies the matrix equation*

$$AA^T = (k-\lambda)I + \lambda J \tag{2.10}$$

where A^T denotes the transpose of A, I is the identity matrix of order v, and J is the matrix of 1's of order v.

A particular case of BIBDs, is when the number of blocks equals the number of objects. In this case, the design is called a *symmetric BIBD* [27, 57].

Definition 2.18 (Symmetric BIBD)**.** A (v,b,r,k,λ)- design is a symmetric BIBD if $v = b$, or equivalently $r = k$.

$$k(k-1) = \lambda(v-1) \tag{2.11}$$

In this case, the parameters r and b can be omitted, and the design is simply denoted as (v,k,λ).

The existence conditions of a BIBD are related to the Fisher's inequality [47] and the theorem of Bruck, Ryser and Chowla [23, 27, 57]. These two theorems are stated below (Theorems 2.5 and 2.6).

Theorem 2.5 (Fisher's inequality)**.** *If a BIBD (v,b,r,k,λ) exists with $2 \leq k \leq v$, then $b \geq v$.*

This theorem simply states that if a BIBD with parameters (v,b,r,k,λ) exists, then the number of blocks b must be greater or equal than the number of objects v.

The existence of a symmetric BIBD is governed by the Bruck, Ryser and Chowla theorem stated below.

Theorem 2.6 (Bruck, Ryser and Chowla Theorem)**.** *Let $n = k - \lambda$. If a symmetric block design exists with parameters v,k,λ, then:*

1. *If v is even, then n is a square.*
2. *If v is odd, then $z^2 = nx^2 + (-1)^{(v-1)/2}\lambda y^2$ has a solution in integers x,y,z not all zero.*

The proof of the Bruck-Ryser and Chowla theorem can be found in [147].

Now let's state a corollaire that establishes the correspondence between combinatorial designs and majority-logic codes [27].

Corollary 2.1. *A linear code whose dual code supports the blocks of a t-design admits one of the simplest decoding algorithms, majority decoding. Essentially, for each symbol y_j of the received codeword $y = (y_0,...,y_{n-1})$, a set of values $y_j^{(i)}, i = 1,...,r$ of certain linear functions defined by the blocks of the design are computed, and the **true** value of y_j is decided to be the one that appears most frequently among $y_j^{(1)},...,y_j^{(r)}$.*

2.1.3 Difference-Sets

When a (v,k,λ) design is a symmetric BIBD, then each block represents a difference-set. For the case of symmetric BIBDs, the construction of such designs only implies the construction of one block (difference-set), as the other blocks can be directly obtained by cyclically permuting the difference-set being constructed. There are many classes of difference-sets, and the understanding of the construction of those objects leads to the design of many classes of cyclic codes that have orthogonality properties that inherit from these designs. We will briefly describe a set of classes of difference-sets, and a classification with the correspondence between these combinatorial objects and codes will be given later in this chapter.

Definition 2.19. Let G be an additively written group of order v. A k-subset D of G is a $(v,k,\lambda;n)$-difference set of order $n = k - \lambda$ if every nonzero element of G has exactly λ representations as a difference $d_i - d_j\,(d_i,d_j \in D)$. The difference set is abelian, cyclic, etc., if the group G has the respective property. The redundant parameter n is sometimes omitted; therefore, the notion of (v,k,λ)-difference sets is also used.

Example 2.2. The set $D = \{0,1,3\}$ is an $(7,3,1)$-difference set in the group $\mathbb{Z}_7$. It corresponds to an hyperplane of the projective plane $PG(2,2)$, called Fano plane, represented in Figure 2.1. The projective geometry $PG(2,2)$ corresponds to a symmetric BIBD with parameters $(v,k,\lambda) = (7,3,1)$.

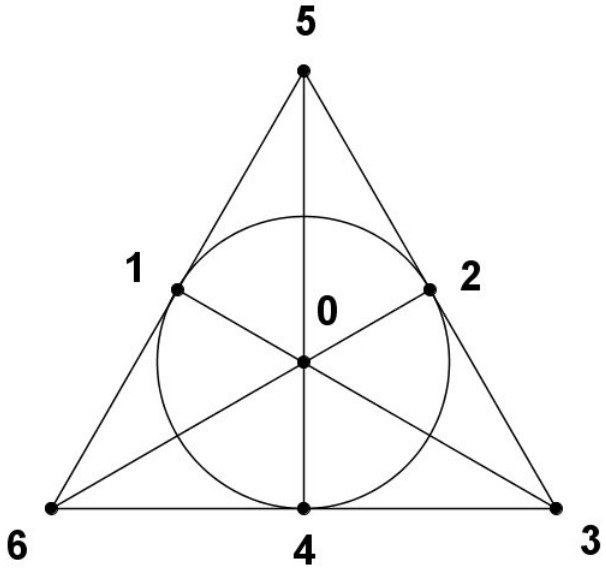

Figure 2.2: The Fano Plane $PG(2,2)$

The incidence matrix of this symmetric BIBD is given by:

$$A_D = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Note that every block of A_D is the incidence block of a difference-set with the same parameters as D. We say that these difference-sets are equivalent.

Example 2.3. The complement of the difference-set D corresponding to $PG(2,2)$ is $D' = \{2,4,5,6\}$, which is again a difference-set in $\mathbb{Z}_7$, called biplane, with parameters $(v,k,\lambda) =$

$(7,4,2)$. The incidence matrix of its corresponding BIBD with parameters $(v,k,\lambda) = (7,4,2)$ is given by:

$$A_{D'} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Similarly to Example 2.2, each block of $A_{D'}$ is a difference-set, and all the blocks are equivalent.

Example 2.4. The set $D = \{1,3,4,5,9\}$ is an $(11,5,2)$-difference set in the group $\mathbb{Z}_{11}$. It is also possible to write the groups multiplicatively, in which case the difference $d_i - d_j$ must be replaced by $d.d^{-1}$.

The *development* of a difference set D is the incidence structure $dev(D)$ whose points are the elements of G and whose blocks are the translates $gD = \{gd : d \in D\}$.

Next we will describe some classes of difference-sets. The construction of each class is different, despite that most of these constructions are based on finite fields.

2.1.3.1 Singer Difference-Sets (Type S)

A Singer difference-set is determined by the Hyperplanes in $PG(m-1,q)$, where $q = p^r$. Its parameters are given by the theorem of Singer [155], which gives:

$$v = \frac{q^m - 1}{q-1}, k = \frac{q^{m-1} - 1}{q-1}, \lambda = \frac{q^{m-2} - 1}{q-1} \tag{2.12}$$

where $q = p^s$ is a prime power.

The construction of Singer Difference-Sets is performed based on the blocks of the projective geometry $PG(m,q)$. Another construction is given as follows:

Let α be a generator of the multiplicative group of F_{q^m}. Then the set of integers $\{i : 0 \leq i < (q^m-1)/(q-1), trace_{m/1}(\alpha^i) = 0\}$ modulo $(q^m-1)/(q-1)$ form a (cyclic) difference set with the classical parameters $[q,m]$. Here the *trace* denotes the usual trace function $trace_{m/1}(\beta) = \sum_{i=0}^{m-1} \beta^{q^i}$ from F_{q^m} onto F_q. These difference sets are *Singer difference sets*.

2.1.3.2 Quadratic Residues Difference-Sets (Type Q)

The parameters of quadratic residues in $GF(q = p^r) \equiv 3(mod\,4)$ are:

$$v = p^r = 4t-1, k = 2t-1, \lambda = t-1 \tag{2.13}$$

Construction: The subset $\mathbb{F}_q^{(2)} = \{x^2 : x \in \mathbb{F}_q \setminus \{0\}\}$ of $\mathbb{F}_q$ is a difference set of type Q.

2.1.3.3 Type H_6

If p is a prime of the form $p = 4x^2+27$, then there will exist a primitive root r modulo p such that $Ind_r(3) = 1(mod\,6)$. The residues $a_1, ..., a_{(p-1)/2}(mod\,p)$ such that $Ind_r(a_i) = 0, 1, \text{or } 3(mod\,6)$ will form a difference set with the following parameters:

$$v = p = 4t-1, k = 2t-1, \lambda = t-1 \tag{2.14}$$

Type H_6 always duplicate the parameters of difference sets of type Q.

2.1.3.4 Twin-Primes Difference-Sets (Type T)

Suppose that p and $q = p+2$ are both primes. Of the $(p-1)(q-1)$ residues modulo pq prime to pq, let $a_1, ..., a_m$, $m = (p-1)(q-1)/2$ be those for which $(a_i/p) = (a_i/q)$, and also let $a_{m+1}, ..., a_{m+p}$ be $0, q, 2q, ..., (p-1)q$. Here $m+p = (pq-1)/2 = k$. Then $a_1, ..., a_k$ form a difference set modulo $v = pq$, with the following parameters:

$$v = pq, k = (pq-1)/2, \lambda = (pq-3)/4 \tag{2.15}$$

Here, necessarily $pq \equiv -1(mod\,4)$, and we have $v = 4t-1, k = 2t-1, \lambda = t-1$. In fact, difference sets of type T, and also type Q and H_6 are **Hadamard difference sets**.

2.1.3.5 Biquadratic Residues Difference-Sets (Type B)

The parameters of biquadratic residues of primes $p = 4x^2 + 1$, where x is odd, are given by:

$$v = p = 4x^2 + 1, k = x^2, \lambda = \frac{x^2 - 1}{4} \tag{2.16}$$

Construction: The subset $\mathbb{F}_p^{(4)} = \{x^4 \ : \ x \in \mathbb{F}_p \setminus \{0\}\}$ of $\mathbb{F}_p$ is a difference set of type B.

2.1.3.6 Biquadratic Residues and Zero Difference-Sets (Type B_0)

The parameters of biquadratic residues and zero modulo primes $p = 4x^2 + 9$, where x is odd, are given by:

$$v = 4x^2 + 9, k = x^2 + 3, \lambda = \frac{x^2 + 3}{4} \tag{2.17}$$

Construction: The subset $\mathbb{F}_p^{(4)} \cup \{0\}$ of $\mathbb{F}_p$ is a difference set of type B_0.

2.1.3.7 Octic Residues Difference-Sets (Type O)

If a and b are odd integers, then the parameters of octic residues of primes $p = 8a^2 + 1 = 64b^2 + 9$ are given by:

$$v = p, k = a^2, \lambda = b^2 \tag{2.18}$$

Construction: The subset $\mathbb{F}_p^{(8)} = \{x^8 \ : \ x \in \mathbb{F}_p \setminus \{0\}\}$ of $\mathbb{F}_p$ is a difference set of type O.

2.1.3.8 Octic Residues and Zero Difference-Sets (Type O_0)

The parameters of octic residues and zero for primes $p = 8a^2 + 49 = 64b^2 + 441$, where a is odd and b is even, is given by:

$$v = p, k = a^2 + 6, \lambda = b^2 + 7 \tag{2.19}$$

Construction: The subset $\mathbb{F}_p^{(8)} \cup \{0\}$ of $\mathbb{F}_p$ is a difference set of type O_0.

2.2 Low Density Parity-Check Codes

2.2.1 Tanner Graph

The Tanner graph of an LDPC code is analogous to the trellis of a convolutional code in that it provides a complete graphical representation of the code and it aids in the description of decoding algorithms. A Tanner graph is a bipartite graph, that is, a graph whose nodes may be separated into two types, with edges connecting only nodes of different types. The two types of nodes in a Tanner graph are the *variable nodes* (or code-bit nodes) and the *check nodes* (or constraint nodes), which we denote by **VNs** and **CNs**, respectively. The Tanner graph of a code is drawn as follows: CN i is connected to VN j whenever element h_{ij} in H is a 1. Observe from this rule that there are m CNs in a Tanner graph, one for each check equation, and n VNs, one for each code bit. Further, the m rows of H specify the m CN connections, and the n columns of H specify the n VN connections. Accordingly, the allowable n-bit words represented by the n VNs are precisely the codewords in the code.

Example 2.5. Consider a $(7,3)$ cyclic linear block code with $\omega_c = 3$ and $\omega_r = 3$ with the following H matrix:

$$H = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Note that the parity-check matrix H is square, and this a consequence of the code being cyclic. We said that there k extra parity-check equations in addition to the trivial $n-k$ check-sums. It is also observed that each row is a cyclic permutation of the previous one to the right. This is only a special case when the code is cyclic, however, in the general case, there are $m \geq n-k$ rows in H. The Tanner graph corresponding to H is depicted in Figure 2.2. Observe that VNs $1,2$, and 4 are connected to CN 0 in accordance with the fact that, in the zeroth row of H, we have $h_{01} = h_{02} = h_{04} = 1$ (all others are zero). Note, as follows from $cH^T = 0$, that the bit values connected to the same check node must sum to zero (mod 2). We may also proceed along columns to construct the Tanner graph. For example, note that VN 0 is connected to CNs $3,5$ and 6 in accordance with the fact that, in the zeroth column of H,

we have $h_{30} = h_{50} = h_{60} = 1$. The matrix H has 4 linearly independent rows. Thus, we have $rank(H) = 4$ and $r = 1 - 4/7 = 3/7$.

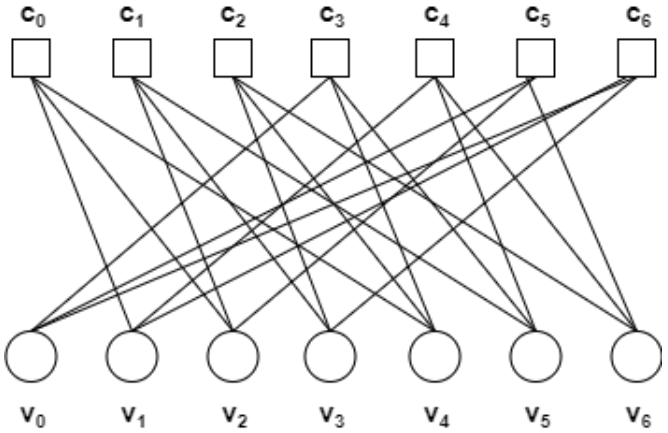

Figure 2.3: The Tanner graph for the LDPC code given in Example 2.5

The Tanner graph of an LDPC code acts as a blueprint for the iterative decoder in the following way. Each of the nodes acts as a locally operating processor and each edge acts as a bus that conveys information from a given node to each of its neighbors. The information conveyed is typically probabilistic information, e.g., log-likelihood ratios (LLRs), pertaining to the values of the bits assigned to the variable nodes. The LDPC decoder is initiated by n LLRs from the channel, which are received by the n VN processors. At the beginning of each half-iteration in the basic iterative decoding algorithm, each VN processor takes inputs from the channel and each of its neighboring CNs, and from these computes outputs for each one of its neighboring CN processors. In the next half-iteration, each CN processor takes inputs from each of its neighboring VNs, and from these computes outputs for each one of its neighboring VN processors. The VN-CN iterations continue until a codeword is found or until the preset maximum number of iterations has been reached.

The effectiveness of the iterative decoder depends on a number of structural properties of the Tanner graph on which the decoder is based. A sequence of edges which form a closed path is called a *cycle*. We are interested in cycles because short cycles degrade the performance of the iterative decoding algorithms employed by LDPC codes. This fact will be made evident in the discussion of the majority-logic decoding algorithms later in this thesis, but it can also be seen from the brief algorithm description in the previous paragraph. It should be clear from the description that cycles force the decoder to operate locally in some portions of the graph

(e.g., continually around a short cycle) so that a globally optimum solution is impossible. Observe also from the decoder description the necessity of a low-density matrix H: at high densities (about half of the entries are 1s), many short cycles will exist, thus precluding the use of an iterative decoder.

The length of a cycle is equal to the number of edges which form the cycle, so the length of the cycle in Figure 2.2 is 6. A cycle of length l is often called an l-cycle. The minimum cycle length in a given bipartite graph is called the graph's *girth*. The girth of the Tanner graph for the example code is clearly 6. The shortest possible cycle in a bipartite graph is clearly a length-4 cycle, and such cycles manifest themselves in the H matrix as four 1s that lie on the four corners of a rectangular sub-matrix of H. Observe that the RC constraint eliminates length-4 cycles, this is the major advantage of OSMLD codes. In the example shown above, the parity-check matrix H is free from cycles of length 4, moreover, this is a completely orthogonalizable difference-set code with minimum distance 4, which is the dual of the $(7,4)$ Hamming code.

The Tanner graph in the above example is completely regular: each VN and CN has 3 edge connections. We say that the degree of each VN and each CN is 3. This is in accordance with the fact that $\gamma = 3$ and $\rho = 3$. It is also clear from this example that $m\rho = n\gamma$ must hold for all regular LDPC codes since both $m\rho$ and $n\gamma$ are equal to the number of edges in the graph.

It is possible to more closely approach capacity limits with irregular LDPC codes than with regular LDPC codes [137], especially for long codes. For irregular LDPC codes, the parameters γ and ρ vary with the columns and rows, so such notation is not useful in this case. Instead, it is usual in the literature to specify the VN and CN degree-distribution polynomials, denoted in the literature by $\lambda(X)$ and $\rho(X)$, respectively.

2.2.2 Random Constructions

Random constructions of LDPC codes include computer-based techniques for constructing LDPC codes based on a set of criteria. In general, computer-generated LDPC are not structured, i.e. they do not have a cyclic or quasi-cyclic structure in their corresponding parity-check matrices. In contrast to structured LDPC codes, the encoding procedure of these codes requires a higher computational complexity compared to their structured counterparts. This is because random LDPC codes require a generator matrix instead of a generator polynomial for performing the encoding. Also, generally the generator matrix associated to a random LDPC codes is not sparse like the parity-check matrix H, and contains a large number of

1s. The generator matrix is generated based on the parity-check matrix, by using Gaussian elimination.

The most known random construction algorithms are the *Progressive-Edge-Growth* (PEG) and the *Approximate Cycle Extrinsic message degree* (ACE) techniques, shortly defined below [146].

1. **The PEG algorithm:** the progressive-edge-growth (PEG) algorithm [66, 67] is very effective and has been used widely for computer-based code design. This technique is essentially based on the fact that short cycles in Tanner graph present problems for iterative decoders. The main principle of this algorithm the is to build the graph one edge at a time, and each edge is added to the graph so that the local girth is maximized. The PEG algorithm is initialized by the number of variable nodes, n, the number of check nodes, m, and a variable node-degree sequence D_v.

2. **The ACE algorithm:** The ACE algorithm [167] accounts for the fact that iterative decoders not only have difficulties with cycles, but they are also impacted by the overlap of multiple cycles. The ACE algorithm was motivated by the following observations made in [167]:

   a) In a Tanner graph for which each VN degree is at least 2, every stopping set contains multiple cycles, except for the special case in which all VNs in the stopping set[1] are degree-2 VNs, in which case the stopping set is a single cycle.

   b) For a code with minimum distance d_{min}, each set of d_{min} columns of H that sum to the zero vector corresponds to a set of VNs that form a stopping set.

   c) The previous result implies that preventing small stopping sets in the design of an LDPC code (i.e., the construction of H) also prevents a small d_{min}.

Note that it is possible to combine the PEG and ACE algorithms to obtain a hybrid PEG/ACE algorithm. This was proposed in [190], where the PEG/ACE algorithm has shown to result in codes with good iterative decoding performance. An application of the PEG/ACE algorithm as well as a detailed description of this algorithm can be found in [146].

[1]

i. A stopping set S is a set of VNs whose neighboring CNs are connected to S at least twice. Stopping sets thwart iterative decoding on erasure channels.

2.2.3 Algebraic Constructions

Most of LDPC codes constructed algebraically or from finite geometries are structured, i.e. they are either cyclic or quasi-cyclic. The parity-check matrix of a structured LDPC code is a circulant if the code is cyclic, or it consists of a set of circulant matrices if the code is quasi-cyclic. These codes are very suitable for the practical use because of the simplicity of their implementation. Additionally, the encoding procedure is simply achieved using a cyclic shift register, as these codes are characterized by a generator polynomial instead of a generator matrix. The most known algebraic constructions of LDPC codes are those derived from finite geometries, namely Euclidean Geometries (EG) and Projective Geometries (PG). Other constructions include those derived from sub-geometries and combinatorial designs. Finite geometries constructions use advanced notions of finite fields and algebra. For details about these constructions, readers are referred to the book [98].

It was shown that LDPC codes derived from finite geometries outperform their randomly generated counterparts, when decoded with the BP-SP algorithm [106]. Also, it was shown in [86] that finite geometry LDPC codes outperform many other families of linear block codes, at the cost of simpler encoding and decoding implementations. In general, finite geometry LDPC codes satisfy the RC constraint, and can be decoded with majority-logic decoding (MLGD) algorithms. When these codes are decoded with majority-logic techniques, they can be comprised into two distinct classes:

- **One-Step Majority-Logic Decodable (OSMLD) codes:** These codes are the most appropriate to use in energy-constrained wireless communication systems, as their corresponding decoding algorithms require a significant lower computational complexity and lower storage requirements compared to other error correcting codes, in addition to a parallelizable implementation. In general, these codes are characterized by a set of orthogonal parity-check equations on each symbol digit, and the decoding process can employed using a simple majority-logic vote based on these equations. These codes are very sparse and they provide very low error floors compared to other algebraic codes.

- **Multi-Step Majority-Logic Decodable (MSMLD) codes:** These codes are generally characterized by a dense (not sparse) parity-check matrix, where the number of 1s entries is relatively high. For this reason, these codes are sometimes considered as Moderate or High density parity-check (MDPC/HDPC) codes. The orthogonalization process is obtained in λ steps, where $\lambda > 1$. Thus, decoding MSMLD codes requires λ majority-logic vote procedures, and it requires relatively higher complexity compared

to the decoding of OSMLD codes. MSMLD codes have shown to provide significant performance degradation, while decoded with the BP-SP algorithm. This degradation is caused by the existence of a large number of short cycles in their corresponding Tanner graphs.

2.3 One-Step Majority-Logic Decodable Codes

Let's first recall the definition of an OSMLD code, before exploring various families of codes that have the orthogonality property. In general, the orthogonalization rules for these codes are derived from notions of finite geometries.

Definition 2.20. Let $\mathscr{C}$ be a code with parameters defined by the triplet (n,k,d_{min}). The code $\mathscr{C}$ is one-step majority-logic decodable, if for each digit position $0 \leq j < n$, a set $\mathscr{H}_j$ of J binary orthogonal parity-check equations exists, with the following properties:

1. The j^{th} component of each vector $\mathscr{H}_{ji}$ is a 1, for $i \in \{0,1,2,...,J-1\}$.
2. For $z \neq j$ there is at most one vector $\mathscr{H}_{ji}$ whose z^{th} component is a 1.

These J vectors are said to be orthogonal on the j^{th} digit position. We call them **orthogonal vectors**.

2.3.1 Reed-Muller Codes

Reed-Muller (RM) codes represent one of the oldest and most understood families of error correcting codes. RM codes have gained the attention of many mathematical communities due to their attractive recursive and symmetric properties. The RM codes were introduced by Muller [118] in 1954, then Reed [133] succeeded in verifying their decoding algorithm in the same year. The RM codes are able to outperform the Hamming (1949) and Gray codes (1950) because they have the capacity to correct multiple errors. RM codes constitute the most prominent examples for which majority-logic decoding is possible. Except for first-order RM codes and codes of modest block lengths, their minimum distance is lower than that of BCH codes, however, the great merit of RM codes is that they are relatively easy to decode, using majority-logic circuits. RM codes were among the first codes to be deployed in space applications, being used in the deep space probes flown from 1969 to 1977. They are also used as components in several other systems. They were probably the first family of

codes to provide a mechanism for obtaining a desired minimum distance. Also, RM codes have a fast maximum likelihood decoding algorithm which is still very attractive. Moreover, with the recent introduction of polar codes, considered as generalized RM codes, these codes have regained the attention of many academic and industrial communities, especially after that it was shown that RM codes are capacity achieving in erasure channels [87].

In [80], [84] and [19], it was simultaneously discovered that RM codes are extended cyclic codes.

There are many ways to describe RM codes, provided that there are a varieties of construction techniques, which has made them useful in many theoretical developments. These codes were well explained and investigated in [117].

Reed-Muller codes are closely tied to functions of Boolean variables and can be described as multinomials over the field $GF(2)$.

Definition 2.21. Consider a Boolean function of m variables, $f(v_1, v_2, ..., v_m)$, which is a mapping from the vector space V, of binary m-tuples to the binary numbers $\{0,1\}$. Such functions can be represented using a truth table, which is an exhaustive listing of the input/output values. Boolean functions can also be written in terms of the variables.

As an example, the table below is a truth table for two functions of the variables v_1, v_2, v_3 and v_4.

$$\begin{array}{lcccccccccccccccc}
v_4 = & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
v_3 = & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
v_2 = & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
v_1 = & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
f_1 = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
f_2 = & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0
\end{array}$$

Clearly we have :

$$f_1(v_1, v_2, v_3, v_4) = v_1 + v_2 + v_3 + v_4$$

and

$$f_2(v_1, v_2, v_3, v_4) = 1 + v_1v_4 + v_1v_3 + v_2v_3$$

The columns of the truth table can be numbered from 0 to $2^m - 1$ using a base-2 representation with v_1 as the least-significant bit. The number of distinct Boolean functions in m variables is the number of distinct binary sequences of length 2^m, which is 2^{2^m}.

The set M of all Boolean functions in m variables forms a vector space that has a basis:

$$\{1, v_1, v_2, ..., v_m, v_1v_2, v_1v_3, ..., v_{m-1}v_m, ..., v_1v_2v_3...v_m\}$$

Every function f in this space can be represented as a linear combination of these basis functions. Below we give some examples of some basic functions and their vector representations:

$$\begin{aligned} 1 &= 1111111111111111 \\ v_1 &= 0101010101010101 \\ v_2 &= 0011001100110011 \\ v_3 &= 0000111100001111 \\ v_4 &= 0000000011111111 \\ v_1v_2 &= 0001000100010001 \\ v_1v_2v_3v_4 &= 0000000000000001 \end{aligned}$$

Now, let's define an $RM(r,m)$ code of order r and length 2^m.

Definition 2.22. The binary Reed-Muller code $RM(r,m)$ of order r and length 2^m consists of all linear combinations of vectors f associated with Boolean functions f that are monomials of degree $\leq r$ in m variables.

Example 2.6. The $RM(1,3)$ code has length $2^3 = 8$. The monomials of degree ≤ 1 are $\{1, v_1, v_2, v_3\}$, with associated vectors:

$$\begin{aligned} 1 &= 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ v_3 &= 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1 \\ v_2 &= 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1 \\ v_1 &= 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1 \end{aligned}$$

The generator matrix is described by:

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

This is the $(n,k,d_{mn}) = (8,4,4)$ RM code, and also the extended Hamming code.

Theorem 2.7. *The parameters of a RM(r,m) code take the following form:*

$$(n,k,d_{min}) = \left(2^m, 1+\sum_{i=1}^{r}\binom{m}{i}, 2^{m-r}\right)$$

One of the most interesting features of RM is that they have many symmetric and recursive properties. There are two other simple construction techniques of RM codes, the first one is a recursive construction, and the other is based on the m-fold Kronecker product of the matrix $G_{2,2}$, just similarly to the polar codes construction.

Lemma 2.1 (The $|u|u+v|$ construction)**.** *RM$(r+1,m+1) = \{(f, f+g)$ for all $f \in$ RM$(r+1,m)$ and $g \in$ RM(*
where its generator matrix is given by:

$$G_{RM}(r,m) = \begin{bmatrix} G_{RM}(r,m-1) & G_{RM}(r,m-1) \\ 0 & G_{RM}(r-1,m-1) \end{bmatrix}$$

Theorem 2.8. *For $0 \leq r \leq m-1$, the RM$(m-r-1,m)$ code is dual to the RM(r,m) code.*

Theorem 2.9 (Kronecker construction)**.** *Let $G_{(2,2)} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. Define the m-fold Kronecker product of $G_{(2,2)}$ as*

$$G_{(2^m,2^m)} = \bigotimes_{i=1}^{m} G_{(2,2)}$$

which is a $2^m \times 2^m$ matrix. Then the generator for the RM(r,m) code is obtained by selecting from $G_{(2^m,2^m)}$ those rows with weight greater than or equal to 2^{m-r}.

In general, a RM code of order r is an $r+1$ majority-logic decodable code. However, as the number of the orthogonalization steps becomes large when the order increases, decoding these codes is usually unpractical, due to the increased memory and computational complexity. But In 1971, Chen [21] showed that many finite geometry codes, among them all Reed–Muller codes RM(r,m), $r \leq \frac{m}{2}$, allow a two-step majority-logic decoding, improving significantly the decoding complexity in comparison with the well-known step-by-step decoding algorithm of Reed [133]. Also, decoding by sequential code reduction (SCR) majority-logic was introduced by Rudolph and Hartmann in [144], in order to reduce the number of the majority-logic gates required by the Reed step by step decoding algorithm, while slightly sacrificing decoding speed.

Recently, many papers have been published in the literature, investigating the decoding of RM codes using diverse variants and generalizations, as well as their performance evaluation and their connection to polar codes [15, 28, 32, 55, 59, 60, 148–150, 193].

Table 2.1 displays a set of Reed-Muller codes with various parameters for orders $1 \leq r \leq 4$.

r	m	n	k	d_{min}
1	3	8	4	4
	4	16	5	8
	5	32	6	16
2	4	16	11	4
	5	32	16	8
	6	64	22	16
3	5	32	26	4
	6	64	42	8
	7	128	64	16
	8	256	93	32
4	6	64	57	4
	7	128	99	8
	8	256	163	16
	9	512	256	32

Table 2.1: A set of Reed-Muller codes

2.3.2 Cyclic OSMLD Codes

2.3.2.1 Difference-Set Codes derived from Projective Geometry

Cyclic Difference-Set codes (DSC) represent an infinite class of powerful OSMLD codes with relatively high minimum distances and increasing code rates. These codes were discovered independently by Rudolph [142, 145]and Weldon [185], where Weldon have defined these codes based on combinatorial background, specifically from difference-sets, while the construction proposed by Rudolph is based on finite geometry, namely the Projective Geometry (PG).

For a (μ, s)th-order $PG(m, 2^s)$ code of length $n = \left(2^{(m+1)s} - 1\right) / (2^s - 1)$, when $\mu = 1$, a class of OSMLD codes is obtained. When $m = 2$, a $(1, s)$th-order PG code becomes a

difference-set code. Thus, DS codes represent a subclass of the class of $(1,s)$th-order PG codes. When $s = 1$,a $(1,1)$th-order PG code is considered a maximum-length code [98].

There are no simple formulas for enumerating the number of parity-check digits of PG codes. However, for $\mu = m-1$, the number of parity-check digits of a $(m-1,s)$th-order PG code was obtained independently by Goethals and Delsarte [49], Smith [156], MacWilliams and Mann [110], and is given by:

$$n-k = 1+\left(\begin{array}{c} m+1 \\ m \end{array}\right)^s \tag{2.20}$$

Difference-Set (DS) codes are well known to be completely orthogonalizable, i.e. each symbol digit contributes in the set of equations that are orthogonal on every digit position j, for $0 \leq j < n$. These codes have shown to outperform many other families of linear block codes, and due to their structured properties, their implementation is easy and requires a low-computational complexity and storage requirements. Additionally, DS codes exhibit low error floors, at a BER of 10^{-15}. Another advantage of these codes is that their minimum distances are well defined and easily calculated, in addition to the regular weight of the parity-check equations, which is given by $J = d_{min} - 1$. Note that the number of the orthogonal equations on a given symbol is equal to J. Thus, cyclic DS codes are completely regular. The expression of the number of information symbols for cyclic DS codes was given independently in [52].

For a positive integer $s > 0$, the parameters of cyclic DS codes over $GF(p)$, where p is a prime, take the following expressions [185]:

$$n = p^{2s} + p^s + 1 \tag{2.21}$$

$$k = n - \left(\left[\left(\begin{array}{c} p+1 \\ 2 \end{array}\right)^s\right] + 1\right) \tag{2.22}$$

$$d_{min} = p^s + 2 \tag{2.23}$$

For the binary case, a difference-set code defined over $GF(p = 2)$ has the following parameters:

$$n = 2^{2s} + 2^s + 1 \tag{2.24}$$

$$k = n - (3^s + 1) \tag{2.25}$$

$$d_{min} = 2^s + 2 \tag{2.26}$$

The formulation of difference-set codes is based on the construction of a *perfect difference-set*. Let $P = \{l_0, l_1, ..., l_q\}$ be a set of $q+1$ non-negative integers such that:

$$0 \leq l_0 < l_1 < l_2 < l_q \leq q(q+1) \tag{2.27}$$

From this set of integers, it is possible to form $q(q+1)$ ordered differences as follows:

$$D = \left\{ l_j - l_i \,|\, j \neq i \right\} \tag{2.28}$$

Obviously, half of the differences in D are positive and the other half are negative. The set P is said to be a *perfect simple difference-set* of order q if and only if it has the following properties [98]:

1. All the positive differences in D are distinct.
2. All the negative differences in D are distinct.
3. If $l_j - l_i$ is a negative difference in D, then $q(q+1) + 1 + (l_j - l_i)$ is not equal to any positive difference in D.

Clearly, it follows from the definition that $P' = \left\{0, l_1 - l_0, l_2 - l_0, ..., l_q - l_0\right\}$ is also a simple perfect difference-set.

As mentioned above, the construction of cyclic and quasi-cyclic DS codes is based on Singer difference-sets. Table 2.2 presents a set of binary difference-set codes with their parameters.

n	k	d_{min}	J	$r = \frac{k}{n}$
7	3	4	3	0.43
21	11	6	5	0.52
73	45	10	9	0.62
273	191	18	17	0.70
1057	813	34	33	0.77
4161	3431	66	65	0.82
16513	14325	130	129	0.87
65793	59231	258	257	0.90

Table 2.2: A set of cyclic binary difference-set codes

2.3.2.2 Euclidean Geometry Codes

Consider the m-tuples

$$(a_0, a_1, ..., a_{m-1}) \tag{2.29}$$

such that $a_i \in GF(q)$ and $q = 2^s$. There are $(2^s)^m = 2^{ms}$ m-tuples that form a vector space over $GF(2^s)$. The 2^{ms} m-tuples over $GF(2^s)$ form an m-dimensional Euclidean Geometry over $GF(2^s)$, denoted $EG(m, 2^s)$.

Each m-tuple is a point in $EG(m, 2^s)$. The m-tuple $0 = (0, 0, ..., 0)$ is called the *origin* of the geometry $EG(m, 2^s)$. There are q^m points and $q^{m-1}(q^m - 1)/(q - 1)$ lines in $EG(m, 2^s)$.

Let a be a non-origin point in $EG(m, 2^s)$ (i.e., $a \neq 0$). The 2^s points $\{\beta a : \beta \in GF(2^s)\}$ constitute a *line* (or 1-flat) in $EG(m, 2^s)$. This line is denoted $\{\beta a\}$.

Let $a_0, a_1, ..., a_\mu$ be $\mu + 1$ linearly independent points in $EG(m, 2^s)$, where $\mu < m$, then the collection of the $2^{\mu s}$ points:

$$\{a_0 + \beta_1 a_1 + ... + \beta_\mu a_\mu\} \tag{2.30}$$

with $\beta_i \in GF(2^s)$ constitute a μ-flat in $EG(m, 2^s)$ that passes through the point a_0. The μ-flats $\{\beta_1 a_1 + ... + \beta_\mu a_\mu\}$ and $\{a_0 + \beta_1 a_1 + ... + \beta_\mu a_\mu\}$ do not have any point in common. They are said to be parallel.

A realization of $EG(m, q)$ is obtained by using the extension field $GF(2^{ms})$. Let α be a primitive element of $GF(2^{ms})$. The 2^{ms} elements in $EG(m, 2^s)$ are expressed as: $\alpha^\infty = 0, \alpha^0 = 1, \alpha^2, ..., \alpha^{2^{ms}-2}$. In this case, $GF(2^s)$ is considered a subfield of $GF(2^{ms})$. Every element α^i of $GF(2^{ms})$ can be expressed as:

$$\alpha^i = a_{i0} + a_{i1}\alpha + a_{i2}\alpha^2 + ... + a_{i,m-1}\alpha^{m-1} \tag{2.31}$$

The μ-flats passing through α^{l_0} can be expressed as follows:

$$\alpha^{l_0} + \beta_1 \alpha^{l_1} + ... + \beta_\mu \alpha^{l_\mu} \tag{2.32}$$

Now let

$$v = (v_0, v_1, ..., v_{2^{ms}-2}) \tag{2.33}$$

be a $(2^{ms}-1)$-tuple over the binary field $GF(2)$. As mentioned above, let α be a primitive element of the Galois field $GF(2^{ms})$. We may number the components of v with the nonzero elements of $GF(2^{ms})$ as follows: the component v_i is numbered α^i for $0 \leq i \leq 2^m - 2$. Hence, α^i is the location number of v_i. We regard $GF(2^{ms})$ as the m-dimensional Euclidean geometry over $GF(2^s)$, denoted as $GF(m, 2^s)$. Let F be a μ-flat in $EG(m, 2^s)$ that does not pass through the origin $\alpha^\infty = 0$. Based on this flat F, we may form a vector over $GF(2)$ as follows:

$$v_F = (v_0, v_1, ..., v_{2^{ms}-2}) \tag{2.34}$$

, whose i^{th} component v_i is 1 if its location number α^i is a point in F; otherwise, v_i is 0. In other words, the location numbers for the nonzero components of v_F form the points of the μ-flat F. The vector v_F is called the *incidence vector* of the μ-flat F [98].

Definition 2.23. A (μ, s)th-order binary Euclidean geometry (EG) code of length $2^{ms}-1$ is the largest cyclic code whose null space contains the incidence vectors of all the $(\mu+1)$-flats that do not pass through the origin.

From Definition 2.23, the null space of the code contains the incidence vectors of all the $(\mu+1)$-flats that do not pass through the origin. There are

$$J = \frac{2^{ms}-1}{2^s-1} \tag{2.35}$$

error sums orthogonal on the error digit $e_{2^{ms}-2}$ at the location $\alpha^{2^{ms}-2}$, corresponding to all the 1-flats passing through the point $\alpha^{2^{ms}-2}$. Thus, $e_{2^{ms}-2}$ can be decoded correctly from these error sums provided that there are no more than $\left[\frac{J}{2}\right]$ errors in the received vector. Since the code is cyclic, decoding other error digits is performed in the same manner successively.

Since the decoding of each error digit requires $\mu+1$ steps of orthogonalization, the (μ, s)th-order EG code of length $n = 2^{ms}-1$ is therefore a $(\mu+1)$ step majority-logic decodable code. The code is capable of correcting

$$t = \left[\frac{2^{(m-\mu)s}-1}{2(2^s-1)} - \frac{1}{2}\right] \tag{2.36}$$

or fewer errors [98].

Note that when $\mu = 0$, then the code is a cyclic OSMLD code. Table 2.3 lists a set of $(\mu = 0, s)$th-order cyclic EG OSMLD codes.

n	k	d_{min}	J	$r = \frac{k}{n}$
15	7	5	4	0.47
63	37	9	8	0.59
255	175	17	16	0.67
1023	781	33	32	0.76

Table 2.3: A set of cyclic $(\mu = 0, s)$th-order EG OSMLD codes

2.3.2.3 Doubly-Transitive Invariant Codes

Doubly-Transitive Invariant (DTI) cyclic codes are a class of OSMLD codes derived from Euclidean geometries. These codes were introduced in 1967 [79], and have shown to provide good performance under iterative majority-logic decoding. In contrast to EG OSMLD codes defined above, DTI codes are characterized by a set of irregular orthogonal parity-check equations on each symbol, where all the orthogonal equations have the same weight q except one of them which has a weight $q-1$, for each symbol position. The construction of DTI codes was well investigated in [79] and [98]. We will briefly define cyclic DTI codes based on the explanations given in [98].

Let $\mathscr{C}$ be an (n,k) cyclic code of length $n = 2^m - 1$. Each vector $v = (v_0, v_1, ..., v_{n-1})$ in $\mathscr{C}$ can be extended by adding an overall parity-check digit to its left, denoted by v_∞, where $v_\infty = v_0 + v_1 + ... + v_{n-1}$. The resulting vector has $n+1 = 2^m$ components. The 2^k extended vectors form an $(n+1,k)$ extended code, denoted by $\mathscr{C}_e$. Let α be a primitive element of the Galois field $GF(2^m)$. Then the components of the extended vector v_e are numbered following the powers of α, where $\alpha^\infty = 0$, and each component v_i is numbered as α^i. These numbers are called *location numbers.* Let Y denote the location of a component of v_e. Let's consider a permutation that carries the component of v_e at the location Y to the location $Z = aY + b$, where $a, b \in GF(2^m)$ and $a \neq 0$. This permutation is called an *affine permutation*.

An extended cyclic code $\mathscr{C}_e$ of length $n = 2^m$ is said to be invariant under the group of affine permutations if every affine permutation carries every code vector in $\mathscr{C}_e$ into another code vector in $\mathscr{C}_e$.

Let h be a nonegative integer less than 2^m. The *radix* 2 (binary) expansion of h is given by

$$h = \delta_0 + \delta_1 2 + \delta_2 2^2 + ... + \delta_{m-1} 2^{m-1} \tag{2.37}$$

where δ_i is binary for $0 \leq i < m$. Let h' be another nonegative integer less than 2^m whose radix 2 expansion is

$$h' = \delta'_0 + \delta'_1 2 + \delta'_2 2^2 + ... + \delta'_{m-1} 2^{m-1}. \tag{2.38}$$

The integer h' is said to be a *descendant* of h if $\delta'_i \leq \delta_i$ for $0 \leq i < m$. The set of all nonzero proper descendants of h is denoted by $\Delta(h)$.

Theorem 2.10 ([98]). *Let $\mathscr{C}$ be a cyclic code of length $n = 2^m - 1$ generated by $g(x)$. Let $\mathscr{C}_e$ be the extended code obtained from $\mathscr{C}$ by appending an overall parity-check digit. Let α be a primitive element of the Galois field GF(2^m). Then the extended code $\mathscr{C}_e$ is invariant under the affine permutations if and only if for every α^h that is a root of the generator polynomial $g(x)$ of $\mathscr{C}$ and for every h' in $\Delta(h)$, $\alpha^{h'}$ is also a root of $g(x)$ and $\alpha^0 = 1$ is not a root of $g(x)$.*

The cyclic DTI code $\mathscr{C}$ is obtained by deleting the first from each vector of $\mathscr{C}_e$. Let J and L be two (odd) factors of $2^m - 1$, where $J.L = 2^m - 1$. The code $\mathscr{C}$ has a set of J orthogonal parity-check equations on each digit position in its dual structure, and is capable of correcting $t = \left[\frac{J}{2}\right]$ errors of fewer. The method of determining the orthogonal equations and the generator polynomial of DTI codes is explained in [98].

Table 2.4 displays a set of cyclic DTI codes with their corresponding parameters.

n	k	d_{min}	$r = \frac{k}{n}$
15	6	6	0.4
63	36	10	0.57
255	174	18	0.69
1023	780	34	0.76

Table 2.4: A set of cyclic DTI codes

2.3.2.4 Maximal-Length Codes

Another class of OSMLD codes is the Maximal-Length codes. These codes have been proved to be completely orthogonalizable, and were shown to be majority-logic decodable independently by Yale [191] and Zierler [206].

For any integer $m \geq 3$, there exists a nontrivial maximum-length code with the following parameters:

$$n = 2^m - 1 \tag{2.39}$$

$$k = m \tag{2.40}$$

$$d_{min} = 2^{m-1} \tag{2.41}$$

The generator polynomial of these codes is given by:

$$g(x) = \frac{x^n + 1}{p(x)} \tag{2.42}$$

where $p(x)$ is a primitive polynomial of degree m. This code consists of the all-zero codeword vector and $2^m - 1$ codeword vectors of weight 2^{m-1}. The dual code of the maximum-length code is a $(2^m - 1, 2^m - m - 1)$ cyclic code generated by the reciprocal of the parity polynomial $p(x)$ calculated as follows [98]:

$$p^*(x) = x^m p(x^{-1}) \tag{2.43}$$

2.3.3 Quasi-Cyclic OSMLD Codes

The construction of Quasi-Cyclic OSMLD codes derived from combinatorial designs has gained many attentions of various research communities. Specifically, attention was addressed to the correspondence between difference-families, including Steiner 2-designs [27, 57], and quasi-cyclic self-orthogonal OSMLD codes. We shall define these notions before presenting these special constructions leading to quasi-cyclic OSMLD codes.

Definition 2.24 (Difference-Families)**.** Let G be a group of order v. A collection $\{B_1, B_2, ..., B_t\}$ of k-subsets of G form a (v, b, r, k, λ) **difference-family** if every non-identity element of G occurs λ times in $\Delta B_1 \cup ... \cup \Delta B_t$. The sets B_i are **base blocks**. A difference family having at least one short block is **partial**.

Special classes of difference families include the *Steiner systems*, constructed based on base blocks of these designs. Historically, Smith *et al.* [157] introduced in 1968 an application of incomplete block designs to the construction of several families of error-correcting codes which may be decoded using a relatively simple majority logic decoding procedure. However any explicit construction for such designs was given. Special cases of these codes are equivalent to the quasi-cyclic self orthogonal codes based on Singer Difference Sets, discussed by Townsend and Weldon in [171]. Chen Zhi *et al.* stated in [204] explicit constructions

of many classes of *difference families*, considered as base blocks for Steiner designs. The authors presented a construction of infinite optimal self-orthogonal quasi-cyclic codes with high rates. Later, these connections was made by MacKay in his 2000 paper using the incidence matrix of Steiner triple system (STS) designs for high-rate LDPC codes [108]. In the same year, Mittelholzer, in an IBM research report [116], demonstrated the link between projective geometry LDPC codes and BIBD designs, and generated new LDPC codes from further BIBDs, extending the work of [108]. The use of *Steiner triple systems* as QC LDPC codes was also the central theme in a presentation by B. Vasic in 2002 at the Information Theory Symposium [176], and later published in [180]. A method of constructing Quasi-cyclic LDPC (OSMLD) codes based on two arbitrary subsets of elements from a given field was proposed in [93]. The authors claimed that their construction technique includes some well known constructions of QC-LDPC codes based on finite fields and combinatorial designs as special cases. Their proposed construction in conjunction with a technique, known as masking, results in codes whose Tanner graphs have girth 8 or larger, and they have shown that the constructed codes perform well and have low error-floors. Recently, a construction of quasi cyclic OSMLD codes (using genetic algorithms) derived from *Disjoint Difference-Sets* was published in [139].

The advantage of Steiner systems is that the number of objects belonging to base blocks is small. This property has motivated many group of researchers to correspond these designs to LDPC codes with sparse dual structure. Generally, the most known Steiner systems are triple Steiner designs with $k = 3$, as well as other Steiner designs with $k = 4$ and $k = 5$.

Details of these constructions based on Steiner systems are not presented in this section, although the main results are depicted in Table 2.5. This table presents the parameters of the Steiner systems, as well as the references where their explicit constructions are given. Note that the parameter λ was omitted because in the case of OSMLD codes, we always have $\lambda = 1$. The *Netto* constructions, *Skolem* and *extended Skolem sequences* are detailed in the Appendix A.

2.3.4 Other OSMLD Codes derived from Combinatorial Designs

Other constructions of OSMLD codes that do not have any particular structure (i.e. cyclic or quasi cyclic) include constructions based on sub-geometries derived from PG and EG, as well as some special combinatorial designs. We briefly review here a non-exhaustive list of previous works already published on this topic.

v	k	b	r	References
$6t+1$	3	$6t^2+t$	$3t$	Netto first construction [180, 204] Skolem Sequences [27]
$6t+3$	3	$(3t+1)(2t+1)$	$3t+1$	Extended Skolem Sequences [27]
$12t+7$	3	$(2t+1)(12t+7)$	$3(2t+1)$	Netto second construction [180]
$12t+1$	4	$t(12t+1)$	$4t$	[204, Lemma 3]
$20t+1$	5	$t(20t+1)$	$5t$	[204, Lemma 4]

Table 2.5: Base blocks constructions of Steiner systems

LDPC codes based on *mutually orthogonal Latin rectangles* were presented in [179]. The same author proposed high-rate LDPC codes based on *anti-Pasch affine geometries* [177]. In [197], regular LDPC codes with girth 12 have been constructed based on BIBDs. In [6], the authors have investigated the construction of LDPC codes with girth at least equal to 6, and they stated that most of these codes derived from block designs are quasi cyclic. In [181], high-rate girth-8 LDPC codes based on *rectangular integer lattices* were proposed. LDPC codes derived from partial geometries as *Unital* and *Oval* designs were presented in [76], and later in [38, 39]. Also, *Hermitian curves* were used to design LDPC codes in [127]. *Semi-partial geometries*, considered as a generalization of partial geometries were investigated in a doctoral thesis in [70] as well as their correspondence with LDPC (OSMLD) codes. In [189], a study on partial geometries from RC-constrained matrices based on *group divisible designs* (GDDs), and relevant constructions of BIBDs and TDs were presented. The authors have proposed a method for constructing LDPC codes with flexible code rate and length parameters by employing the resolvability of GDDs.

2.4 Multi-Step Majority-Logic Decodable Codes

2.4.1 Background and Definitions

Multi-Step Majority-Logic Decodable (MSMLD) codes represent a generalization of OSMLD codes, in a manner that the notion of the orthogonality on a given symbol is extended to a set of symbols. Therefore, the decoding process of these codes requires many orthogonalization steps, where each step consists of decoding a given set of symbols, until reaching a

set of equations orthogonal on the symbol being decoded. It is straightforward that decoding MSMLD codes needs a higher computational complexity than that required for decoding OSMLD codes, due the number of orthogonalization steps involved.

Generally, MSMLD codes are defined by a parity-check matrix H which is not sparse, i.e. the number of nonzero entries is very large. Thus, unlike OSMLD codes, the structure of H contains a large number of short cycles. As a consequence, MSMLD codes provide bad performance when decoded with the BP-SP decoding algorithm, in addition to a large storage requirement. This performance degradation is obvious, as the number of short cycles is significantly large. Therefore, an interesting research direction consists of devising suitable decoding algorithms that exploits the orthogonality structure of MSMLD codes, in order to achieve good performance.

The decoding complexity of MSMLD codes is relatively higher, this is a major drawback of these codes. However, this drawback can be overcame by considering only cyclic OSMLD codes, which helps to significantly reduce the encoding and decoding complexity of these codes. Additionally, MSMLD codes have shown to provide high coderates and low error floors due because of a relative large column degree.

Thus, cyclic MSMLD codes derived from finite geometries represent the most interesting subclass of these codes. Moreover, it was shown in [96] that MSMLD codes derived from Euclidean geometries are the most powerful class of this family of codes. Other MSMLD codes are those derived from Projective geometries, as well as those derived from combinatorial designs, with the parameter $\lambda > 1$. We will briefly define the notion of multi-steps, as well as a special class of MSMLD codes derived from EG, namely $(\mu > 0, s)$th-order EG cyclic codes and $(\mu > 0, s)$th-order multifold EG cyclic codes. As mentioned in 2.4.1.2, the parameter μ is the number of steps required for achieving the orthogonality, and we say that the EG cyclic code is a μ-step majority-logic decodable code.

First, before defining MSMLD codes derived from EG, let's introduce a definition on the notion of the multi step orthogonality in Definition 2.25. Next we will assume that the notions of lines and flats in EGs are already defined in 2.4.2.1, so that we will skip these definitions.

Let $E = \{e_{i_1}, e_{i_2}, ..., e_{i_M}\}$ be a set of M error digits where $0 \leq i_1 < i_2 < ... < i_M < n$. The integer M is called the **size** of E.

Definition 2.25. A set of J parity-check sums $A_1, A_2, ..., A_J$ is said to be orthogonal on the set E if and only if:

- Every error digit e_{i_l} in E is checked by every check-sum A_j for $1 \leq j \leq J$.
- No other error digit is checked by more than one check-sum.

  Similarly to One-Step MLG decoding, the sum of error digits in E can be determined correctly from the J check-sums orthogonal on E, provided that there are $\left[\frac{J}{2}\right]$ errors or fewer in the error pattern e.

  This **orthogonalization** process is additionally used to decode correctly all the error digits.

Now let $L_{u,j}$ denotes the u^{th} error set (Line) of $EG(m,q)$ orthogonal on the error digit e_j, for $0 \leq u < J_2$ and $0 \leq j < n$. Let $S_{L_{u,j}}$ denotes its corresponding check-sum. Let $S_{L_{u,i,j}}$ denotes the i^{th} check-sum orthogonal on the line $L_{u,j}$ for $0 \leq i < J_1$, where $J_1 < J_2$.

Then, for sake of simplicity, consider that $\mu = 1$, so that the code $\mathscr{C}$ is a two-step majority-logic decodable (TSMLD) code (i.e. $(\mu+1)$-step MLD code). Based on this assumption, and for $0 \leq j < n$, $0 \leq u < J_2$ and $0 \leq i < J_1$, the decoding process includes the following steps:

1. **First orthogonalization step:** Decoding each line $L_{u,j}$ based on the J_1 check-sums $S_{L_{u,i,j}}$ orthogonal on it.
2. **Second orthogonalization step:** Decoding the digit e_j based on the J_2 check-sums $S_{L_{u,j}}$ orthogonal on it.

Figure 2.3 illustrates the principle of two-step majority-logic decoding. For instance, it represents the decoding tree of the last symbol of the Hamming code with parameters $(n,k,d_{min}) = (7,4,3)$. For decoding the 6^{th} digit, the decoding process starts from the set of 2-flats $S_{L_{u,i,j}}$ orthogonal on each 1-flat (line) $S_{L_{u,j}}$ containing the last symbol, where each line is decoded by a majority-logic vote based on the 2-flats orthogonal on it, until all these lines of 1-flats are decoded, then a second majority-logic vote is decided on the symbol. As this code is cyclic, the same decoding procedure is employed for all the remaining symbols, by cyclically shifting the sequence being decoded. It is clear that for this case, we have $J_2 = 5$ and $J_1 = 2$ provided that this code is capable of correcting at most $\frac{J_1}{2} = 1$ error.

Details on decoding algorithms for TSMLD codes are included in Chapter 5, dedicated for this topic. Note that many known algebraic codes can be considered as MSMLD codes, for instance, Reed Muller (RM), Quadratic Residue (QR) and Golay codes can be decoded with

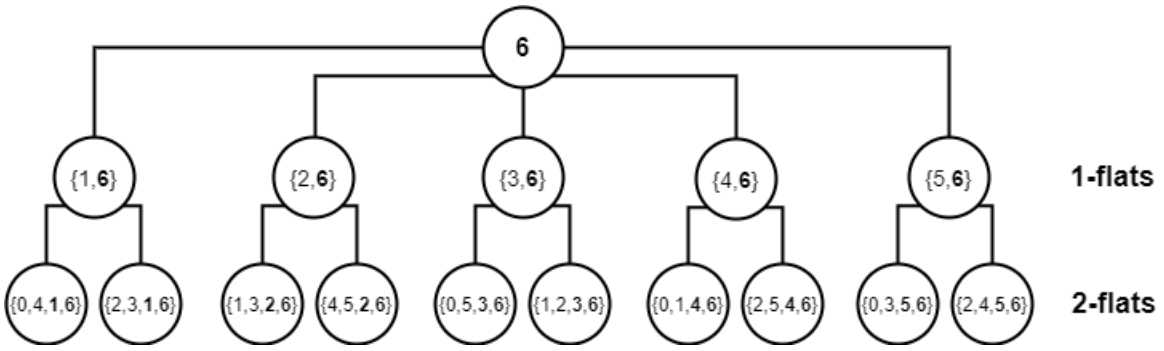

Figure 2.4: Two-Step Majority-Logic Decoding Tree of the $(7,4,3)$ Hamming code

majority-logic in multiple steps (L steps), as well as many families of self-dual codes have shown to have orthogonal equations on sets of digit positions in their dual structure, leading to consider them as MSMLD codes. In addition, it was shown in [98] that for $m \geq 3$, the cyclic $(n,k,d_{min}) = (2^m-1, 2^m-m-1, 3)$ Hamming codes are $m-1$ steps majority-logic decodable. Cyclic MSMLD codes derived from Euclidean geometries are actually known to be the most efficient class of MSMLD codes, including a subset of codes belonging to the family of BCH codes, and this is due to their geometrical structure and to the fact that a subclass of these codes includes MSMLD codes that are completely orthogonalizable. Table 2.6 lists a set of $L = m-1$ steps majority-logic decodable cyclic Hamming codes.

m	(n,k,d_{min})	J_2	J_1	L
3	$(7,4,3)$	5	2	2
4	$(15,11,3)$	13	2	3
5	$(31,26,3)$	29	2	4
6	$(63,57,3)$	61	2	5
7	$(127,120,3)$	125	2	6
8	$(256,248,3)$	254	2	7

Table 2.6: A set of $L = m-1$ steps majority-logic decodable cyclic Hamming codes

2.4.2 MSMLD Codes derived from Euclidean Geometries

2.4.2.1 (μ, s)th-order EG cyclic EG codes

Given a μ-flat $F^{(\mu)}$ in $EG(m, q = 2^s)$ passing through $\alpha^{2^{ms}-2}$, the number of $(\mu+1)$-flats not passing through the origin in $EG(m, 2^s)$ that intersect on $F^{(\mu)}$ is:

$$J_1 = \frac{2^{(m-\mu)s} - 1}{2^s - 1} \tag{2.44}$$

The incidence vectors of these J_1 $(\mu+1)$-flats are orthogonal on the digits corresponding to the points in $F^{(\mu)}$. This is the first orthogonalization step (the sum $S(F^{(\mu)})$ is determined)

Let $F^{(\mu-1)}$ be a $(\mu-1)$-flat passing through $\alpha^{2^{ms}-2}$. There are:

$$J_2 = \frac{2^{(m-\mu+1)s} - 1}{2^s - 1} - 1 > J_1 \tag{2.45}$$

μ-flats not passing through the origin which intersect on $F^{(\mu-1)}$. This process is the second orthogonalization step. We say that a (μ, s)-th order Euclidean geometry (EG) code of length $2^{ms} - 1$ is a $L = \mu + 1$ steps majority-logic decodable code, capable of correcting any combination of $\left[\frac{J_1}{2}\right]$ or fewer errors.

Definition 2.26 ([98]). A (μ, s)-th order binary Euclidean geometry (EG) code of length $2^{ms} - 1$ is the largest cyclic code whose null space contains the incidence vectors of all the $(\mu+1)$-flats of $EG(m, 2^s)$ that do not pas through the origin.

The determination of the generator polynomial of these codes is given in the following theorem [98].

Theorem 2.11 ([97]). *Let α be a primitive element of the Galois field $GF(2^{ms})$. Let h be a nonnegative integer less than $2^{ms} - 1$. The generator polynomial $g(x)$ of the (μ, s)-th order EG code of length $2^{ms} - 1$ has α^h as a root if and only if*

$$0 < \underset{0 \le l < s}{max} W_{2^s}(h^{(l)}) \le (m - \mu - 1)(2^s - 1) \tag{2.46}$$

where $W_{2^s}(h)$ is the 2^s radix-expansion of the integer h.

Example 2.7. Let $m = 4$, $s = 1$ and $\mu = 1$. Thus $EG(4, 2)$ is considered with $J_1 = 6$ and $J_2 = 13$.

- The $J_2 = 13$ lines (1-flats) of $EG(4,2)$ passing through α^{14} are obtained by $\alpha^{14} + a\alpha^i$, yielding:
  $\{\alpha^{13}, \alpha^{14}\}, \{\alpha^{12}, \alpha^{14}\}, \{\alpha^{11}, \alpha^{14}\}, \{\alpha^{10}, \alpha^{14}\}, \{\alpha^{9}, \alpha^{14}\}, \{\alpha^{8}, \alpha^{14}\}, \{\alpha^{7}, \alpha^{14}\},$
  $\{\alpha^{6}, \alpha^{14}\}, \{\alpha^{5}, \alpha^{14}\}, \{\alpha^{4}, \alpha^{14}\}, \{\alpha^{3}, \alpha^{14}\}, \{\alpha^{2}, \alpha^{14}\}, \{\alpha^{1}, \alpha^{14}\}.$
- The $J_2 = 6$ 2-flats passing through $\{\alpha^{13}, \alpha^{14}\}$ are obtained by $\alpha^{14} + a\alpha^i + b\alpha^j$ yielding:
  $\{\alpha^{4}, \alpha^{10}, \alpha^{13}, \alpha^{14}\}, \{\alpha^{7}, \alpha^{12}, \alpha^{13}, \alpha^{14}\}, \{\alpha^{9}, \alpha^{11}, \alpha^{13}, \alpha^{14}\},$
  $\{\alpha^{0}, \alpha^{8}, \alpha^{13}, \alpha^{14}\}, \{\alpha^{1}, \alpha^{5}, \alpha^{13}, \alpha^{14}\}, \{\alpha^{3}, \alpha^{6}, \alpha^{13}, \alpha^{14}\}.$
  $\vdots$

The rest of the 2-flats are obtained in the same manner, and the obtained code is a $(15,5,7)$ cyclic TSMLD code capable of correcting $\left\lceil \frac{J_1}{2} \right\rceil = 3$ errors or fewer. This code is also the triple error correcting BCH codes.

Remark 2.3 ([98]). A $(\mu, s = 1)$th-order EG code is called a μ-th order *Reed Muller (RM)* code.

A set of (μ, s)th-order EG cyclic codes is listed in Table 2.7. These codes are $(\mu + 1)$ majority-logic decodable, and they contain Hamming codes with $J = 2$.

2.4.2.2 (μ, s)th-order Twofold cyclic EG codes

Here we will define a special class of MSMLD codes, called (μ, s)th-order twofold EG cyclic codes. Note that this notion is generalizable to *multifold*. However, we will here focus our interest to twofold codes, as it was stated that these codes are the most powerful MSMLD codes [96, 98].

Let's denote by $EG^*(m,q)$ the subgeometry obtained by removing the origin of $EG(m,q)$ and all the lines passing through it.

A line of $EG^*(m,q)$ has q points. Every point of $EG^*(m,q)$ is contained in

$$J_2 = \frac{n}{(q-1)} - 1 \tag{2.47}$$

lines.

Let $L_{u,j} = \{\alpha^{j_0}, \alpha^{j_1}, ..., \alpha^{j_{q-1}}\}$ denotes the u^{th} line of $EG^*(m,q)$ orthogonal on α^j for $0 \leq u < J_2$. There are

m	s	μ	n	k	J
3	1	1	7	4	2
4	1	2	15	11	2
4	1	1	15	5	6
5	1	3	31	26	2
5	1	2	31	16	6
6	1	4	63	57	2
6	1	3	63	42	6
6	1	2	63	22	14
3	2	1	63	48	4
7	1	5	127	120	2
7	1	4	127	99	6
7	1	3	127	64	14
8	1	5	255	219	6
8	1	4	255	163	14
4	2	2	255	231	4
4	2	1	255	127	20
10	1	6	1023	848	14
10	1	5	1023	638	30
5	2	2	1023	748	20
5	2	1	1023	288	84

Table 2.7: A set of (μ, s)th-order EG cyclic codes

$$J_1 = q^{m-1} - 2 \tag{2.48}$$

$(1,2)$-frames parallel to it, denoted as $L_{u,i,j}$ where $0 \leq i < J_1$.

A $(1,2)$-frame in $EG^*(m,q)$ intersecting on α^j is denoted as $\left\{L_{u,j}, L_{u,i,j}\right\}$. There are J_1 $(1,2)$-frames $\left\{L_{u,j}, L_{u,0,j}\right\}, \left\{L_u, L_{u,1,j}\right\}, ..., \left\{L_u, L_{u,J_1-1,j}\right\}$ orthogonal on the line $L_{u,j}$.

The total number of $(1,2)$-frames contained in $EG^*(m,q)$ is given by:

$$J_0 = \frac{n\left(q^{m-1}-1\right)\left(q^{m-1}-2\right)}{2\left(q-1\right)} \tag{2.49}$$

Each line that passes through α^{q^m-2} consists of the following points:

$$\alpha^{q^m-2} + \eta\alpha^j \tag{2.50}$$

where $\eta \in GF(q)$.

A $(1,2)$-frame that contains the line $\{\alpha^{q^m-2} + \eta\alpha^j\}$ is of the form:

$$(\{\alpha^{q^m-2} + \eta\alpha^j\}, \{\alpha^{q^m-2} + \alpha^l + \eta\alpha^j\}) \tag{2.51}$$

where α^l is not in $\{\alpha^{q^m-2} + \eta\alpha^j\}$.

A (μ,s)th-order twofold EG code of length $n = 2^{ms}-1$ is a $L = \mu+1$ steps majority-logic decodable code, capable of correcting any combination of $\left[\frac{J_1}{2}\right]$ or fewer errors.

Definition 2.27. A (μ,s)th-order twofold EG code of length $n = 2^{ms}-1$ is the largest cyclic code whose null space contains the incidence vectors of all the $(\mu,2)$-frames in $EG(m,2^s)$ that do not pass through the origin.

Theorem 2.12. *Let α be a primitive element of the Galois field $GF(2^{ms})$. Let h be a nonnegative integer less than $2^{ms}-1$. The generator polynomial $g(x)$ of the (μ,s)th-order twofold EG code of length $2^{ms}-1$ has α^h as a root if and only if*

$$0 < \max_{0\le l<s} W_{2^s}(h^{(l)}) < (m-\mu)(2^s-1) \tag{2.52}$$

A set of (μ,s)th-order Twofold EG cyclic codes is displayed in Table 2.8. These codes are $(\mu+1)$ majority-logic decodable.

2.5 Conclusion

In this chapter, we have presented some interesting notions on balanced incomplete block designs as well as difference-sets, then we have established a literature review on the different classes of majority-logic decodable codes derived from various block designs and finite geometries. We particularly discussed the methods of constructing cyclic, quasi-cyclic and non structured OSMLD and MSMLD codes. For cyclic OSMLD codes, we have seen that most of them are derived from Singer difference-sets and Golomb ruler, as well as Euclidean geometries. The quasi-cyclic OSMLD codes are essentially developed from difference-families and various Steiner systems. Base blocks of these combinatorial systems are the key for generating quasi-cyclic OSMLD codes, where the other lines of the parity-check matrix are obtained by cyclically shifting the base blocks t times, yielding a quasi circulant parity-check

m	s	μ	n	k	J
3	2	1	63	24	14
2	3	1	63	45	6
4	2	1	255	45	62
4	2	2	255	171	14
2	4	1	255	191	14
3	3	1	511	184	62
3	3	2	511	475	6
5	2	1	1023	76	256
5	2	2	1023	438	62
5	2	3	1023	868	14
2	5	1	1023	813	30

Table 2.8: A set of (μ, s)th-order Twofold EG cyclic codes

matrix. For the non structured OSMLD codes, we have shown that they are derived from subgeometries of PG and EG, as well as other block designs not investigated in this thesis.

Our summary of this classification is presented in Table 2.9. The first column includes the structure of codes, the second column displays different families of MLGD codes, while the third column includes the combinatorial or geometric construction that leads to these codes. Note that this classification is not exhaustive, as we are actually pursuing these investigations towards finding new families of MLGD codes derived from other types of combinatorial designs.

Structure	Combinatorial Construction	Codes
Cyclic OSMLD	EG (Type 0, Type 1)	EG, DTI
	PG, Singer Difference-Sets, Golomb Rulers	DS codes, other OSMLD codes
Cyclic MSMLD	(μ,s)-th order EG and PG, other Difference-Sets	(μ,s)-th order (Multifold) EG, RM, (μ,s)-th order PG
Quasi-Cyclic OSMLD	Difference-Families and Steiner Systems	SOQC and other QC OSMLD codes
Non Cyclic OSMLD	Sub-geometries of PG, other designs and geometries	Ovals, Unitals, PBDs Quadric geometries, PBIBDs

Table 2.9: Classification of MLGD codes derived from combinatorial designs and finite geometries

Cyclic MSMLD codes are principally derived from Euclidean and Projective geometries, and they include various classes, depending on the parameters of the geometries involved in the construction. Other cyclic MSMLD codes can be derived from the other families of difference-sets that were reviewed in this chapter. Similarly to OSMLD codes, quasi-cyclic MSMLD codes can be derived from BIBDs that are similar to Steiner systems and difference-families, with a parameter $\lambda > 1$. MSMLD codes that do not have any particular structure may be constructed from other families of block designs, although non-cyclic MSMLD codes are not interesting for wireless communication systems. Besides that, however, non-cyclic codes can be very beneficial for other applications, including low-complexity authentication systems and McEliece cryptosystems. This chapter is essential in order to develop basic notions about the constructions of various MLGD codes as well as the attractive structural properties that these codes inherit from their corresponding combinatorial constructions.

Chapter 3

Genetic Algorithms for the Discovery of New Cyclic One-Step Majority-Logic Decodable Codes

3.1 Introduction

The construction of One-Step Majority-Logic Decodable codes represents an important topic in coding theory, that have gained the attention of various research communities since the 1950s. Most of the construction methods of OSMLD codes are derived from finite geometries, finite fields and combinatorial designs. After the rediscovery of LDPC codes [109], many research papers have investigated the construction of regular LDPC codes without short cycles, namely with a minimum girth of 6 [68, 75, 76, 92, 116, 180, 186]. Most of these construction techniques are derived from combinatorial designs, where each combinatorial design leads to a set of OSMLD codes with properties that inherit those designs. The oldest construction of OSMLD codes was that of Difference-Set Codes, discovered independently by Rudolph [145] and Weldon [185], respectively derived from Projective geometries and Singer Difference-Sets. These codes are known to be completely orthogonalizable and represent a particular class of OSMLD codes where the code length is in the form $n = J(J-1)+1$. Weldon has shown that these codes are approximately as powerful as the best cyclic codes for given values of efficiency and length, and are very easily implemented. The formula for calculating the number of parity-check equations was introduced by Graham [52] and MacWilliams [112]. Other OSMLD codes derived from finite geometries are those based on Euclidean geometries. Also, in [86] it was shown that cyclic OSMLD codes are finite

geometry LDPC codes with a Tanner graph that has no cycles of length 4. In general, these codes have a regular structure in their parity-check matrix and due to their cyclic structure, it is assumed that they have a great benefit for the use in energy-constrained and low-latency modern wireless communication and data storage systems.

Since the rediscovery of LDPC codes in the late 1990s, research communities have focused their interest on binary LDPC codes, because long binary LDPC codes can achieve performance approaching the Shannon limit, especially when the irregularity is introduced into their graph structure [137], at the cost of practical and low-cost implementations. But in 1998, Davey and MacKay [30] showed that non-binary LDPC codes over $GF(q)$ for $q > 2$ perform well compared to their binary counterparts. Moreover, non-binary LDPC codes outperform binary codes on channels with burst errors and are very suitable for high order modulation schemes. All these advantages have motivated many works on the design of non-binary codes [24, 74, 169, 196, 205]. Similarly to their binary counterparts, non-binary LDPC codes with no short cycles in their corresponding Tanner Graph are OSMLD codes. Although non-binary codes have some advantages over their binary counterparts, unfortunately their decoding complexity is a significant challenge. The iterative hard and soft reliability based majority-logic decoding algorithms are very attractive decoding schemes for non-binary LDPC codes, intuitively, the construction of algebraic non-binary OSMLD codes is a challenging task for future industrialization over many standards of telecommunications.

Based on the pioneering works of MacWilliams on the idempotents and the Mattson-Solomon polynomials [111, 112], it was shown that every cyclic code may be generated from its dual code, using a parity-check Idempotent polynomial $E(x)$, built from the cyclotomic cosets modulo n, as an alternative to its generator polynomial $g(x)$. In fact the generation of a cyclic code is based on a single cyclotomic coset, or the union (modulo 2 addition) of a set of cyclotomic cosets, with the property $E(x) = E(x)^2$ over $GF(2^s)$, where s is a positive integer such that $s > 0$. This parity-check idempotent is a multiple of a polynomial $h(x)$ such that $deg(h(x)) = k$, and a zero minimal degree, that is $E(x) = m(x)h(x)$, k being the code dimension and $m(x)$ is a monomial. The parity-check idempotent polynomial $E(x)$ directly allows the calculation of the generator polynomial of the code $\mathscr{C}$ and its dual parity-check matrix. However, in order to achieve high code lengths with good minimum distances, the search for a feasible idempotent $E(x)$ may becomes more difficult due to the significant increase of the number of cyclotomic cosets modulo n, thus an increase in the search space dimension.

Based on this method, an algebraic exhaustive search based construction algorithm was pro-

posed in [64, 168–170] for the construction of binary and non-binary cyclic codes. However, several limitations were observed for achieving diverse codes especially for high code lengths. For this purpose, we propose to design a flexible construction algorithm able to reduce the search space dimension in order to construct new moderate and high dimensional cyclic OSMLD codes with increasing correction capacities.

This chapter is organized as follows. In Section 3.2, background and definitions are introduced. Section 3.3 presents the constraints imposed on the construction of a feasible parity-check idempotent $E(x)$ to generate cyclic OSMLD Codes. A literature review on the existing exhaustive search based construction is presented in Section 3.4. The problematic is explained in Section 3.5 as well as the analysis of the search space. We introduce the proposed construction algorithm in Section 3.6, where the problem modeling is explained as well as the genetic algorithm proposed to solve the construction problem. Section 3.7 presents the obtained results. Finally, in Section 3.8, our conclusions of this chapter are drawn.

3.2 Cyclotomic Cosets and Parity-Check Idempotents

This section is dedicated to the introduction of the background and some definitions that will be useful throughout this chapter. We will define some combinatorial objects, namely Singer Difference-Sets [155] and Golomb rulers [35]. Additionally, we will go through the introduction of the algebraic background, used for the construction of cyclic codes. Note that for sake of consistency, we will use different notations of these objects in contrast to the literature, in order to simplify the correspondence between these designs and the codes.

3.2.1 Singer Difference Sets and Golomb Rulers

Singer Difference-Sets represent a particular class of combinatorial designs. These objects are derived from projective geometries, and are defined as blocks of symmetric balanced incomplete block designs. Each block of a symmetric BIBD is a difference-set, and optimal difference-sets are called perfect difference-sets. Their construction is based on the theorem of Singer, which is used to design projective geometries.

Definition 3.1 (Difference-Sets)**.** Let G be an additively written group of order n. A J-subset D of G is a $(n,J,\lambda;v)$-difference set of order $v = J - \lambda$ if every nonzero element of G has exactly λ representations as a difference $d_i - d_j (d_i, d_j \in D)$. The difference set is

abelian, cyclic, etc., if the group G has the respective property. The redundant parameter v is sometimes omitted; therefore, the notion of (n,J,λ)-difference sets is also used.

Fact 3.1. *If D is a difference-set of size J over a group of order n, then the following equality must be satisfied:*

$$J(J-1) = n-1 \tag{3.1}$$

Example 3.1. The set $D = \{0,1,3\}$ is an $(7,3,1)$-difference set in the group $\mathbb{Z}_7$. It corresponds to an hyperplane of the projective plane $PG(2,2)$, called Fano plane.

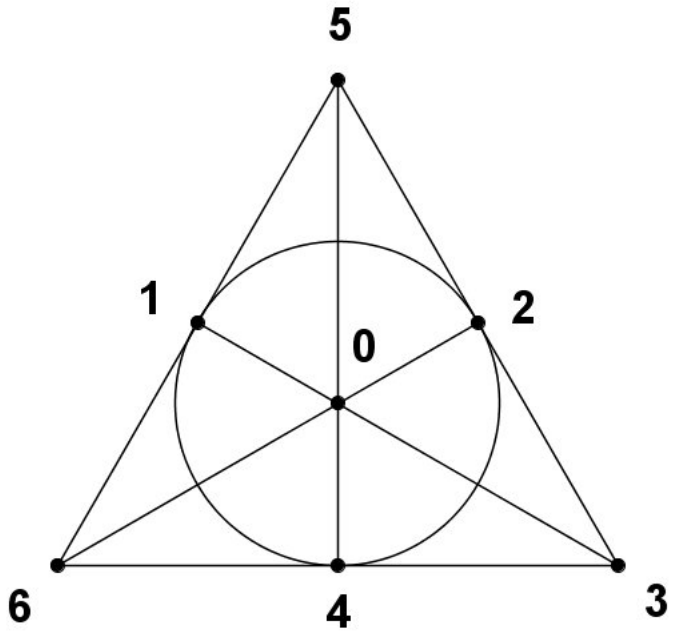

Figure 3.1: The Fano Plane $PG(2,2)$

The incidence matrix of the projective plane $PG(2,2)$ is given by:

$$A_D = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Note that every block of A_D is the incidence block of a difference-set with the same parameters as D. We say that these difference-sets are equivalent.

Example 3.2. The complement of the difference-set D corresponding to $PG(2,2)$ is $D' = \{2,4,5,6\}$, which is again a difference-set in $\mathbb{Z}_7$, called biplane, with parameters $(v,k,\lambda) = (7,4,2)$. Its incidence matrix is given by:

$$A_{D'} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Similarly to Example 3.8, each block of $A_{D'}$ is a difference-set, and they are all equivalent.

Definition 3.2. A set of integers $\{a_i : 0 \leq a_i < n, 1 \leq i \leq J\}$ is called a **Golomb ruler** modulo n with J marks, or simply a modular Golomb ruler when n and J are clear from the context, if the differences $(a_i - a_j) mod\, n$ are distinct for all ordered pairs (i, j) with $i \neq j$.

- A modular Golomb ruler modulo n with J marks satisfies the following inequality:

$$J(J-1) \leq n-1$$

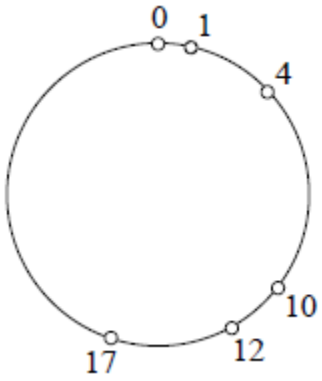

Figure 3.2: A modular Golomb ruler modulo $n = 31$ with $J = 6$ marks

3.2.2 Cyclotomic Cosets and Parity-check Idempotents

We will introduce here the notion of cyclotomic cosets and parity-check idempotents, as well as other definitions useful for this purpose. Note that these definitions will be used in the next sections of this chapter in the construction of cyclic OSMLD codes. Basic algebraic background on the groups, rings and finite fields are out of scope of this chapter, and they can be found in [98, 146].

Definition 3.3 (q-th order Cyclotomic Cosets)**.** Let u and n be two odd positive integers, the q^{th} order cyclotomic cosets associated with u modulo n is defined by:

$$C_u = \left\{ q^i u \, mod\,(n) \,|\, 0 \leq i \leq t \right\} \tag{3.2}$$

where t represents the integer for which $q^{t+1} u \, mod\,(n) = u$. The integer u is the smallest element of the coset C_u and is called Coset Leader.

Definition 3.4. Let $\mathscr{F}$ be the set that contains the leaders of all cyclotomic cosets modulo n, thus the property $\bigcup_{u \subseteq \mathscr{F}} C_u = \{0, 1, 2, \ldots, n-1\}$ holds.

Definition 3.5 (Primitive idempotent)**.** All the cyclotomic cosets modulo n corresponds to idempotent polynomials $C_u(x)$, called **Primitive idempotents**, with the property:

$$C_u(x) = C_u(x^2) \text{ for all } u \in \mathscr{F} \tag{3.3}$$

Theorem 3.1. *Let's define the quotient ring* A_m[1] *by* $A_m = GF(2^m)[x]/(x^n - 1)$. *Let* $\{C_i(x)\}_{i=1,\ldots,n_c}$ *be the set of primitive idempotents of* A_m. *Then* $x^n - 1$ *has* n_c *distinct monic irreducible factors*

[1]When $m = 1$, so that binary codes are considered, the notation A_m is simplified to A.

over $GF(2^m)$*, namely*

$$\gcd(x^n-1, C_i(x)-1), \quad i=1,...,n_c \tag{3.4}$$

Example 3.3. Let $n=7$ and $A=GF(2)[x]/(x^7-1)$. Then the polynomial x^7+1 has the following irreducible factors :

$$x^7+1=(x+1)\left(x^3+x+1\right)\left(x^3+x^2+1\right)$$

Clearly, here we have $n_c=3$ monic irreducible factors, corresponding to n_c cyclotomic cosets of order 2 modulo $n=7$. These cyclotomic cosets are those illustrated in Example 3.10. It shows that the size of each cyclotomic coset corresponds to the order of its associated monic irreducible polynomial of x^7-1.

Definition 3.6. The number n_c of primitive idempotents of A is equal to

$$n_c=\sum_{d/n}\frac{\varphi(d)}{o_2(d)} \tag{3.5}$$

where φ is the Euler's function, $o_2(d)$ stands for the order of q in the multiplicative group $\mathbb{Z}/d\mathbb{Z}$, and d are the the integers that divide n.

Example 3.4. Let's consider $n=7$ and $q=2$. The set of 2-th order cyclotomic cosets modulo 7 is given by:

- $C_0=\{0\}$
- $C_1=\{1,2,4\}$
- $C_3=\{3,5,6\}$

Their associated primitive idempotents over the quotient ring A are given by:

- $C_0(x)=1$
- $C_1(x)=x+x^2+x^4$
- $C_3(x)=x^3+x^5+x^6$

Clearly, we have $n_c=3$ and $\mathscr{F}=\{0,1,3\}$. Definition 3.31 holds; i.e. $\sum_{u\subseteq\mathscr{F}} C_u=\{0,1,2,3,4,5,6\}$.

Definition 3.7 (Parity-check idempotent)**.** Let Ω be a subset of $\mathscr{F}$. The polynomial $E(x)$ such that:

$$E(x) = \sum_{u \in \Omega} C_u(x) \tag{3.6}$$

can be considered as a **parity-check idempotent polynomial** of a cyclic code $\mathscr{C}$.

Definition 3.8 (Difference-enumerator polynomial)**.** Let $D(E(x))$ be the **Difference Enumerator Polynomial** of the parity-check idempotent $E(x)$ defined by:

$$D(E(x)) = E(x)E(x^{-1}) = d_0 + d_1 x + ... + d_{n-1}x^{n-1} \tag{3.7}$$

where $d_0 = J$, the terms d_i denote the integer coefficients of $D(E(x))$ for $0 \leq i \leq n-1$, and x is the indeterminate. Thus, another expression of the difference-enumerator polynomial takes the following form:

$$D(E(x)) = J + \sum_{i=1}^{n-1} d_i x^i \tag{3.8}$$

Definition 3.9 (Mattson-Solomon Polynomial)**.** If $a(x) \in A$, the Mattson-Solomon polynomial of $a(x)$ is the mapping from $a(x)$ to $A(z)$ and is defined by [111]:

$$A(z) = MS(a(x)) = \sum_{j=0}^{n-1} a(\alpha^{-rj}) z^j \tag{3.9}$$

where α is a primitive element over the finite field $GF(2^m)$, the field of the quotient ring A, and z is the indeterminate.

Note that the Mattson-Solomon transform is the equivalent of Fourrier Transform in finite fields.

3.2.3 Constraints on the Parity-Check Idempotent

For the construction of cyclic OSMLD codes based on the parity-check idempotent, a set of conditions on $E(x)$ must be satisfied. As shown in Definition 3.34, the parity-check idempotent polynomial is designed based on the union (addition over $GF(q)$) of a set of primitive idempotents. As primitive idempotent are directly related to cyclotomic cosets modulo n, the construction of the set Ω is out interest. Note that every set Ω can lead to generate a cyclic code of length n. However, for the generation of OSMLD codes with the orthogonality property, many mathematical conditions are imposed on the construction of the parity-check idempotent polynomial $E(x)$. For this purpose, and based on the works launched in [64, 168–170],

we will briefly describe the mathematical constraints for generating a feasible parity-check idempotent $E(x)$ for the generation of a cyclic OSMLD code.

Next Lemma describes the constraints that control the feasibility of a parity-check idempotent polynomial $E(x)$ for generating a cyclic OSMLD code.

Lemma 3.1. *We call a parity-check idempotent polynomial $E(x) \in GF(q)^n$* ***feasible*** *if the following conditions are satisfied (proof in [170]):*

1. *$E(x)$ must be an idempotent. i.e. $E(x) = E(x^2)$.*
2. *The weight J of $E(x)$ must satisfy $J(J-1) \leq n-1$ for the construction of a OSMLD code of length n.*
3. *Exponents of $E(x)$ must not contain a common factor of n other than unity. Example: $n = 63$ and $E(x) = C_9(x) = x^9 + x^{18} + x^{36} = x^9(1 + x^9 + x^{27})$ contains factors of 63.*
4. *All differences between the exponents of $D(E(x))$ must be distinct. i.e. $d_i \in \{0,1\}$ for all $i \in \{1,2,...n-1\}$.*
   a) *If $d_i = 1$ for all $i \in \{1,2,...,n-1\}$, then $E(x)$ can be used to generate a cyclic DS code.*

The proof of Lemma 3.2 can be found in [170].

Following MacWilliams, a code $\mathscr{C}$ may be defined by the parity-check idempotent $E(x)$ as an alternative to its generator polynomial $g(x)$. Since $E(x)$ is in the dual code, it an be written as

$$E(x) = m(x)h(x) \tag{3.10}$$

, and the generator polynomial $g(x)$ of the code $\mathscr{C}$ is simply deduced by the euclidean division of the unity $x^n - 1$ and the generating idempotent $E_c(x) = 1 + E(x)$ as follows:

$$g(x) = gcd(x^n - 1, E_c(x)) \tag{3.11}$$

Note that cyclic OSMLD codes can be considered and used as LDPC codes, due to sparsity of the parity-check matrix H. A parity-check matrix could be obtained directly by performing n cyclic shifts of the polynomial $x^{deg(E(x))}E(x^{-1})$. The parity-check matrix H of a cyclic OSMLD code contains for every symbol index $0 \leq j < n$, a set of parity-check equations

orthogonal on it, denoted as $\mathscr{M}(j)$. Due to the orthogonality property, the minimum distance of the OSMLD code $\mathscr{C}$ is easily defined, and is given by:

$$d_{min} = J + 1 \tag{3.12}$$

such that:

$$J = wt(E(x)) \tag{3.13}$$

The mathematical consequences of the orthogonality property is given in the following lemma [168]. The proof of Lemma 3.3 was given in [128, Theorem 10.1].

Lemma 3.2. *Let d_i for $0 \leq i < n$ denote the coefficients of $D(E(x))$. If $d_i \in \{0,1\}$, for all $i \in \{1,2,...,n-1\}$, the parity-check polynomial derived from $E^*(x)$ is orthogonal on each .position in the n-tuple. Consequently:*

1. *the minimum distance of the resulting OSMLD code is $d_{min} = 1 + wt(E^*(x))$, and*
2. *the underlying Tanner Graph of its equivalent LDPC code has girth of at least* 6.

Proof. 1) Let a codeword $c(x) = c_0 + c_1x + ... + c_{n-1}x^{n-1}$ and $c(x) \in GF(2^m)$. For each non zero bit position c_j of $c(x)$, where $j \in \{0,1,...,n-1\}$, there are $wt_H(E(x))$ parity-check equations orthogonal to position c_j. Each of the parity-check equations must check another non zero bit c_l, where $l \neq j$, so that the equation is satisfied. Clearly, $wt_H(c(x))$ must equal to $1 + wt_H(E(x))$ and this is the minimum weight of all codewords.
2) The direct consequence of having orthogonal parity-check equation is the absence of cycles of length 4 in the Tanner Graphs. Let a, b and c, where $a < b < c$, be three distinct coordinates in an n-tuple, since $d_i \in \{0,1\}$ for $1 \leq i \leq n-1$, this implies that $b - a \neq c - b$. It is known that $q(b-a)\ (mod\, n) \in \{1,2,...,n-1\}$ and thus, $q(b-a)\ (mod\, n) \equiv (c-b)$ for some integer $q \in \{1,2,...,n-1\}$. If the integers a, b and c are associated with some variable vertices in the Tanner graph, a length 6 cycle is formed. □

In the next example, we give the construction of a binary cyclic OSMLD code with parameters $(n,k,d_{min}) = (21,11,6)$, following the method described above.

Example 3.5. Let's fix a code length $n = 21$. From the the code length n, it is possible to construct a completely orthogonalizable difference-set code with a parity-check idempotent of weight $J = 5$ and a code length $n = 21$, due to the fact that the equality $J(J-1) = n-1$ is verified.

The classical construction of the generator polynomial of this cyclic code comes from the factorization of $x^{21}-1$ over $A = GF(2)[x]/(x^{21}-1)$. The factorization is given as follows:

$$x^{21}+1 = (x+1)\left(x^2+x+1\right)\left(x^3+x^2+1\right)\left(x^3+x+1\right)\left(x^6+x^4+x^2+x+1\right)\left(x^6+x^5+x^4+x^2+1\right)$$

By following the dual code construction technique, the set C of 2-th order cyclotomic cosets modulo $n = 21$ is given by:

- $C_0 = \{0\}$
- $C_1 = \{1,2,4,8,11,16\}$
- $C_3 = \{3,6,12\}$
- $C_5 = \{5,10,13,17,19,20\}$
- $C_7 = \{7,14\}$
- $C_9 = \{9,15,18\}$

The primitive idempotent polynomials associated to the set C is given by:

- $C_0(x) = 1$
- $C_1(x) = x+x^2+x^4+x^8+x^{11}+x^{16}$
- $C_3(x) = x^3+x^6+x^{12}$
- $C_5(x) = x^5+x^{10}+x^{13}+x^{17}+x^{19}+x^{20}$
- $C_7(x) = x^7+x^{14}$
- $C_9(x) = x^9+x^{15}+x^{18}$

Clearly, the number of cyclotomic cosets modulo $n = 21$, or equivalently, the number of primitive idempotent polynomials over $A = GF(2)[x]/(x^{21}-1)$ is $n_c = 6$. Moreover, the set $\mathscr{F}$ is defined by $\mathscr{F} = \{0,1,3,5,7,9\}$. Following the respective weights of the n_c primitive idempotents, it is straightforward that we have two possibilities for the construction of a parity-check idempotent $E(x)$ with a weight $J = 5$. These two possible choices are:

$$\begin{aligned} E_1(x) &= C_3(x)+C_7(x) \\ &= x^3+x^6+x^7+x^{12}+x^{14} \end{aligned}$$

and

$$\begin{aligned} E_2(x) &= C_7(x) + C_9(x) \\ &= x^7 + x^9 + x^{14} + x^{15} + x^{18} \end{aligned}$$

In other words, we have: $\Omega_1 = \{3,7\}$ and $\Omega_2 = \{7,9\}$. By calculating the difference enumerator polynomials $D(E_1(x))$ and $D(E_2(x))$, we find that:

$$D(E_1(x)) = D(E_2(x)) = 5 + x + x^2 + ... + x^{20}$$

which shows that these difference-enumerator polynomials take the following form given in (3.8):

$$D(E(x)) = J + \sum_{i=1}^{n-1} x^i$$

Consequently, the parity-check idempotents $E_1(x)$ or $E_2(x)$ may be used to generate a cyclic binary DS code with parameters $(n,k,d_{min}) = (21,11,6)$. If we consider the parity-check idempotent $E_1(x)$, the header of its corresponding parity-check matrix H is obtained as follows:

$$E^*(x) = x^{deg(E(x))} E(x^{-1}) = 1 + x^2 + x^7 + x^8 + x^{11} \tag{3.14}$$

Thus, the parity-check matrix H is obtained by applying n cyclic shifts to the incidence vector of the reciprocal polynomial $E^*(x)$. The parity-check matrix corresponding to the code $\mathscr{C}$ is sparse and completely regular, where the row and column weights are equal to $J = 5$. Also, all the orthogonal parity-check equations orthogonal on each position $0 \leq j < n$ are included, such that each symbol has $J = 5$ orthogonal equations on it. The subset of the equations that are orthogonal on the last position $j = n-1$ is given by:

$$H_{n-1} = \begin{cases} x^9 + x^{11} + x^{16} + x^{17} + x^{20} \\ x^2 + x^{12} + x^{14} + x^{19} + x^{20} \\ 1 + x^3 + x^{13} + x^{15} + x^{20} \\ x^4 + x^5 + x^8 + x^{18} + x^{20} \\ x + x^6 + x^7 + x^{10} + x^{20} \end{cases}$$

The set H_{n-1} includes the orthogonality property on the last symbol, where the later is checked in all the orthogonal equations, while the other digits are not checked more than

once. Due to the cyclic structure of the OSMLD code, the set H_{n-1} can be used to decode all the symbol digits, by applying n cyclic permutations to the received binary sequence being decoded. Note that all the symbols x^j for $0 \leq j < n$ participate in the orthogonal equations of the set H_{n-1}, thus the OSMLD code $\mathscr{C}$ is said to be **completely orthogonalizable**.

3.3 Exhaustive Search based Construction

The authors in [168,170] have developed an exhaustive search idempotent based construction method of binary LDPC codes with no cycles of length 4. The key parameters they took into consideration in devising this technique are the code rate and the minimum distances. This construction method results in many OSMLD codes with various code rates and minimum distances. However, due to the exhaustive search, a key limitation of this method is the difficulty to achieve high minimum distances, especially for high code lengths, as high minimum distances directly implies a larger size of the Ω constructed upon the union of cyclotomic cosets. From their obtained results, it was shown that the authors were not successful to achieve a diversity in code lengths and minimum distances, provided that the construction algorithm requires a higher time and computational complexities for choosing different sets Ω and testing the conditions stated in Lemma 3.2.

A key feature of the cyclotomic coset-based construction is the ability to increment the minimum Hamming distance of a code by adding further weight from other idempotents and so steadily decrease the sparseness of the resulting parity-check matrix. Despite the construction method has a feature of producing LDPC codes with no cycles of length 4, it is important to remark that codes that have cycle of length 4 in their parity-check matrix do not necessarily have bad performance under iterative decoding and a similar finding has been demonstrated in [164]. It has been observed that there are many cyclotomic coset-based LDPC codes that have this property and the constraints given in Lemma 3.2 can be easily relaxed to allow the construction of cyclic LDPC codes with girth 4 [168, 169].

Similarly to the binary case, authors in [64, 169] introduced another exhaustive search construction method of non-binary LDPC codes over $GF(2^m)$ with no short cycles. In contrast to binary OSMLD codes, the non-binary codes are based on the non-binary cyclotomic cosets modulo n, constructed following the Lemma 31.

Before introducing the non-binary primitive idempotent, we will briefly give some preliminaries that will be useful. Let's denote by $\mathrm{A_m(x)}$ the set of polynomials of degree at most $n-1$ with coefficients in $GF(2^m)$. Let m and m' be two positive integers such that $m \mid m'$,

where $GF(2^m)$ is a sub-field of $GF(2^{m'})$. Let n be a positive odd integer, representing the code length, and conversely, let $GF(2^{m'})$ be the splitting field for $x^n - 1$ over $GF(2^m)$, so that $n \mid 2^{m'} - 1$. Let $r = (2^{m'} - 1)/n$, $l = (2^{m'} - 1)/(2^m - 1)$, α a primitive element and a generator for $GF(2^{m'})$, and β be a generator for $GF(2^m)$, where $\beta = \alpha^l$.

With the given preliminaries, the construction of non-binary primitive idempotents over $A_m(x)$ is given in Lemma 3.4.

Lemma 3.3. *The non-binary primitive idempotent polynomial* $C_u(x) \in \mathrm{A_m(x)}$ *is given by:*

$$C_u(x) = \sum_{0 \le i \le |C_u|-1} e_{C_{u,i}} x^{C_{u,i}} \tag{3.15}$$

where $|C_u|$ *is the number of elements in* C_u *and* $e_{C_{u,i}}$ *is defined by:*

- $i = 0$: $e_{C_{u,i}} \in \{1, \beta, \beta^2, ..., \beta^{2^m-2}\}$
- $i > 0$: $e_{C_{u,i}} = e^2_{C_{u,i-1}}$

Definition 3.10. If $e(x) \in \mathrm{A_{m'}(x)}$ then the Mattson-Solomon finite field transform of $e(x)$ is:

$$E(z) = MS(e(x)) = \sum_{j=0}^{n-1} e(\alpha^{-rj}) z^j \tag{3.16}$$

where $E(z) \in \mathrm{A_{m'}(z)}$. The inverse Mattson-Solomon transform is given by:

$$e(x) = MS^{-1}(E(z)) = \frac{1}{n} \sum_{j=0}^{n-1} E(\alpha^{rj}) x^i \tag{3.17}$$

This transform is widely known in coding theory. The number of non-zero elements of the Mattson-Solomon polynomial of an idempotent gives the dimension k of the cyclic code $\mathscr{C}$. Also, this transform is usual for the determination of the BCH lower-bound of the minimum distance d_{min} of the code, which is equal to the maximal number of consecutive ones in $E(z)$. It is known that if $E(x)$ is an idempotent, then $E(z) = MS(E(x)) \in \mathrm{A_1(z)}$ is a binary polynomial [128, Chapter 8]. From the Mattson-Solomon polynomial, the bound on the minimum distance of a code $\mathscr{C}$ is determined following Definition 3.38 [64, 128].

Definition 3.11. The minimal distance of a cyclic code $\mathscr{C}$ over $GF(2^m)$ is bounded by:

$$d_0 \le d_{min} \le \min\left(wt(g(x)),\ J+1\right) \tag{3.18}$$

where d_0 denotes the maximum number of the consecutive run of ones in $MS(E(x))$ taken cyclically modulo n.

Similarly to cyclic binary OSMLD codes, we state in the next proposition the correspondence between a feasible parity-check idempotent and combinatorial designs, namely difference-set and modular Golomb rulers.

Proposition 3.1. *The parity-check idempotent $E(x) \in A_m(x)$ is the parity-check polynomial for a cyclic (binary/non-binary) OSMLD code $\mathscr{C}$ if the set of the exponents of $E(x)$ forms a modular Golomb Ruler, or a Singer difference-set.*

The construction of such designs with long parameters for finding a feasible $E(x)$ has been widely investigated in the literature, due to the difficulty provided by large space dimensions. This problem is addressed especially for cyclic OSMLD codes that are not derived from PG or EG.

The algebraic construction method of cyclic OSMLD over $GF(2^m)$ is based on the following steps:

1. Fix the integers m and n.
2. If $m = 1$, go to step 6, otherwise continue.
3. Find the splitting field $GF(2^{m'})$ of $x^n - 1$ over $GF(2^m)$, such that $m \mid m'$.
4. Generate the cyclotomic cosets modulo $2^{m'} - 1$ and denote it C'.
5. Construct all elements of $GF(2^m)$ using $p(x)$ as primitive polynomial and β as a primitive element. The primitive polynomial $p(x)$ is derived from C' by choosing the smallest integer $u \in \mathscr{F}$ such that $|C'_u| = m$. Then the minimal polynomial of α^u is given by:
$$p(x) = \prod_{0 \le i < m} (x + \alpha^{C'_{u,i}})$$
6. Generate the set of cyclotomic cosets modulo n over $\mathrm{A_m(x)}$ following Lemma 3.4, and denote it C. Then, generate the solution, which is a non-empty set $\Omega \subset \mathscr{F}$, giving the parity-check idempotent polynomial $E(x)$.
7. Calculate the Mattson-Solomon transform $MS(E(x)) \in A_1(z)$ and obtain the lower-bound of the minimal distance d_{min} and the dimension of the code (optional).
8. Calculate the generator polynomial $g(x) \in \mathrm{A_m(x)}$ of the resulting code, given by:
$$g(x) = gcd(x^n - 1, 1 + E(x))$$

9. Generate the set ω of $J = wt(E(x))$ dual code orthogonal parity-check equations on the last symbol position $n-1$, given by, for $1 \leq j \leq J-1$

$$\omega_j(x) = (\beta x)^i \omega_0(x)$$

where $\omega_0(x)$ is the first orthogonal parity-check equation given by $\omega_0(x) = x^{deg(E(x))}E(x^{-1})$ and i is an index running over all the exponents of $h(x)$ such that $E(x) = m(x)h(x)$, such that $m(x)$ is a monomial.

The step 7 that includes the calculation of the Mattson-Solomon (MS) transform of the parity-check idempotent is optional, since the dimension of the code $\mathscr{C}$ is obtained in the next step (Step 8), and its minimum distance can be correctly lower-bounded by $J+1$. This step will be effective when linear (including LDPC) codes construction is of interest, excluding the orthogonality property, and hence the conditions on the design of the idempotent can be relaxed.

Next we give an example (Example 3.13) of the construction of a non-binary BCH code over $GF(4)$ with a code length $n = 15$. We will show that this codes is a cyclic NB-OSMLD code. This code is also derived from the Euclidean geometry.

Example 3.6. Let $m = 2$,$m' = 4$ and $n = 15$. The field $GF(16)$ is the splitting field of $x^n - 1$ over $GF(4)$. We generate the set C' of cyclotomic cosets modulo $2^{m'} - 1 = 15$. The smallest coset leader u such that $|C'_u| = m$ is $u = 5$. Then we derive a minimal polynomial from C'_1, defined by

$$p(x) = (x+\alpha^5)(x+\alpha^{10}) = 1+x+x^2$$

Now, we can generate all elements of $GF(4)$. The elements of $GF(4)$ are given by:

$$\begin{aligned} 0: &\quad 0 \quad 0 \\ 1: &\quad 1 \quad 0 \\ \beta: &\quad 0 \quad 1 \\ \beta^2: &\quad 1 \quad 1 \end{aligned}$$

Next, we generate the cyclotomic cosets modulo $n = 15$. There are $n_c = 5$ cosets. Then we assign coefficients from $\{1, \beta, \beta^2\}$ following Lemma 3.4. Now we have to find the set Ω in order to construct a feasible parity-check idempotent $E(x)$ for the construction of a cyclic NB-OSMLD code. Suppose that we have chosen initial coefficients $e_{C_{u,0}}$ following

Lemma 3.4 as: $\{1,\beta,\beta^2,1\}$, respectively excluding the cyclotomic coset $C_0=\{0\}$. Hence, the primitive idempotents are defined by:

$$\begin{aligned} C_1(x) &= x+x^2+x^4+x^8 \\ C_3(x) &= \beta x^3+\beta^2 x^6+\beta^2 x^9+\beta x^{12} \\ C_5(x) &= \beta^2 x^5+\beta x^{10} \\ C_7(x) &= x^7+x^{11}+x^{13}+x^{14} \end{aligned}$$

We introduce the idempotent weight enumerator $\psi(x)$ here defined by

$$\psi(x)=1+x^2+3x^4$$

, such that its coefficients represent the number of occurrences of the weights represented by the exponents, in the cyclotomic cosets modulo n. This polynomial indicates that there is one cyclotomic coset with weight 1, one with weight 2 and three primitive idempotents with weight 4. An essential condition to construct OSMLD codes, is that the weight of $E(x)$, which is a sum of primitive idempotents indexed by Ω, must satisfy the condition $J(J-1)\leq n-1$. Clearly, there are 4 choices of the set Ω, and two possible weights of $E(x)$, given by $J=2$ and $J=4$.

- Suppose we fix $J=2$, and $\Omega=\{5\}$. In fact, we have a parity-check idempotent $E(x)=\beta^2x^5+\beta x^{10}$. Following Definition 3.37, its Mattson-Solomon transform is equal to:

  $$E(z)=1+x+x^3+x^4+x^6+x^7+x^9+x^{10}+x^{12}+x^{13}$$

  and shows that the number of consecutive ones is equal to $d_0=2$, therefore $d_{min}\geq 2$, which is the lower-bound on the minimal distance of the code. Also, the weight of $E(z)$ shows that it is possible to construct a NB-OSMLD code with dimension $k=n-wt(E(z))=5$. In fact, the BCH code with parameters $(n,k,d_{min})=(15,5,3)$ can be constructed from $E(x)$ with $J=2$ orthogonal equations. Its generator polynomial is of degree $deg(g(x))=n-k=10$, and is defined by:

  $$g(x)=gcd(x^n-1,E(x)-1)$$

- Now suppose that we fix $J=4$, and $\Omega=\{1\}$. The obtained parity-check idempotent is $E(x)=x+x^2+x^4+x^8$, and its MS transform is:

  $$E(z)=MS(E(x))=x+x^2+x^3+x^4+x^6+x^8+x^9+x^{12}$$

Clearly, $E(x)$ is the parity-check idempotent of a cyclic NB-OSMLD code with parameters $(n,k,d_{min}) = (15,7,5)$ over $GF(4)$.

Authors in [64,168,169] have developed a construction algorithm which exhaustively searches for all non-degenerate cyclic LDPC codes of length n which have orthogonal parity-check polynomial. The exhaustive search algorithm is described in Algorithm 3.1.

Algorithm 3.1 Exhaustive Search cyclotomic cosets based construction algorithm

Input: $n \leftarrow$ block length (odd integer)
index $\leftarrow$ an integer initialized to -1
$V \leftarrow$ a vector initialized to $\oslash$
$S \leftarrow \mathscr{F}$ excluding 0
Output: CodesList contains set of cyclic codes which have orthogonal parity-check polynomial

1: $T \leftarrow V$
2: **for** $i = index+1; i \leq |S|; i++$ **do**
3: $T_{prev} \leftarrow T$
4: **if** ($\sum_{\forall t \in T} |C_{S_t}| \leq \sqrt{n}$, S_t is the t^{th} element of S) **then**
5: Append i to T
6: $u(x) = \sum_{\forall t \in T} e_{S_t}(x)$
7: **if** ($u(x)$ is non degenerate) **and** ($u(x)$ is orthogonal on each position) **then**
8: $U(z) = MS(u(x))$
9: $k = n - wt_H(U(z))$
10: $C \leftarrow$ an $[n,k,1+wt_H(u(x))]$ cyclic code defined by $u(x)$
11: **if** ($k \geq \frac{1}{4}$) **and** ($C \notin CodeList$) **then**
12: Add C to $CodeList$
13: **end if**
14: **end if**
15: $CodeSearch(T,i)$
16: **end if**
17: $T \leftarrow T_{prev}$
18: **end for**

3.4 Problematic and Analysis of the Search Space

As shown in the previous section, the construction of a feasible parity-check idempotent $E(x)$ for generating a cyclic OSMLD code over $GF(2^m)$ is constrained by a set of mathematical

conditions. The design of $E(x)$ is particularly based on finding a set Ω of primitive idempotents indexes. The choice of Ω must be carried out appropriately following Lemma 3.2.

Consequently, we will show that the exhaustive search construction algorithm leads to many limitations. The number n_c of cyclotomic cosets modulo n, or equivalently, the number of primitive idempotents over A is a key parameter for analyzing the space search of a feasible parity-check idempotent. Tables 3.1 illustrate the evolution of the number n_c of cyclotomic cosets modulo n for a set of cyclic OSMLD EG and DS codes, respectively. Table 3.3 presents for a set of DS codes the corresponding space search dimension S for a given value of J. In addition, Figure 3.3 depicts the evolution of n_c with respect to the code length n. The selected values of n include all odd integers such that $7 \leq n \leq 4161$, and the analytical results using Definition 3.33 are compared with those obtained experimentally.

n	k	d_{min}	J	n_c
63	37	9	8	**12**
255	175	17	16	**34**
511	199	19	18	**58**
1023	781	33	32	**106**
2047	1023	34	33	**186**
4095	2135	38	37	**350**
8191	4199	40	39	**630**
16383	8207	57	56	**1180**

Table 3.1: A set of cyclic EG OSMLD codes with their corresponding number of cyclotomic cosets n_c

Indeed, we can observe from Figure 3.3 that the parameter n_c increases significantly with the code length n. This assumption is straightforward as the number of irreducible factors over A of the polynomial $x^n - 1$ increase with n. As a consequence, an exhaustive search for the feasible set Ω leads to a combinatorial explosion due to the large search space dimension S as shown in Table 3.3. These limitations explain why the design of long algebraic cyclic OSMLD codes is actually a difficult task.

Also, from Figure 3.3, it is shown that the value of n_c increases significantly from moderate to long code lengths. We particularly observe the first peak of the curve at $n = 127$, followed by several peak transitions, then from $n = 2000$, the value of n_c begins to increase exponentially. This figure also shows that the analytical expression for calculating the number of cyclotomic cosets modulo n coincides exactly with the values obtained by numerical simulations.

n	k	d_{min}	J	$r=\frac{k}{n}$	n_c
7	3	4	3	0.43	**2**
21	11	6	5	0.52	**5**
73	45	10	9	0.62	**8**
273	191	18	17	0.70	**26**
1057	813	34	33	0.77	**72**
4161	3431	66	65	0.82	**236**
16513	14325	130	129	0.87	**788**
65793	59231	258	257	0.90	**2756**

Table 3.2: A set of cyclic DS OSMLD codes with their corresponding number of cyclotomic cosets n_c

n	J	n_c	S	Approximate value of S
7	3	3	2	2
21	5	6	3	3
73	9	9	8	8
273	17	27	23	23
1057	33	73	$2\times\binom{2}{70}$	4830
4161	65	237	$8\times\binom{3}{228}$	1.55×10^{7}
16513	129	789	$2\times\binom{12}{786}$	2.13×10^{26}
65793	257	2757	$2\times\binom{12}{2730}$	6.98×10^{32}

Table 3.3: Space search evolution for different values of n for a set of cyclic DS codes

Intuitively, for simplifying the construction, we can derive the idempotent weights enumerator polynomial with coefficients that represent the number of primitive idempotents over T_m with a weight equal to their exponents.

We state the definition of the primitive idempotent weights enumerator as follows:

Definition 3.12. Let $T_m = GF(2^m)[x]/(x^n-1)$. The distribution of the weights of all primitive idempotents over T_m is defined by:

$$\psi(x) = x + \sum_{u\in\mathscr{F}-\{0\}} a_{|C_u|}x^{|C_u|} \tag{3.19}$$

where $|C_u|$ denotes the weight of the cyclotomic coset C_u for $u\in\mathscr{F}$, and $a_{|C_u|}$ is the number of cyclotomic cosets with a weight equal to $|C_u|$, such that $\sum_{u\in\mathscr{F}} a_{|C_u|} = n_c$.

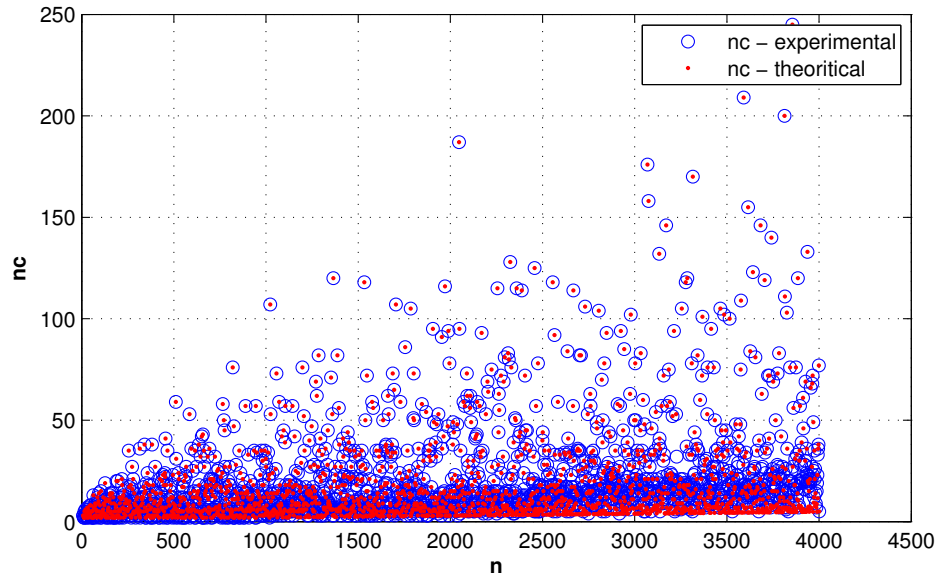

Figure 3.3: The evolution of n_c (analytical versus experimental) with respect to the code lengths n for $7 \leq n \leq 4161$

This polynomial will be useful for enumerating the corresponding weights of primitive idempotents over T_m, in addition to the enumeration of equivalent cyclic OSMLD codes over T_m. We note that with the given definition, $\psi(x)$ is a polynomial representation of a partition of the integer n. We will use this polynomial later for the reduction of the dimension of the search space.

Tables 3.4 and 3.5 display a set of code lengths associated to difference-set codes and EG OSMLD codes respectively, over $GF(2)$, with their corresponding primitive idempotent weights polynomials.

n	n_c	$\psi(x)$
7	3	$x+2x^3$
21	6	$x+x^2+2x^3+2x^6$
73	9	$x+8x^9$
273	27	$x+x^2+2x^3+2x^6+21x^{12}$
1057	73	$x+2x^3+70x^{15}$
4161	237	$x+x^2+8x^9+227x^{18}$

Table 3.4: A set of DS code lengths with their associated idempotent weight polynomials

n	n_c	$\psi(x)$
15	5	$x+x^2+3x^4$
31	7	$x+6x^5$
63	13	$x+x^2+2x^3+9x^6$
127	19	$x+18x^7$
255	35	$x+x^2+3x^4+30x^8$
1023	107	$x+x^2+6x^5+99x^{10}$

Table 3.5: A set of EG OSMLD code lengths with their associated idempotent weight polynomials

3.5 Construction of cyclic OSMLD codes using Genetic Algorithms

Motivated by the fact that a more sophisticated search strategy must be introduced to solve the problem of designing a feasible parity-check idempotent for generating cyclic OSMLD codes, we address our interest to apply an evolution strategy for the exploration of the space search. Intuitively, we propose the use of genetic algorithms to address this problem.

Genetic algorithms (GAs) were first invented by Holland in [63]. Inspired by the natural mechanisms of adaptation and biological evolution, GAs provide good performances in solving global optimization problems. With genetic operators, it is possible to optimize the process of finding a global optimum of a function, by overcoming the local optimums using the diversity provided by these operators.

Holland's GA is a method for moving from one population of chromosomes (e.g., strings of ones and zeros, or bits) to a new population by using a kind of natural selection together with the genetic-inspired operators of **crossover**, **mutation**, and **inversion**. Each chromosome consists of genes (e.g., bits), each gene being an instance of a particular allele (e.g., 0 or 1). The selection operator chooses those chromosomes in the population that will be allowed to reproduce, and on average the fitter chromosomes produce more offspring than the less fit ones. Crossover exchanges sub-parts of two chromosomes, roughly mimicking biological recombination between two single chromosome (haploid) organisms; mutation randomly changes the allele values of some locations in the chromosome; and inversion reverses the order of a contiguous section of the chromosome, thus rearranging the order in which genes are arrayed [115].

3.5.1 Problem modeling

The mathematical modeling of the problem of the parity-check idempotent design is a key parameter for applying a genetic algorithm. We propose an objective function $f(x_1,x_2,...,x_M)$ with $M = card(\Omega)$ variables, such that $x_i \in \mathscr{F}$ for $i \in \{1,2,...,M\}$. This means that our individuals are non zero integers vectors $(x_1,x_2,...,x_M)$ of size M. The chromosomes (elements x_i for $i \in \{1,2,...,M\}$) represent cosets leaders of the corresponding cyclotomic coset C_{x_i} contained in the parity-check idempotent $E(x)$. Therefore, individuals and their corresponding fitness functions may be formulated as follows:

Proposition 3.2. *For an individual $(x_1,x_2,...,x_M)$ such that $J = wt(E(x))$, it is possible to correspond a fitness function defined by:*

$$f(x_1,x_2,...,x_M) = J(J-1) - wt_1(D(E(x))) \tag{3.20}$$

to the construction problem of the parity-check idempotent for OSMLD codes, where $wt_1(D(E(x)))$ denotes the number of differences d_i such that $d_i = 1$ for $i \in \{1,2,...,n-1\}$.

The objective function f is a metric that indicates the number of repeated differences in the exponents of the parity-check idempotent $E(x)$, which occurs more than once. According to Lemma 3.2, the constraint $d_i \in \{0,1\}$ for $i \in \{1,2,...,n-1\}$ must be satisfied for cyclic OSMLD codes and in this case, it is clear that the value of the evaluation function $f(x_1,x_2,...,x_M)$ must be equal to zero. Therefore, the construction of the parity-check idempotent $E(x)$ for generating a cyclic binary OSMLD code is equivalent to the following minimization problem:

$$\begin{aligned} &\text{Find } \Omega = (x_1,x_2,...,x_M),\ x_i \in \mathscr{F} \\ &\text{s.t. } f(x) = J(J-1) - wt_1(D(E(x))) = 0 \end{aligned} \tag{3.21}$$

where $E(x) = \sum_{u\in\Omega}\sum_{i\in C_u} x^i$ is the parity-check idempotent constructed from the sum of the cyclotomic cosets modulo n associated to the subset Ω.

3.5.2 OSMLD-GA construction algorithm for cyclic OSMLD codes over $GF(2^{m\geq 1})$

The proposed algorithm here aims to exploit the standard steps in the process of genetic algorithms. For each code length n, the algorithm searches for feasible cyclic OSMLD codes, with incremental value of M. When M increases, then J also tends to increase. In fact,

we obtain codes with higher minimum distances by increasing M. When the parameter M reaches a saturated value, then the algorithm stops. A saturated value of M means that it is impossible to find a set of M primitive idempotents contained in $E(x)$ such that the constraint $J(J-1) \leq n-1$ is satisfied. For high code lengths, we introduce a parameter N_{tr} which represents the number of trials of the genetic algorithm in each individual value M for a given code length n. This parameter helps to improve the performance and to obtain diversified results in each instance of the algorithm.

For the GA process, the principle is to generate an initial population of N_i individuals. The parameter N_i is called the *population size*. Then the value of the fitness function $f(x_1, x_2, ..., x_M)$ is calculated for each individual $E^{(i)}(x)$. If one of the individuals has a zero fitness value, the algorithm returns the current individual as solution, then extracts the code parameters. Alternatively, if the initial population does not contain a solution, the genetic algorithm is executed according to the following steps:

- Selection of parent individuals from the current population.
- Crossing parents for creating children. The crossing is performed according to a crossover function $f_{crossover}$ and a crossover probability p_c.
- Mutation of the individual children to ensure a genetic diversity for future generations. The mutation is performed according to a mutation function $f_{mutation}$ and a mutation probability p_m.
- Calculation of the value of the fitness function for each child.
- New generation based on elitism. The elitism is an operation that selects the best individuals of the current generation based on their fitness values.

The same steps are repeated for different values of M to produce different weights of $E(x)$. For each value of M, we perform N_{tr} iterations of the algorithm, in order to find a larger set of possible combinations of the weights of primitive idempotents. For the mutation operator, as individuals are not binary elements, we have used a custom mutation function. The proposed mutation consists of selecting chromosomes following the mutation probability p_m, then for each selected chromosome $c^{(i)}$, a random value is chosen from the set $\{\mathscr{F}^{(i)} \setminus v(c^{(i)})\}$, where $v(c^{(i)})$ denotes the value of the current chromosome $c^{(i)}$.

Stopping Criteria:

For a given value of M, the construction algorithm OSMLD-GA stops when the maximum number of generations N_g is achieved, or when a feasible solution is found for the current value of M. Note also that when the fitness function is zero, the parity-check idempotent is checked whether it leads to a degenerated code, in which case the solution is discarded.

In order to reduce the complexity of the problem, i.e. the search space dimension, we discard the cyclotomic cosets whose size J does not meet the condition 2 in Lemma 3.2, i.e. a new reduced set $\dot{C}$of cyclotomic cosets is formed such that $\dot{C} = \{C_i, |C_i|.(|C_i| - 1) \leq n - 1\}$.

The hyper-parameters of the proposed algorithm (OSMLD-GA) are illustrated in Table 3.6. Note that these parameters are optimized by simulations for each code length. For each interval of code length, we have found that the hyper-parameters may vary. As a consequence, we have chosen to represent here a set of sub-optimal hyper-parameters that give satisfactory results, without representing the exact values for each code length interval.

Parameter	**Value**
Fitness function	$f(x_1,...,x_M) = J(J-1) - wt_1(D(E(x)))$
Population size N_i	512
Mutation operator	Random $p_m = [0,...,1]$
Crossover operator	Random $p_c = [0,...,1]$
Selection operator	Roulette
Elite count	$\left[N_i \times \frac{20}{100}\right]$
Max generations N_g	$100 \times M$

Table 3.6: The hyper-parameters of the construction algorithm OSMLD-GA

For the cyclic non-binary OSMLD codes, we propose two different construction approaches. The first one (Construction A) is that derived from extension fields, and explained in 3.4, which provides some limitations on the possible codelengths. The second one (Construction B) is based on the cyclotomic cosets of order q modulo n, and leads to a large set of non-binary OSMLD codes over $GF(q)$. In addition, the constructed NB-OSMLD codes using

this method have binary polynomial generators and dual orthogonal equations, which is very a useful feature for reducing the data storage requirements of non-binary codes, in contrast to most NB codes known in the literature. We summarize these two methods as follows:

1. **Method A:** Based on the extension fields (few code lengths).
2. **Method B:** Based on the cyclotomic cosets of order q modulo n (very large set of code lengths). The generator polynomial and the orthogonal equations are binary, which is very advantageous in reducing the storage requirements of non-binary codes.

For illustrating the construction method B, we give an explanatory example below.

Example 3.7. Let's consider a code length $n = 15$ and $A_2(x) = GF(2^2)[x]/(x^{15}-1)$, so that $q = 4$. We would like to construct an OSMLD code of length $n = 15$ defined over $GF(4)$. The set of 4-th order cyclotomic cosets modulo 15 is given by:

- $C_0 = 0$
- $C_1 = \{1,4\}$
- $C_2 = \{2,8\}$
- $C_3 = \{3,12\}$
- $C_4 = \{5\}$
- $C_5 = \{6,9\}$
- $C_6 = \{7,13\}$
- $C_7 = \{10\}$
- $C_8 = \{11,14\}$

Clearly, we have $n_c = 9$, and we observe that the union of all the cosets leads the set $\{0,1,...,n-1\}$, i.e. $\underset{u\subseteq\mathscr{F}}{\cup} C_u = \{0,1,2,...,n-1\}$. Now, let $\Omega = \{1,2\}$, so that the idempotent weight is $J = 2+2 = 4$. The parity-check idempotent is given by $E = \{1,2,4,8\}$, and its associated polynomial is $E(x) = x+x^2+x^4+x^8$. We have $E(x) = E(x^2) \bmod 15$, thus $E(x)$ is an idempotent polynomial. Moreover, all the differences between the exponents of $E(x)$ are distinct,

thus $E(x)$ is a feasible parity-check idempotent for generating an OSMLD code with parameters $(n,k,d_{min})_4 = (15,7,5)$ over $GF(4)$. The generator polynomial of the code is simply $1+E(x)$ and is given by:

$$g(x) = 1+x+x^2+x^4+x^8$$

The $J = 4$ dual parity-check equations orthogonal on the last digit (the 14^{th} digit) are given in the binary form below:

$$H_{n-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

As shown, despite that the code is defined over $GF(4)$, the generator polynomial and the dual parity-check equations are binary. This is a major advantage of this construction, in addition to the fact that it leads to a large set of cyclic NB-OSMLD codes. As the code is cyclic, only the set H_{n-1} is used to decode all the digit positions, by cyclically shifting the decoded sequence. Note that this construction requires few storage memory due to the cyclic structure and the binary form of the code generators.

The proposed construction algorithm OSMLD-GA is described in details in Algorithm 3.2. Note that both the binary and non-binary constructions (construction A and B) are consolidated in the same algorithm, as the only difference between these approaches is the order of the cyclotomic cosets as well as the method of choosing their coefficients.

Algorithm 3.2 OSMLD-GA

Input: $n, m, q = 2^m, M_{max}, N_{trials}, N_i, N_G, p_c, p_m, F_{limit} = 0, CodesList = \oslash$
Output: *CodesList*

/*- - - - - - - - - - - - - Initialization - - - - - - - - - - - - -*/
if $q > 2$ **then**
  if Construction = "A" **then**
    Find m', then generate the extension field $GF(2^{m'})$.
    Generate the set C of cyclotomic cosets of order 2 modulo n.
  else if Construction = "B" **then**
    Generate the set C of cyclotomic cosets of order q modulo n.
  end if
else
  Generate the set C of cyclotomic cosets of order 2 modulo n.
end if
Form the reduced set $\dot{C}$ of cyclotomic cosets and set $M_{max} = n_c$, where $n_c = card(\dot{C})$.
Construct the primitive idempotents over T_m.
/*- - - - - - - - - - - - - - - Process - - - - - - - - - - - - - - -*/
for $M = 1 : M_{max}$ **do**
  for $trials = 1 : N_{trials}$ **do**
    for $g = 1 : N_G$ **do**
      Generate the population P_g of the current generation
      for all $\Omega_z \in P_g$ **do**
        Set $E_z(x)$
        $J_z \leftarrow wt(E_z(x))$
      end for
      $cpt \leftarrow false$
      for all $E_z(x)$ **do**
        if $J_z(J_z - 1) \leq n - 1$ **then**
          $D(E_z(x)) \leftarrow E_z(x)E_z^{-1}(x)$
          $F_z = J_z(J_z - 1) - wt_1(D(E_z(x)))$
          if $F_z = F_{limit}$ and *isNoDegenerated* **then**
            $cpt \leftarrow true$
            Add $E_z(x)$ to E_{list}
          end if
        end if
      end for
      if $cpt = false$ **then**
        Selection of parents
        Crossover to produce children
        Mutation of children
        Calculate the Fitness of each child
        Elitism to generate the next population
      else
        break
      end if
    end for
    for all $E(x) \in E_{list}$ **do**
      $E(z) = MS(E(x))$
      $k = n - wt(E(z))$
      $\mathscr{C} \leftarrow$ an $[n, k, 1 + wt(E(x))]$ cyclic OSMLD code
      if $(\frac{k}{n} > \frac{1}{2})$ **and** $(\mathscr{C} \notin CodesList)$ **then**
        Add $\mathscr{C}$ to *CodesList*
      end if
    end for
  end for
  if $CodesList = \oslash$ **then**
    Exit the algorithm
  end if
end for

Note that we restrict the construction to only cyclic OSMLD codes with rates $r > \frac{1}{2}$, such that the coderate is calculated directly after computing the Mattson-Solomon transform, from which the code dimension is deducted by $k = wt(E(z))$. The Mattson-Solomon calculation procedure is optional (only used in Construction A) as the code dimension can be directly deducted from the generator polynomial $g(x)$ of the constructed code.

3.6 Construction results

The construction results are presented in this section. The proposed construction algorithm OSMLD-GA has resulted in many new binary and non-binary cyclic OSMLD codes with various parameters, as well as a large set of equivalent cyclic OSMLD codes for the same parameters. Equivalent OSMLD codes were obtained because of the global optimization aspect of genetic algorithms, where different regions of the search space are explored in the same instance. Note that the execution time of the OSMLD-GA algorithm is very fast and requires very low time complexity compared to the existing construction techniques. We highlight the fact that long cyclic OSMLD codes were easily constructed using the OSMLD-GA algorithm.

Table 3.7 displays an illustrative and non-exhaustive set of the obtained cyclic binary OSMLD (and LDPC) codes using the OSMLD-GA algorithm. Tables 3.8 and 3.9 present a set of cyclic non-binary OSMLD codes constructed using the NB-OSMLD-GA algorithm, using Contruction A, over $GF(4)$ and $GF(8)$, respectively. In Tables 3.10 to 3.14, a set of cyclic non-binary OSMLD codes constructed by Construction B are illustrated, with alphabets defined over $GF(4)$, $GF(8)$, $GF(16)$, $GF(32)$ and $GF(64)$, respectively. The parameters of these codes are also included, namely the codelength, dimension, minimum distance and the code rate as well as the parity-check idempotent index set Ω. Note that this displayed set of codes was chosen just for sake of illustration, as we have obtained a very large number of new cyclic OSMLD codes. Is is noted from the results that the coderate generally increases with the minimum distance for short codes, while the opposite is observed for long codes. However, this observation is not general. In our case the most interesting cyclic OSMLD codes are those with higher rates. Note also that cyclic difference-set codes and cyclic EG codes were obtained using this construction, and they are included on the Appendix B. For the same codelength, we have obtained several codes that are equivalent as well as nonequivalent with different coderates and correction capacities. For spatial constraints, we have only selected to represent one OSMLD code for each n.

The complete list of the constructed cyclic OSMLD codes will be presented in an online database in the near future, including also a set of quasi-cyclic OSMLD codes derived from combinatorial designs. A longer list of a set of the designed cyclic OSMLD codes with various codelengths is included in the Appendix B.

n	k	d_{min}	J	$r=\frac{k}{n}$	$\Omega_{E(x)}$
315	213	19	18	0.68	$\{3,35\}$
357	227	20	19	0.64	$\{51,85,119,133\}$
765	637	25	24	0.83	$\{127\}$
803	559	29	28	0.70	$\{33,73,275\}$
1157	815	35	34	0.71	$\{39,89,429\}$
1287	819	37	36	0.64	$\{117,275,319,429\}$
1335	947	37	36	0.71	$\{89,195,267,285,445,623\}$
1519	1095	40	39	0.72	$\{21,31,217\}$
1683	1251	41	40	0.74	$\{153,165,187,297,363\}$
1755	1453	43	42	0.83	$\{59,195\}$
2415	1995	50	49	0.83	$\{105,161,253,345,525,575,805\}$
2937	2135	55	54	0.73	$\{33,89,267,363,445,979\}$
3471	2873	60	59	0.83	$\{39,89,195,623,1157,1287\}$
4005	3303	64	63	0.82	$\{75,89,267,445,801,1485,1869\}$
4539	3797	68	67	0.84	$\{51,89,255,445,979,1513,1683,1691\}$
5073	4259	72	71	0.84	$\{57,89,445,513,741,1691\}$
5225	4865	73	72	0.93	$\{95,275,285,475,1045\}$
7353	5993	81	80	0.81	$\{19,129,645,2451\}$
7755	6375	85	84	0.82	$\{165,235,517,705,825,1175,3619\}$
8855	7567	92	91	0.85	$\{161,253,385,483,805,1771,1925,3795\}$
9597	7677	97	96	0.80	$\{105,457,1371,2285,3199,4113\}$
9821	6887	100	99	0.70	$\{61,161,1403,4209\}$
10341	8785	103	102	0.84	$\{171,183,915,1159\}$
12533	9871	113	112	0.79	$\{151,1245,2905\}$
16513	14325	130	129	0.87	$\{1,89,947,965,1699,2737,7077\}$
65793	59231	258	257	0.90	$\{1,1205,1309,1395,1749,2209,3713,4427,6931,9399,9553,11529,21931\}$

Table 3.7: A set of cyclic binary OSMLD codes constructed using the OSMLD-GA algorithm

m'	n	k	d_{min}	$r = \frac{k}{n}$	Ω	$e_{C_{u,0}}$
12	7	3	4	0.43	$\{1\}$	$\{1\}$
	15	7	5	0.47	$\{7\}$	$\{1\}$
	21	11	6	0.52	$\{3,7\}$	$\{1,1\}$
10	31	15	6	0.48	$\{11\}$	$\{1\}$
6	63	53	8	0.84	$\{5,21\}$	$\{\beta^2,\beta\}$
12		37	9	0.59	$\{1,21\}$	$\{\beta^2,1\}$
8	85	37	9	0.44	$\{7\}$	$\{1\}$
10	93	47	8	0.51	$\{3,31\}$	$\{1,\beta^2\}$
8	255	223	13	0.87	$\{17,43\}$	$\{\beta^2,\beta^2\}$
		175	17	0.69	$\{27,43\}$	$\{1,1\}$
12	315	279	11	0.89	$\{21,115\}$	$\{1,\beta\}$
10	341	320	11	0.94	$\{17\}$	$\{\beta^2\}$
		205	16	0.60	$\{35,55\}$	$\{1,1\}$
12	819	689	21	0.84	$\{65,273,307\}$	$\{\beta^2,\beta,1\}$
	1365	1327	28	0.97	$\{149,293,585\}$	$\{\beta^2,1,\beta^2\}$

Table 3.8: A set of cyclic non-binary OSMLD codes over $GF(4)$ constructed using the NB-OSMLD-GA algorithm (Construction A)

m'	n	k	d_{min}	$r = \frac{k}{n}$	Ω	$e_{C_{u,0}}$
15	7	3	4	0.43	$\{1\}$	$\{\beta^4\}$
12	15	7	5	0.47	$\{1\}$	$\{1\}$
12	21	11	6	0.52	$\{3,7\}$	$\{\beta^6,1\}$
15	31	15	6	0.48	$\{3\}$	$\{1\}$
12	63	47	7	0.75	$\{13\}$	$\{\beta^3\}$
6		37	9	0.59	$\{1,21\}$	$\{\beta,1\}$
9	73	45	10	0.62	$\{17\}$	$\{1\}$
12	105	97	5	0.92	$\{45,49\}$	$\{\beta^4,1\}$
		53	8	0.50	$\{15,49\}$	$\{1,1\}$
15	217	201	6	0.93	$\{77,93\}$	$\{1,\beta^6\}$
		109	9	0.50	$\{31,77\}$	$\{\beta,1\}$
12	273	239	18	0.88	$\{39,41,91\}$	$\{\beta^5,\beta^2,1\}$
12	819	789	21	0.96	$\{143,273,275\}$	$\{\beta^3,\beta^3,\beta^2\}$
12	4095	2199	31	0.54	$\{85,311,845\}$	$\{\beta^3,1,1\}$

Table 3.9: A set of cyclic non-binary OSMLD codes over $GF(8)$ constructed using the NB-OSMLD-GA algorithm (Construction A)

n	k	d_{min}	J	$r = \frac{k}{n}$	$\Omega_{E(x)}$
93	61	11	10	0.66	$\{5,10\}$
273	191	18	17	0.70	$\{39,91,97,101,182\}$
465	337	21	20	0.72	$\{53,106\}$
803	559	29	28	0.70	$\{55,73,146,187\}$
945	777	31	30	0.82	$\{5,10,21,42\}$
1661	1231	41	40	0.74	$\{11,151,302,407\}$
1963	1471	43	42	0.75	$\{91,151,302,481\}$
2313	1929	49	48	0.83	$\{191,271\}$
2667	2163	49	48	0.81	$\{95,127,190,254\}$
2945	2195	53	52	0.74	$\{155,310,437,551,589,665,1045,1178\}$
3311	2371	59	58	0.72	$\{11,22,301,473,602,1419\}$
3995	2605	63	62	0.65	$\{85,141,282,425,705,1410\}$
4305	3157	64	63	0.73	$\{43,86,1845\}$
4743	3351	70	69	0.71	$\{69,138,341,459,682\}$
4781	3545	70	69	0.74	$\{9,18,683\}$
5083	3581	71	70	0.70	$\{115,221,230,1105,1265,2139\}$
5635	3991	74	73	0.71	$\{105,345,483,966,1127,2254,2415\}$
5735	4005	76	75	0.70	$\{155,185,310,333,555,666,1147,1295,2294\}$

Table 3.10: A set of cyclic non-binary OSMLD codes over $GF(4)$ constructed using the OSMLD-GA algorithm (Construction B)

n	k	d_{min}	J	$r = \frac{k}{n}$	$\Omega_{E(x)}$
255	175	17	16	0.69	$\{31,39\}$
279	167	17	16	0.60	$\{31,45,62,99,124\}$
595	403	25	24	0.68	$\{3,6,12\}$
651	491	26	25	0.75	$\{99,161,198,291\}$
715	497	27	26	0.70	$\{11,22,44,65,143\}$
889	697	29	28	0.78	$\{11,22,44,49\}$
935	697	31	30	0.75	$\{33,85,143,187\}$
1157	815	35	34	0.70	$\{13,89,143,178,356\}$
1419	1029	39	38	0.73	$\{99,121,129\}$
1495	1073	39	38	0.72	$\{65,69,138,276,299,325\}$
1677	1229	41	40	0.73	$\{39,91,129,258,516\}$
1905	1521	43	42	0.80	$\{57,355\}$
2451	1795	49	48	0.73	$\{95,129,171,258,516,817\}$
2871	2313	55	54	0.81	$\{87,99,319,435,638,1276\}$
3519	2837	60	59	0.81	$\{153,255,437,575,1127,1173\}$
3999	3291	60	59	0.82	$\{93,279,645,651,989,1333\}$
4895	4153	70	69	0.85	$\{363,445,715,979\}$
5313	3591	74	73	0.68	$\{77,115,161,230,231,299\}$

Table 3.11: A set of cyclic non-binary OSMLD codes over $GF(8)$ constructed using the OSMLD-GA algorithm (Construction B)

n	k	d_{min}	J	$r = \frac{k}{n}$	$\Omega_{E(x)}$
341	205	16	15	0.60	$\{39,55,78\}$
381	255	17	16	0.67	$\{71,127,142,254\}$
651	491	26	25	0.75	$\{33,35,70\}$
803	559	29	28	0.70	$\{73,121,146,143\}$
889	697	29	28	0.78	$\{7,97\}$
1617	1191	41	40	0.74	$\{33,77,147,154,231,294\}$
2117	1531	47	46	0.72	$\{73,87,146,292,584,725\}$
2759	2003	53	52	0.73	$\{89,93,267,445,623,979,1023,1335\}$
3311	2371	59	58	0.72	$\{297,301,451,473,602,1419\}$
4347	3071	63	62	0.71	$\{23,46,189,297\}$
4781	3545	70	69	0.74	$\{89,178,2049\}$
5037	3783	72	71	0.75	$\{365,511,621,759,943,1095,1679,1886,3358\}$

Table 3.12: A set of cyclic non-binary OSMLD codes over $GF(16)$ constructed using the OSMLD-GA algorithm (Construction B)

n	k	d_{min}	J	$r=\frac{k}{n}$	$\Omega_{E(x)}$
315	195	19	18	0.62	$\{5,45,63,105,135\}$
357	227	20	19	0.64	$\{17,35,119,153\}$
585	377	25	24	0.64	$\{61,201\}$
657	451	27	26	0.69	$\{9,73,81,219\}$
945	657	31	30	0.70	$\{9,155\}$
1235	881	35	34	0.71	$\{19,65,247\}$
1285	833	37	36	0.65	$\{37,109,257\}$
1691	1211	41	40	0.72	$\{19,89,209\}$
1755	1459	37	36	0.83	$\{217\}$
2313	1929	49	48	0.83	$\{283\}$
2967	2187	53	52	0.74	$\{23,69,129,645,989\}$
3213	2701	43	42	0.84	$\{21,221\}$
3309	2205	59	58	0.67	$\{559\}$
3471	2873	60	59	0.83	$\{39,89,117,623,741,1157\}$
4347	2793	64	63	0.64	$\{115,189,207,345,483,621,945,1449\}$
4361	3103	66	65	0.71	$\{245,267,441,539,637\}$
4945	3449	69	68	0.70	$\{161,215,805,989,1075\}$
5035	3869	71	70	0.77	$\{265,361\}$

Table 3.13: A set of cyclic non-binary OSMLD codes over $GF(32)$ constructed using the OSMLD-GA algorithm (Construction B)

n	k	d_{min}	J	$r=\frac{k}{n}$	$\Omega_{E(x)}$
217	153	16	15	0.70	$\{27,54,108\}$
465	337	21	20	0.72	$\{23,46\}$
1513	1079	39	38	0.71	$\{85,89,153,178,267,534\}$
1581	1069	41	40	0.68	$\{35,70\}$
2225	1607	47	46	0.72	$\{75,89,178,445,825,890\}$
2263	1815	46	45	0.80	$\{227,235,470\}$
2635	1977	51	50	0.75	$\{59,85,118,1275\}$
2759	2003	53	52	0.73	$\{31,89,267,341,445,623,979,1335\}$
2829	1897	54	53	0.67	$\{23,46,205,287,615\}$
3995	2965	63	62	0.74	$\{47,85,94,425,1363,2679\}$
4945	3449	69	68	0.70	$\{115,215,230,897,989,1075,1081,1978\}$
	3155	70	69	0.64	$\{115,215,230,473,301\}$
4991	3231	71	70	0.65	$\{69,105,138,276\}$
5429	3393	72	71	0.62	$\{89,178,356,445,671,712,890\}$
5969	3907	75	74	0.65	$\{127,235,635,987,1081,1457\}$

Table 3.14: A set of cyclic non-binary OSMLD codes over $GF(64)$ constructed using the OSMLD-GA algorithm (Construction B)

3.7 Conclusion

In this chapter, we have presented a new construction algorithm based on genetic algorithms, for the design of cyclic OSMLD codes using the parity-check idempotent technique. The proposed algorithm has shown to result in a large number of new cyclic OSMLD codes with various coderates and code lengths, where long codes were constructed easily in a low time complexity, in contrast to previous exhaustive search based techniques that were presented in this chapter. The proposed construction technique has an advantage in the flexibility of parameters, where it is possible to relax the mathematical constraints easily for design other cyclic codes not necessarily with the orthogonality property.

The same construction algorithm was used to design binary and non-binary OSMLD codes, where in the NB-OSMLD-GA construction, two different methods were proposed, the method A which requires extension fields and leads to few code length possibilities, and the method B which exploits the cyclotomic cosets of order q modulo n for designing cyclic OSMLD codes over $GF(q)$. The later has two main interesting features, firstly, it leads to a very

large set of new cyclic NB-OSMLD codes, secondly, it provides an advantage in low storage requirements, as the codes are cyclic and their generator polynomials and dual orthogonal parity-check equations are written in the binary field.

The proposed formulation of the problem as an objective function leads to new investigations on how to extend this contribution to constructing irregular OSMLD codes as well as non-cyclic OSMLD codes by exploiting global optimization algorithms, and appropriately modifying the objective function by adding additional structural constraints. We note that the large database of the constructed codes are very useful for energy-constrained wireless communication systems as well as systems that require limited storage memory. For instance, one such application is the massive Machine-Type Communications (mMTC) including drone communications and Internet of Things (IoT) devices in the 5G NR cellular mobile networks, as well as other communication systems that may use these codes in concatenated (serial or parallel) forward error correction schemes, including spatial, underwater and optical communications.

Chapter 4

A New Gradient-Descent based One-Step Majority-Logic Decoding Algorithm for LDPC Codes

4.1 Introduction and Preliminaries

4.1.1 Introduction

During last decades, coding theory has seen many ever-growing efforts from academic and industrial research communities towards reaching fast and reliable communication systems for a wide range of propagation channels, while providing an efficient trade-off between performance and computational complexity. LDPC codes, first introduced by [48], and later rediscovered in [86, 109], have shown to form a class of error correcting codes approaching the theoretical capacity, while decoding with the Belief Propagation Sum-Product (BP-SP) algorithm. The BP-SP algorithm [109], along with its variants, has shown to provide a large number of arithmetic operations required for the decoding process. The most efficient LDPC codes are whose Tanner graph structure does not contain short cycles, i.e. cycles of length 4. These LDPC codes can be simply decoded with majority-logic algorithms, instead of the BP algorithm which requires more computational complexity. LDPC codes with no short cycles represents a sub-class of OSMLD codes.

OSMLD codes [145, 185] represent an interesting class of error correcting codes, due to their structured combinatorial properties providing orthogonality in the dual structure, which helps to decode these codes with a simple majority-logic decoding (MLGD) procedure. A

linear code $\mathscr{C}$ with a length, dimension and minimal distance respectively denoted by the triplet (n,k,d_{min}), is a OSMLD code if for each symbol index $0 \leq j < n$, there is a set of J orthogonal parity-check equations belonging to its dual code, such that every symbol apart the symbol j is contained in no more than one equation, and all the J equations intersect on the symbol index j. These equations are said to be orthogonal on the j^{th} symbol. As a consequence, the code $\mathscr{C}$ has a structured minimum distance given by $d_{min} \geq J+1$. When $\mathscr{C}$ is completely orthogonalizable (Difference-set codes [185]), i.e. all the digit positions participate in the decoding process of a symbol index j, its corresponding minimum distance is exactly determined by $d_{min} = J+1$. Due to the orthogonality property, decoding OSMLD codes is simpler and requires lower computational complexity compared to decoding other families of codes.

The first majority-logic decoder was devised by Reed [133] for decoding Reed-Muller codes, and was later extended by Massey [113], who proposed the first Soft-Input Hard-Output (SIHO) one-step majority-logic scheme. Majority-logic decoding and threshold decoding were also extensively studied by Massey, as well as Kolesnik and Mironchikov [84]. The weighted one-step majority-logic decoding was introduced by Rudolph [142, 143]. Lucas *et al.* [106] has shown that OSMLD codes derived from finite geometries outperforms their equivalent randomly generated LDPC codes, when decoded with the BP algorithm. Several majority-logic decoding algorithms have been proposed in the literature. In general, decoding OSMLD codes can be classified into 2 distinct categories, hard-decision and soft-decision decoding algorithms. The hard-decision MLGD algorithm (OSMLGD) introduced in [133] is the most simple decoder, and can be realized with simple logical operations. However, its error rate performances provides bad convergence, due to the absence of the soft channel output information. The use of hard-decision MLGD algorithms can only be useful for data storage systems, or when a simple decoder is needed. In counterpart, soft-decision MLGD algorithms use the channel information, and exploit it generally in an iterative scheme, with an relative increase in complexity. Several soft-reliability based MLGD algorithms have been proposed for finite geometry LDPC codes and OSMLD codes in the literature [10, 11, 69, 88, 105, 120, 158, 173, 174, 192, 207]. The comparison that will be presented in this paper in terms of performances and complexity of these algorithms, shows that MLGD schemes suffer from performance degradation compared to the BP decoding of equivalent LDPC codes, especially for long codes.

In this chapter, we propose a new MLGD algorithm, based on the Gradient-Descent (GD) optimization technique (GD-MLGD). A suitable model of the decoding problem of OSMLD

codes is proposed as a first-order derivable multi-variable objective function. An investigation of the designed objective function is addressed, and the decoding algorithm consists of iteratively maximizing this function, based on the first-order partial derivatives until achieving convergence. Also, a quantized version of the proposed algorithm (QGD-MLGD) is proposed, aiming to reduce the computational complexity. A comparison between the proposed decoding algorithms and the relevant MLGD algorithms in the literature is presented. This comparison is based on error rates performances and computational complexity. A statistical analysis of the wrong decisions of the decoder is also established, and some eventual strategies and perspectives that can be employed to improve decoding performance are investigated.

The rest of this chapter is organized as follows. In Section 2, preliminaries and basic background are briefly presented, and a literature review on the existing majority-logic decoders is provided. Section 3 presents related works on gradient-descent decoding, and introduces the proposed decoding algorithms (GD-MLGD and QGD-MLGD). A Complexity analysis and comparison with previous works is also given. Performance results are presented is Section 4, with remarks and discussions, followed by a statistical analysis and investigations on performance improvement, before concluding remarks in Section 5.

4.1.2 Preliminaries

Let $\mathscr{C}$ be a regular binary OSMLD code, with code length, dimension and minimum distance denoted respectively by (n,k,d_{min}). Let $c \in \mathbb{F}_2^n$ be a codeword, i.e. $c \in \mathscr{C}$, where $\mathbb{F}_2$ denotes the Galois field of order 2, i.e. $GF(2)$, such that $cH^T = 0$, where H is the parity-check matrix of $\mathscr{C}$ with dimension $m \times n$, such that its row and column weights are respectively denoted by ρ and γ, and $m \geq n-k$ is the number of parity-check equations of H. The null space of H is the code $\mathscr{C}$. Since $\mathscr{C}$ is an OSMLD code, its parity-check matrix H contains for each symbol index $0 \leq j < n$, a set of J binary orthogonal equations on the j^{th} digit position. Let's denote the set containing the indexes of parity-check equations orthogonal on the j^{th} digit by $\mathscr{M}(j)$, where $\mathscr{M}(j) = \{i : h_{ij} = 1, 0 \leq i < m\}$. When H is a circulant matrix, then $\mathscr{C}$ is a cyclic OSMLD code. If H consists of a set of circulant matrices, then $\mathscr{C}$ is a quasi-cyclic code. For the case of cyclic regular OSMLD codes derived from finite geometries that are considered in this paper (type-1 EG and difference-set codes), row and column weights of H are similar and equal to J [146].

Let i be an index running through the set $\mathscr{M}(j)$, and let $\mathscr{N}(i)$ be the index of the non zero digits of the i^{th} orthogonal equation on the j^{th} symbol, i.e. $\mathscr{N}(i) = \{j : h_{ij} = 1, 0 \leq j < n\}$.

We consider OSMLD codes that have the same number of orthogonal equations for each symbol. Thus, let $J = |\mathscr{M}(j)|$ denote the cardinal of the set $\mathscr{M}(j)$.

We assume a transmission over an additive white Gaussian noise (AWGN) channel using a BPSK modulation (0 is mapped to 1, and 1 to -1) such that $x_j = (-1)^{c_j}$ for $0 \leq j < n$, where $x \in \{-1,1\}^n$ is the modulated (bipolar) codeword. The transmission is performed assuming the model $y = x + e$, where $y \in \mathbb{R}^n$ is the received signal, and $e \in \mathbb{R}^n$ is the AWGN vector with zero mean and variance $\sigma^2 = N_0/2$, where $N_0/2$ is the single-sided noise-power spectral density.

Let $z = (z_0, z_1, ..., z_{n-1}) \in \mathbb{F}_2^n$ denote the hard decision of the received signal y, defined by $z_j = (1 - sgn(y_j))/2$ for each $0 \leq j < n$, where $sgn(.)$ is the sign operator. The syndrome vector $S(z) = (s_0, s_1, ..., s_{m-1}) \in \mathbb{F}_2^m$ is calculated by the scalar product $s = zH^T$ over $\mathbb{F}_2$.

Let $A_{ji} = \bigoplus_{s \in \mathscr{N}(i)} z_s$ and $B_{ji} = A_{ji} \oplus z_j$ denote respectively the i^{th} parity checksum and the i^{th} binary estimator orthogonal on the symbol z_j. Let $E_j^{(q)}$ and $R_j^{(q)}$ represent the extrinsic information and the soft-reliability on the symbol z_j at the q^{th} iteration, respectively. In general, the extrinsic information $E_j^{(q)}$ is calculated by the summation of the bipolar estimators orthogonal on the symbol index j, where a bipolar estimator is given by $1 - 2B_{ji}$, and its reliability is denoted by ω_{ji}. The reliability of the i^{th} bipolar checksum of H is denoted by ω_i.

4.2 The Existing Majority-Logic Decoding Algorithms

4.2.1 Hard Decision Majority-Logic Decoding

Majority-logic decoding (OSMLGD) was first devised by Reed [133] to decode Reed-Muller (RM) codes step by step, and was later extended by Massey in [113], by devising the threshold decoding, namely the Hard-In Hard-Out (HIHO) and Soft-In Hard-Out (SIHO) threshold majority-logic decoding algorithms. The simplest hard decision based Majority-logic decoder (OSMLGD) can be implemented with simple logic gates [133], and was proposed in two variants, A and B, where respectively, either the error or the symbol value is estimated via the orthogonal equations on each symbol. The OSMLGD algorithm was well explained and investigated in [97].

Recently in [69], an iterative hard reliability-based majority-logic decoder (HRBI-MLGD) has been proposed. Based on the same decoding rule used in OSMLGD, authors established an initialization of the hard reliability R of the received sequence, with either $-J$ or $+J$,

provided that a received bit z_j is more reliable if the value of R_j is close to $-J$ or J. Then the reliability R is updated in each iteration by accumulating the extrinsic information provided by the orthogonal parity-check equations.

4.2.2 Soft Decision Majority-Logic Decoding

The first soft reliability-based majority-logic decoder was introduced by Massey in [113]. He proposed the APP threshold decoding scheme, which is essentially developed based on the mathematical expression of the a posteriori probability (APP). In other part, the weighted one-step majority-logic decoding algorithm (WOSMLGD) was introduced by Rudolph [142,143]. Independently, Kolesnik [83] presented the WOSMLGD algorithm, as a simplification of the APP decoding rule of the threshold decoding [113], by weighting the orthogonal equations in the decoding function. Independently in [163], The WOSMLGD algorithm was also proposed with an analytical closed form of its bit error probability. The weighted OSMLGD (WOSMLGD) is described in Algorithm 4.1.

Algorithm 4.1 WOSMLGD

Input: Received signal y
Output: Binary Decoded sequence z

1: **for** $j = (0 : n-1)$ **do**
2: **for** $i = (0 : J-1)$ **do**
3: Calculate σ_{ji}
4: $w_{ji} = min_{t \in \omega_{ji} \setminus \{j\}} \{|y_t|\}$
5: **end for**
6: $E_j = \sum_{i=0}^{J-1} (2\sigma_{ji} - 1) w_{ji}$
7: Calculate the hard decision z_j of E_j
8: **end for**

Later, Lucas *et al.* [105] proposed an Iterative Decoding Algorithm (IDA) for decoding linear block codes, and has shown that OSMLD codes can be decoded with the IDA giving better performance than other families of linear codes. Their proposed algorithm uses a soft reliability which is accumulated by iterations. The calculation of the soft reliability in the IDA is derived, similarly to the APP threshold algorithm, from the development of the conditional probability expression.

Let $\sigma^2 = \frac{N_0}{2}$ denotes the single-sided noise-power spectral density. Hagenauer *et al.* [56] have interpreted the MAP probability as an extrinsic part, and an intrinsic part, namely:

$$\ln \frac{\Pr(x_j = +1)}{\Pr(x_j = -1)} = LLR_j = \gamma_{ch}.y_j + E_j(C^{\perp}) \tag{4.1}$$

where

$$\gamma_{ch} = \frac{2}{\sigma^2} \tag{4.2}$$

and the probability $\Pr(x_j = +1)$ is given by:

$$\Pr(x_j = +1) = \frac{e^{LLR_j}}{1 + e^{LLR_j}} \tag{4.3}$$

By induction and using the identity $\tanh(\frac{x}{2}) = \frac{(e^x - 1)}{(e^x + 1)}$, the extrinsic information σ_j for decoding the j^{th} symbol based on the the dual code takes the following form [56, 105]:

$$E_j(C^{\perp}) = \ln \left[\frac{1 + \sum_{b \in C^{\perp}} \prod_{s=1, s \neq j}^{n} \tanh(\frac{\gamma_{ch} y_s}{2})^{b_s}}{1 - \sum_{b \in C^{\perp}} \prod_{s=1, s \neq j}^{n} \tanh(\frac{\gamma_{ch} y_s}{2})^{b_s}} \right] \tag{4.4}$$

This extrinsic information is clearly optimal with respect to its sign. Here, all codewords of the dual code are taken into account. By the use of a set of parity-check vectors H_j orthogonal on the coordinate j, (4.4) takes the form that was derived by Gallager [48] and Massey [113], given by:

$$E_j(\mathscr{M}(j)) = \ln \left[\frac{1 + \sum_{i \in \mathscr{M}(j)} \prod_{s \in \mathscr{N}(i)} \tanh(\frac{\gamma_{ch} y_s}{2})}{1 - \sum_{b \in \mathscr{M}(j)} \prod_{s \in \mathscr{N}(i)} \tanh(\frac{\gamma_{ch} y_s}{2})} \right] \tag{4.5}$$

Using the following identity:

$$\log \frac{1 + x}{1 - x} = 2 \arctan(x) \tag{4.6}$$

, the extrinsic information is simplified and takes the new form:

$$E_j(\mathscr{M}(j)) = 2 \arctan \left(\sum_{i \in \mathscr{M}(j)} \prod_{s \in \mathscr{N}(i) \setminus \{j\}} \tanh(\frac{\gamma_{ch} y_s}{2}) \right) \tag{4.7}$$

which can be further simplified [56] to take the following final form:

$$\sigma_j(H_j) = \sum_{i \in \mathscr{M}(j)} \prod_{s \in \mathscr{N}(i) \setminus \{j\}} \tanh(\frac{\gamma_{ch} y_s}{2}) \tag{4.8}$$

At the initialization step of the IDA, the soft reliability vector $R \in \mathbb{R}^n$ is set to $R_j^{(0)} = \gamma_{ch} y_j$ for all $0 \leq j < n$. Then at each iteration $0 \leq q < I_{max}$, the soft reliability is updated by accumulating the extrinsic information E_j calculated at each iteration as follows:

1. For $0 \leq j < n$, compute the extrinsic information

$$E_j^{(q)} = \sum_{i \in \mathscr{M}(j)} \prod_{s \in \mathscr{N}(i) \setminus \{j\}} \tanh(R_s^{(q)}) \tag{4.9}$$

2. For $0 \leq j < n$, update the vector of soft reliabilities

$$R_j^{(q+1)} = R_j^{(q)} + E_j^{(q)} \tag{4.10}$$

The Iterative Decoding Algorithm (IDA) is summarized in Algorithm 4.2.

Algorithm 4.2 IDA

Input: Received signal y, $R^{(0)} = y$, I_{max}
Output: Binary Decoded sequence $z^{(q)}$

- - - - - - - - - - - Initialization - - - - - - - - - - -
$R_j^{(0)} = y_j$ for all $0 \leq j < n$
- - - - - - - - - - - Iterations - - - - - - - - - - -
for $q = (0 : I_{max})$ **do**
  Calculate s, If ($s = 0$ or $q = I_{max}$), Stop decoding
  for $(j = 0 : n-1)$ **do**
    $E_j = \sum_{i \in \mathscr{M}(j)} \prod_{s \in \mathscr{N}(i) \setminus \{j\}} tanh(R_s^{(q)})$
    $R_j^{(q+1)} = R_j^{(q)} + E_j$
  end for
end for

An iterative threshold decoding algorithm (ITD) was proposed separately for decoding OSMLD product codes in [10], and parallel concatenated OSMLD codes in [11], and later for decoding simple OSMLD codes [88]. The ITD is based on a development of the log-likelihood mathematical expression using the Bayes rule. For a transmission over an AWGN channel using a BPSK modulation, the soft-output of the j^{th} bit of the received signal y, for $0 \leq j < n$ is given by:

$$LLR_j = \ln \frac{\Pr(c_j = 1/y)}{\Pr(c_j = 0/y)} \tag{4.11}$$

where LLR_j is the Log-Likelihood Ratio of the symbol c_j. Given an OSMLD code C with J orthogonal parity-check equations, the expression (4.11) can be rewritten as follows:

$$LLR_j \simeq \ln \frac{\Pr(c_j = 1/\{B_i\})}{\Pr(c_j = 0/\{B_i\})} \tag{4.12}$$

where B_i denotes the i^{th} orthogonal estimator on the j^{th} bit, for $i \in [0,...,J]$, and B_0 is the hard decision h_j of y_j. By using Bayes theorem, (4.12) takes the form:

$$LLR_j \simeq \ln \frac{\Pr(\{B_i\}/c_j = 1) \times \Pr(c_j = 1)}{\Pr(\{B_i\}/c_j = 0) \times \Pr(c_j = 0)} \tag{4.13}$$

As the estimators are orthogonal on the j^{th} bit, the individual probabilities are all independent, and therefore (4.13) takes the following form:

$$LLR_j \simeq \sum_{i=0}^{J} \ln \frac{\Pr(\{B_i\}/c_j = 1)}{\Pr(\{B_i\}/c_j = 0)} + \ln \frac{\Pr(c_j = 1)}{\Pr(c_j = 0)} \tag{4.14}$$

Equation (4.14) can be expanded as follows:

$$LLR_j \simeq \left(\sum_{i=1}^{J} \ln \frac{\Pr(\{B_i\}/c_j = 1)}{\Pr(\{B_i\}/c_j = 0)} \right) + \left(\ln \frac{\Pr(\{B_0\}/c_j = 1)}{\Pr(\{B_0\}/c_j = 0)} \right) + \left(\ln \frac{\Pr(c_j = 1)}{\Pr(c_j = 0)} \right) \tag{4.15}$$

where the first term acts as an extrinsic information on the symbol c_j, the second term is related to the soft channel information and the third term is the a priori information.

Following [26], (4.15) can be written as follows:

$$LLR_j \simeq \sum_{i=1}^{J} (1 - 2B_i)\omega_i + ((1 - 2B_0)\omega_0) + \ln \frac{\Pr(c_j = 1)}{\Pr(c_j = 0)} \tag{4.16}$$

where

$$(1 - 2B_0)\omega_0 = \frac{4E_s}{N_0} y_j \tag{4.17}$$

and the weighting coefficients ω_i of each estimator is given by:

$$\omega_i = \ln \left[\frac{1 + \prod_{s=1}^{n_i} \tanh(\frac{L_s}{2})}{1 - \prod_{s=1}^{n_i} \tanh(\frac{L_s}{2})} \right] \tag{4.18}$$

where n_i denotes the size of the i^{th} orthogonal equation on c_j, and s is the s^{th} element of the i^{th} orthogonal equation and

$$L_s = \frac{4E_s}{N_0} . |y_s| \tag{4.19}$$

Thus, the soft output of the first iteration of the ITD algorithm is given by (4.16), which corresponds to a normalized version of the soft input, and an extrinsic information E_j estimated from the orthogonal equations on the symbol c_j, in addition to an a priori information that was omitted in the algorithm.

The Iterative Threshold Decoding (ITD) algorithm is described in details in Algorithm 4.3.

Algorithm 4.3 ITD

Input: y, $R^{(0)} = 0$, I_{max}, $\gamma = 4E_s/N_0$
Output: Binary Decoded sequence $z^{(q)}$

- - - - - - - - - - - Initialization - - - - - - - - - - -
1: $R_j^{(0)} = 0$ for all $0 \leq j < n$
- - - - - - - - - - - Iterations - - - - - - - - - - -
2: **for** $q = (0 : I_{max})$ **do**
3: $y_j^{(q)} = y_j + R_j^{(q)}$ for all $0 \leq j < n$
4: **for** $j = (0 : n-1)$ **do**
5: Calculate σ_{ji} for all $0 \leq i < J$
6: $w_{ji} = ln(\frac{1+\prod_{t \in \omega_{ji} \setminus \{j\}} tanh(\frac{\gamma}{2}|y_j^{(q)}|)}{1-\prod_{t \in \omega_{ji} \setminus \{j\}} tanh(\frac{\gamma}{2}|y_j^{(q)}|)})$ for all $0 \leq i < J$
7: Calculate $(2B_0 - 1)W_0 = 4\frac{E_s}{N_0} y_j^{(q)}$
8: $E_j = \sum_{i=0}^{J-1} (2\sigma_{ji} - 1) w_{ji}$
9: Calculate $z_j^{(q)}$ based on E_j
10: Update $R_j^{(q+1)} = \alpha(E_j + (2B_0 - 1)W_0)$
11: **end for**
12: **end for**

The soft reliability-based iterative majority-logic decoder (SRBI-MLGD) was also proposed in [69] as a soft-reliability version of the HRBI-MLGD. The SRBI algorithm exploits the soft information from the channel by using an uniform quantization function, which maps each received real value to an interval of integers. Thus, the quantization helps to reduce the computational complexity by keeping an integer-valued decoding reliability. The SRBI decoder is described in Algorithm 4.4.

Later in [120], authors presented an improved version of the SRBI-MLGD algorithm, which is the improved soft reliability-based majority-logic decoder (ISRBI-MLGD). In the ISRBI, the computation of the extrinsic information which is used to update the reliability measures of the orthogonal check-sums is improved, by weighting each orthogonal estimator. Also, a modification to the update rule was also proposed, by introducing a scaling factor α. The ISRBI-MLGD has shown to outperform the SRBI-MLGD with a slight increase in the computational complexity. The ISRBI decoder is described in Algorithm 4.5.

More recently in [174], an iterative decoding scheme was proposed, named multi-threshold decoding (MTD) algorithm, which use a difference register, and a soft-reliability related to information and parity bits in order the estimate the error. The MTD was first proposed for self-orthogonal convolutional codes [174], then improved later by the same authors in

Algorithm 4.4 SRBI

Input: Received signal y, I_{max}, b and Δ
Output: Binary Decoded sequence $z^{(q)}$

- - - - - - - - - - - Initialization - - - - - - - - - - -
Init $Q_j^{(0)}$, and $R_j^{(0)} = Q_j^{(0)}$ for all $0 \leq j < n$
- - - - - - - - - - - Iterations - - - - - - - - - - -
for $q = (0 : I_{max})$ **do**
  Calculate s, If ($s = 0$ or $q = I_{max}$), Stop decoding
  for $j = (0 : n-1)$ **do**
    Calculate σ_{ji} for all $0 \leq i < J$
    $E_j = \sum_{i=0}^{J-1}(1 - 2\sigma_{ji})$
    $R_j^{(q+1)} = R_j^{(q)} + E_j$
  end for
end for

Algorithm 4.5 ISRBI

Input: Received signal y, I_{max}, α, b and Δ
Output: Binary Decoded sequence $z^{(q)}$

- - - - - - - - - - - Initialization - - - - - - - - - - -
Init $Q_j^{(0)}$, $R_j^{(0)} = Q_j^{(0)}$ and $w_{ji} = min_{t \in \omega_{ji} \setminus \{j\}} |R_t^{(0)}|$ for all $0 \leq j < n$
- - - - - - - - - - - Iterations - - - - - - - - - - -
for $q = (0 : I_{max})$ **do**
  Calculate s, If ($s = 0$ or $q = I_{max}$), Stop decoding
  for $j = (0 : n-1)$ **do**
    Calculate σ_{ji} for all $0 \leq i < J$
    $E_j = \sum_{i=0}^{J-1}(1 - 2\sigma_{ji})\omega_{ji}$
    $R_j^{(q+1)} = R_j^{(q)} + \alpha E_j$
  end for
end for

[173]. The MTD was recently proposed for self-orthogonal block codes for the use in optical channels [207]. The MTD algorithm is described in Algorithm 4.6.

Recently in 2015, a proportional majority-logic decoding scheme was proposed [158]. Authors introduced a proportionality factor for computing the quantized reliability of symbols estimated from a proportion of the orthogonal equations.

Table 4.1 summarizes the overall decoding operations of the existing majority-logic decoding algorithms.

Algorithm 4.6 MTD

Input: Received signal y, I_{max}
Output: Binary Decoded sequence $z^{(q)}$

- - - - - - - - - - - Initialization - - - - - - - - - - -
Init $D_j = 0$ and w_{ji} for all $0 \leq j < n$
- - - - - - - - - - - Iterations - - - - - - - - - - -
for $q = (0 : I_{max})$ **do**
  Calculate s, If ($s = 0$ or $q = I_{max}$), Stop decoding
  for $j = (0 : n-1)$ **do**
    Calculate σ_{ji} for all $0 \leq i < J$
    $E_j = \sum_{i=0}^{J-1}(1-2\sigma_{ji})\omega_{ji}$
    $R_j^{(q)} = |y_j|(1-2D_j) + E_j$
    if $R_j^{(q)} > 0$ **then**
      Flip z_j and D_j
    end if
  end for
end for

Table 4.1: Decoding operations comparison of the existing MLGD algorithms

Algorithm	Initialization	E_j	$R_j^{(q)}$ update
SRBI	$R_j^{(0)} = q_j$	$E_j = \sum_{i=0}^{J-1}(1-2B_{ji})$	$R_j^{(q+1)} = R_j^{(q)} + E_j$
ISRBI	$R_j^{(0)} = q_j$	$E_j = \sum_{i=0}^{J-1}(1-2B_{ji})w_{ji}$	$R_j^{(q+1)} = R_j^{(q)} + \alpha E_j$
MTD	$D_j = 0$	$E_j = \sum_{i=0}^{J-1}(1-2A_{ji})w_{ji}$	$R_j^{(q+1)} = E_j + \|y_j\|(1-2D_j)$
IDA	$R_j^{(0)} = \gamma_{ch} y_j$	$E_j = \sum_{i=0}^{J-1}\prod_{t\in\bar{H}_{ji}\setminus\{j\}}\tanh(y_t^{(q)})$	$R_j^{(q+1)} = R_j^{(q)} + E_j$
ITD	$R_j^{(0)} = 0$	$E_j = \sum_{i=0}^{J-1}(1-2B_{ji})w_{ji}$	$R_j^{(q+1)} = \alpha(E_j + \gamma_{ch} y_j^{(q)})$ $y_j^{(q)} = y_j^{(0)} + R_j^{(q)}$

4.2.3 Gradient-Descent based Decoding

4.2.3.1 The Gradient-Descent Optimization technique

The Gradient-Descent (GD) is well known as one of the simplest local optimization techniques. GD is widely used to solve local non linear optimization problems, and recently has seen a deep interest from industrial communities due to its use in the optimization of artificial neural networks in various applications. The convergence of the GD depends essentially on the initial solution of the problem, and on the nature of the problem itself. The GD has shown to perform better for the optimization of convex objective functions.

Suppose that we look for the optimum (minimum/maximum) of an objective function $f(x)$ with n variables $x = (x_0, x_1, ..., x_{n-1})$. For the case of a maximization, the optimization problem is stated as follows:

$$\max_{x \in \mathbb{R}^n} f(x) \tag{4.20}$$

Given an initial solution $x^{(0)} = (x_0^{(0)}, x_1^{(0)}, ..., x_{n-1}^{(0)})$, the GD algorithm performs in each iteration, an update of the solution following the direction given by the gradient vector already calculated in the previous iteration, based on the following update rule:

$$x^{(q)} = x^{(q-1)} \pm \alpha \nabla f(x^{(q-1)}) \tag{4.21}$$

where q is an iteration index, $\alpha \in [0, 1]$ is the descent step, and $\nabla f(x) = (\frac{\partial f(x)}{\partial x_0}, \frac{\partial f(x)}{\partial x_1}, ..., \frac{\partial f(x)}{\partial x_{n-1}})$ is the gradient vector of partial derivatives with respect to each variable x_j for $0 \leq j < n$.

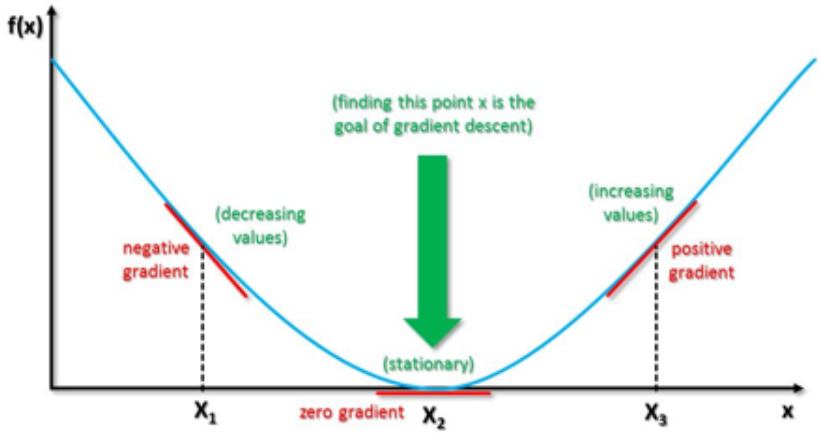

Figure 4.1: The Gradient-Descent Optimization Technique

4.2.3.2 Gradient-Descent Decoding

The first application of the GD in decoding error correcting codes was introduced by [105] for decoding linear block codes. The authors proposed the IDA and have introduced an interpretation of this algorithm as a gradient descent optimization. The extrinsic information used in IDA has shown to coincide with the first partial derivative of the considered objective function, excluding a term to be investigated. The objective function used for this purpose is

the *generalized syndrome weight*, denoted by $W_g(S_w)$, introduced in [105]. For each $0 \leq i < m$, the weighted syndrome component S_{w_i} corresponding to the i^{th} check-sum is defined by:

$$S_{w_i}(R) = \prod_{s \in \mathscr{N}(i)} \tanh(R_s) \tag{4.22}$$

and the *generalized syndrome weight* is given by the sum of all S_{w_i}'s, which takes the following form:

$$W_g(S_w) = \sum_{0 \leq i < m} S_{w_i}(R) = \sum_{0 \leq i < m} \left(\prod_{s \in \mathscr{N}(i)} \tanh(R_s) \right) \tag{4.23}$$

The evolution of the generalized syndrome weight function with various error weights and signal to noise ratio (SNR) values, for the Difference-Set (DS) code with parameters $(n,k,d_{min}) = (21,11,6)$ is illustrated in Figure 4.2.

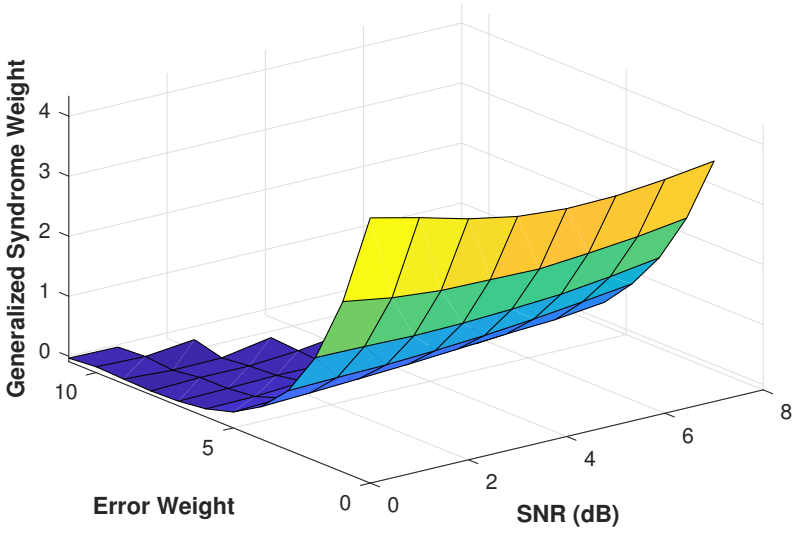

Figure 4.2: Evolution of the generalized syndrome weight with various error weights and SNRs, for the DS code $(n,k,d_{min}) = (21,11,6)$

Note that the number of codewords of this code is $2^{k=11} = 2048$. Thus for each distinct SNR and error weight, the set of 2048 codewords is transmitted multiple times over an AWGN channel, in order to guarantee that different configurations of errors has occurred, and $W_g(S_w)$ is calculated by averaging all the obtained values for each SNR value. It is shown in Figure

4.2 that when the error weight is zero (transmission without errors), then $W_g(S_w)$ takes a maximal value (yellow surface), especially in high SNR regions, and it decreases proportionally when the error weight increases.

Independently in [94], the gradient-descent algorithm was proposed for solving the *t-bounded Coset Weights problem* (t-CWP) for decoding linear codes. The authors have introduced two different variants, supported by a gradient-like decoding function. Later, an algebraic investigation of these algorithms was proposed in [132], using the Grobner representation of linear codes. After these works, a Gradient-Descent Bit Flipping (GDBF) algorithm with two variants was proposed for decoding LDPC codes [184]. The authors proposed the use of an inversion function based on the first-order Taylor development of an objective function derived from the maximum likelihood (ML) argument, and penalized by the syndrome weight, in contrast to [105] where the ML argument was not considered. Later, an improved version of GDBF, namely the Noisy Gradient-Descent Bit-Flipping (NGDBF) was proposed in [159] with its variants, where the authors introduced weighting coefficients in the objective function. Additionally, a noise perturbation was introduced in the inversion function, in order to help the algorithm to escape from local optimums. The NGDBF and its variants has shown to outperform other bit-flipping decoding algorithms in the literature. The initialization step of the GDBF decoding algorithm includes setting $x_j = sgn(y_j)$ for each $0 \leq j < n$, and is based on the following objective function:

$$f(x) = \sum_{j=0}^{n-1} x_j y_j + \sum_{i=0}^{m-1} \prod_{s \in \mathscr{N}(i)} x_s \tag{4.24}$$

where $x \in \{-1,1\}^n$ is the bipolar solution at the current iteration q.

The partial derivative of the objective function $f(x)$ with respect to the variable x_j is given by:

$$\frac{\partial}{\partial x_j} f(x) = y_j + \sum_{i \in \mathscr{M}(j)} \prod_{s \in \mathscr{N}(i) \setminus j} x_s \tag{4.25}$$

Then for each symbol x_j for $0 \leq j < n$, the flipping decision of the GDBF algorithm is based on the sign of the following inversion function:

$$x_j \frac{\partial}{\partial x_j} f(x) = x_j y_j + \sum_{i \in \mathscr{M}(j)} \prod_{s \in \mathscr{N}(i)} x_s \tag{4.26}$$

If $x_j \frac{\partial f(x)}{\partial x_j} < 0$, then the phase of the symbol x_j is flipped, providing that x_j must follow the same direction provided by the partial derivative of $f(x)$ with respect to it. The same process is repeated for all the symbols at each iteration, until reaching a solution, or achieving the maximum number of allowed decoding iterations.

For the IDA [105], the considered objective function is the generalized syndrome weight given in (4.23), and the decoding process is given in (4.9) and (4.10).

4.3 A New Gradient-Descent Majority-Logic Decoding Algorithm

4.3.1 GD-MLGD

Inspired by the works launched in [105, 159, 184], we propose to devise a majority-logic decoder based on the gradient-descent optimization technique. We will show that the gradient-descent is suitable for modeling and solving the decoding problem of OSMLD codes.

In contrast to the IDA, where only the generalized syndrome weight function was considered, we propose the use of an objective function that includes the soft syndrome weight, which is relatively simpler to calculate, in addition to a term related to the correlation between the solution and the received signal. The proposed objective function is similar to that used in the GDBF algorithm, excluding an over-scaling factor that we will introduce in order to scale between the two terms, which is beneficial for improving performance, in addition to an offset factor used to improve the accuracy of each orthogonal estimator, in order to enhance the strategy used to calculate the extrinsic information. Also, similarly to the IDA, and in contrast to the GDBF algorithm, the objective function is directly maximized using the gradient-descent technique, allowing the soft reliability calculated using the first-order partial derivatives to evolve and to be updated during each decoding iteration, until either achieving convergence, or the maximum number of allowable decoding iterations.

It is known that decoding error correcting codes is an NP-Complete problem [12, 175]. The optimal decoding rule is derived from the ML decoding. Let $\bar{\mathscr{C}}$ be the set of all bipolar codewords of the code $\mathscr{C}$. The standard ML decoding problem consists of finding the nearest bipolar codeword $x_{ML} \in \bar{\mathscr{C}}$ from to received signal y:

$$x_{ML} = \arg_{x \in \bar{\mathscr{C}}} max(\sum_{j=0}^{n-1} x_j y_j) \tag{4.27}$$

The ML argument in (4.27) is a correlation between the solution and the received signal. This function can be penalized by a term related to the syndrome weight. In [105], the generalized syndrome weight defined in (4.23) was used as the objective function of the decoding problem. We propose the use of the *soft syndrome weight* $W_m(S_w(x))$ as a penalty term to be added to the ML argument, which we define as follows:

$$W_m(S_w) = \sum_{0 \leq i < m} \bar{S}_i(x) = \sum_{0 \leq i < m} \omega_i \left(\prod_{s \in \mathscr{N}(i)} sgn(x_s) \right) \tag{4.28}$$

where $\bar{S}_i(x)$ is the i^{th} weighted bipolar syndrome component of the vector x, such that each bipolar syndrome component is weighted by a term ω_i, which represents the minimum magnitude of the soft symbols that participate in the i^{th} parity-check, which is given by:

$$\omega_i = \max(\min_{s \in \mathscr{N}(i)} \{ \mid x_s \mid \} - \theta, 0) \tag{4.29}$$

such that $\theta \in [0,1]$ is an offset parameter [73] that controls the minimum reliability ω_i at which the i^{th} decision (vote) is canceled, and $sgn(X)$ for $X \in \mathbb{R}$ is the sign operator defined by:

$$sgn(X) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{4.30}$$

Note that in (4.29), when $\min_{s \in \mathscr{N}(i)} \{ \mid x_s \mid \} < \theta$, we obtain $\omega_i = 0$, which means that the reliability of the i^{th} orthogonal equation is zero, and does not contribute in the majority vote process. This technique is useful because in some decoding cases, the reliability ω_i is very weak, however it can contributes to the emergence of erroneous decoding decisions.

The soft syndrome weight $W_m(S_w(x))$ is a decreasing function as the error weight increase. Thus, the maximization of (4.28) leads to a solution for which the hard decision is a codeword of $\mathscr{C}$. If $sgn(\hat{x}) \in \bar{\mathscr{C}}$ is a bipolar codeword, then $\prod_{s \in \mathscr{N}(i)} sgn(\hat{x}_s) = 1$ for all $0 \leq i < m$, and the soft syndrome weight function takes the following value:

$$W(S_m(\hat{x})) = \sum_{i=0}^{m-1} \omega_i \tag{4.31}$$

Figure 4.3 represents the evolution of the soft syndrome weight function with the error weight and the SNR, for a DS code $\mathscr{C}$ with parameters $(n,k,d_{min}) = (21,11,6)$. It is shown that

when the error weight is zero, i.e. transmission without errors, then $W_m(S_w)$ is maximal. The function decreases when the SNR decreases, and when the error weight is higher. We deduce that the function $W_m(S_w)$ is suitable to add to the objective function as a penalty term, for modeling the decoding problem. Note that the calculation of $W_m(S_w)$ requires less computational complexity than the generalized syndrome weight.

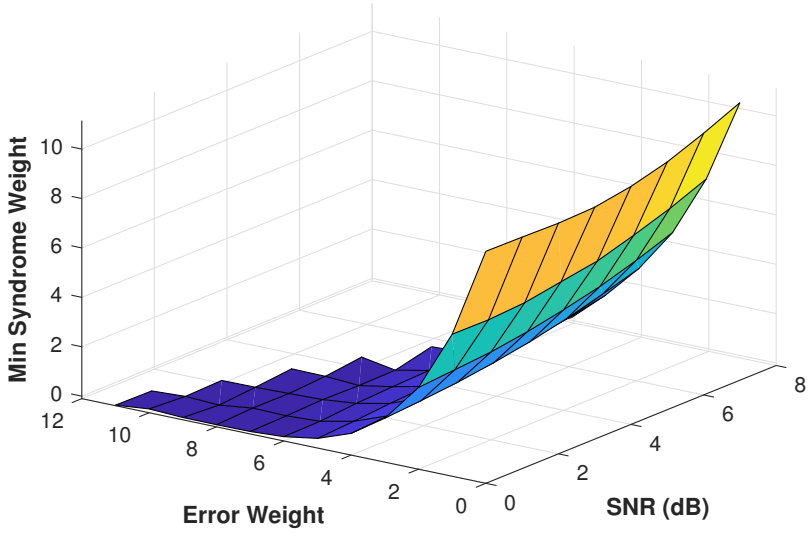

Figure 4.3: Evolution of the soft syndrome weight function $W_m(S_w)$ with the error weight and various SNRs for the DS code $(n,k,d_{min}) = (21,11,6)$

Intuitively, the decoding problem may be formulated by the maximization, with respect to $x \in \mathbb{R}^n$, of the objective function:

$$f(x) = C(x,y) + \beta W_m(S_w(x)) \tag{4.32}$$

where $C(x,y)$ is the correlation of the solution x and the received signal y, and β is an over-scaling parameter used to scale between the correlation $\mathscr{C}(x,y)$ and the soft syndrome weight function $W_m(S_w)$. The expression (4.32) takes the following form:

$$f(x) = \sum_{j=0}^{n-1} sgn(x_j)y_j + \beta \sum_{i=0}^{m-1} \omega_i \prod_{s \in \mathscr{N}(i)} sgn(x_s) \tag{4.33}$$

Note that the objective function given in (4.33) is monotone and derivable. Note also that $f(x) \in \mathbb{R}$.

The partial derivative of $f(x)$ with respect to each variable x_j is given by:

$$\frac{\partial f(x)}{\partial x_j} = y_j + \beta \sum_{i \in \mathscr{M}(j)} \omega_{ji} \prod_{s \in \mathscr{N}(i) \backslash j} sgn(x_s) \tag{4.34}$$

The first term of the partial derivative of the objective function $f(x)$ is an intrinsic information related to the channel, while the second term corresponds to the sum of the weighted bipolar estimators that are orthogonal on the j^{th} symbol.

By inspecting the expression 4.34, the bipolar syndrome components summation are weighted by the coefficients $\omega_{ji} \in \mathbb{R}$, related to the minimal magnitude of the symbols involved in the i^{th} row of H, excluding the j^{th} symbol being decoded. The weighting coefficients ω_{ji} acts here as a reliability of the i^{th} bipolar estimator on the j^{th} symbol. For decoding a symbol x_j, given the offset factor θ, the weighting coefficients are determined by:

$$\omega_{ji} = \max(\min_{s \in \mathscr{N}(i) \backslash \{j\}} \{ \mid y_s \mid \} - \theta, 0) \tag{4.35}$$

Note that the expression of ω_{ji} given in (4.35) corresponds to the partial derivative of ω_i with respect to x_j. Indeed, it is known that $\forall (a,b) \in \mathbb{R}^2$, we have:

$$f_{min}(a,b) = \min(a,b) = \begin{cases} a & \text{if } a \leq b \\ b & \text{if } a > b \end{cases} \tag{4.36}$$

The partial derivatives of the $\min(.)$ function with respect to its variables a and b are given by:

$$\frac{\delta f_{min}(a,b)}{\delta a} = \begin{cases} 1 & \text{if } a < b \\ 0 & \text{if } a > b \end{cases} \tag{4.37}$$

and

$$\frac{\delta f_{min}(a,b)}{\delta b} = \begin{cases} 0 & \text{if } a < b \\ 1 & \text{if } a > b \end{cases} \tag{4.38}$$

Similarly, the partial derivatives of the max function are obtained in the same manner. Consequently, the partial derivative of ω_i with respect to the variable x_j is given by:

$$\begin{aligned}\frac{\delta\omega_i(x)}{\delta x_j} &= \frac{\delta(\max(\min_{s\in\mathscr{N}(i)}\{|\,x_s\,|\}-\theta,0))}{\delta x_j}\\ &= \max(\min_{s\in\mathscr{N}(i)\setminus\{j\}}\{|\,x_s\,|\}-\theta,0)\\ &= \omega_{ji}\end{aligned} \tag{4.39}$$

For sake of simplicity, we will consider that each ω_{ji} term is a constant, which further simplifies the expression of the first-order partial derivative as given in (4.34).

With the given formulation, finding the maximum of (4.33) requires to solve the unconstrained optimization problem stated in the next proposition.

Proposition 4.1. *The decoding problem of OSMLD codes is equivalent to the following optimization problem:*

$$\begin{aligned}\textit{Maximize } f(x) \quad &= \quad \sum_{j=0}^{n-1} sgn(x_j)y_j + \beta\sum_{i=0}^{m-1}\omega_i\prod_{s\in\mathscr{N}(i)} sgn(x_s)\\ \textit{Subject to} \quad & x\in\mathbb{R}^n\end{aligned} \tag{4.40}$$

Proof. An optimal solution $\hat{x}$, i.e. a solution that maximizes $f(x)$ in Proposition 1 is a bipolar codeword $\hat{x}\in\bar{\mathscr{C}}$ with all the bipolar parity-check sums satisfied. Thus, the objective function value at $\hat{x}$ takes the following form:

$$f(\hat{x}) = \sum_{j=0}^{n-1}\hat{x}_j y_j + \beta\sum_{i=0}^{m-1}\omega_i \tag{4.41}$$

The objective function $f(x)$ is monotone and derivable. Its partial derivative with respect to each variable x_j is given by:

$$\frac{\partial f(x)}{\partial x_j} = y_j + \beta\sum_{i\in\mathscr{M}(j)}\omega_{ji}\prod_{s\in\mathscr{N}(i)\setminus j} sgn(x_s) \tag{4.42}$$

where the first term coincides with the intrinsic information of the j^{th} symbol being decoded, while the second term represents the sum of the weighted bipolar estimators orthogonal on the j^{th} symbol. Thus, this partial derivative can be considered as a soft reliability on the j^{th} symbol, and can be exploited in each iteration in order to correctly decode the received sample y, by including it in the gradient-descent solution update. □

The proposed gradient-descent decoding algorithm (GD-MLGD) performs in each iteration, the calculation of the gradient of (4.40) in order to exploit it for the next iteration, where the real solution $y^{(q)}$ at the q^{th} iteration is updated to a new value, following the direction given by the calculated gradient. As the updated solution $y^{(q)}$ is real-valued, we consider that the binary estimated solution is the hard decision of $y_j^{(q)}$ for all $0 \leq j < n$.

The GD-MLGD initialization step includes the initialization of the descent step parameter $\alpha \in [0,1]$, the received vector at the first iteration $y_j^{(0)} = \gamma_{ch} y_j$ where $\gamma_{ch} = \frac{2}{\sigma^2}$ is the channel reliability, and the reliability vector $R \in \mathbb{R}^n$ by a zero vector $R_j^{(0)} = 0$ for each $0 \leq j < n$.

Let I_{max} be the maximum number of allowed decoding iterations, and let q be the iteration index such that $0 \leq q < I_{max}$. At the beginning of each iteration q, the solution update is performed following:

$$y^{(q)} = y^{(q-1)} + \alpha R^{(q)} \tag{4.43}$$

The syndrome vector is then calculated based on the hard decision $z^{(q)}$ of $y^{(q)}$, and as a stopping criteria, if $S(z^{(q)}) = 0$ then we stop the decoding process and $z^{(q)}$ is returned as the decoded word. Otherwise, for each variable y_j, the soft reliability $R_j^{(q+1)}$, which is the partial derivative $\frac{\delta f(y^{(q)})}{\delta y_j^{(q)}}$ at the q^{th} iteration is calculated based on (4.42), and is given by:

$$R_j^{(q+1)} = y_j^{(0)} + \beta \left(\sum_{i \in \mathscr{M}(j)} \omega_{ji} \prod_{s \in \mathscr{N}(i) \setminus j} sgn(y_s^{(q)}) \right) \tag{4.44}$$

The GD-MLGD decoding procedure is summarized as follows:

1. Initialize the initial solution vector $y_j^{(0)} = \gamma_{ch} . y_j$ and the gradient vector $R_j^{(0)} = 0$ for all $0 \leq j < n$.
2. At each iteration q:
   a) Update the solution $y_j^{(q)} = y^{(q)} + \alpha R_j^{(q)}$ for all $0 \leq j < n$.
   b) Compute the hard decision $z^{(q)}$ of $y^{(q)}$.
   c) Calculate the syndrome vector $S(z^{(q)})$.
      i. Check if the stopping criterion $S(z^{(q)} = 0)$ or $q = I_{max}$ are satisfied.
      ii. If a stopping criteria is satisfied, then output $z^{(q)}$ as a solution.

d) For each symbol index $0 \leq j < n$, compute the partial derivative $R_j^{(q+1)}$ using (4.44).

The update rule given by (4.43), which gives the estimation on the solution $y^{(q)}$ at the q^{th} iteration, causes a degradation at the decoding process. This effect has a significant impact providing slow convergence, in addition to a degradation in error rates. This is explained by the consideration of the term $y^{(q-1)}$, corresponding to the solution update calculated in the previous iteration. Considering the previous update causes the q^{th} update value to be significantly increased, and sometimes it can push the decoder to make an erroneous update of some symbols. This remark can be simply deduced from the small value of the descent step α required to achieve decoding convergence in this case. From (4.43), the update rule at the iteration $q > 1$ can be rewritten as follows:

$$y^{(q)} = \alpha R^{(q)} + \sum_{i=0}^{q-1} y^{(i-1)} + \alpha R^{(i-1)} \tag{4.45}$$

The summation in (4.45) is accumulated at each iteration, providing an over-scaling between the first and second term. At the first iteration $q = 0$, the reliability is a zero vector $R^{(0)} = 0$. Hence, the expression (4.45) takes the following form:

$$y^{(q)} = y^{(0)} + \alpha R^{(q)} + \sum_{i=1}^{q-1} y^{(i-1)} + \alpha R^{(i-1)} \tag{4.46}$$

Henceforth, we propose to omit the summation term in (4.46) in order to reduce the over-scaling effect between the two terms. Therefore, beginning from $q = 0$, the update rule at the q^{th} iteration can be replaced by:

$$y^{(q)} = y^{(0)} + \alpha R^{(q)} \tag{4.47}$$

This modification leading to (4.47) provides a considerable improvement of the GD-MLGD algorithm. As an alternative to the standard GD update rule, the updated value of $y^{(q)}$ consists of jumping at each iteration from the initial solution $y^{(0)}$, towards the direction provided by the gradient$R^{(q)}$ at the q^{th} iteration, with respect to a fixed descent step value α. This improvement results in less decoding iterations and lower error rates.

The proposed GD-MLGD algorithm is described in details in Algorithm 4.7.

Algorithm 4.7 GD-MLGD

Input: Received signal $y^{(0)} = y$, I_{max}, $\alpha, \beta, \theta, \gamma_{ch} = 2/\sigma^2$
Output: z

Init $R_j^{(0)} = 0$ for all $0 \leq j < n$
for $q = (0 : I_{max})$ **do**
  $y_j^{(q)} = \gamma y_j^{(0)} + \alpha R_j^{(q)}$ for all $0 \leq j < n$
  Calculate the hard decision $z^{(q)}$ of $y^{(q)}$
  Calculate $S(z^{(q)})$, If ($S(z^{(q)}) = 0$ or $q = I_{max}$), Stop decoding
  for $j = (0 : n-1)$ **do**
    $\omega_{ji} = max(min_{s \in \mathcal{N}(i) \setminus \{j\}} |y_s^{(q)}| - \theta, 0)$
    $R_j^{(q+1)} = y_j^{(0)} + \beta \sum_{i \in \mathcal{M}(j)} \omega_{ji} \prod_{s \in \mathcal{N}(i) \setminus \{j\}} sgn(y_s^{(q)})$
  end for
end for

4.3.2 GD-MLGD with Quantization (QGD-MLGD)

In order to further reduce the computational complexity, the Quantized Gradient-Descent Majority-Logic Decoder (QGD-MLGD) is proposed here. The quantization function aims to map each real symbol of the GD-MLGD algorithm into an integer, belonging to a certain range. The main provided advantage is to restrict the decoding operations to only integer operations, instead of operations on real valued symbols. As in [69, 120], we propose the use of an uniform quantization function with b quantization levels (bits). The quantization interval length is denoted by Δ, and is given, for the case of an uniform quantization, by:

$$\Delta = \frac{1}{2^{b-1}} \tag{4.48}$$

The quantization function $Q(x) : \mathbb{R}^n \longrightarrow \mathbb{Z}^n$ is defined for each $0 \leq j < n$ as follows:

$$Q(y_j) = \begin{cases} -(2^b - 1), & \frac{y_j}{\Delta} \leq -(2^b - 1) \\ \left[\frac{y_j}{\Delta}\right], & |\frac{y_j}{\Delta}| < (2^b - 1) \\ +(2^b - 1), & \frac{y_j}{\Delta} > +(2^b - 1) \end{cases} \tag{4.49}$$

where $[.]$ denotes the rounding operator, which selects the nearest integer.

Each quantized symbol belongs to the interval $Q(y_j) \in [-(2^b - 1), (2^b - 1)]$. The behavior of the quantization function $Q(x)$ is illustrated in Figure 4.4, for various quantization bits $b = 2, 3, ..., 5$ and with x taking real values from the interval $[-4, 4]$ with a step equal to 0.1.

Clearly, the accuracy of the quantization function increases with the quantization bits b, at the cost of a slightly higher storage and computational complexity.

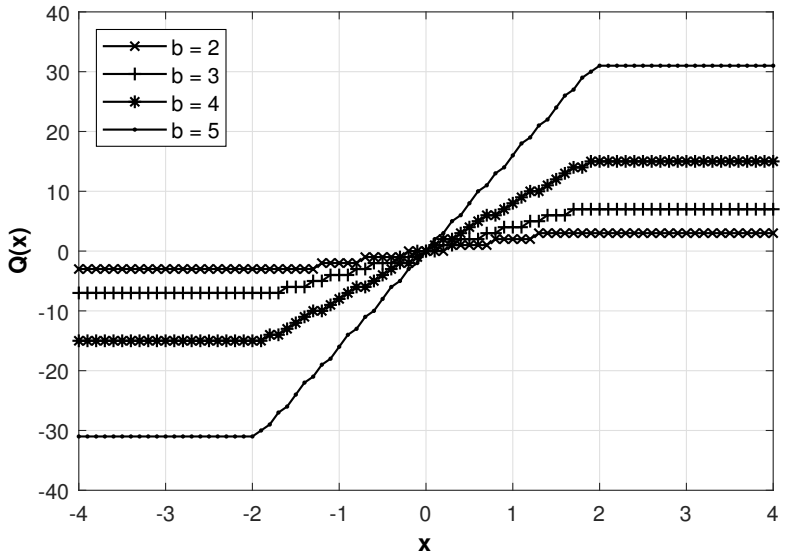

Figure 4.4: The uniform quantization function behavior with various quantization bits

In the QGD-MLGD algorithm, the initialization step consists of computing the quantized received values $y_j^{(0)} = Q(y_j^{(0)})$ for every $j \in [0, n-1]$. Then at the beginning of each iteration, the solution update at the q^{th} iteration given in Algorithm 1, Line 3, can be written for every $j \in [0, n-1]$ as follows:

$$y_j^{(q)} = y_j^{(0)} + \left[\alpha R_j^{(q)}\right] \tag{4.50}$$

As the descent step parameter α is real, the product $\alpha R_j^{(q)}$ results in a real value. Thus, the rounding operator $[.]$ is used to convert the result to an integer. Note that the over-scaling parameter β is omitted. Also, in contrast to GD-MLGD where the offset parameter θ is real, here θ takes an integer value, as the updated solution is an integer. Note that the QGD-MLGD algorithm does not need any knowledge on the channel reliability γ_{ch}.

The QGD-MLGD algorithm is summarized in Algorithm 4.8.

Algorithm 4.8 QGD-MLGD

Input: Received signal y, I_{max}, $\alpha \in \mathbb{R}$, $\theta \in \mathbb{N}$, b

Output: z

Init $R_j^{(0)} = 0$ and $y_j^{(0)} = Q(y_j)$ for all $0 \leq j < n$
for $q = (0 : I_{max})$ **do**
    $y_j^{(q)} = y_j^{(0)} + [\alpha R_j^{(q)}]$ for all $0 \leq j < n$
    Calculate the hard decision $z^{(q)}$ of $y^{(q)}$
    Calculate $S(z^{(q)})$, If ($S(z^{(q)}) = 0$ or $q = I_{max}$), Stop decoding
    for $j = (0 : n-1)$ **do**
        $\omega_{ji} = max(min_{s \in \mathscr{N}(i) \setminus \{j\}} |y_s^{(q)}| - \theta, 0)$
        $R_j^{(q+1)} = y_j^{(0)} + \sum_{i \in \mathscr{M}(j)} \omega_{ji} \prod_{s \in \mathscr{N}(i) \setminus \{j\}} sgn(y_s^{(q)})$
    end for
end for

4.4 Performance Analysis and Comparison with previous works

This section is dedicated to the complexity analysis as well as the performance evaluation of the proposed decoding algorithms. For simulations, we consider cyclic OSMLD codes derived from finite geometries, namely difference-set codes (DSC) derived from Projective Geometries [145, 185], and OSMLD codes derived from Euclidean Geometries (EG). Table 4.2 presents the parameters of the selected OSMLD codes for performance evaluation. The first column includes the code parameters, while the three columns that follows includes the number of orthogonal parity-check equations, the code rate and the code sub-class, respectively.

(n, k, d_{min})	J	$r = \frac{k}{n}$	Sub-Class
$(255, 175, 17)$	16	0.69	EG
$(273, 191, 18)$	17	0.70	DSC
$(1023, 781, 33)$	32	0.76	EG
$(1057, 813, 34)$	33	0.77	DSC

Table 4.2: A set of cyclic OSMLD codes derived from Euclidean and Projective Geometries

Note that the simulations of all the decoding algorithms were performed after a numerical optimization of their associated input parameters.

Simulation results were carried out by the Monte-Carlo method, using a BPSK modulation and a transmission over an AWGN channel with zero mean and variance σ^2. For the quantized decoding algorithms, the number of quantized bits considered for simulations is $b = 7$ with 127 quantization intervals. The simulations were executed in C++, while prototyping was carried out using the simulation platform MATLAB. Simulation parameters are displayed in Table 4.3.

Parameter	Value
Modulation	BPSK
Channel	AWGN
Number of quantization bits (for the quantized algorithms)	7 bits
Minimum transmitted blocks	1000
Minimum residual errors	200

Table 4.3: Simulation parameters

4.4.1 Complexity Analysis

We evaluate the computational complexity provided by the proposed decoding schemes ((Q)GD-MLGD). The idea of using the gradient-descent algorithm for decoding OSMLD codes is mainly motivated by the simplicity of the GD technique. Let's first consider the initialization step. The initialization of the vector $R^{(0)}$ of the soft-reliabilities included in Algorithm 4.7, line 1, requires simple affectations without additional operations. Let's now analyze the complexity per iteration required by the GD-MLGD algorithm.

Without loss of generality, we will conventionally denote row and column weights of the parity-check equations respectively by ρ and γ, as mentioned in the preliminaries. Each iteration needs at the beginning n real additions to compute the updated solution $y^{(q)}$ at line 3 in Algorithm 4.7. The syndrome calculation of $s^{(q)}$ at line 5 requires n sign tests, which are logical operations, and $(\rho-1)m$ other binary operations. For the evaluation of the J orthogonal equations for each symbol, the weighting coefficients ω_{ji} needs a total of $3m(\rho-1)$ real comparisons for the process of all the n symbols. For each orthogonal equation, the binary estimator B_{ji} requires $\rho-2$ binary operations, thus a total of $(\rho-2)\gamma n$ binary operations are needed for all the estimators. $(\gamma-1)n$ real additions are needed for the calculation of the extrinsic information vector E. Finally, the vector of soft-reliabilities $R^{(q)}$ is calculated by n real additions and n real multiplications.

Then, assuming that $\delta = \rho m = \gamma n$, to carry out one iteration of the GD-MLGD algorithm, a total of $(\rho-1)\delta + n - m$ logical operations, $\delta + n$ real additions and $3(\delta - m)$ real comparisons are required.

The QGD-MLGD algorithm require less computational complexity than the GD-MLGD. The real operations are simplified to integer operations, providing a faster decoding speed.

Table 4.4 presents and compares the computational complexities of the proposed algorithms GD-MLGD and QGD-MLGD with the existing soft-reliability-based majority-logic decoding algorithms reviewed in Section II. In Table 4.4, let BO, IA, RA, IC, RC and RM denote binary operations, integer additions, real additions, integer comparisons, real comparisons and real multiplications, respectively. We see that the GD-MLGD algorithm requires a lower computational complexity compared to the IDA and ITD algorithms. Additionally, it is shown that the QGD-MLGD algorithm provides an additional reduction of the required complexity.

Decoding	Computational cost per iteration						
algorithm	BO	IA	RA	IC	RC	RM	Log
SRBI	$2\delta + n - m$	δ					
ISRBI	$2\delta + n - m$	$(\delta - 1)n$	n				
MTD	$\delta\rho + n - m$		δ				
IDA	$\delta + n - m$		δ			$(\rho - 2)\delta$	
ITD	$(\rho - 2)\delta + n$		$\delta + n$		$3(\delta - m)$	n	n
GD	$(\rho - 1)\delta + n - m$		$\delta + n$		$3(\delta - m)$	n	
QGD	$(\rho - 1)\delta + n - m$	$\delta + n$		$3(\delta - m)$		n	
SPA						6δ	n

Table 4.4: Computational complexity per iteration of various majority-logic decoding algorithms

Besides the computational complexity, the (Q)GD-MLGD algorithms need the storage of the gradient vector $R^{(q)}$ and the updated signal $y^{(q)}$. Thus, for the GD-MLGD algorithm, n real numbers are needed to store both $R^{(q)}$ and $y^{(q)}$. Then a total storage of $2n$ real numbers are required for the GD-MLGD algorithm. For the QGD-MLGD algorithm, the storage requirements depends on the number b of quantization bits. Thus, the gradient vector $R^{(q)}$, which contains n real reliabilities, can be stored in bn bits, and similarly for the vector $y^{(q)}$. Thus the QGD-MLGD algorithm requires a total storage cost of $2bn$ units (bits).

Table 4.5 gives the memory requirements per iteration of the proposed algorithms (Q)GD-MLGD, compared to previous works in the literature. Clearly, the GD-MLGD and QGD-MLGD algorithms require a reasonable storage complexity compared to the other decoding approaches.

Decoding	Memory Requirement	
Algorithm	Units (bits)	RN
MTD	$2n$	
IDA		n
SRBI	$m+bn$	
ISRBI	$m+bn$	n
ITD	m	n
GD-MLGD		n
QGD-MLGD	bn	
BP		δ

Table 4.5: Memory Requirements for Decoding a OSMLD Codes with various Decoding Algorithms

4.4.2 Parameters Optimization

Our purpose here is to perform a numerical optimization of the parameters involved in the GD-MLGD algorithm, in order to find the best set of parameters. The parameters of GD-MLGD are the descent step α, the over-scaling factor β and the offset parameter θ. Thus, we present below the results obtained by an exhaustive numerical optimization of a set of values of α and β, then the best values are chosen for optimizing the offset parameter θ individually.

Figures 4.5 and 4.6 represent the obtained optimization results, for a set of values given in the range $\alpha \in \{0.05, 0.2, 0.4, 0.6, 0.8, 1\}$ and $\beta \in \{0.05, 0.2, 0.4, 0.6, 0.8, 1\}$ for various SNRs, and for the cyclic OSMLD code with parameters (n, k, d_{min}) given by $(255, 175, 17)$, for SNR values of 3 dB and 3.25 dB, respectively.

Clearly, the optimal values of α and β are weakly dependent on the SNR, and strongly depends on the considered code itself. It is shown from Figures 4.5 and 4.6 that the optimal values for the code $(255, 175, 17)$ are given by $(\alpha, \beta) = (0.2, 1.0)$. Similarly, we have performed the same optimization for all the OSMLD codes displayed in Table 4.2.

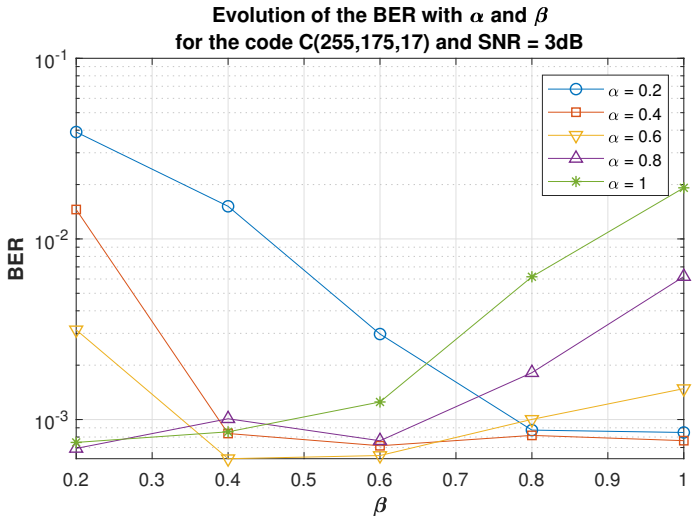

Figure 4.5: Optimization of the descent step α and the over-scaling factor β for the code $(255, 175, 17)$ in $SNR = 3$ dB

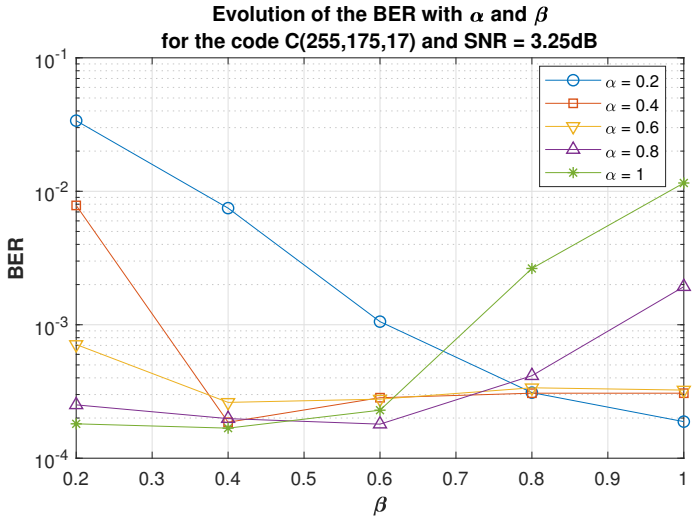

Figure 4.6: Optimization of the descent step α and the over-scaling factor β for the code $(255, 175, 17)$ in $SNR = 3.25$ dB

Given the optimal values of α and β, the optimization of the offset parameter is performed individually and is represented in Figures 4.7, 4.8 and 4.9, for a set of OSMLD codes with parameters $(255, 175, 17)$, $(273, 191, 18)$, and $(1057, 813, 34)$, respectively. A set of values in the range $[0, 1]$ are assigned to θ with a step equal to 0.2, and the BER is determined for each value over various SNR values. For each code, the optimal value of the offset parameter is obtained based on the BER performance.

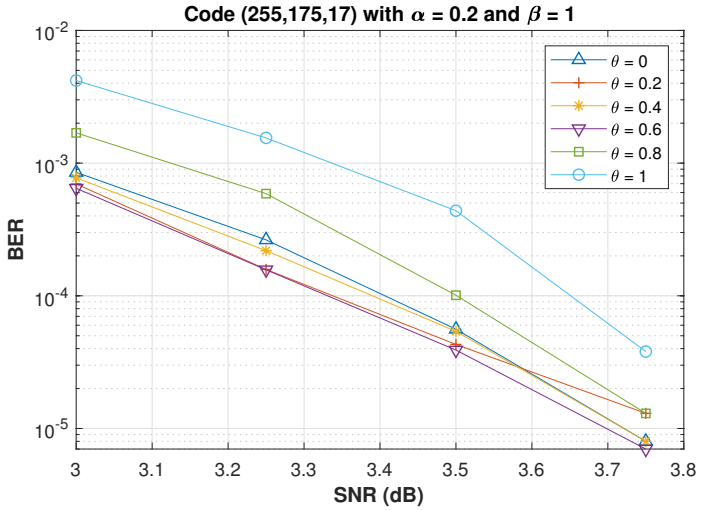

Figure 4.7: Optimization of the offset factor θ over various SNRs for the OSMLD code $(255, 175, 17)$

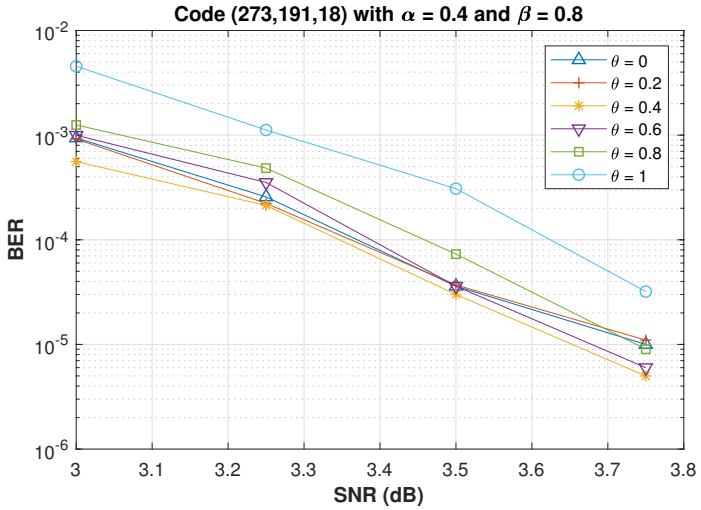

Figure 4.8: Optimization of the offset factor θ over various SNRs for the OSMLD code $(273, 191, 18)$

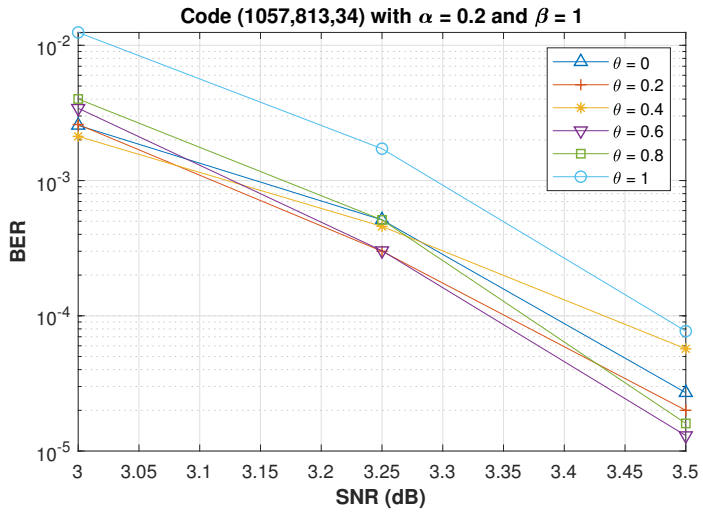

Figure 4.9: Optimization of the offset factor θ over various SNRs for the OSMLD code $(1057, 813, 34)$

(n, k, d_{min})	GD-MLGD (α, β, θ)	QGD-MLGD (α, θ)	Family
$(255, 175, 17)$	$(0.2, 1.0, 0.6)$	$(0.2, 4)$	Cyclic EG
$(273, 191, 18)$	$(0.4, 0.8, 0.4)$	$(0.2, 4)$	Cyclic DSC
$(1023, 781, 33)$	$(0.6, 0.4, 0.6)$	$(0.2, 4)$	Cyclic EG
$(1057, 813, 34)$	$(0.2, 1.0, 0.6)$	$(0.3, 0)$	Cyclic DSC

Table 4.6: Optimal values of α, β and θ for a set of cyclic OSMLD codes

Based on this optimization, Tables 4.6 displays the optimal values of the descent step α, the over-scaling β and the offset θ, for the set of OSMLD codes displayed in Table 4.2, and for the algorithms GD-MLGD and QGD-MLGD. Note that the obtained values are suitable for the case of a transmission over an AWGN channel with a BPSK modulation, and may be different for other transmission channels and modulation schemes.

4.4.3 Average Number of Iterations

In order to perform a numerical convergence analysis of the proposed decoding scheme, we propose to analyze the average number of iterations with respect to the SNR. If the number of simulated transmitted blocks is N, then the average iterations number I_{avg} is obtained by the ratio $I_{avg} = T/N$, where T is the total number of iterations used for decoding all the N

blocks, and N is chosen such that at least 200 erroneous decoded words are observed for each SNR.

The curves corresponding to the average number of iterations for the codes displayed in Table 4.2 with respect to different SNRs are illustrated in Figures 4.10-4.13, respectively.

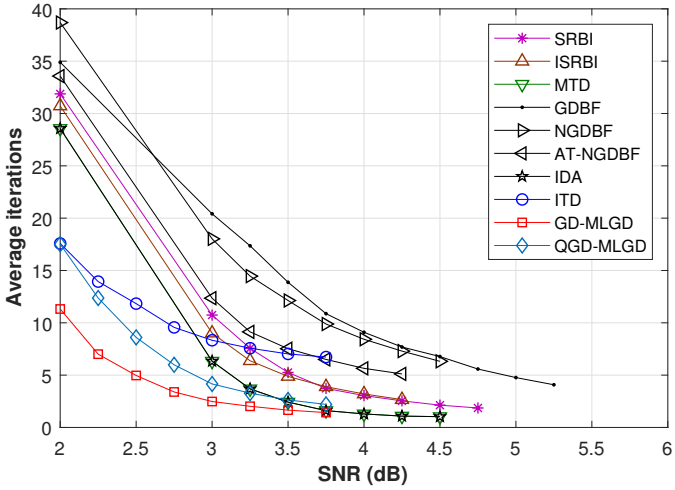

Figure 4.10: Comparison of the average iterations number of various MLGD algorithms for the cyclic OSMLD code $(255, 175, 17)$

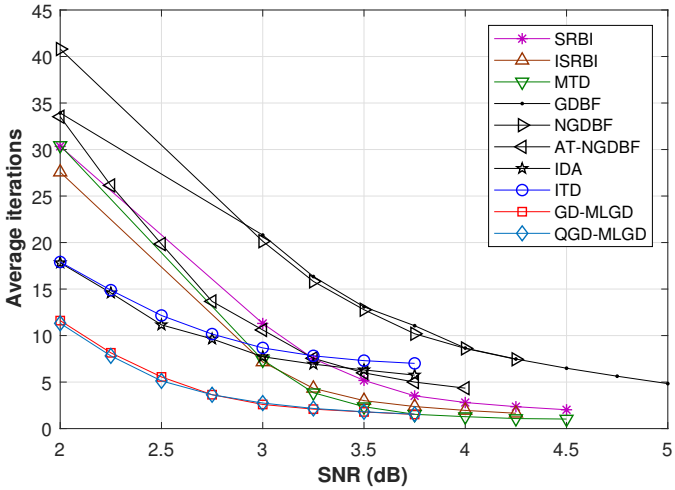

Figure 4.11: Comparison of the average iterations number of various MLGD algorithms for the cyclic OSMLD code $(273, 191, 18)$

For the moderate length OSMLD codes presented in Figures 4.10 and 4.11 corresponding to the OSMLD codes $(255, 175, 17)$ and $(273, 191, 18)$, we see that in the entire range of SNRs,

the GD-MLGD needs less iterations to achieve convergence, followed by the QGD-MLGD (Figure 4.10), the IDA, then the ITD, which in turns outperforms the MTD, ISRBI, and the SRBI. Bit-Flipping algorithms always require more iterations to achieve convergence. In moderate SNRs, the GD-MLGD requires approximately 50% fewer iterations than the IDA and ITD. From Figure 4.11, it is shown that the proposed algorithms GD-MLGD and QGD-MLGD requires the same number of decoding iterations, which shows the advantage of the proposed quantized algorithm QGD-MLGD.

The average number of decoding iterations of long OSMLD codes is represented in Figures 4.12 and 4.13 corresponding to the codes $(1023, 781, 33)$ and $(1057, 813, 34)$, respectively.

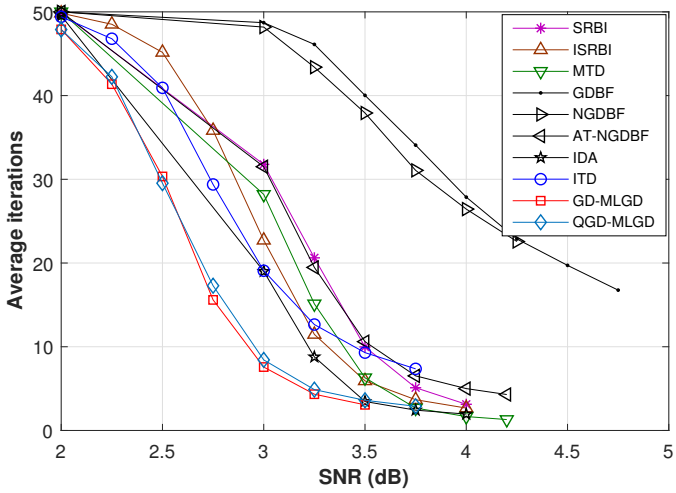

Figure 4.12: Comparison of the average iterations number of various MLGD algorithms for the cyclic OSMLD code $(1023, 781, 33)$

Clearly, the proposed decoding algorithms require a lower number of decoding iterations compared to the other schemes, where it requires about 50% fewer iterations than the ITD and IDA. The ISRBI requires slightly fewer iterations than the IDA and ITD, followed by the MTD, then the SRBI. The GD-MLGD and its quantized version QGD-MLGD requires the same number of decoding iterations.

Decoding latency can be analyzed using the required number of decoding iterations for achieving a given BER value, as well as the SNR value at which this BER value can be achieved. Tables 4.7 and 4.8 depict the SNR and the average decoding iterations required for achieving a BER of 10^{-5}, for the cyclic OSMLD codes $(255, 175, 17)$ and $(1057, 813, 34)$ respectively. It is shown that the proposed decoding algorithms achieve a BER of 10^{-5} at

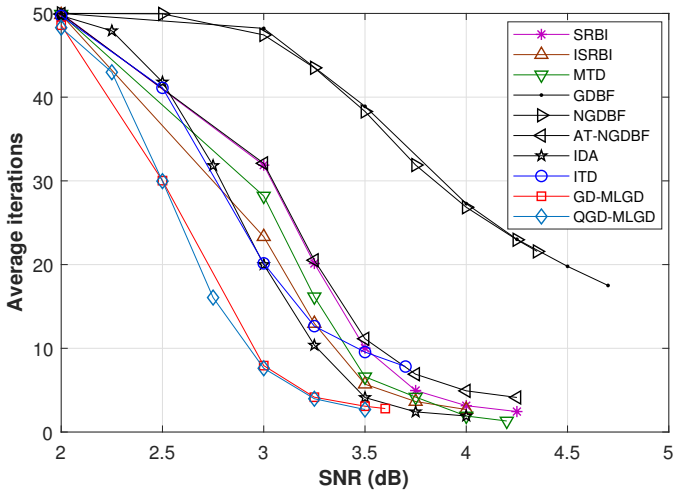

Figure 4.13: Comparison of the average iterations number of various MLGD algorithms for the cyclic OSMLD code $(1057, 813, 34)$

lower SNR values, compared to the other decoding algorithms. Also, with the obtained SNR values, a reasonable average number of decoding iterations is required, relatively lower than the other decoding techniques. Note that in the case where an algorithm requires lower decoding iterations corresponds to a higher SNR value than that required by the proposed (Q)GD-MLGD algorithms. This established analysis emphasizes the fact that the proposed algorithms provide lower decoding latency compared to the state of art.

Decoding algorithm	SNR (dB)	Average decoding iterations
GDBF	5.3	4
MTD	4.5	1.03
SRBI	4.75	1.86
ISRBI	4.15	2.96
IDA	3.85	1.44
ITD	3.83	6.12
GD-MLGD	3.65	1.5
QGD-MLGD	3.69	2.2

Table 4.7: Comparison of the SNR and the average decoding iterations required to achieve a BER of 10^{-5} for the cyclic OSMLD code $(255, 175, 17)$

Generally, it is observed that the proposed algorithms achieve a fast convergence, and provide lower latency compared to previous MLGD techniques. It is also shown that the bit-flipping

Decoding algorithm	SNR (dB)	Average decoding iterations
GDBF	4.73	16.8
MTD	4.2	1.32
SRBI	4.3	2.15
ISRBI	4	2.7
IDA	3.86	2.15
ITD	3.7	7.82
GD-MLGD	3.51	3.08
QGD-MLGD	3.51	2.68

Table 4.8: Comparison of the SNR and the average decoding iterations required to achieve a BER of 10^{-5} for the cyclic OSMLD code $(1057, 813, 34)$

algorithms requires more decoding iterations for achieving convergence.

An overall comparison shows that the QGD-MLGD algorithm is more suitable for applications requiring high-speed decoding and low latency. For instance, the QGD-MLGD algorithm is suitable for the use in Ultra-Reliable Low-Latency (URLLC) and massive Machine-Type Communications (mMTC) use cases in the 5G NR mobile networks.

4.4.4 Error rates

The Block Error Rate (BLER) simulation results of the proposed algorithms GD-MLGD and QGD-MLGD, compared with the ITD, IDA, MTD, ISRBI, SRBI, and WOSMLGD algorithms for the cyclic OSMLD codes displayed in Table 4.2, are shown in Figures 4.14-4.17, respectively for each code. Note that these figures include additionally the performance of the GDBF and NGDBF decoding algorithms.

Figures 4.14 and 4.15 represents the BLER performance corresponding to the cyclic OSMLD codes $(255, 175, 17)$ and $(273, 191, 18)$, respectively. The ITD and IDA algorithms provide approximately the same performance. The benefits of the quantization function is clearly illustrated, especially for the low and moderate SNR values, where the QGD-MLGD outperforms slightly the GD-MLGD, and in the remaining SNR regions the two algorithms provide approximately the same BLER performances. The GD-MLGD algorithm achieves a coding gain of about 0.2 dB versus the ITD and IDA algorithms. A significant improvement compared to ISRBI, MTD, SRBI and WOSMLGD algorithms is shown, given by a coding gain of about 0.45 dB, 0.75 dB, 1.1 dB and 1.75 dB, respectively. Clearly the proposed algorithm

outperforms all the existing algorithms, at the cost of a low computational complexity and decoding latency.

The BLER performance of the cyclic OSMLD codes $(1023, 781, 33)$ and $(1057, 813, 34)$ are displayed in Figures 4.16 and 4.17, respectively. We see that the proposed algorithms outperform all the existing MLGD algorithms. The coding gain is further improved for the case of long OSMLD codes compared to moderate code lengths depicted in Figures 4.14 and 4.15. In figure 4.16, the OSMLD code $(1023, 781, 33)$ achieves a coding gain of 0.25 dB at a BER of 10^{-5} compared to the ITD algorithm. The IDA has a performance loss of about 0.25 dB versus the ITD. A general comparison shows that the GD-MLGD algorithm achieve a coding gain of 0.5 dB versus the the IDA and ISRBI, which has similar performances in this case. GD-MLGD outperforms the MTD, SRBI and WOSMLGD by a coding gain of 0.75 dB, 0.85 dB and 1.7 dB, respectively. In general, we see that the GD-MLGD algorithm represents the best BLER performances compared to the existing MLGD algorithms. Note also that GD-MLGD achieves performances that are nearly comparable with the performances of the BP-SP decoding algorithm for LDPC codes, given in [86], such that a lower computational complexity is required.

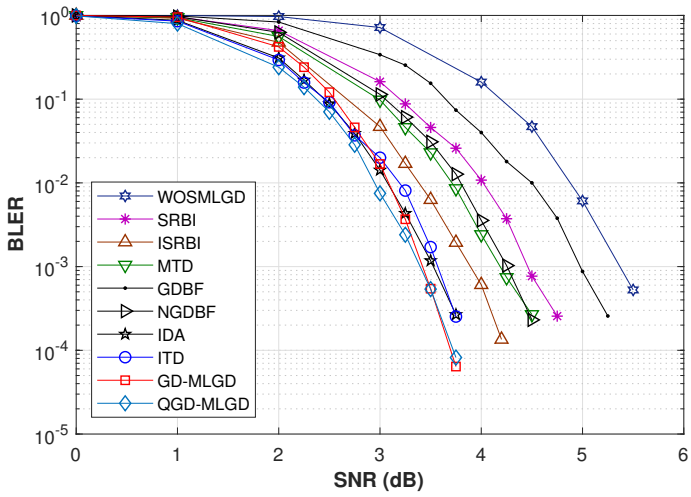

Figure 4.14: BLER performance of various MLGD algorithms for decoding the cyclic OSMLD code $(255, 175, 17)$

An overall comparison of the BLER performances of the existing MLGD algorithms shows that in the general case, the proposed decoding algorithm outperforms all the existing algorithms, especially for long codes. GD-MLGD outperforms ITD, followed by the IDA. The

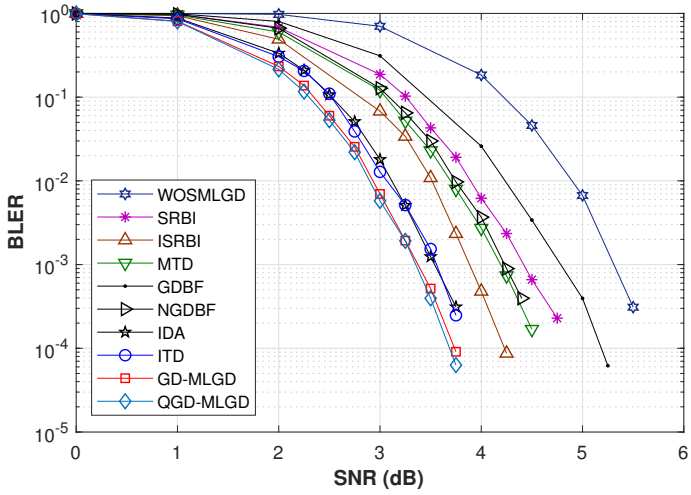

Figure 4.15: BLER performance of various MLGD algorithms for decoding the cyclic OSMLD code $(273, 191, 18)$

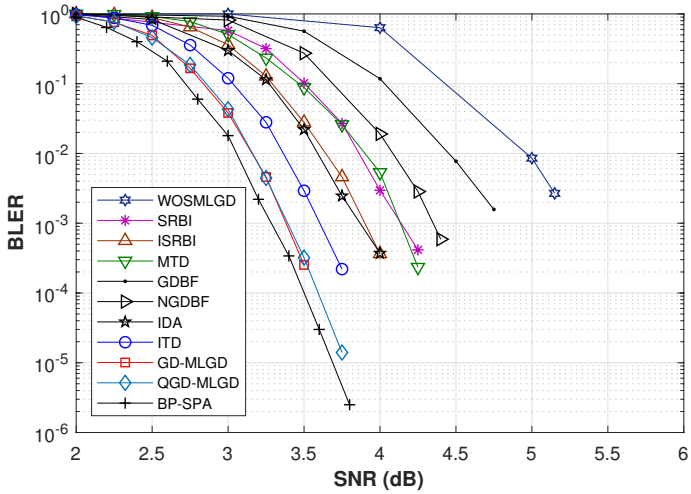

Figure 4.16: BLER performance of various MLGD algorithms for decoding the cyclic OSMLD code $(1023, 781, 33)$

ISRBI provides better performance than the MTD, which in turns outperforms the SRBI and the WOSMLGD algorithms.

It is also shown that the QGD-MLGD algorithm provides robust performances compared to the expected performances given its computational complexity. Consequently, simulation results shows that the proposed approach results in an efficient trade-off between performance and complexity, where the obtained performances has shown that the QGD-MLGD algorithm

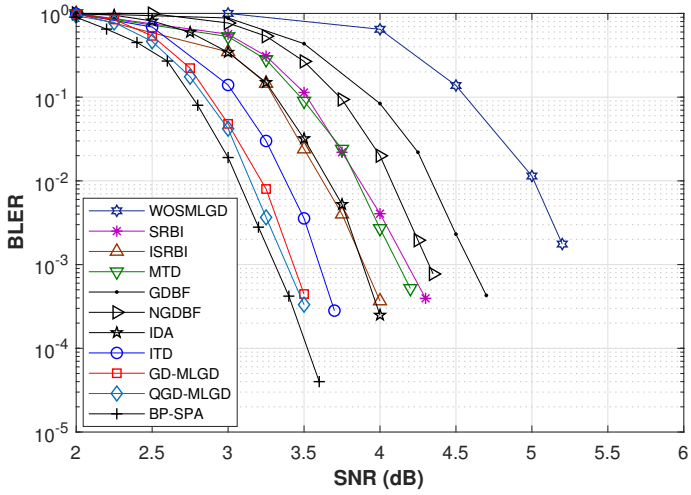

Figure 4.17: BLER performance of various MLGD algorithms for decoding the cyclic OSMLD code $(1057, 813, 34)$

represents the most suitable decoding scheme for applications that require low-complexity and latency in addition to robust performance.

4.5 Analysis of the False Decoding Decisions

The study and analysis of the erroneous decisions of the decoder is primordial in order to understand the behavior of the decoder, and eventually improve its performance. In the most cases, the decoding decision is successful, however, wrong decisions provide a major impact on the decoding error rates.

[107] claimed that a typical error correcting code, when decoded using a message-passing algorithm, can result in two types of errors:

1. Undetected errors: When the decoder converges to a wrong codeword, which is not the transmitted codeword, we say that the error configuration is undetected. We can also refer to these types of errors as false converged codewords. Typically, these errors imply the existence of low-weight codewords. Some codes (for example long codes, and Gallager regular codes) do not contain low-weight codewords. However, for many families of codes and decoding algorithms, these errors occur frequently, which indicates that they should not be considered as other errors.

2. Detected errors: When the decoder fails to converge to a codeword. Typically, the decoder reaches the maximum number of iterations, and outputs an erroneous vector which is not a codeword. Generally these error configurations are associated to a large number of errors.

Given the stopping criteria of the GD-MLGD algorithm, it is easy to distinguish a detected error, as the decoder reaches the last iteration and the syndrome weight is not zero. However, the errors of type 1 are undetected and cannot be distinguished from a successful decoding decision. A partial solution to detect these errors consists of the use of a cyclic redundancy check (CRC) as in [124, 200], at the cost of a reduced coding rate. However, the use of a CRC does not always guarantees that the error is detectable, especially if the redundancy associated to the CRC is erroneous.

Many papers have addressed the problem of detected errors of LDPC codes in the literature [5, 31, 81, 89, 126, 135, 178]. Typically, these errors are associated to some special combinatorial error patterns that exist and are related to the graph structure of the codes. These special configurations are called Trapping Sets [135].

It has been shown that the degree of influence of the trapping sets is only of importance in the case of diverged decoding decisions (detected error). As will be shown in the following, most of the erroneous decisions of the GD-MLGD algorithm are due to the false converged codewords phenomenon. Therefore, we will henceforth not focus our attention on trapping sets, despite their influence on the error-floor region. Thus, error floors are not investigated in this analysis and can be considered as a perspective of this work.

Table 4.9 illustrates the obtained results from a statistic on decoding the DS code with parameters $(n, k, d_{min}) = (73, 45, 10)$ with the GD-MLGD algorithm. The first and second columns represent the SNR value and the success rate, respectively. The percentage of undetected errors and uncorrected errors are also represented in the third and fourth columns, respectively. Note that these results were obtained by simulations, where for each SNR value, a set of 10000 randomly generated codewords had been transmitted.

We see that the undetected errors corresponding to false converged codewords are dominant, compared to the uncorrected errors. These errors are responsible for all the decoding failures in the moderate and high SNR regions. As displayed in Table 4.9, the percentage of the number of uncorrected errors is negligible as these errors generally occur when the number of errors in the received signal is significantly greater than the correction capacity of the code, and the decoder fails to find any close codeword to the received sequence. This statement is

SNR (dB)	Success rate	% of undetected errors	% of uncorrected errors
2.5	0.959	98.78%	1.22%
2.75	0.976	98%	2%
3	0.983	97.96%	2.04%
3.25	0.990	95%	5%
3.5	0.994	97.87%	2.12%
3.75	0.998	100%	0%
4	0.998	100%	0%
4.25	0.999	100%	0%
4.5	0.999	100%	0%

Table 4.9: Statistics on error types in GD-MLGD using the DS code with parameters $(n,k,d_{min}) = (73,45,10)$

straightforward as a large number of errors is contained in the symbols participating in the parity-check equations, thus many of these check-sums are not satisfied even after that the maximum number of allowed decoding iterations is achieved. A check-sum is satisfied if there is an even number of errors, therefore, when the number of errors is large, many check-sums will contain an odd number of errors, which causes the decoder to fail in finding a valid codeword that has a zero syndrome vector.

Our current investigations and perspectives of this work aim to establish an analysis of the behavior and dynamic of the GD-MLGD algorithm, where the variations of the syndrome weight and the error weight during decoding iterations are of interest. By denoting the sequence of the syndrome weight and the error weight during iterations respectively by $W_S(q)$ and $W_e(q)$, where q is the iteration index, current investigations consist of the analysis of the correlation between these two distributions, considered as stochastic processes.

Moreover, a regression analysis of these processes is also established, but not presented in this report. The expectations of this analysis includes the detection of the iterations at which the correlation between $W_S(t)$ and $W_e(t)$ is inversely proportional. At these particular points, the decoder begins to minimize the syndrome weight towards an erroneous decision, and the weight of the error is systematically increased. These errors are said to be punctual, and they represent the most interesting case of local maximums for the GD-MLGD algorithm. Other types of observed errors are those where the decoder oscillates during iterations around an erroneous decision. These errors are said to be periodic. The last type of errors is the

aperiodic errors, where the oscillation of the error weight takes an aperiodic form.

This study has currently shown that the false converged codewords are especially caused by the punctual errors, which are the most dominant error types in GD-MLGD decoding, and occur in most cases where the GD-MLGD algorithm converges to a local maximum. An open challenge consists of devising a suitable strategy for detecting these local maximums, based on the analysis briefly explained above. The accurate detection of false converged codewords in the GD-MLGD algorithm will open some attractive research directions for devising different strategies for escaping from these local maximums, and therefore decoding performances can be drastically improved.

For this purpose, based on our current investigations, we have found that for the case of punctual errors, the decision of the decoder begins to diverge directly after the occurrence of periodic points in the syndrome weight distribution. These periodic points are those iterations at which the syndrome weight has the same value, and generally, based on our observations, these iterations are consecutive. Before these periodic points, the error and syndrome weights are correlated, and in the iterations that follow, the correlation is inversely proportional, which means that the error weight increases, while the syndrome weight has a decreasing behavior. In fact, at these particular points, the decoder begins to diverge towards an erroneous codeword, other than that being transmitted.

As a consequence, we are currently investigating a solution for classifying these periodic points. Based on this classification, a potential solution consists of storing the symbol indexes that have a phase been flipped during the iterations that follow the periodic point being detected. After storing these indexes, the phase inversion of these symbols is canceled, and the magnitude of these symbols can be increased by multiplying there values by a corresponding factor that needs to be optimized. Using this strategy, the decoder can be forced to retry the decoding process based on the modified reliabilities retained at the iteration in which the periodic point occurs. Thus, the decoder will try to converge to another codeword, by correcting other symbols until the correct codeword is achieved. This analysis will be published soon in as a journal paper.

4.6 Conclusion

In this chapter, a new gradient-descent based majority-logic decoding algorithm with low-complexity has been proposed for decoding OSMLD codes. The quantized version of the

proposed algorithm has surprisingly shown to provide robust performances and fast convergence. The proposed algorithm achieves lower error rates and faster decoding speed compared to the existing MLGD algorithms in the literature, and can be easily scalable for the use in various communication channels and higher order modulation schemes, by suitably optimizing its parameters. Simulation results showed that (Q)GD-MLGD outperforms the previous algorithms, especially for long codes, proving that the proposed decoding scheme is suitable for decoding moderate and long OSMLD codes in high speed wireless communication systems requiring ultra-high reliability and low latency, while providing an efficient trade-off between performance and complexity. The obtained results emphasize the benefit of the use of the proposed decoding algorithms in the 5G NR URLLC and mMTC use cases as well as in other wireless communication systems, where both performance and complexity are required. Additionally, we have presented an analysis on the erroneous decisions of the GD-MLGD algorithm, where we have studied the impact of the syndrome weight distribution along decoding iterations on the emergence of false converged codewords, this study was not completely presented in this chapter and will be published soon.

Chapter 5

Unified Models for Majority-Logic Decoding Algorithms using Local Optimization Techniques and a New Constrained Optimization based Decoding Algorithm for Linear Block Codes

5.1 Introduction

Nowadays, with the emergence of many use cases of modern wireless communication systems, a significant challenge in coding theory is to find good error correcting codes as candidates that meet the technical requirements of these systems, as well as adequate decoding algorithms capable of providing high reliability, low latency and simple hardware implementations. An important class of codes capable of facing these challenges are Low-Density Parity-Check (LDPC) codes, initially introduced by Gallager in [48], and later rediscovered in [86, 109]. LDPC codes have demonstrated their ability to approach the fundamental limits promised by Shannon in his seminal work [153]. The most powerful algorithm suitable for decoding these codes is the Belief Propagation (BP) algorithm. It was shown in many papers in the literature that the most efficient LDPC codes are those whose Tanner graph does not contain short cycles of length 4. Equivalently, the parity-check matrices of LDPC codes must

satisfy the row-column (RC) constraint, and in this case, they can be simply decoded using majority- logic.

Majority-Logic Decoding (MLGD) is known to be the simplest approach for decoding LDPC codes, and is very suitable for energy-constrained and real-time wireless communication systems, due to the low-complexity and low-cost hardware implementation involved in this decoding approach. LDPC codes that satisfy the RC constraint are so-called One-Step Majority-Logic Decodable (OSMLD) codes. One-Step Majority-Logic Decodable codes [49, 78, 145, 185] represent an interesting class of codes, due to their structured combinatorial and geometrical properties, including the orthogonality on each digit in their dual structure, which allows the design of low-complexity MLGD algorithms. This feature is very beneficial in achieving efficient implementations for real-time and low-complexity wireless communications. The most known OSMLD codes are those derived from finite geometries, such as Difference-Set code [185], and Euclidean Geometry (EG) codes [49, 78, 145].

As mentioned in previous chapters, the principle of majority-logic decoding was first introduced by Reed [133] for decoding Reed-Muller codes. Majority-logic decoding and threshold decoding were extensively studied by Massey [113] who proposed the first unified formulation of majority-logic decoding, as well as in [84]. The weighted one-step majority-logic decoding was introduced by Rudolph in [142, 143]. It was shown by Lucas et al. [106] that OSMLD codes derived from finite geometries outperform their equivalent randomly generated LDPC codes counterparts, when decoded with the BP algorithm. Several iterative majority-logic decoding algorithms have been proposed in the literature [10, 11, 69, 88, 105, 120, 158, 173, 174, 207]. One distinctive feature of these iterative decoding algorithms, is that they follow the same logic and rule, and differ only in the initialization of the soft-input information, the calculation of the extrinsic information and the update of the soft reliabilities.

In this chapter, we propose a unified view of majority-logic decoding algorithms as an unconstrained minimization/maximization of a derivable objective function. We show that for each MLGD algorithm, it is possible to associate a suitable objective function representing the decoding problem. Our study reveals that each derived objective function can be maximized using the Gradient-Descent (GD) optimization technique. The calculations of the corresponding first-order partial derivatives of these functions are investigated, and have shown to coincide with their corresponding soft-reliabilities. Moreover, we propose enhanced versions of these algorithms based on introducing new update rules derived from other variants of the Gradient-Descent. In addition, we extend our mathematical formulation to constrained optimization models, where we propose a new decoding algorithm, so-called the Augmented

Lagrangian Multiplier Method Decoder (ALMM-D), derived from the ALMM constrained optimization technique. Our contribution aims to establish a solid mathematical foundation of the decoding problem, and to facilitate the task of devising a decoding algorithm suitable for decoding LDPC codes, especially those with no short cycles (considered as OSMLD codes) as well as other linear block codes, by adequately choosing an appropriate mathematical model, in which the involved complexity only depends on the arithmetical calculations of the first-order partial derivatives of the incorporated objective function which represents the decoding problem. Furthermore, the extension of these models as well as the employed optimization techniques for solving it depend on the use case, where many compromises between performance, complexity and latency can be mathematically jointly optimized.

The rest of this chapter is organized as follows. In Section 2, basic preliminaries are given and a brief review on the existing majority-logic decoders is presented. Section 3 presents a new interpretation of the MLGD algorithms as a gradient-descent optimization, as well as a short discussion on the computational complexity of these algorithms. Section 4 introduces improved variants of the GD as well as a background on the ALMM constrained optimization technique. Section 5 includes improved versions of the considered MLGD algorithms based on new update rules derived from some variants of the GD, in addition to an extension to a new simplified constrained optimization model for devising the ALMM-D algorithm. Finally, Section 6 concludes this chapter.

5.2 Interpretation of Majority-Logic Decoding Algorithms as a Gradient-Descent Optimization

In this section, we propose an unified view of majority-logic decoding algorithms as an unconstrained minimization/maximization of a derivable objective function. We show that for each MLGD algorithm, it is possible to correspond a suitable objective function representing the decoding problem. Based on the established literature review, our study reveals that each derived objective function can be maximized using the Gradient- Descent (GD) optimization technique. The calculations of the corresponding first-order derivatives of these functions are investigated, and has shown to coincide with their corresponding soft-reliabilities.

As shown in Table 4.1, the main steps involved in all the existing MLGD algorithms in the literature are identical. These steps are as follows:

1. Initialization of the soft-reliability vector $R^{(0)}$ at the iteration $q = 0$.

2. Calculation of the extrinsic information $E_j^{(q)}$ for all $0 \leq j < n$.

3. Update of the soft-reliability values $R_j^{(q)}$ for all $0 \leq j < n$.

4. Update of the solution.

Intuitively, we figured out that each MLGD algorithm coincide with a gradient-descent maximization of a particular first-order derivable objective function. The reason behind this finding is that the GD optimization technique also includes an initialization of the solution, which is considered as an initial solution of the problem, then in each iteration, the GD computes the vector of partial derivatives with respect to each variable of the problem, then finally update the solution by following the direction provided by the gradient. For decoding OSMLD codes, each codeword symbol is a variable of the problem, thus there are n variables denoted as $x = (x_0, x_1, ..., x_{n-1})$, and the objective function $f(x)$ of each MLGD algorithm either includes only a term related to the syndrome weight, which can take different forms, or includes the later in addition to a term related to the correlation between the solution and the received signal y. Also, after devising each objective function, we have shown that each soft-reliability term R_j coincides with the partial derivative of the objective function $f(x)$ with respect to each variable x_j. We also investigated the value of the descent step that corresponds to each decoding algorithm. These results are consolidated in the following propositions below.

The IDA can be considered as an application of the gradient-descent algorithm to maximize the function $W_g(S_w)$. The optimization aims to find a real vector $R \in \mathbb{R}^n$ of soft-reliability that maximize $W_g(S_w(R))$. The problem can be stated as follows:

Proposition 5.1. *The Iterative decoding algorithm (IDA) is equivalent to the following gradient-descent optimization problem:*

$$\begin{aligned} &\textit{Maximize } f(x) = \sum_{0 \leq i < m} \left(\prod_{s \in \mathcal{N}(i)} \tanh(x_s) \right) \\ &\textit{subject to } x \in \mathbb{R}^n \end{aligned} \tag{5.1}$$

Proof. The partial derivative of $f(x)$ with respect to each variable x_j is given by:

$$R_j = \frac{\partial f(x)}{\partial x_j} = (1 - \tanh^2(x_j)) \sum_{i \in \mathcal{M}(j)} \left(\prod_{s \in \mathcal{N}(i) \setminus \{j\}} \tanh(x_s) \right) \tag{5.2}$$

Note that the partial derivative in (5.2) coincides with the extrinsic information E_j used in the IDA, except the first term $(1-\tanh^2(x_j))$ that was not considered. Given the initial solution $x_j^{(0)} = \gamma_{ch} y_j$, where $\gamma_{ch} = \frac{2}{\sigma^2}$ is the channel reliability, then the solution update at the q^{th} decoding iteration is given by:

$$x_j^{(q+1)} = x_j^{(q)} + \alpha_j R_j \tag{5.3}$$

which coincides perfectly with the update of the IDA, with the assumption that the descent step parameter $\alpha = (\alpha_0, ..., \alpha_{n-1})$ takes for each $0 \leq j < n$ the following value:

$$\alpha_j = \frac{1}{1-\tanh^2(x_j)} \tag{5.4}$$

As $-1 \leq \prod_{s \in \mathscr{N}(i)} \tanh(x_s) \leq 1$, the maximum and minimum of the objective function $f(x)$ are given by $\max(f(x)) = m$ and $\min(f(x)) = -m$. This shows that the IDA is a gradient-descent optimization of the objective function given in (5.1), which proves the proposition 4.4. □

Similarly for the SRBI algorithm proposed in [69], we observe that the extrinsic information E_j corresponds to the first-order partial derivation of the objective function $f(x)$ that we can define for each $x \in \{-1,1\}^n$ by:

$$f(x) = \sum_{0 \leq i < m} \prod_{s \in \mathscr{N}(i)} x_s \tag{5.5}$$

This objective function represents the bipolar syndrome weight.

Proposition 5.2. *The Soft Reliability based Iterative Majority-Logic Decoder (SRBI) is equivalent to the following gradient-descent maximization problem:*

$$\begin{aligned} &\textit{Maximize } f(x) = \sum_{0 \leq i < m} \prod_{s \in \mathscr{N}(i)} x_s \\ &\textit{subject to } x \in \{-1,1\}^n \end{aligned} \tag{5.6}$$

Proof. The partial derivative of $f(x)$ with respect to each variable x_j is given by:

$$R_j = \frac{\partial f(x)}{\partial x_j} = \sum_{i \in \mathscr{M}(j)} \prod_{s \in \mathscr{N}(i) \setminus \{j\}} x_s \tag{5.7}$$

Given the initial solution $x_j^{(0)} = Q(y_j)$, where $Q(y_j)$ is the quantized valuey_j, the solution update takes the following form:

$$x_j^{(q+1)} = x_j^{(q)} + \alpha R_j \tag{5.8}$$

where in the SRBI, the descent step is $\alpha = 1$. Clearly the SRBI is a gradient-descent optimization of the function $f(x)$. The minimum and maximum values of this objective function are given by $\max(f(x)) = m$ and $\min(f(x)) = -m$. □

The SRBI algorithm was improved later in [120] to give the ISRBI-MLGD algorithm. In the calculation of the extrinsic information E_j in the SRBI, the bipolar estimators have the same contribution, which is either $+1$ or -1. As a consequence, the iterative scheme does not exploit the soft output information of the channel, except in the initialization step, where $x_j^{(0)} = Q(y_j)$. The ISRBI adds weighting coefficients $w_i \in \mathbb{Z}$ to each estimator, acting as reliabilities of the i^{th} orthogonal equation, which results in improved performance. This means that the objective function $f(x)$ is obtained by adding the weighting coefficients to the function $f(x)$. Thus, the modeling of the ISRBI as a gradient-descent is given in the following proposition.

Proposition 5.3. *The Improved Soft Reliability based Iterative Majority-Logic Decoder (ISRBI) is equivalent to the following gradient-descent maximization problem:*

$$\begin{aligned} &\textit{Maximize } f(x) = \sum_{0 \le i < m} w_i \prod_{s \in \mathscr{N}(i)} x_s \\ &\textit{subject to } x \in \{-1, 1\}^n \end{aligned} \tag{5.9}$$

Proof. The partial derivative of $f(x)$ with respect to each variable x_j is given by:

$$R_j = \frac{\partial f(x)}{\partial x_j} = \sum_{i \in \mathscr{M}(j)} w_{i,j} \prod_{s \in \mathscr{N}(i) \setminus \{j\}} x_s \tag{5.10}$$

and similarly to the SRBI, given the initial solution $x_j^{(0)} = Q(y_j)$, the solution update takes the following form:

$$x_j^{(q+1)} = x_j^{(q)} + \alpha R_j \tag{5.11}$$

where the descent step parameter $\alpha = 1$. Clearly, the solution update in (5.11) corresponds to the update of the ISRBI algorithm. Note that the objective function $f(x)$ represents the Quantized weighted bipolar syndrome weight, and that $\max(f(x)) = \sum_{0 \le i < m} w_i$ and $\min(f(x)) = -\sum_{0 \le i < m} w_i$. □

By using the same reasoning as above, the ITD algorithm [88] modeling is given in the next proposition.

Proposition 5.4. *The Iterative Threshold Majority-Logic Decoder (ITD) is equivalent to the following gradient-descent maximization problem:*

$$\begin{aligned} &\textit{Maximize } f(x) = \frac{\gamma_{ch}}{2} \parallel x \parallel_2^2 + \sum_{0 \leq i < m} \prod_{s \in \mathcal{N}(i)} \tanh(x_s) \\ &\textit{subject to } x \in \mathbb{R}^n \end{aligned} \tag{5.12}$$

Proof. The partial derivative of $f(x)$ given above with respect to each variable x_j is given by:

$$R_j = \frac{\partial f(x)}{\partial x_j} = \gamma_{ch} x_j + \sum_{i \in \mathcal{M}(j)} w_{i,j} \prod_{s \in \mathcal{N}(i) \setminus \{j\}} sgn(x_s) \tag{5.13}$$

where

$$w_{i,j} = \ln \left[\frac{1 + \prod_{s \in \mathcal{N}(i) \setminus \{j\}} \tanh(|\frac{L_s}{2}|)}{1 - \prod_{s \in \mathcal{N}(i) \setminus \{j\}} \tanh(|\frac{L_s}{2}|)} \right] \tag{5.14}$$

and $L_s = \gamma_{ch} x_s^{(q)}$, with the assumption that since $|\tanh(.)| < 1$, then the approximation $\ln(\frac{1+x}{1-x}) \approx x$ can be used.

The solution update takes the following form:

$$x_j^{(q+1)} = x_j^{(0)} + \alpha R_j \tag{5.15}$$

where the descent step parameter $\alpha = 1/J$, and the update given in (5.15) is modified from the standard GD by updating only the first solution $x^{(0)} = \gamma_{ch} y$. □

Table 5.1 summarizes our contribution, where for each MLGD algorithm, namely the IDA, ITD, SRBI and ISRBI algorithms, the associated objective function expression and descent step value are presented.

Note that this result is very interesting in the way that a universal interpretation of MLGD algorithms is established, which further simplify and unify the current investigations on how it is possible to improve the performance of MLGD algorithms, as well as how it is possible to devise an efficient tradeoff between performance and complexity, where here we have shown that both the complexity and performance depend on the mathematical expression of the objective function representing the decoding problem of OSMLD LDPC codes.

MLGD algorithm	Objective function $f(x_0, x_1, ..., x_{n-1})$	Descent Step
IDA	$\sum_{0\leq i<m}\left(\prod_{s\in\mathcal{N}(i)}\tanh(x_s)\right)$	$\alpha_j = \frac{1}{1-\tanh^2(x_j)}$
ITD	$\frac{\gamma}{2}\parallel x\parallel_2^2 + \sum_{0\leq i<m}\prod_{s\in\mathcal{N}(i)}\tanh(x_s)$	$\alpha = \frac{1}{J}$
SRBI	$\sum_{0\leq i<m}\prod_{s\in\mathcal{N}(i)} x_s$	$\alpha = 1$
ISRBI	$\sum_{0\leq i<m} w_i \prod_{s\in\mathcal{N}(i)} x_s$	$\alpha = 1$

Table 5.1: MLGD algorithms and their corresponding objective functions to maximize with the Gradient-Descent

5.3 Improved local optimization techniques

After the proposition of a universal view of the majority-logic decoding algorithms in the literature as a gradient-descent optimization, a straightforward perspective is to exploit other improved variants of the GD algorithm in order to result in enhanced versions of these decoding algorithms. Indeed, many variants of the GD were already proposed in the literature, where each variant tends to address some limitations of the standard GD optimization technique. One such limitation is the problem of local optimums, where the standard GD update has shown that sometimes is unable to escape from these local optimums. Another limitation of the standard GD is the oscillations of the convergence process across the slopes of the ravine while making hesitating progress towards the local optimum.

We shall first introduce some variants of the GD algorithm, namely 3 different variants, where we will emphasize the benefits of each approach compared to the standard GD. Then we propose to reformulate the majority-logic decoding of LDPC codes problem following the formalism of these improved variants, and we propose to investigate the mathematical model of each MLGD algorithm defined in the previous subsection following the considered variants of the GD. Moreover, we transform the unconstrained minimization problem of decoding LDPC into the contrained optimization formalism, by investigating the according objective function as well as the constraints, in order to exploit the Augmented Lagrange Method of Multipliers (ALMM) technique for solving the decoding problem. The aims of this contribution is to provide several mathematical models for the decoding problem of LDPC codes, and then to evaluate the performance of the proposed models in future research directions. The

proposed models are enhanced versions of the existing MLGD algorithms in the literature.

5.3.1 Gradient-Descent with Momentum (MGD)

The standard Gradient-Descent technique suffers in some use cases from the problem of navigating the surfaces that curve more steeply in one dimension than another [141,160]. The GD performs many oscillations in these scenarios and makes hesitant convergence towards the local optimum.

Momentum Gradient-Descent [131] is a method that helps accelerate GD in the relevant direction and dampens oscillations, by adding a fraction γ of the update vector of the past time step to the current update vector [141]. The Momentum GD update rule (in vectorial notation) takes the following form:

$$\begin{aligned} v^t &= \gamma v^{t-1} + \alpha \nabla_x f(x) \\ x &= x - v^t \end{aligned} \tag{5.16}$$

The momentum factor is usually equal to $\gamma = 0.9$ [141]. This phenomena is illustrated by pushing a ball down a hill. While rolling downhill, the ball accumulates momentum and increases its speed until reaching its terminal velocity. Thus, similarly, the momentum term increases for dimensions whose gradients evolve in the same directions and reduces updates for dimensions with directions that vary significantly, which results in a faster convergence of the algorithm and more reduced oscillations compared to the standard GD.

5.3.2 Nesterov Accelerated Gradient (NAG)

When the ball is rolling down a hill, it is blindly following the slope. A better strategy would be to have a smarter ball which anticipates the slopes of the surface, so that it could slow down before the hill slopes up again. Nesterov Accelerated Gradient (NAG) [119] is an improved variant that allows to anticipate the approximate future position of the convergence, by predicting the surface slopes.

The momentum term γv^{t-1} is essentially used to move the parameters. Thus, an approximation of of the next position of the parameters is obtained by $x - \gamma v^{t-1}$, and the NAG technique can effectively look ahead by calculating the gradient with respect to the approximate future position of the parameters. The NAG update rule is given by:

$$v^t = \gamma v^{t-1} + \alpha \nabla_x f(x - \gamma v^{t-1}) \tag{5.17}$$
$$x = x - v^t$$

Similarly to the MGD, the momentum term is usually set to a value around $\gamma = 0.9$. From (5.17), the NAG method perform a big jump in the direction of the previous accumulated gradient, measures the gradient and then perform an anticipatory correction which prevents it to converge too fast in the wrong direction. This anticipatory process results in an increased responsiveness and improved performance compared to the other GD variants.

5.3.3 Adaptive Gradient (Adagrad)

Another interesting desired feature of the GD is to adapt the updates to each individual parameter to perform larger or smaller updates depending on their importance. Adagrad [36] enables the adaptation of the descent step (or learning rate in machine learning) to the parameters, by allowing larger updates for infrequent and smaller updates for frequent parameters.

The update rule of Adagrad consists of the modification of the descent step at each time step t for every parameter j based on the past gradients that have been computed for x_j, and is given, for every $j \in [0, ..., n-1]$ by:

$$x_j^{t+1} = x_j^t - \frac{\alpha}{\sqrt{G_{jj}^t + \varepsilon}} \nabla_{x^t} f(x_j^t) \tag{5.18}$$

where $G^t \in \mathbb{R}^{n \times n}$ is a diagonal matrix where each diagonal element j, j is the sum of the squares of the gradients w.r.t. x_j up to time step t, while ε is a smoothing term used for avoiding division by zero.

The key advantage of Adagrad is the elimination of manual tuning of the descent step α, in addition to the adaptive update of each parameter. It is shown that most implementations use a default value of $\alpha = 0.01$ [141]. However, for our case it might strongly impact the performance of the decoder. On the other hand, the weakness of Adagrad is that the accumulated squared gradients in the denominator of (5.18) results in making the descent step infinitesimally small. This limitation is resolved by another variant of the GD so-called Adadelta [195], which is not invoked in this study.

5.3.4 The Augmented Lagrangian Method of Multipliers

The Augmented Lagrange Method of Multipliers (ALMM) is a powerful technique for solving constrained minimization (or maximization) problems. The Augmented Lagrangian and the Method of Multipliers were initially introduced in the late 1960s by Hestenes and Powell [62, 129]. Much of the early work was recently consolidated and generalized by Bertsekas [14]. The ALMM is a penalty-based local optimization technique, widely used in many applications, such as statistical problems, constrained sparse regression, sparse signal recovery, image restoration and denoising, etc. The key principle of this technique consists of adding a penalty term to the objective function, so that the penalty term determines the severity of the penalty by prescribing a high cost to infeasible points. The resulting function is the Augmented Lagrangian function which is essentially used to approximate the original constrained problem as an unconstrained problem. Note there are many different penalty functions that can be used for this purpose. Also, there are different possible constraints, for instance equality and inequality constraints.

Let's consider the following constrained minimization problem:

$$\begin{aligned} &\text{Minimize } f(x) \\ &\text{subject to } x \in X \text{ and } h(x) = 0 \end{aligned} \tag{5.19}$$

where $f:\ \mathbb{R}^n \to \mathbb{R}$ and $h:\ \mathbb{R}^n \to \mathbb{R}^m$ are continuous, and X is a closed set.

The Augmented Lagrangian of (5.19) is given by:

$$L_\rho(x,\lambda) = f(x) + \lambda^T h(x) + \frac{\rho}{2}\|h(x)\|_2^2 \tag{5.20}$$

where ρ is the *penalty parameter*, and λ is the *dual variable* or *Lagrange multiplier*.

If the penalty ρ is large, then the objective function (5.20) is strongly convex. The principle of ALMM is to solve the unconstrained minimization problem of the Augmented Lagrangian in (5.20), as follows:

$$x^* = \arg\min_{x\in X} L_\rho(x,\lambda) \tag{5.21}$$

If ρ is larger than a threshold, then x^* is a strict local minimum of $L_\rho(.,\lambda^*)$ corresponding to λ^*. Moreover, if we set λ close to λ^* and perform unconstrained minimization of the A.L., then we can find x close to x^* [14].

Now given ρ, x^0 and λ^0, applying the ALMM technique to (5.21) yields to the following algorithm :

$$x^{q+1} := \arg\min_x L_\rho(x^{(q)}, \lambda^q) \tag{5.22}$$

$$\lambda^{q+1} := \lambda^q + \rho h(x^{(q)}) \tag{5.23}$$

Generally, the A.L. given in (5.20) is first-order derivable if the objective function $f(x)$ is, thus the minimization problem (5.22) can be solved iteratively using the Gradient-Descent technique, taking into account the update rule in (5.23) of the Lagrange multiplier.

For achieving convergence and finding a strict local minimum, two first-order necessary conditions must be satisfied. The solution x^* is a strict local minimum if $\nabla_x L(x^*, \lambda^*) = 0$ and $\nabla_\lambda L(x^*, \lambda^*) = 0$ for $x^* \in \mathbb{R}^n$ and $\lambda^* \in \mathbb{R}^m$.

5.4 Improved update rules using variants of the Gradient-Descent

We will present various new mathematical models for the decoding problem of LDPC codes, inspired from the existing MLGD algorithms. Our aim is to propose new enhanced versions of each reviewed algorithm using the mentioned variants of the GD technique.

Based on Table 4.1, we propose here to introduce modifications to the standard update rule of each MLGD decoding algorithm, using the update rule associated to each variant of the GD optimization technique. Note that the main modifications of these GD variants consist of changing the update operation in order to quickly achieve the convergence and to avoid local optimums. In this case, a decoder falls in a local optimum when it outputs a non-valid codeword, different from that being transmitted.

As the key modification includes the update rule in each decoding algorithm, the objective functions presented in Table 5.1 and their first-order partial derivatives (corresponding to the soft reliability R_j) remain unchangeable. In Table 5.2, we represent for each MLGD algorithm, the mathematical expression of the soft-reliability R_j of the j^{th} symbol, which corresponds to the partial derivative of the objective function $f(x)$ w.r.t each variable x_j, for $j \in [0, ..., n-1]$. Note that $x_j \in \mathbb{R}$ and $x_j \in \mathbb{Z}$ for the unquantized and quantized algorithms, respectively. We also indicate in the same table the initialization step of each decoding algorithm. The new update rules derived from the variants of the GD technique are summarized in

Table 5.3. Note that the descent step α of each algorithm must be numerically optimized for performance improvement, as well as the momentum term γ. For spatial constraints, let's assume the following notations: $S_{i,j}^{(q)} = \prod_{s \in \mathcal{N}(i) \setminus \{j\}} sgn(x_s^{(q)})$ and $\widetilde{S}_{i,j}^{(q)} = \prod_{s \in \mathcal{N}(i) \setminus \{j\}} \tanh(x_s^{(q)})$.

MLGD algorithm	Initialization	Partial derivative $\nabla_{x^{(q)}} f(x_j^{(q)})$
IDA	$x_j^{(0)} = \gamma_{ch} y_j$	$(1-\tanh^2(x_j^{(q)})) \sum_{i \in \mathscr{M}(j)} \widetilde{S}_{i,j}^{(q)}$
ITD	$x_j^{(0)} = y_j$	$\gamma_{ch} x_j^{(q)} + \sum_{i \in \mathscr{M}(j)} \omega_{i,j} S_{i,j}^{(q)}$
SRBI	$x_j^{(0)} = Q_j$	$\sum_{i \in \mathscr{M}(j)} S_{i,j}^{(q)}$
ISRBI	$x_j^{(0)} = Q_j$	$\sum_{i \in \mathscr{M}(j)} \omega_{i,j} S_{i,j}^{(q)}$

Table 5.2: Initialization and partial derivatives associated to various MLGD algorithms

New update		
MGD	NAG	Adagrad
$v_j^{(q)} = \gamma v_j^{(q-1)} + \alpha \nabla_{x^{(q)}} f(x_j^{(q)})$ $x_j^{(q+1)} = x_j^{(q)} - v_j^{(q)}$	$v_j^{(q)} = \gamma v_j^{(q-1)} + \alpha \nabla_{x^{(q)}} f(x_j^{(q)} - \gamma v_j^{(q-1)})$ $x_j^{(q+1)} = x_j^{(q)} - v_j^{(q)}$	$x_j^{(q+1)} = x_j^{(q)} - \frac{\alpha}{\sqrt{G_{jj}^{(q)} + \varepsilon}} \nabla_{x^{(q)}} f(x_j^{(q)})$

Table 5.3: Improved update rules for various MLGD algorithms using some variants of the Gradient-Descent

5.5 A New Universal Decoding Approach for Linear Block Codes using the Augmented Lagrange Method of Multipliers

We extend our modeling to a more general decoding problem, that of decoding every linear code characterized by a parity-check matrix, and we use the ALMM technique for addressing this problem and devising a new algorithm capable of decoding every class of linear codes. The main objective is to devise a set of decoding algorithms for opening new perspectives towards a universal improvement and comparisons of the various methods, as well as providing a facility to establish analytical convergence and performance studies of the decoding problem using these proposed frameworks. The difference between each proposition relies on the considered objective function, the constraints (if considered), and the employed optimization technique for solving the decoding problem.

Here we propose a new decoding model based on the ALMM constrained optimization technique. This model is associated to the ITD (and IDA) algorithm [88,105], despite it is possible to devise its reduced complexity version inspired from the objective function introduced in the Gradient-Descent MLGD (GD-MLGD) algorithm [194], providing an efficient tradeoff between performance and complexity. However, we address our interest here to the general model, and the reduced complexity version will be also addressed for solving a more general problem for decoding linear codes. Our main purpose is to transform the unconstrained objective functions corresponding to these algorithms into a constrained optimization problem. The key advantage of this purpose is to help the decoding algorithm to escape from local optimums, and to prevent solutions that violate the considered constraints, resulting in guaranteed convergence properties to the ML codeword, which inherit essentially from the optimization technique. Note that when the constraints are linear, a better choice would be the use of Linear Programming (LP) decoding techniques.

Indeed, decoding LDPC codes or more generally binary linear codes using the LP technique was first introduced in 2003 by Feldman in his doctoral thesis [44], and much of the obtained results were published later in [45]. These starter works have initially led the research community to investigate new perspectives for decoding linear block codes and LDPC codes using constrained optimization techniques. Independently in [182], a comparison and the relation between LP decoding and the Min-Sum (MS) algorithm was investigated. In [22], an efficient pseudo-codeword search algorithm was proposed for LP decoding of LDPC codes. In [183] and [162], decoding complexity reduction and adaptive methods for LP decoding were investigated. Later in [9], LP decoding of LDPC codes was improved by the use of the Alternative Direction Method of Multipliers (ADMM) constrained optimization technique. A projection algorithm was proposed in [203] for reducing the complexity of the ADMM decoding algorithm. More recently, the penalized ADMM decoder (ADMM-PD) was proposed with investigations on different penalty functions for achieving performance improvement [102]. In [101], the same authors proposed the ADMM-PD technique for decoding non-binary LDPC codes.

Let's consider the constrained model inspired from the ITD and IDA algorithms. We would like to devise a decoding algorithm suitable to decode every class of linear block codes defined by a parity-check matrix. As shown in Table 5.1, the objective function associated to the ITD algorithm includes a term related to the l_2-norm of the variable, and a term related to the generalized syndrome weight. In contrast, the IDA includes only the generalized syndrome weight. Intuitively, it is possible to add to the objective function a term related to the ML

argument, defined by the distance between the solution and the received signal y. Here, there are many distance metrics that can be considered. A suitable model for the decoding problem would be the consideration of the ML argument as a term to minimize, and the constraints can be described by the minimum value that the generalized syndrome weight must take.

Next proposition states the constrained model for the decoding problem based on the generalized syndrome weight.

Proposition 5.5. *The iterative decoding problem of LDPC codes can be stated as follows:*

$$\textit{Minimize } f(x) = -\sum_{j=0}^{n-1} sgn(x_j)y_j \tag{5.24}$$

$$\textit{subject to } x \in \mathbb{R}^n \textit{ and } \prod_{s\in\mathscr{N}(i)} \tanh(x_s) - 1 = 0,\ \forall i \in [0, m-1] \tag{5.25}$$

The objective function $f(x)$, such that $f:\ \mathbb{R}^n \to \mathbb{R}$, includes a correlation between the solution vector x and the received signal y. Since $\tanh(x_s) < 1$, the optimal value of the sum in the constraint cannot be greater than m, i.e. when the generalized syndrome weight is equal to m, then all the parity-checks are satisfied, which means that $sgn(x)$ is a bipolar codeword of $\mathscr{C}$. The Augmented Lagrangian (A.L.) corresponding to (5.25) takes the following form:

$$L_\rho(x,\lambda) = -\sum_{j=0}^{n-1} sgn(x_j)y_j + \lambda \sum_{0\le i<m} \left(\prod_{s\in\mathscr{N}(i)} \tanh(x_s) \right) - m + \frac{\rho}{2} \sqrt{\sum_{0\le i<m} \left| \left(\prod_{s\in\mathscr{N}(i)} \tanh(x_s) \right) - 1 \right|^2} \tag{5.26}$$

where λ is the Lagrange Multiplier, and ρ is the penalty factor. For sake of simplicity, we consider that λ is a real number instead of being a vector, thus $\lambda \in \mathbb{R}$.

Therefore, given $x^{(0)}$, $\lambda^{(0)}$ and ρ, the decoding problem now consists of solving the following optimization problem:

$$x^{(q+1)} := \arg\min_{x\in\mathbb{R}^n} L_\rho(x^{(q)}, \lambda^q) \tag{5.27}$$

$$\lambda^{(q+1)} := \lambda^q + \rho \left(\sum_{0\le i<m} \left(\prod_{s\in\mathscr{N}(i)} \tanh(x_s^{(q)}) \right) - m \right) \tag{5.28}$$

The minimization of the A.L. in (5.27) can be solved iteratively using the gradient-descent. However, the main drawback of this model is the computational complexity involved in the calculation of the partial derivatives $\frac{\partial L_\rho(x^{(q)},\lambda^{(q)})}{\partial x_j^q}$ for solving (5.27), which results in an unpractical decoding algorithm due to the high complexity.

One alternative for reducing the computational complexity consists of the consideration of another metric for the calculation of the syndrome weight. Similarly to the SRBI algorithm, the bipolar syndrome weight can be considered instead of the generalized syndrome weight. Thus, the problem takes the following simplified form stated in the next Proposition.

Proposition 5.6. *The iterative decoding problem of LDPC codes can be stated as follows:*

$$\textit{Minimize } f(x) = -\sum_{j=0}^{n-1} sgn(x_j) y_j \tag{5.29}$$

$$\textit{subject to } x \in \mathbb{R}^n \textit{ and } \prod_{s \in \mathscr{N}(i)} sgn(x_s) - 1 = 0,\ \forall i \in [0, m-1] \tag{5.30}$$

The augmented Lagrangian corresponding to (5.30) takes the following form:

$$L_\rho(x,\lambda) = -\sum_{j=0}^{n-1} sgn(x_j) y_j + \lambda \sum_{i=0}^{m-1} \left(\prod_{s \in \mathscr{N}(i)} sgn(x_s) \right) - m + \frac{\rho}{2} \sqrt{\varepsilon + \sum_{i=0}^{m-1} | \left(\prod_{s \in \mathscr{N}(i)} sgn(x_s) \right) - 1|^2} \tag{5.31}$$

where ε is a smoothing term used for avoiding the division by zero.

Now, with the given A.L. in (5.31), the decoding problem consists of solving the following optimization problem:

$$x^{(q+1)} := \arg\min_{x \in \mathbb{R}^n} L_\rho(x^{(q)}, \lambda^q) \tag{5.32}$$

$$\lambda^{(q+1)} := \lambda^q + \rho \left(\sum_{i=0}^{m-1} \left(\prod_{s \in \mathscr{N}(i)} sgn(x_s^{(q)}) \right) - m \right) \tag{5.33}$$

Note in (5.31) that when the constraint is satisfied, i.e. $sgn(x)$ is a bipolar codeword, then all the m parity-check equations are satisfied, therefore $\prod_{s \in \mathscr{N}(i)} sgn(x_s) = 1$ for each $0 \leq i < m$, thus the penalty argument inside the square root is zero. Otherwise, we have $\prod_{s \in \mathscr{N}(i)} sgn(x_s) = -1$.

Let $|\mathscr{N}(i)|$ denotes the weight of the i^{th} row of the parity-check matrix. The partial derivative of the A.L. can be simplified for the codes with a regular row weight. In this case, we denote by $J = |\mathscr{N}(i)|$ the row weight of the parity-check matrix. Thus the partial derivative takes the following form:

$$\begin{cases} \frac{\partial L_\rho(x^{(q)},\lambda^{(q)})}{\partial x_j^q} = -y_j + \lambda^{(q)} \sum_{i\in\mathscr{M}(j)} \left(\prod_{s\in\mathscr{N}(i)\setminus\{j\}} sgn(x_s^{(q)})\right) & \text{if } \sum_{0\le i<m} \prod_{s\in\mathscr{N}(i)} sgn(x_s^{(q)}) = m \\ \frac{\partial L_\rho(x^{(q)},\lambda^{(q)})}{\partial x_j^q} = -y_j + \left(\sum_{i\in\mathscr{M}(j)} \prod_{s\in\mathscr{N}(i)\setminus\{j\}} sgn(x_s^{(q)})\right)\left(\lambda^{(q)} - \frac{\rho J}{\sqrt{\varepsilon+4m}}\right) & \text{otherwise} \end{cases} \tag{5.34}$$

Note that the obtained final form of the partial derivative in (5.34) requires relatively low computational operations, which results in a practical low-complexity decoding algorithm, which can also be used to decode every linear block code defined by a parity-check matrix. Now, the gradient-descent final solution update takes the following vectorial form:

$$x^{(q+1)} := x^{(q)} + \alpha \frac{\partial L_\rho(x^{(q)},\lambda^{(q)})}{\partial x_j^q} \tag{5.35}$$

$$\lambda^{(q+1)} := \lambda^q + \rho \left(\sum_{i=0}^{m-1} \left(\prod_{s\in\mathscr{N}(i)} sgn(x_s^{(q)}) \right) - m \right) \tag{5.36}$$

where $0 < \alpha \le 1$ is the descent step, which can be optimized separately for improving performance. Note that it is optional to use a variable penalty factor ρ which evolves with decoding iterations, however, for sake of optimization simplicity, we consider a fixed penalty factor during all the decoding process.

Therefore, with the given formulation, the initialization of the proposed algorithm (ALMM-D) includes fixing a maximum number of allowed decoding iterations I_{max}, the penalty factor ρ, the Lagrange multiplier $\lambda^{(0)}$ and the descent step α. The partial derivative $R_j = \frac{\partial L_\rho(x^{(q)},\lambda^{(q)})}{\partial x_j^q}$ is initialized by zero for $0 \le j < n$. The input of the algorithm is the received real (or complex) signal $y \in \mathbb{R}^n$ in the case of an AWGN or a fading channel, or a binary sequence $y \in \mathbb{F}_2^n$ in the case of a Binary Symmetric Channel (BSC). The input signal is considered as an initial solution, i.e. $x^{(0)} = y$. The ALMM-D algorithm performs the decoding process iteratively until a stopping criteria is met. The stopping criteria are defined by reaching the maximum number of iterations, i.e. $q = I_{max}$, or when the fluctuation of the solution becomes sufficiently small, i.e. $\sum_j \|z_j^{(q)} - z_j^{(q-1)}\|_2^2 < \varepsilon^2$, where $z^{(q)}$ is the hard decision vector of the solution $x^{(q)}$.

The ALMM-D algorithm is described in Algorithm 5.1.

Algorithm 5.1 ALMM-D

Input: Received signal y, I_{max}, ρ and α.
Output: Binary Decoded sequence $z^{(q)}$.

- - - - - - - - - - - Initialization - - - - - - - - - - -
$x_j^{(0)} = y_j$ and $\lambda_j^{(0)} = 0$ for all $0 \leq j < n$.
- - - - - - - - - - - Iterations - - - - - - - - - - - - -
for $q = (0 : I_{max})$ **do**
    Calculate $z^{(q)}$, If $(\sum_j \|z_j^{(q)} - z_j^{(q-1)}\|_2^2 < \varepsilon^2$ or $q = I_{max})$, then stop decoding.
    for $(j = 0 : n-1)$ **do**
        Calculate the partial derivative R_j of the A.L. using (5.34).
    end for
    Update the solution vector $x^{(q+1)}$ using (5.35).
    Update the Lagrange multiplier $\lambda^{(q+1)}$ using (5.36).
    Update the penalty parameter by $\rho^{(q+1)} = \beta\rho^{(q)}$ where $\beta \geq 1$. (Optional)
end for

Note that if the channel is symmetric, then the probability that the ALMM-D algorithm fails is independent of the codeword that was transmitted. As mentioned in [102], it is generally difficult to determine whether or not a feasible point is a global minimizer of a non-convex optimization problem, such as the problem in Proposition 4.9. However, one can still use the ML certificate property of LP decoding [44] to test whether or not the obtained solution is a ML solution.

5.6 Conclusion

A unified view of majority-logic decoding algorithms was presented in this paper. We have shown that all these decoding techniques are derived from a gradient-descent maximization of a particular derivable objective function. A suitable correspondence between these algorithms and the gradient-descent was given, with an investigation on the descent-step value and first-order derivatives for each decoding algorithm. Moreover, we have introduced a more general decoding model based on constrained optimization techniques, namely the Augmented Lagrangian Multiplier Method, for addressing the problem of decoding both LDPC (and OSMLD) codes and generally all linear block codes characterized by a parity-check matrix. This approach has resulted in a low-complexity decoding algorithm (ALMM-D) that we have proposed based on this mathematical constrained model.

An important future direction of this work is the performance evaluation of the proposed enhanced versions of majority-logic decoders as well as the ALMM-D decoding algorithm

for various OSMLD, LDPC and linear block codes with an adequate investigation on the optimization of the parameters involved in these decoding techniques. Additionally, the investigation of other mathematical expressions of the objective functions that represent these decoding problems is considered as another interesting future direction.

Chapter 6

Evaluation of OSMLD codes in DNA Data Storage and Modern Wireless Communication Systems

6.1 Cyclic Ternary Difference-Set Codes for DNA Data Storage Systems

6.1.1 Introduction

Recently, DNA based data storage systems has seen a large interest from research and industrial communities, due to the ever growing data density required by the latest technologies, as big data, IoT, high quality streaming, immersive technologies (VR, AR, holographs). Indeed, the current magnetic based data storage mediums, namely optical, digital and cloud data storage, has proven their density limitations, and needs to be maintained regularly. It was reported by [134] that the total amount of data generated in 2016 has reached 16.1 ZetaBytes, and will exponentially grow to reach 163 ZetaBytes in the horizon of 2025. Clearly, the classical data storage mediums will fail to handle this challenge and must furthermore be replaced by other storage candidates. Indeed, many researchers developed a natural approach beyond the classical storage mediums, towards storing massive data into the DNA molecule. As reported by many scientific papers, the DNA information contained in the Neanderthal bones has been recovered successfully even if it has been emerged at least 200,000 years ago. This natural approach for data storage has proven the low-density, scalability and long-term stability and storage that can provide DNA-based data storage systems.

A DNA molecule is composed from a sequence of nucleotides, each one taking a value from the four bases (A: Adenine, C: Cyanine, G: Guanine, T: Thymine). Nucleotides are organized into chains of DNA chunks, called oligonucleotides (or oligos). Theoretically, 1 gram per DNA can store 455 EB (Exabytes) of information in low-maintenance environments, with a long-term storing longevity [51], which outperforms by far the current classical digital storage mediums.

Many efforts have been devoted recently for storing data into the DNA code. In 2012, Georges Church and his team [25] proved experimentally the concept of storing data in DNA molecules, by storing 22 MB of data in DNA. In 2013, N. Goldman *et al.* [51] made the breakthrough by proposing and testing a new efficient approach for storing data of size 739 KB on DNA and retrieving it back successfully. They proposed an efficient DNA encoding scheme with the given sequencing and synthesizing technologies in 2013 for storing archives of several MegaBytes (MB) in a $500-5000$ years horizon, in low-maintained environments. Authors have used four folds redundancy to retrieve data from one of the DNA strands, and proposed the use of the ternary Huffman code and differential encoding for avoiding homopolymer runs. However, this approach provides an increase in the DNA length, which limits it for the commercial use due to the cost increase. Later, the synthesizing and reading-access techniques have been improved, giving the possibility to synthesize a larger amount of DNA data. Based on the Goldman's approach, many different approaches were proposed in the literature for error control coding in DNA data storage. Authors in [71] used an approach where each group of five bits is followed by one parity-bit for error detection. In order to enhance reliability, they proposed an encoding approach based on inserting multiple copies of data into multiple regions of the genome of the host organism. In [61], the extended $(8,4,4)$ binary Hamming code, or repetition coding were used for encoding data. In [95], the authors proposed the use the ternary non-linear Golay code with parameters $(n,k,d_{min}) = (11,6,5)_3$, and theoretically, a high density was reached, giving 115 exabytes (EB) that can be stored in one gram of DNA.

In this section, we propose to further enhance and simplify the encoding approach in DNA data storage, by the use of ternary cyclic Difference-Set Codes (DSC). We show that efficient storage density and error rates can be achieved using the proposed scheme, in addition to a simple majority-logic decoding procedure for retrieving data.

This section is organized as follows. In Subsection 6.2.2, we briefly describe the DNA channel model, where we present the calculation of the Shannon capacity, and the coding potential associated to constrained DNA channels. Also, we discuss the proposed modulation schemes

for this channel. In Subsection 6.2.3, previous works on DNA forward error correction and the Goldman's encoding scheme [51] are reviewed. Subsection 6.2.4 introduce the proposed scheme, where cyclic ternary DS codes are presented followed by the proposed DNA encoding approach. Subsection 6.2.5 presents performance analysis of the proposed DNA data storage scheme, with a comparison with the relevant previous works. Finally concluding remarks are presented in Section 6.3.

6.1.2 DNA data storage Channel Model

6.1.2.1 Channel Model and Capacity

DNA data storage systems can be considered as a classical digital transmission over a noisy channel. The DNA information is transmitted over the channel by synthesizing DNA oligos. The information is received by sequencing the DNA oligos and decoding the sequenced sequence. The channel noise is caused by various experimental factors, including DNA synthesis imperfections, PCR dropouts, degradation of DNA molecules over time, and sequencing errors. In contrast to other classical theoretic channels, where the noise is identical and independently distributed, the error patterns in DNA depends essentially on the input sequence [40, 41].

It was shown that biological, bio-chemical and bio-physical processes are causing errors, while the physical and chemical effects introduces by itself errors to DNA oligos by the time [16]. Church and his team [25] gathered DNA channels characteristics by conducting several experimental analysis. Technically, 3 types of errors were observed in DNA channels. First, flipping errors (swapping) occur when a DNA nucleotide symbol is replaced by another one. Additionally, insertion and deletion errors were also detected. Oligos that were not found in the DNA are called missing oligos. The obtained experimental results showed that the swap error rate lies approximately between $6.0 \cdot 10^{-4}$ and $1.4 \cdot 10^{-3}$, while insertion and deletion errors are $1.0 \cdot 10^{-3}$ and $5.0 \cdot 10^{-3}$ respectively [25]. Authors in [16] claimed that the DNA channel is a data memory-less channel. In [40], authors described the DNA channel as a constrained channel, concatenated to an erasure channel. Previous studies [7, 42, 43, 85, 140, 152] has identified that homopolymer runs and GC content are the major constraints that impacts synthesis and sequencing errors.

Figure 6.7 illustrates the transmission model applied to data storage systems.

The transmitter generate a synthesized DNA oligo of length N, which include information data and an indexing header. As DNA are organized in a mixed pool due to the multiplexing

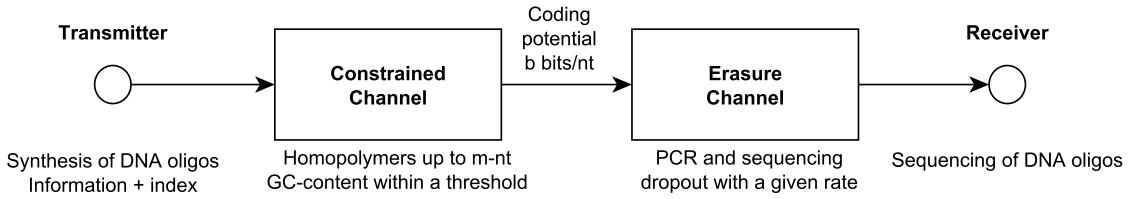

Figure 6.1: DNA Data Storage Channel Model

architecture of synthesis reactions and high throughput sequencing, it is important to index each oligo before transmitting it. We will denote the appended index header oligo by h_{index}. The synthesized oligo is transmitted over a constrained channel. Here an oligo is considered a valid sequence if its GC content is within $0.5 \pm c_{gc}$, and its longest homopolymer length is up to m nucleotides. Otherwise, the sequence is considered invalid and cannot be transmitted. Then valid sequences are exposed to an erasure channel with low dropout rate δ_v. Finally, the transmitted DNA oligos are sequenced for reading the received data. Additionally, forward error correction can be introduced, by appropriately choosing a code C for encoding data before the transmission, and decoding the received data after sequencing DNA oligos. Erlich *et al.* [40] established a theoretical study of the DNA channel capacity, by assigning realistic values to the parameters m, c_{gc}, h_{index} and δ_v into his model.

Let's denote by A_X the set of all possible transmitted DNA sequences of length N, and by A_Y the set of all possible received sequences. Let $X \in A_X$ and $Y \in A_Y$ represents a random DNA transmitted sequence and received sequence respectively. The information capacity of each oligo is given by [40]:

$$C = \max_{p_X} I(X;Y) \tag{6.1}$$

where $I(X;Y)$ is the mutual information of X and Y, defined by:

$$I(X;Y) = H(X) - H(X \mid Y) \tag{6.2}$$

Thus, the information capacity per nucleotide C_{nt} is obtained as follows:

$$C_{nt} = \frac{C}{N} \tag{6.3}$$

In a constrained channel, the mutual information is maximized by transmitting equiprobable valid sequences [107], therefore $H(X) = \log_2 |A_X|$, where $|.|$ denotes the size of a set.

A sequence is dropped out with a probability δ_v, and is received without dropping with a probability $1-\delta_v$. Thus, the capacity per nucleotide is defined as:

$$C_{nt} = \frac{(1-\delta_v)\log_2 |A_X|}{N} \tag{6.4}$$

6.1.2.2 The homopolymer constraint

The size of valid codewords, given the homopolymer constraint is given by:

$$|A_X^h| = Q(m,l)4^l \tag{6.5}$$

where $|A_X^h|$ is the set of valid codewords under the homopolymer constraint, and $Q(m,l)$ is the probability to observe up to an m-nt homopolymer run in a random l-nt sequence.

Let $q_m(p,l)$ denotes the probability of not observing m or more successes in l Bernoulli trials with a success probability p and a failure probability $q = 1-p$. In [46], a tight approximation was proposed:

$$q_m(p,l) \approx \frac{\beta}{x^{l+1}} \tag{6.6}$$

where

$$x = 1 + qp^m + (m+1)(qp^m)^2 \tag{6.7}$$

and

$$\beta = \frac{1-px}{(m+1-mx)q} \tag{6.8}$$

For practical purposes, authors in [40] have proposed to approximate the distribution of observing up to m-nt homopolymer runs as the product of four independent events:

$$Q(m,l) = \left[q^{m+1}(p=0.25,l)\right]^4 \tag{6.9}$$

Thus for any $m \geq 3$ and $l \geq 50$, combining (6.4), (6.5) and (6.9) yields to:

$$\frac{\log_2 |A_X^h|}{l} = 2 - \frac{3\log_2 e}{4^{m+1}} \tag{6.10}$$

Clearly, the information capacity under the homopolymer constraint does not depend on the length of oligos, but depends especially on the maximal length of homopolymers.

6.1.2.3 The GC content constraint

Under the GC content constraint, the probability p_{gc} that a sequence of l nucleotides is within $0.5 \pm c_{gc}$ is given by:

$$p_{gc} = 2\Phi(2\sqrt{l}c_{gc}) - 1 \tag{6.11}$$

where Φ is the cumulative function of a standard normal distribution. Therefore, the number of bits that can be transmitted per nucleotide takes the form:

$$\frac{\log_2 |A_X^{gc}|}{l} = 2 + \frac{\log_2 \left[2\Phi(2\sqrt{l}c_{gc}) - 1\right]}{l} \tag{6.12}$$

It was noted by Erlich and Zielinski [40] that the impact of the GC content constraint on reducing the information capacity per nucleotide is negligible when $c_{gc} \geq 0.05$ and $l \geq 50$.

6.1.2.4 The Coding Potential

When the biochemical constraints are taken together into consideration, namely the homopolymer and GC content, then the coding potential per nucleotide b is a metric that represents the output of the constrained channel (see Figure 6.1) , which takes the following form:

$$\begin{aligned} b &= \frac{\log_2 |A_X|}{l} \text{ (bit/nt)} \\ &= \frac{\log_2 |A_X^h \cup A_X^{gc}|}{l} \text{ (bit/nt)} \\ &= 2 - \frac{3\log_2 e}{4^{m+1}} - \frac{\log_2 \left[2\Phi(2\sqrt{l}c_{gc}) - 1\right]}{l} \text{ (bit/nt)} \end{aligned} \tag{6.13}$$

In order to select suitably b, a practical and realistic set of constraints m, c_{gc} and l must be used.

As mentioned before, it is crucial to add an indexing header to encoded data, as the oligos are not linearly sequenced. By adding a header of size K to oligos, the coding potential b is further reduced, and can be rewritten as follows:

$$b = 2 - \frac{3\log_2 e}{4^{m+1}} - \frac{\log_2 \left[2\Phi(2\sqrt{l}c_{gc}) - 1\right]}{l} - \frac{\log_2 K}{l} \text{ (bit/nt)} \tag{6.14}$$

6.1.2.5 Modulation

From the DNA channel characteristics mentioned above, a suitable modulation must be applied in order to limit the error propagation phenomena. A feasible modulation must take into consideration the following conditions:

- When a nucleotide symbol is erroneous, the error should be propagated to the minimum number of digits after the demodulation.
- In order to avoid homopolymers, the maximal run-length of similar nucleotides should be limited to 3.
- Self-complementary DNA sections has to be avoided because it causes amplification issues of the corresponding oligo, and also a significant information density loss.

Based on these constraints, an efficient modulation scheme was proposed in [16] for handling these limitations. Interested readers are referred to [16]. Also, the differential encoding (base 3 to DNA) technique used by Goldman [51], presented next in Table 6.1, provides satisfying results. However, this modulation technique is only useful when ternary symbols are considered for modulation.

6.1.3 DNA Forward Error Correction

Many academic and industrial research communities have proposed error-correcting schemes for DNA data storage, after that the pioneering works of Church's [25] and Goldman's [51] teams made a remarkable advance in long-term data storage. Using the next generation synthesizing and sequencing technology in 2012, Church [25] proposed on an efficient one bit per base encoding algorithm for storing digital information into a fixed length of DNA chunks (99 bases). For the header, flanking primers (headers) were inserted at the beginning and the end of information data in order to indicate the specific DNA segment in which the information data was encoded. A net information density of 0.83 bits per nucleotide was achieved. However, Church's approach suffers from the existence of homo polymers repeated DNA sequences that introduces writing and reading errors. This problem was resolved by Goldman in 2013 [51], by introducing the improved base-3 Huffman code with 3 symbols called trits (0, 1 and 2) in addition to differential encoding for data modulation. Thus, using Goldman's approach, binary data (one byte) was first encoded by the Huffman code, and then converted (modulated) to its corresponding DNA triplet where each trit digit was converted to one of the

3 nucleotides different from the last one, in order to avoid homopolymers. The differential encoding (modulation) technique used by Goldman to avoid homopolymers is presented in table 6.2.

Table 6.1: Goldman's base-3 to DNA modulation for avoiding repeated nucleotides

Previous *nt*	Next trit to encode		
	0	1	2
A	*C*	*G*	*T*
C	*G*	*T*	*A*
G	*T*	*A*	*C*
T	*A*	*C*	*G*

In the Goldman's scheme, a four redundancy protection system was used, where for each DNA strand of length 117 nt, comprising 114 nt for information and indexing data, and one parity symbol, plus a beginning and ending symbol. Two similar copies of the DNA strand were added and two odd indexed strands were reverse complemented. Additionally, prepending paddings and parity were used to allow error detection. The obtained net information density in this scheme was 0.34 bits per nucleotide.

Clearly, the Goldman's approach uses a large amount of redundancy to information data, providing lower DNA information density and higher costs. The Goldman's DNA data storage scheme is represented in Figure 6.8.

This approach can be efficiently improved by introducing error correcting codes for the protection of oligos. In [16], authors has achieved a net information density of 0.92 bits per nucleotide by encoding information data using the Reed Solomon (RS) code $(n,k,d_{min}) = (255,223,31)$ over $GF(2^8)$, and separately encode the header using a strong binary BCH code with parameters $(63,39,9)$. In [61], the extended $(8,4,4)$ binary Hamming code, or repetition coding were used for encoding data. A subcode $(11,256,5)_3$ of the ternary Golay code $(11,6,5)$ was proposed in [95] for encoding information data, and a DNA storage capacity of 115 ExaBytes (EB) was achieved. However, the indexing header remains unprotected, which impact error rates performance of the DNA storage medium. In [53], authors has introduced a double protection scheme, which consists of an inner Reed Solomon (RS) code over $GF(47)$ with parameters $(n,k) = (39,33)$ for correcting individual errors, and an outer RS code over the extension field $GF(47^{30})$ with parameters $(n,k) = (713,594)$ for the correction of erasures and errors from the inner decoder. The field $GF(47)$ was used in order

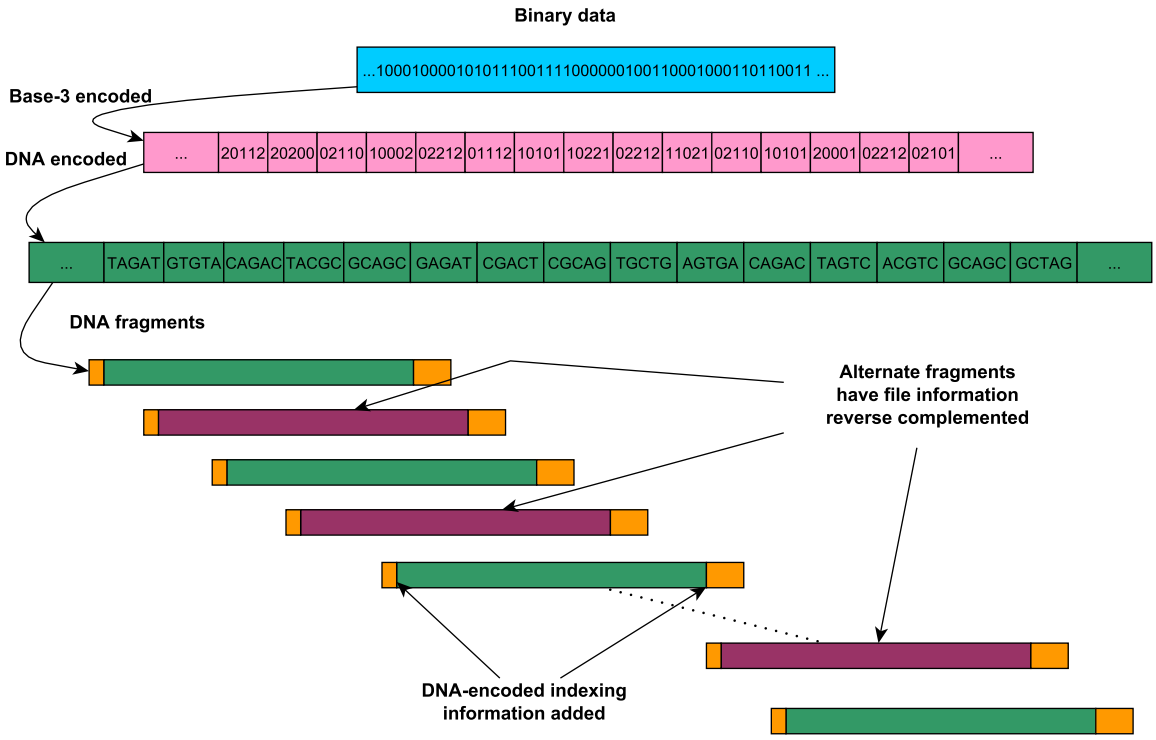

Figure 6.2: Goldman's Scheme for DNA Data Storage

to avoid homopolymers of length $m > 3$. Authors achieved a net information density of 1.14 bits per nucleotide, and claimed that their proposed encoding scheme is robustly suitable for digital data storage in DNA for thousands of years. Later, in [41], a Fountain Code technique was proposed to screen potential valid oligos to reach the maximal coding capacity of $b = 1.98$ bit/nt. 2 bytes of RS code redundancy was additionally added to protect both the seed and data payload. A net information density of 1.55 bits per nucleotide was achieved in this scheme.

6.1.4 The proposed DNA data storage scheme

6.1.4.1 Cyclic Difference-Set codes over $GF(3)$

Difference-Set codes represents an infinite class of algebraic majority-logic decodable codes derived from finite geometries, namely from projective geometry. DS codes were discovered in their cyclic form independently by Weldon [185] and Rudolph [145]. Weldon has shown that these codes are approximately powerful as BCH codes, with simpler decoding

implementation. Later in [86, 106], it was shown that OSMLD codes derived from finite geometries (DSC and EG) can be considered as finite geometry (PG, EG) LDPC codes, and they perform better than BCH codes and other linear block codes under the BP iterative decoding algorithm.

DS codes are completely orthogonalizable, as a consequence, the minimal distance is exactly defined by $d_{min} = J + 1$, where J is the number of orthogonal parity-check equations on each symbol being decoded, providing that the code is capable to correct each error pattern of weight t (t erroneous symbols) or less, by a simple majority-logic vote. Due to the orthogonality provided by the dual code, the length n of a cyclic DS code takes the following form:

$$n = J(J-1)+1 \tag{6.15}$$

Weldon has given the expression of the code length n and the minimum distance d_{min}, while Graham *et al.* [52] has devised in an accompanying paper the exact enumeration of the number of parity check symbols for cyclic DS codes.

For a prime p, and a positive integer $s > 0$, the code length, dimension, and minimum distance of a cyclic DS code over $GF(p)$ are given by:

$$n = p^{2s} + p^s + 1 \tag{6.16}$$

$$k = n - \left(\left[\begin{pmatrix} p+1 \\ 2 \end{pmatrix}^s \right] + 1 \right) \tag{6.17}$$

$$d_{min} = p^s + 2 \tag{6.18}$$

When $p = 3$, a cyclic difference-set code over $GF(3)$ is defined by the following parameters:

$$n = 3^{2s} + 3^s + 1 \tag{6.19}$$

$$k = n - 6^s - 1 \tag{6.20}$$

$$d_{min} = 3^s + 2 \tag{6.21}$$

The construction of cyclic DS codes is derived from combinatorial objects called Difference-Sets, introduced by Singer [155], which in turn are derived from finite geometry, namely Projective Geometries (PG), well investigated later in [57, 98].

Definition 6.1 (Difference-Sets)**.** A difference-set D is a collection of J integers $(d_0, d_1, ..., d_{J-1})$ modulo n, taken from the set $\{0, 1, ..., n-1\}$ such that $n = J(J-1)+1$, and all the $J(J-1)$ differences $(d_i - d_j)$ between the elements of D are distinct for $i \neq j$. That is, each difference occurs once.

A difference-set D can be considered in its polynomial form $D(x) \in GF(p)^n$, in the algebra of polynomials modulo $x^n - 1$. Thus, the difference-set polynomial $D(x)$ is written as follows:

$$D(x) = x^{d_0} + x^{d_1} + ... + x^{J-1} \tag{6.22}$$

From the difference-set properties involved above, differences between the exponents of $D(x)$ can be written as:

$$D_e(x) = D(x)D(x^{-1}) = J + x + x^2 + ... + x^{n-1} \tag{6.23}$$

where $D_e(x)$ represents the difference enumerator polynomial. Therefore, $D_e(x)$ may be rewritten as:

$$D_e(x) = J + j_n(x) \tag{6.24}$$

where

$$j_n(x) = \sum_{i=1}^{n-1} x^i \tag{6.25}$$

The construction of Singer difference-sets is based on the theorem of Singer based on the hyperplanes of projective geometries.

Suppose $\mathbb{F}$ is any finite field. The space of all vectors $(a_0, ..., a_m)$, $a_i \in \mathbb{F}$ is called the projective geometry of dimension m over $\mathbb{F}$. The zero vector $(0, ..., 0)$ is the void space of dimension -1. A point P, of dimension 0, is the set of vectors

$$(bx_0, ..., bx_m) \tag{6.26}$$

such that $(x_0, ..., x_m) \neq (0, ..., 0)$ and b ranges over the elements of $\mathbb{F}$. A subspace of dimension $(m-1)$ is called an hyperplane. The points in common in two distinct hyperplanes form a subspace of dimension $(m-2)$.

Now let the finite field $\mathbb{F}$ be $GF(q)$, i.e. the Galois field of order q, such that $q = p^r$ and p is a prime. There are $v = (q^{m+1}-1)/(q-1)$ points obtained from the $(q^{m+1}-1)$ points, and

v hyperplanes. Each hyperplane has $J = (q^m - 1)/(q-1)$ different points, and a space S_{m-2} has $\lambda = (q^{m-1} - 1)/(q-1)$ points. We denote the geometry by $PG(m, p^r)$. [57]

Theorem 6.1 (Theorem of Singer)**.** *The hyperplanes of $PG(m, p^r)$, $q = p^r$ as blocks, points as objects, form a symmetric block design with*

$$v = \frac{q^{m+1}-1}{q-1},\ J = \frac{q^m-1}{q-1},\ \lambda = \frac{q^{m-1}-1}{q-1} \tag{6.27}$$

This design is cyclic, and the points in any hyperplane determine a (v, J, λ) difference-set.

Indeed, for DS codes over $GF(3)$, the code length n given by expression (6.19), and the minimum distance in (6.21) are derived respectively from the expression of a special case of the expressions of v and J in 6.27, where $PG(m,3)$ is especially considered.

Suppose that a difference-set D with J points is given, designed from the projective geometry $PG(m,3)$. Let $D(x)$ be the polynomial form of D, in the algebra of polynomials modulo $(x^n - 1)$, when the arithmetic is performed over $GF(3)$. Let $D(x)$ be the difference-set polynomial associated to D. From the code properties, a set ω_j of J orthogonal equations on each symbol position j for $0 \leq j < n$ exists, which belong to the dual code. The parity-check header polynomial $h(x)$ has degree k and is obtained by:

$$h(x) = gcd(D(x), x^n - 1) \tag{6.28}$$

Thus, ω_{j0} is containing the last symbol denoted x_{n-1}. The remaining $(J-1)$ orthogonal equations on x_{n-1} are obtained by shifting ω_0 with respect to the $(J-1)$ non zero exponents of the parity-check header polynomial $h(x)$ [185].

The generator polynomial $g(x)$ of the DS code C is then simply obtained by:

$$g(x) = (x^n - 1)/h(x) \tag{6.29}$$

Example 6.1. Let $s = 1$. The set $D = \{0, 1, 3, 9\}$ is a difference-set of modulo $n = 13$ of order $J = 4$. We propose to design a cyclic difference-set code C over $GF(3)$ with parameters $(n, k, d_{min}) = (13, 6, 5)$. The parity-check polynomial associated to D takes the form:

$$D(x) = 1 + x + x^3 + x^9$$

By calculating its difference enumerator polynomial following (6.23), it is clear that the condition (6.24) is satisfied and thus $D(x)$ is a difference-set. By calculating the greatest common

divisor $h(x)$ of $D(x)$ and x^n-1 in the algebra of polynomials modulo x^n-1 following (6.28), the generator polynomial of the code C has a degree $n-k=7$ and is obtained as follows:

$$g(x)=(x^{13}-1)/h(x)=1+x+2x^4+x^5+2x^6+2x^7$$

The reciprocal polynomial $h^*(x)$ of $h(x)$ is the header of the null space of the code C. Consequently, as the code is cyclic, the null space is given by the n cyclical shifts of the header. A set of $J=4$ orthogonal parity-check sums on every symbol is included in the null space. The parity-check matrix of the ternary DSC code $(13,6,5)$ is given by:

$$H=\left[\begin{array}{cccccccccccc}
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\
\mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0
\end{array}\right]$$

where the rows in bold face are the orthogonal equations on the last symbol.

The set of the indexes of the orthogonal equations of the code $(13,6,5)$ on the last symbol can be written as follows:

$$\omega_{12}=\begin{cases}
x^3+x^9+x^{11}+x^{12} \\
1+x^4+x^{10}+x^{12} \\
x+x^2+x^6+x^{12} \\
x^5+x^7+x^8+x^{12}
\end{cases}$$

Due to the cyclic structure of the code, the set ω_{12} is used to decode all the symbols x^j for $0\leq j<n$ by applying n cyclic shift to the sequence being under decoding. From the set

ω_{n-1}, the code C is capable of detecting 4 error trits, and correcting $t = 2$ errors or less by a simple majority-logic vote on each trit.

Erasure-burst-correction capabilities of cyclic LDPC codes derived from two-dimensional finite geometries (EG and PG) was investigated in [146]. Let σ be the zero-covering-spam of maximum length contained in the regular parity-check matrix H associated to a DS code C (considered also as an LDPC code). Then any erasure-burst of length $\sigma+1$ or less is guaranteed to be recoverable regardless of its starting position. In fact, the erasure-burst capacity l_b of a DS code is lower-bounded by $l_b \geq \sigma+1$. The erasure-burst-correction efficiency of the code C is given by:

$$\eta = \frac{l_b}{n-k} \tag{6.30}$$

When $\eta = 1$, and therefore $l_b = n-k$, then the code is said to be optimal for erasure-burst-correction.

Table 6.3 displays a set of cyclic Difference-Set codes over $GF(3)$, constructed from $PG(m, q = 3)$ based on Theorem 2. The difference-set D is only displayed for small codes due to spatial constraints.

Table 6.2: A set of ternary cyclic difference-set codes

s	(n,k,d_{min})	$r = \frac{k}{n}$	D
1	$(13,6,5)$	0.46	$\{0,1,3,9\}$
2	$(91,54,11)$	0.59	$\{0,1,37,39,51,58,66,69,82,86\}$
3	$(757,540,29)$	0.71	-
4	$(6643,5346,83)$	0.80	-

6.1.4.2 Encoding information data block using the code $(91,54,11)_3$

We propose a double protection to data and header chunks using a set of DS codes presented in Table 6.2. In this work, each byte is mapped to 6 ternary symbols (trits), that will be modulated to a DNA sequence following Table 6.1. Note that the use of ternary symbols avoids automatically the homopolymer runs constraint. To realize this scheme, a mapping table of ASCII symbols and a set of 256 ternary sequences of length 6 must be fixed from the set of possible sequences (729 sequences). Therefore, each datablock (DB) of length $k_1 = 54$ trits corresponds to 9 bytes of data information. Then each DB is encoded using the DS code C_1 with parameters $(n,k,d_{min}) = (91,54,11)$, which adds 37 trits of redundancy for protecting data.

For sake of scalability of the proposed scheme, we introduce a positive integer parameter $\lambda > 0$, which acts as a multiplier for defining the length N of oligos such that the available and current synthesis and sequencing technologies are capable to handle. Thus, the total length of encoded data to modulate into DNA is given by:

$$N = \lambda n \tag{6.31}$$

Thus, in each data block of length n, the code C_1 is capable to detect 10 error trits, and to correct any error pattern of 5 trits or less. Additionally, the code C_1 has an erasure-burst capacity for correcting configurations of burst erasures in the stored nucleotides. We note that the code C_1 can be replaced by a longer DS code if the available technologies are suitable for it.

6.1.4.3 Encoding the index header using the code $(13,6,5)_3$

Information data is lost if the indexing headers are corrupted. Henceforth, a strong protection to headers must be set up. In this work, we propose, for each oligo of length N (given by (6.31)), a prep-ending index of length $n_H = 13$, which corresponds to an encoded sequence of $k_H = 6$ trits using the DS code C_2 over $GF(3)$ with parameters $(n_H, k_H, d_{min_H}) = (13,6,5)$. Thus, with $k_H = 6$ trits, it is possible to index $3^6 = 729$ distinct oligos for each processed file. Note that, similarly to C_1, the code C_2 is also capable of correcting erasure-bursts. The weight enumerator polynomial $A_{C_2}(x)$ of the code C_2 is:

$$A_{C_2}(x) = 1 + 156x^6 + 494x^9 + 78x^{12} \tag{6.32}$$

With the given proposition, the total length l of protected oligos including its indexing headers is given by:

$$l = N + n_H \tag{6.33}$$

Consequently, with the proposed protection scheme, robust forward error correction is ensured for both data and indexing headers, while efficiency and scalability are provided.

The proposed encoding scheme for DNA data storage systems is fully described in Figure 6.9.

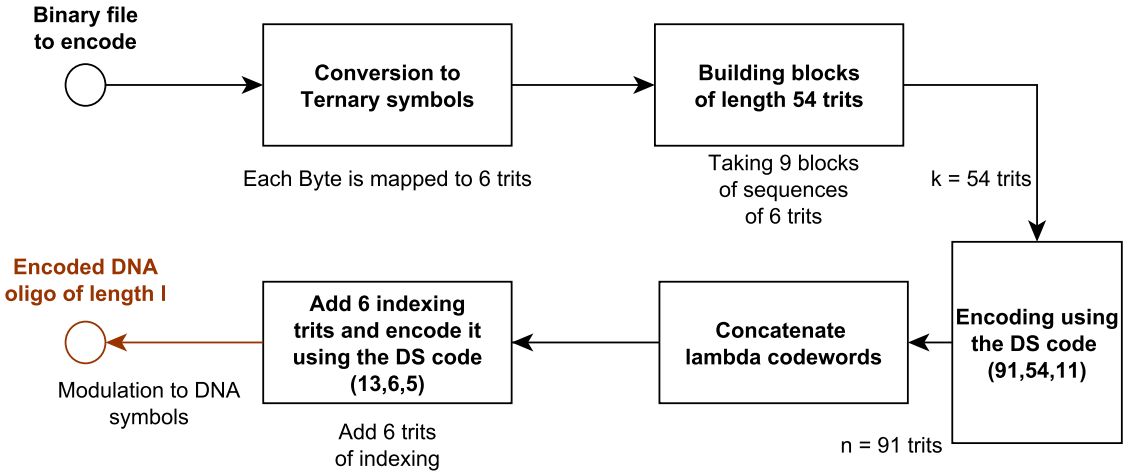

Figure 6.3: The proposed DNA data encoding scheme

6.1.4.4 Decoding DNA data

For decoding sequenced DNA oligos, each DNA chunk is demodulated to base-3, and the header is decoded first using C_2. Syndrome calculation can be performed using a simple division of the encoded sequence by the generator polynomial $g_2(x)$ of the code C_2. If errors are detected (non-zero syndrome), a majority-logic decoding is performed for correcting the header, or it can be decoded using the Maximum Likelihood. Then, decoding of data information is performed by using the majority-logic decoding algorithm (OSMLGD), for decoding each of the λ codewords of C_1 contained in the oligo of length N.

6.1.5 Performance analysis

We propose to evaluate the proposed forward error correcting scheme for DNA data storage. Performance analysis of the proposed scheme is performed based on the coding potential and the net information density. Note that, due to the absence of the experimental materials, the obtained results are based on an analytical study as in [40], which tends to approximate the realistic case.

6.1.5.1 Coding Potential

Suppose that a DNA sequence is valid only if its GC content is within $0.5 \pm c_{gc}$, and its longest homopolymer length is up to m nucleotides.

The coding potential, b, represents the entropy of each nucleotides in valid sequences. The coding potential per nucleotide is defined based on the GC and homopolymer constraints, and is given by:

$$b = 2 - \frac{3\log_2 e}{4^{m+1}} - \frac{\log_2\left[2\Phi(2\sqrt{l}c_{gc}) - 1\right]}{l} - \frac{\log_2 K}{l} \text{ (bit/nt)} \tag{6.34}$$

where l is the oligo length, m is the maximum homopolymer length, K is the index length and $\phi(.)$ is the cumulative function of a standard normal distribution. In this study, a conservative set of constraints has been chosen, as in [40], given by the following parameters: $m = 3$ and $c_{gc} = 0.05$. An oligo length of $l = (3 \times 91) + 13 = 286$ (nts) and $\lambda = 3$ are set to match our proposed scheme.

By the given parameters, the obtained coding potential is $b = 1.97$ bits per nucleotide.

6.1.5.2 Net Information Density

In the proposed scheme, each byte is encoded into 6 nucleotides with a coding potential of $b = 1.97$ bit/nt. A set of λ blocks of nine bytes is encoded into $\lambda n + n_H$ nucleotides. Therefore, the net information density D_n of the proposed scheme is calculated by:

$$D_n = \frac{72\lambda}{\lambda n + n_H} \text{ (bit/nt)} \tag{6.35}$$

In this work, we have $n = 91$, $n_H = 13$ and λ is a parameter to be optimized used for improving the net density, which depends on the current sequencing technologies. When using $\lambda = 2$, a net density of 0.74 bit per nucleotide is achieved. For $\lambda = 3$, the net density is 0.75 bit/nt. Note that the use of longer codes with higher rates can further increase the net information density of the proposed scheme.

6.1.5.3 Comparison with previous works

Table 6.4 depicts a comparison between the proposed scheme and various DNA storage encoding schemes proposed in the literature. The comparison is performed based on a set of parameters, such that a subset of these parameters are defined based on realistic experience. Due to the absence of experimentation for the proposed scheme, the fields associated to these parameters in Table 6.4 are omitted. Table 6.4 includes the length of input data in the experiment, the coding potential, redundancy, robustness of dropouts, error detection and correction existence in the scheme, and the full recovery parameter which indicates if the DNA oligos

were completely recovered in the experiment. Also the net and physical information densities are included, in addition to the realized capacity and the number of oligos used in the experiment.

6.2 The Constructed OSMLD Codes for the 5G NR Mobile Networks

6.2.1 Introduction

After many decades of continuous efforts devoted for the design of reliable mobile networks, future generations of wireless communication systems have received a deep attention from various academic and industrial research communities towards ultra fast, high capacity, low latency and reliable communications over the air interface. Due to the requirements of next generation wireless communications, including interconnection of various technologies and spectrums, achieving ultra high throughput and low latencies, fast mobility, low-cost radio network architecture, efficient waveforms and optimal radio-frequency models, many challenges are actually open for meeting all these requirements. The 5^{th} generation New Radio (5G NR) represents the latest standard for wireless communication systems, initially completed in June 2018 by the 3GPP in the release 15 of the RAN technical specifications [4]. In contrast to previous generations of mobile networks (2G GSM, 3G CDMA, 4G LTE), the 5G NR requires many new specifications for operating over many various frequency bands, and for allowing the interconnection of different technologies. New radio propagation models are supported in the 5G NR, in addition to new waveforms and modulation schemes, and efficient channel coding schemes that support a wide range of coderates and blocklengths.

The timeline of the 5G standardization [2] is depicted in Figure 6.1.

6.2.2 Background on the 5G NR Mobile networks

In contrast to previous generations of mobile networks, the new 5G NR standard requires the support of many upcoming new technologies operating over different spectrum frequency bands, as well as the support of high capacity and ultra high throughput communications with low latencies, in order to allow current and future connected user equipments (UEs) to communicate with a high reliability and low latency. The 5G NR will allow to redefine and connect various industries, including the automotive industry, e-Health, industrial au-

Table 6.3: Comparison of DNA storage encoding schemes

Parameter	Church *et al.* [25]	Goldman *et al.* [51]	Grass *et al.* [53]	Bornhort *et al.* [18]	Blawat *et al.* [16]	Erlich *et al.* [41]	This work
Input data (Mbytes)	0.65	0.75	0.08	0.15	22	2.15	-
Coding potential (bits/nt)	1	1.58	1.78	1.58	1.6	1.98	1.96
Redundancy	1	4	1	1.5	1.13	1.07	1.05
Robustness to dropouts	No	Repetition	RS	Repetition	RS	Fountain	DSC
Error detection and correction	No	Yes	Yes	No	Yes	Yes	Yes
Full recovery	No	No	Yes	No	Yes	Yes	-
Net information density (bits/nt)	0.83	0.33	1.14	0.88	0.92	1.57	0.75
Realized capacity	45%	18%	62%	48%	50%	86%	
Number of oligos	54,898	153,335	4,991	151,000	1,000,000	72,000	-
Physical density (Pbytes/g)	1.28	2.25	25	-	-	214	-

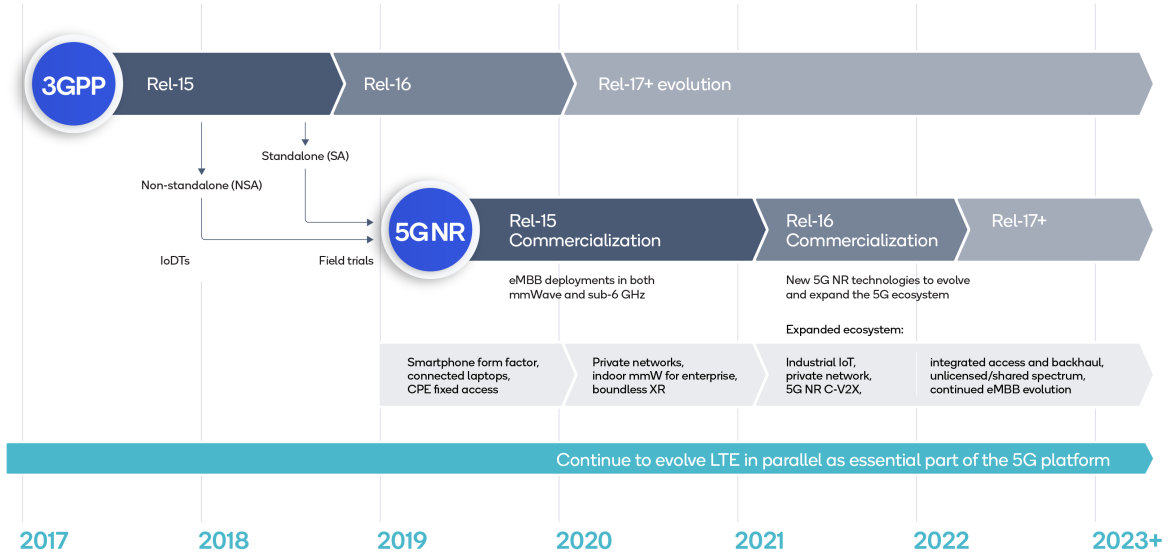

Figure 6.4: 5G standardization timeline

tomation, immersive applications, the Internet of Things (IoT), smart homes, smart cities and smart farming, etc.

Actually, besides academic communities, there are various industries that lead and drive the current evolution of the 5G NR, including Qualcomm, Nokia, Ericsson, Huawei, and many other companies, whose contributions are done in a collaborative business strategy in order to meet the requirements of these challenges and to drive the ecosystem towards next generation mobile communications. Additionally, automotive and smartphone form factors industries are currently including modems that support the 5G NR, as well as network operators that are continuously working on building the required radio access networks and infrastructures. It is known that this generation of mobile communications will change how people connect and communicate, as there are not only mobile phones and computers that will use this technology, but everything is supposed to connect to the 5G networks, leading the entire ecosystem to a massive and fast connectivity, implying the extension of our vision to many new technologies and business models, and creating a lot of new job opportunities. It is known that the 5G NR technology will be the key enabler and driver of the innovation age in the next few years.

Current 5G development focuses on enabling technologies such as flexible baseband and RF technologies, hybrid beamforming, massive MIMO systems, rapid prototyping and field trials, and verification of compliance with the new standard specifications.

The most distinctive features of the PHY/MAC design of the 5G NR are not exhaustively listed below:

- Network densification: In order to allow high capacity connectivity, the current radio architecture must support a high density in both urban and rural areas, which requires many efforts from network operators to meet these requirements. The use of massive Multiple Input Multiple Output (massive MIMO) antenna arrays is a key to achieve this task, including related beamforming issues. Also, network densification enables to overcome the limitations of high frequency communications using milimetric waves, as these waves are unable to travel long distances and are easily attenuated by obstacles.
- Ultra-high throughputs: Achieving ultra-high throughputs is a key requirement of the 5G networks, and this can be achieved by using wider spectrum bands and high bandwidths, in addition to the inclusion of carrier aggregation. High frequencies imply the use of milimetric waves (mm waves), which leads to the investigation of new radio propagation models that support these high spectrum bands, including tapped delay line (TDL) and clustered delay line (CDL) channel models as specified in 3GPP TR 38.901. In the 5G NR standard, there are two frequency ranges that must be supported, namely:
  - FR 1: This frequency band includes frequencies below 6 GHz, that are supposed to support high capacity and long range communications.
  - FR 2: This frequency band supports the mm waves propagations, which is above 24 GHz, used for low latency and ultra-high throughput wireless communications.
- New waveforms: New flexible waveforms that support different technologies and spectrum bands, including the OFDM with a scalable multiplexing numerology [2] for downlinks, and SC-FDMA for uplinks. More advanced multiplexing techniques are still investigated for their eventual use [29, 151, 201].
- High-order modulations: For allowing high throughputs, various and high-order modulations are supported in the 5G NR, including the BPSK, QPSK, 16-QAM, 32-QAM, 64-QAM and 256-QAM modulations.
- Software-defined core networks: SDNs allow the smart management of the switching between different technologies and spectrum bands as well as an efficient mobility management, where the 5G core network is basically inspired from that of 4G LTE but is much smarter and simpler.

- Flexible Time Division Duplex (TDD): In order to allow a flexibility in multiple-access techniques, a different TDD numerology is adopted in the 5G NR that allows the use of various slots durations including mini-slots for allowing short data communications.
- New channel coding schemes: With a wide range of coderates and blocklengths that must be supported, in addition to a high data reliability, the 5G NR employs new channel coding schemes, including polar codes [8] for control channels, and **protograph LDPC codes** for data channels. Note that these coding schemes were only adopted for the eMBB applications of the 5G NR.

Figure 6.2 illustrates the considered frequency spectrum bands in the 5G NR standard.

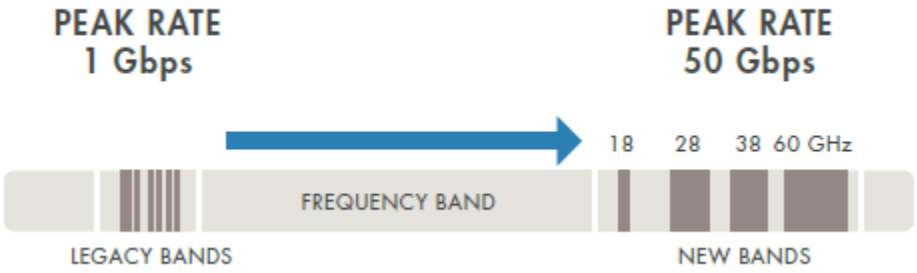

Figure 6.5: 5G spectrum bands

The 5G NR supports many different applications that require different technical specifications. The most expected applications are as follows [65]:

- **eMBB—Enhanced Mobile Broadband:** For high-capacity and ultrafast mobile communications for phones and infrastructure, virtual and augmented reality, mobile gaming, ultra-HD video broadcast and streaming, haptic feedback.
- **URLLC—Ultrareliable and Low Latency:** For vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), cellular vehicle-to-everything (cV2X) communications, autonomous driving, industrial automation, remote surgeries.
- **mMTC—Massive Machine-Type Communications:** For consumer and industrial IoT, Industry 4.0 mission-critical machine-to-machine (MC-M2M) communications.

Thus, by providing higher bandwidth capacity than current 4G–supporting broadband, 5G will enable a higher density of mobile broadband users and support ultra-reliable device-to-

device and massive machine-type communications. The key 5G parameters are displayed in Table 6.1.

Parameter	Value
Latency in the air link	< 1 ms
Latency end-to-end	< 10 ms
Connection density	$100\times$ vs current 4G LTE
Area capacity density	1 (Tbit/s)/km^2
System spectral efficiency	10 (bit/s)/Hz/cell
Peak throughput (downlink) per connection	10 Gbit/s
Energy efficiency	$> 90\%$ improvement over LTE

Table 6.4: Key 5G Parameters

6.2.3 Protograph based LDPC Codes for Data Channels in 5G NR eMBB applications

As mentioned before, two channel coding schemes were adopted in the 5G NR eMBB use cases, namely polar codes for control channels, and multi-edge (protograph) LDPC codes for data channels [136]. The main difference between protograph LDPC codes and other coding schemes used in previous wireless communication systems, is that, protograph LDPC codes are targeted to support a wide range of blocklengths and coderates, which is a key requirement of the 5G systems.

Indeed, protograph LDPC codes [33, 34, 104, 138, 166] have shown to provide near capacity performances over various propagation channels while resulting in diverse code parameters. The main concept behind their construction is to encapsulate the macroscopic structure of the desired LDPC codes in a compressed form, which is the *base graph*, also called *protograph*. The graphical construction of the base graph, which is based on algebra and graph theory,

includes the most relevant obtained results from many decades of research in the topic of the construction of powerful error correcting codes. The construction of efficient base graphs for the 5G NR LDPC codes takes into consideration the following properties:

- Large girths, or equivalently, the absence of short cycles.
- Low-density, given by a small number of 1s in the structure of the parity-check matrices derived from the protograph.
- The irregularity in the parity-check matrix helps to achieve capacity approaching LDPC codes, especially for large blocklengths.
- Low error floors, governed by the existence of trapping sets in the structure of the resulting parity-check matrices.
- The support of puncturing and shortening, in order to allow an efficient use of the incremental redundancy HARQ (IR-HARQ).
- The support of various coderates by allowing the puncturing and shortening of some columns.
- Including degree 1 and 2 variable nodes (VNs) have shown to provide performance improvement.
- The use of a Repeat Accumulate (RA) structure in the base graph (i.e. dual diagonal matrix) allows the simplification of the encoding process.
- The use of a quasi-cyclic structure of the base graph will allow to reduce the decoding implementation complexity, by providing parallelizable layered decoding, as well as to reduce and simplify the storage requirement of the base graph.

In the 5G NR, two base graphs were adopted for meeting all these requirements [136]. The base graph 1 (BG1) is targeted to support large information blocklengths, ranging between $500 \leq k \leq 8448$, and high coderates between $\frac{1}{3} \leq r \leq \frac{8}{9}$, while the base graph 2 (BG2) is targeted to support smaller information blocklengths $40 \leq k \leq 2560$, and lower coderates $\frac{1}{5} \leq r \leq \frac{2}{3}$. Additionally, rate matching, multiplexing and bit selection and block interleaving techniques are designed to be used with these codes, where these procedures are well presented in the standard [4].

The structure of the base graphs standardized for the 5G NR eMBB contains two sets of parity-checks, namely the core check rows and extension check rows. The structure is a concatenation of an LDPC code and a low-density generator matrix (LDGM) code. The structure begins with a relatively high rate "core"; this is the LDPC part. The base graph for the core has a small number of parity checks and some number (e.g. $k_{b_{max}} = 22$) of information variable nodes and m_{core} parity variable nodes. All additional variable nodes are extension degree one variable nodes each connected to a unique check node whose other variable node neighbors are taken from the core; this is the LDGM part. In general, the degree one variable nodes are the extension nodes used for IR-HARQ but the first degree one variable node is special for reasons of performance improvement, and is included in all code rates. For structural reasons, the core portion of the graph does not perform very well without including at least some of the first degree one parity bits [136]. The structure of the base graphs of the 5G NR eMBB [136] is illustrated in Figure 6.3.

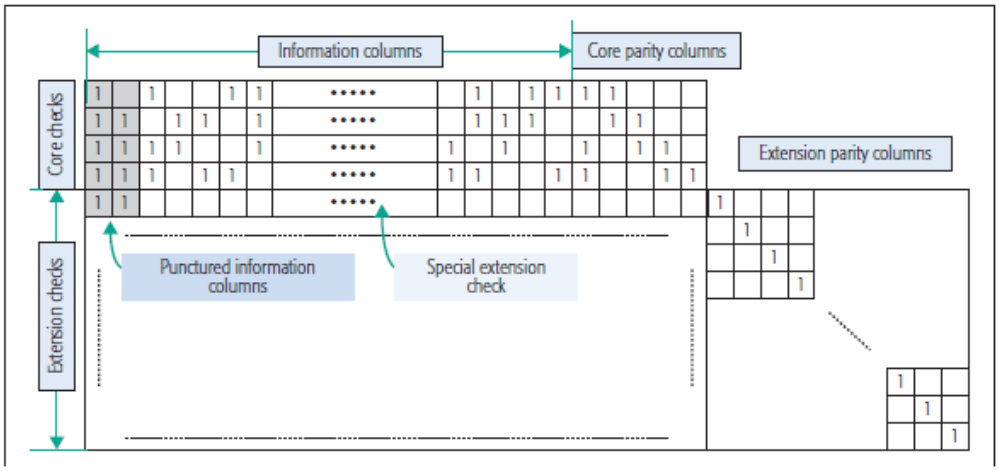

Figure 6.6: Sketch of base parity check structure for the 5G NR LDPC code

The base graph includes integer values typically lower than 384, than can be took modulo Z, where Z is the lifting or expansion factor. Thus, each value of the base graph, after performing the rate-matching, will correspond to a $Z \times Z$ circulant matrix, permuted by the value of the base graph component taken modulo Z. The value of Z is calculated based on the considered codelength n and coderate r for the desired parity-check matrix. There are 8 sets of lifting sizes Z considered in the 5G NR, leading to different constructions of the parity-check matrix [4]. The lifting sets and values considered in the 5G eMBB as well as the complete PR-LDPC encoding procedure is included in [4].

In a typical hardware implementation of a quasi-cyclic LDPC decoder, there are Z processors, where Z corresponds to the maximum lift value. These Z processors perform variable node and check node update operations. More precisely, in each clock cycle, the Z processors are working on one layer (row in the base graph) and processing Z edges in parallel. The decoder performs Z variable node updates and at the same time performs Z check node updates of the next layer. If the edge connectivity of the base graph is such that variable nodes are connected to two consecutive layers, there is potential for a negative impact to decoder performance. Indeed, when the second such layer is processed, the updated variable node operation may not yet be available, so the potential gain from the previous layer processing is not available to and does not benefit the performance of the current layer. This can be circumvented by introducing additional delay in the update process, but that results in a slowing down of the decoder, also potentially degrading performance. This problem can be avoided if the consecutive layers are orthogonal, meaning they have no variable node neighbors in common. Typically, this is not feasible due to the structure provided by the design constraints explained above, however a quasi row-orthogonality is achieved with these standardized base graphs.

In the 5G NR LDPC codes, the encoding process is performed on-fly by using matrix decomposition techniques. An efficient encoding method and a high-throughput low-complexity encoder architecture for quasi-cyclic LDPC codes for the 5G NR standard was proposed in [123]. By storing the quantized value of the permutation information for each submatrix instead of the whole parity check matrix, the required memory storage size was considerably reduced. In addition, sharing techniques were employed to reduce the required hardware complexity.

In [91], an algebra-assisted method for constructing QC-LDPC codes with the properties explained above was proposed. The authors presented a review of the encoding mechanism and requirements of 5G LDPC codes, and presented cycle analysis for such emerging codes. Then they have proposed a metric, referred to as weighted average number of cycles (WANC), from the perspective of cycle structure for constructing the QC-LDPC codes that can support multiple lifting sizes.

6.2.4 The Constructed OSMLD Codes for the URLLC and mMTC use cases

The eMBB 5G NR is different from the URLLC and mMTC use cases, in the way that for the later ones, both reliability and low-complexity are key requirements, while short information

blocklengths must be employed in transmissions. Additionally, memory storage requirements of these scenarios are very limited, especially for the mMTC use case, where the connected devices do not support a large memory storage. As a consequence, the 5G NR LDPC base graphs 1 and 2 are not longer useful for these applications. In fact, new coding schemes must be investigated for meeting these requirements, while good performances should be obtained at the cost of low complexity and memory usage.

Typically, for the URLLC and mMTC applications, data length of dozen or few hundred bits must be supported. However, it is very known that LDPC codes achieve near capacity performance only for long datablocks, and when decoded with the BP-SP algorithm. The BP-SP algorithm requires relatively a large memory storage and computational complexity, especially in the calculation of the CN update. Also, BG1 and BG2 require an on-fly encoding procedure that require an additional latency. Moreover, the irregularity in the parity-check matrices obtained by BG1 and BG2 is only useful for large blocklengths. Thus, the investigation of other candidate channel coding schemes is an interesting current research challenge, as well as devising suitable decoding algorithms that provide an efficient tradeoff between performance and complexity.

In [114], a white paper was published by R.G Maunder, actually founder and CTO of AccelerComm [3], who outlined a vision for 5G, in which channel coding is provided by a flexible turbo code for most use-cases, but which is supported by an inflexible LDPC code for 20 Gbps downlink use-cases, such as fixed wireless broadband. Additionally, recently in 2019, AccelerComm has achieved unprecedented performance in industry, and was successful to reduce the 5G latency by up to 16x with the NR LDPC channel coding and to deliver high-performance throughput thanks to their innovative FPGA implementation solutions. In [37], short-packet data communications were investigated over MIMO fading channels. The same authors have presented in a tutorial of IEEE Globecom 2018, a talk entitled “Short-packet communications: fundamentals and practical coding schemes”. The authors have presented a fundamental study of candidate coding schemes for the URLLC and mMTC 5G NR standards, where a comparison of efficient short channel coding schemes was presented. They have emphasized their study on modern channel codes such as binary and non-binary LDPC codes, tail-biting convolutional codes, turbo-codes and polar codes. They have concluded that polar codes have shown very good performance in the short blocklength regime and also for higher-order modulation. However, for achieving good performance, polar codes require CRC aided successive cancellation list (CRC-SCL) decoding algorithm, which requires a large computational complexity, where most connected devices operating in the URLLC and

mMTC use cases could not support those required resources. Later, an evaluation of LDPC and polar coding schemes for the mMTC 5G terminology was presented in [130].

It is known that mMTC is about wireless connectivity to tens of billions of machine-type terminals, and about availability, low latency, and high reliability. The main challenge in mMTC is scalable and efficient connectivity for a massive number of devices sending very short packets, which is not done adequately in cellular systems designed for human-type communications. Some Physical and MAC layer solutions were proposed in [17] for the mMTC applications.

In [161], performance comparison between different channel coding schemes was presented for the use in URLLC applications in the 5G NR. It was shown that it is not possible to find one coding scheme that outperforms all others for all considered block sizes and coding rates. For short blocklengths, it was shown that LDPC and polar codes outperform turbo-codes. However, the opposite was true for medium blocklengths. Other aspects play a key role in the selection of suitable coding schemes, as implementation complexity, latency and flexibility. Due to the implementation issues of list decoding and to the sequential nature of the SC decoding algorithm, it was stated that the use of polar codes is uncertain at this stage. Independently in [154], an overall study and comparison of short blocklength channel coding schemes for the URLLC 5G NR applications was presented. The authors have established a comparison between binary and non-binary LDPC codes, convolutional codes, turbo-codes, polar codes and BCH codes, and they have identified that BCH codes provide the best error rate performances when decoded with of the Ordered-Statistic Decoding (OSD) algorithm. This result is straightforward as BCH codes are characterized by large minimum distances compared to other codes, for the same codelengths and coderates. However, the OSD algorithm requires a huge computational complexity which is not adequate for the URLLC applications.

Intuitively, and based on all these studies and investigations in the literature, we emphasize the benefits of the use of OSMLD codes as a promising candidate for the URLLC and mMTC 5G applications. The motivation behind our proposition is that, as shown in previous chapters, OSMLD codes have the property of simple encoding and decoding algorithms, with small storage requirements and low latency. It is also shown in the literature that for short blocklengths, the introduction of irregularities in the parity-check matrix is no longer useful, oppositely to long codes. Therefore, the use of regular structured OSMLD codes, along with the proposed quantized gradient-descent majority-logic decoder (QGD-MLGD) in Chapter 4, is an interesting proposition with great potential for these applications. However, and

similarly to other coding schemes outlined by the previous studies above, the flexibility in coderates and blocklengths is an important issue, in contrast to protograph LDPC codes. Besides that, it is not sure whether URLLC and mMTC should benefit from this flexibility, as it is possible that for many applications, only a set of dozen OSMLD codes should be stored in the memory for meeting the requirements of the application.

As a consequence, we are currently working on the graphical construction of a new base graph (BG3), suitable for short blocklength transmissions, and capable to generate a set of OSMLD codes, similarly to BG1 and BG2. However, in our proposition, we will focus our interest on short information blocklengths, and regular structure of parity-check matrices, as well as a quasi-cyclic, or possibly, a cyclic structure of the generated parity-check matrices of the designed BG3. The aim of this proposition is the construction of a new base graph, with a minimum degree of irregularity (for maintaining good performance at short blocklenghs), and with emphasizing on the orthogonality property that should be present for all the set of symbols. Also, we will maintain the compatibility of our base graph BG3 with the standardized lifting sizes already used for BG1 and BG2 in eMBB applications, and we will also include the punctured columns and rows for compatibility with various coderates and with the IR-HARQ.

Our proposition will be realized by the use of constrained optimization techniques, where an objective function is devised in order to minimize it. The objective function to minimize will consist of the bit error rate of the generated OSMLD codes with a given set of parameters, namely a codelength n and coderate r, as well as a lifting size index. We will eventually extend the minimization to other sets of parameters, iteratively, and possibly introduce a multi-objective function that also try to minimize other parameters, as the weight of the overall base graph. The constrained minimization will take into consideration a set of constraints, especially devised based on the cycles constraints, as well as the existence of punctured columns and extension rows. This proposition is actually under investigation, and the mathematical modeling of this construction problem is currently under study. The main goal is to propose a new base graph that competes the existing base graphs (BG1 and BG2) for short information blocklengths, and giving better performance than polar codes and BCH codes that are actually the major candidates for the URLLC and mMTC 5G NR applications.

For illustrating the benefits of cyclic OSMLD codes and their robustness compared to the standardized LDPC codes in the 5G NR eMBB, Figure 6.4 depicts the simulation results of a comparison at a coderate of $r \approx 0.61$ of the BLER performance of the cyclic OSMLD code with parameters $(n,k) = (73,45)$ decoded with the proposed QGD-MLGD algorithm, versus

the 5G NR LDPC code with parameters $(74,45)$, generated with the standardized BG2 and decoded with the BP-SP algorithm, for a transmission over an AWGN channel with a QPSK modulation. For the simulated OSMLD code, the parameters of the QGD-MLGD algorithm are $\alpha = 0.4$ and $\theta = 7$, with an uniform quantization function with $b = 7$ quantization bits.

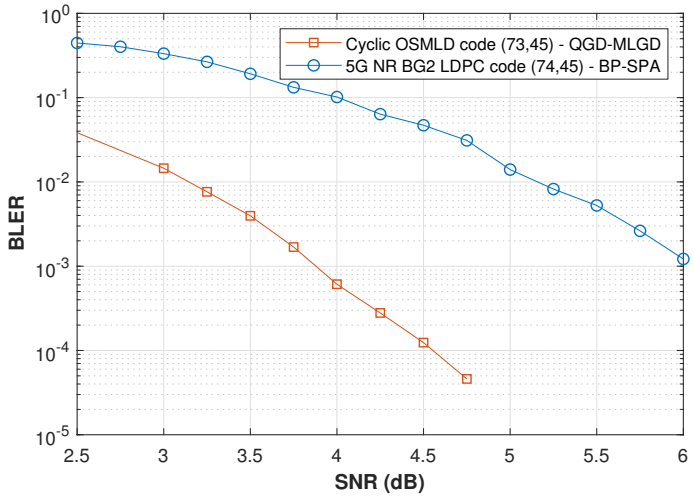

Figure 6.7: BLER performance of the cyclic OSMLD code with parameters $(73,45)$ decoded with the QGD-MLGD algorithm versus the 3GPP 5G NR LDPC code with parameters $(74,45)$ generated with BG2 and decoded with the BP-SPA, for a coderate $r \approx 0.61$

Its it shown in Figure 6.4 that the cyclic OSMLD code $(73,45)$ outperforms significantly its counterpart 5G NR LDPC code $(74,45)$ generated with BG2, such that a lower decoding complexity is employed using the QGD-MLGD algorithm, versus the BP-SP algorithm. At a BLER of 10^{-3}, our scheme achieves a gain up to 2.1 dB. This comparison shows that cyclic OSMLD codes are better than the current standardized LDPC codes in the 5G NR, especially for coderates above 0.5. The regular structure of the OSMLD code is a key advantage, in addition to its higher minimum distance, and its structure which requires lower encoding and decoding complexities.

Now for a coderate $r \approx 0.64$, we depict in Figure 6.5 a BLER comparison between the cyclic OSMLD code with parameters $(357,227)$ and decoded with the QGD-MLGD algorithm, versus the 3GPP 5G NR LDPC code $(356,227)$ generated with BG2 and decoded with the BP-SPA. The same quantization levels and transmission parameters are considered as in the previous comparison, and the parameters of the QGD-MLGD algorithm are $\alpha = 0.2$ and $\theta = 8$ with $b = 7$ quantization bits.

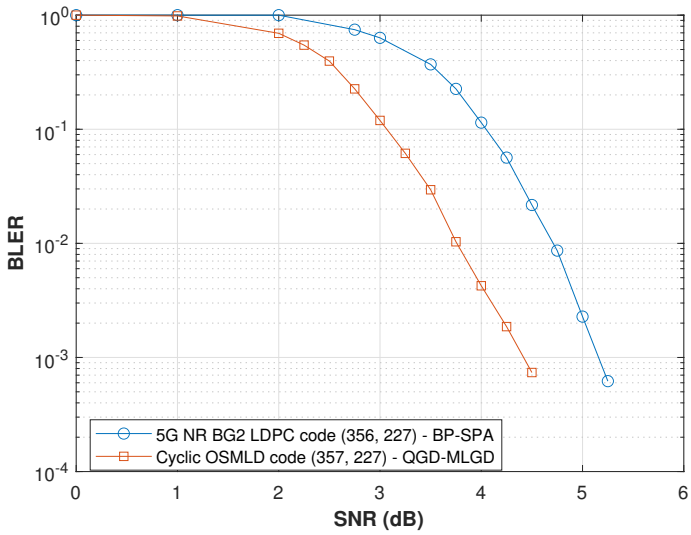

Figure 6.8: BLER performance of the cyclic OSMLD code with parameters $(357,227)$ decoded with the QGD-MLGD algorithm versus the 3GPP 5G NR LDPC code with parameters $(356,227)$ generated with BG2 and decoded with the BP-SPA, for a coderate $r \approx 0.64$

Figure 6.5 shows that the channel coding scheme based on Cyclic OSMLD codes decoded with the QGD-MLGD algorithms provides a significant performance improvement over the 3GPP 5G NR quasi-cyclic LDPC code generated with BG2. A significant coding gain of up to 0.75 dB is achieved by our scheme at a BLER of 10^{-3}. Note also that the waterfall region begins earlier than the 5G LDPC code, where the later suffers from a extended erroneous region until 3 dB, compared to less than 2 dB for our proposed scheme. The minimum channel SNR that supports reliable iterative decoding of asymptotically large codes is called *decoding threshold*. Similarly to the previous figure, this considerable performance improvement is obtained at a cost of lower encoding and decoding complexities, by employing the proposed QGD-MLGD algorithm, which requires lower computational complexity than the BP-SP decoding algorithm.

For a higher coderate $r \approx 0.70$, Figure 6.6 presents the BLER comparison of the cyclic OSMLD code $(803,559)$ decoded with the QGD-MLGD algorithm, with parameters $\alpha = 0.1$ and $\theta = 0$, compared with the 3GPP 5G NR LDPC code $(804,559)$ generated with BG2 and decoded with the BP-SP algorithm.

As shown in Figure 6.6, the proposed OSMLD code $(803,559,29)$ decoded with the QGD-MLGD algorithm provides a significant improvement over the 3GPP LDPC code generated

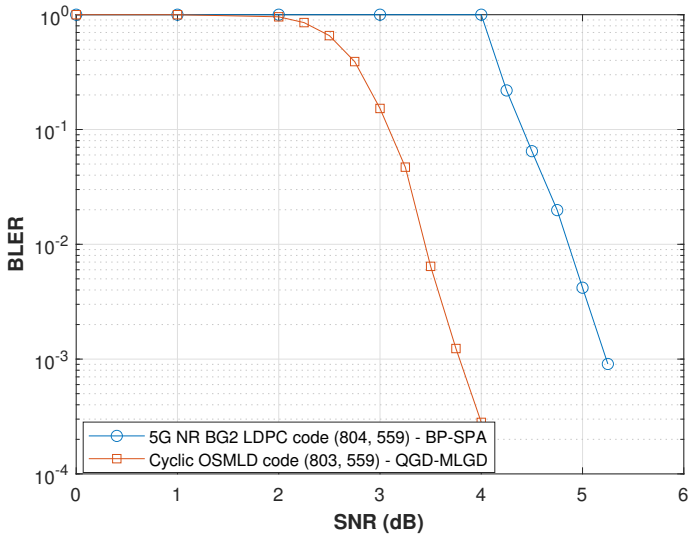

Figure 6.9: BLER performance of the cyclic OSMLD code with parameters $(803, 559)$ decoded with the QGD-MLGD algorithm versus the 3GPP 5G NR LDPC code with parameters $(804, 559)$ generated with BG2 and decoded with the BP-SPA, for a coderate $r \approx 0.70$

with BG2 with parameters $(804, 559)$ and decoded with the BP-SP algorithm. Clearly, a significant coding gain of up to 1.45 dB is obtained at a BLER of 10^{-3}. The iterative decoding threshold of our proposed scheme is observed approximately at an SNR of 2.25 dB versus 4.25 dB in the 3GPP 5G NR quasi-cyclic LDPC scheme.

Now for a coderate $r \approx 0.83$, Figure 6.7 illustrates the BLER comparison of the cyclic OSMLD code $(3471, 2873)$ decoded with the QGD-MLGD algorithm, with parameters $\alpha = 0.05$ and $\theta = 3$, compared with the 3GPP 5G NR LDPC code $(3472, 2873)$ generated with BG1 and decoded with the BP-SP algorithm.

Figure 6.7 shows that the constructed cyclic OSMLD code with parameters $(3471, 2873)$ outperforms the 3GPP LDPC code generated with BG1 with parameter $(3472, 2873)$, where the former is decoded with the proposed QGD-MLGD algorithm, while the later is decoded with the BP-SPA. An approximate coding gain of about 0.86 dB is obtained at a BLER of 10^{-2}, and approximately the same coding gain is observed at a BLER of 10^{-3}. The proposed scheme shows an iterative decoding threshold at 3.5 dB, while in the 5G 3GPP scheme the threshold is observed at 4.5 dB, providing a gain of 1 dB.

The presented performance improvements of OSMLD codes when decoded with the low-complexity QGD-MLGD algorithm compared to the 5G NR LDPC codes emphasize the

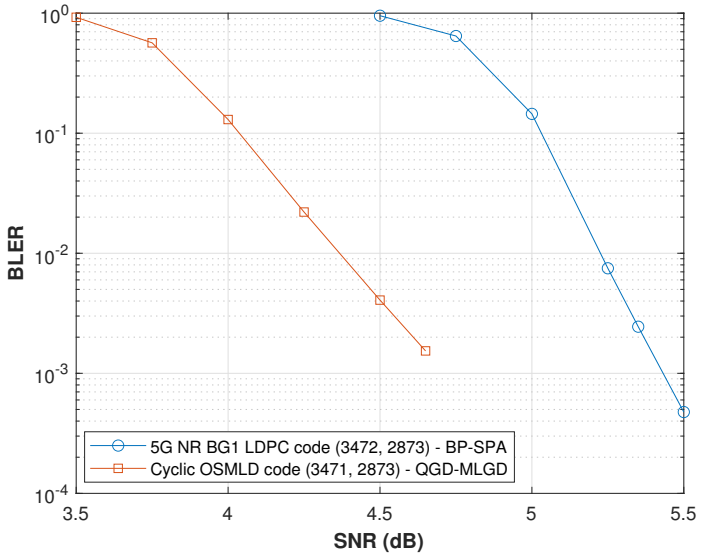

Figure 6.10: BLER performance of the cyclic OSMLD code with parameters $(3471, 2873)$ decoded with the QGD-MLGD algorithm versus the 3GPP 5G NR LDPC code with parameters $(3472, 2873)$ generated with BG1 and decoded with the BP-SPA, for a coderate $r \approx 0.83$

benefits of the construction of new base graphs for the 5G NR that include the characteristics of structured OSMLD codes, and show that these later codes are the most promising candidates for high coderate channel coding schemes in the next generation wireless communication systems. This motivate also our current project to include our new constructed base graphs in the URLLC and mMTC use cases in the 5G NR, to meet all the standard requirements of these communication systems.

6.3 Conclusion

In the first part of this chapter, we have emphasized that DNA data storage systems represent the most suitable candidates for high reliability and long-term (thousands years) storage of massive governmental, research and industrial archival data in the next few years. We have presented an overview of the DNA channels and modulation schemes, as well as a literature review on the forward error correcting schemes already proposed for DNA data storage systems. We have proposed a robust cost-effective DNA encoding scheme with a net information density of 0.75 bit per nucleotide, under realistic parameters, allowing to detect and correct errors in DNA oligos, using cyclic ternary difference-set codes. The proposed system aims to ensure the protection of both information and indexing headers from symbol-flipping and

erasure errors by the use of two different DS codes, with less redundancy compared to many previous works, and with a simple majority-logic decoding process providing a significant reduced data storage and decoding cost, and a high DNA information density. As shown in this chapter, the proposed scheme is scalable and may be improved by different ways, based on the optimization of the parameters involved in the designed system. With the current evolution of synthesizing and sequencing technologies, longer codes with higher correction capacities can be considered in the proposed scheme for future DNA data storage mediums.

In the second part of this chapter, we have presented a brief overview of the features and current evolution of the 5G NR standard, as well an outline on the standardized protograph LDPC codes in the eMBB 5G use cases. We have established a state of art on the current evolution of the propositions related to candidate channel coding schemes for the URLLC and mMTC applications, where comparison between different schemes and the relevant results found in the previous works were presented comprehensively. We finally described our current contribution in progress, which consists of the construction of a new base graph for generating regular OSMLD codes for short blocklength transmissions suitable for the use in the URLLC and mMTC 5G NR applications, and adequate to be decoded using the proposed QGD-MLGD algorithm. This construction is essentially based on constrained optimization techniques as well as various combinatorial design objects, that take into consideration a set of constraints imposed on the BG3 in order to meet the requirements of these next generation applications. The motivations behind the use of OSMLD codes for the URLLC and mMTC applications were also given in this part, based on the established literature review of the current results in the state of art, as well as some performance comparisons between OSMLD codes and the current 3GPP standardized 5G NR LDPC codes, which have shown that cyclic OSMLD codes outperform significantly the 5G NR quasi-cyclic LDPC codes standardized by the 3GPP for the eMBB use cases, at the cost of lower decoding and encoding computational complexities, and lower latency.

Chapter 7

Conclusions and future directions

7.1 Concluding remarks

This thesis has focused on majority-logic decodable codes, their construction based on finite geometries and combinatorial designs, their iterative decoding as well as some of their applications in wireless communication systems and DNA data storage technologies. The major benefits on considering structured majority-logic decodable codes with mathematical constructions are their guaranteed code properties, simplicity in encoding and decoding requiring low-complexity, and low storage requirements, where these features are very beneficial for modern communication, data storage and security systems. These properties are not easily achieved using random constructions, and require many difficult constraints to take into considerations when designing these codes.

We have presented in the first part of this thesis a classification of majority-logic decodable codes, the cyclic, quasi-cyclic and non cyclic ones, as well as those that require only one-step orthogonalization process and two orthogonalization steps, while the generalization to multi-step is straightforward. The background used for this classification includes finite geometries and balanced incomplete block designs, as well as other designs including various types of difference-sets and difference-families. This classification was useful for a complete understanding of different mathematical constructions of majority-logic decodable codes for an eventual future exploration of other advanced combinatorial and geometrical design techniques.

In the second part of this thesis, and based on the first part, we have proposed to investigate the construction of new cyclic OSMLD codes based on algebraic background related to cyclo-

tomic cosets and the parity-check idempotents. A literature review of the existing exhaustive search constructions was presented, and the construction problematic was detailed for further simplification and for more flexibility. A mathematical modeling of the construction problem as a fitness function was introduced for exploiting global optimization techniques for solving the problem, and for reducing the search space dimension. Then a flexible construction based on genetic algorithms was proposed, consisting of minimizing the devised objective function that represents the construction problem. This proposition had led us to a large database of new cyclic binary and non-binary OSMLD codes, never presented in the literature before. The main advantage of our new design algorithm is a diversity in code lengths and coderates, ranging from short, moderate to very long code lengths.

The third part of this thesis was dedicated to the iterative decoding of OSMLD codes, where an extensive literature review of the existing decoding algorithms was presented, along with an overall comparison between previous decoding techniques in terms of computational complexity, average number of decoding iterations and error rates. We have reviewed gradient-descent decoding of error correcting codes, and we have devised a new objective function that represents the decoding problem of OSMLD codes. We have proposed a gradient-descent majority-logic decoder (GD-MLGD) along with its quantized version (QGD-MLGD), and we have shown by simulations that proposed schemes outperform significantly the state of art, at the cost of a reasonable complexity, emphasizing the benefits of the proposed decoding schemes for energy-constrained and low-complexity communication systems, that require both high performance and low-complexity. We have also presented an analysis of erroneous decoding decisions of the proposed algorithms, along with a statistical study where false converged codewords are of interest. We have concluded with some interesting perspective towards exploiting artificial neural networks for devising a classifier of false converged codewords, by suitably studying the dynamic and evolution of the syndrome weight during decoding iterations, where these distributions are considered as stochastic processes.

The contribution is Chapter 4 had led us to a new general interpretation of all the existing majority-logic decoding algorithms as a gradient-descent maximization of a particular objective function, here we have devised for each algorithm an appropriate objective function that coincides exactly with the reliability update in the original proposition. Furthermore, we have devised new update rules derived from other variants of the gradient-descent. Also, a more general decoding framework for linear block codes was proposed, based on a constrained optimization mathematical model, where the Augmented Lagrange Method of Multiplier algorithm was employed for addressing the decoding problem. This contribution results in a

new direction toward a universal improvement approach of the existing decoding algorithms, based on the mathematical model of the problem.

Another contribution that was not presented in this report is a new iterative decoding of TSMLD codes derived from Euclidean geometries. A literature review on the existing previous works and an overall comparison of these decoding algorithms was established. We have proposed a new decoding algorithm, namely the two-step iterative threshold decoding algorithm (TS-ITD) with is quantized version, which aims to optimize the trade-off between performance and complexity for decoding TSMLD codes. The proposed algorithm exploits the two-step orthogonality structure of TSMLD codes, and overcomes the performance degradation provided by the belief-propagation sum-product decoding algorithm, due to the large number of short cycles in the dual structure of these codes. Simulation results have shown that the proposed algorithms outperform all the previous decoding approaches, at the cost of a relatively low-computational complexity. This contribution was not presented in this report as it is not yet published, and we are working on more improvements of this proposition by proposing a custom deep neural network to be hybridized with the proposed decoder.

The last part of this thesis included two applications of OSMLD codes, the first one in some applications of the 5G NR, and the second one in DNA digital data storage systems. In the 5G NR, especially in the URLLC and mMTC use cases where short information block lengths and low-computational complexity are required, we have emphasized the benefits of cyclic OSMLD codes in meeting the requirements of these standards. We have presented a brief literature review on previous forward error correction schemes, considered as candidates for their eventual use in these standards, along with a comparison of the performance of these schemes with those provided by cyclic OSMLD codes. We have shown that for high coderates, cyclic OSMLD codes are more advantageous and provide better error rate performances with lower encoding and decoding complexity, in addition to lower storage requirements. These results show that cyclic OSMLD codes are appropriate for energy-constrained communication systems in 5G NR mMTC applications, as well as for the 5G NR URLLC where both high performance and low-complexity are of interest. OSMLD codes can also be key candidates for other applications such as IEEE 802.15.6 standards for body area networks, for UAVs and drones, defense systems, satellite and underwater communications.

The second application includes the application of cyclic ternary difference-set codes in DNA data storage systems. This technology represents a promising candidate for next generation massive data storage, achieving long-term storage and maximum security, without requiring high-cost and regular maintenance. We have outlined some preliminaries about this data stor-

age technologies, by establishing a comprehensive literature review on the evolution of this topic, along with the introduction of DNA channels and modulations in an information theory point of view. We have also provided the historical evolution of forward error correction schemes already proposed for DNA data storage, with the obtained performance for each system. Then we have proposed a new forward error correction scheme aiming to protect DNA oligos against symbol flipping and burst-erasure errors, where both information and indexing headers data are protected. The new proposed scheme has an advantage of low-complexity encoding and decoding, making synthesizing and sequencing encoded symbols easier and requiring lower costs compared to previous schemes. The proposed system is scalable with regards to the current synthesizing and sequencing technologies, and the presented comparison with previous works has shown that the obtained performances are competitive with the current state of art.

7.2 Future research directions

The classification of MLGD codes led us to consider the decoding of many known algebraic error correcting codes using majority-logic decoding algorithms, instead of other decoding algorithms that require more computational complexity. For instance, we have figured out that Quadratic Residues (QR) codes can be considered cyclic MSMLD codes, where their construction is derived from Quadratic residues difference-sets. Other residues codes can be derived from biquadratic residues, biquadratic residues with zero, octic residues and octic residues with zero difference-sets. One interesting future direction is the exploration of further combinatorial designs for constructing various new OSMLD and MSMLD codes, and considering many known families of codes as MLGD codes. Further combinatorial designs can be explored from the books [27,57], where many advanced designs were presented. Such combinatorial designs include Bhaskar Rao designs, self-orthogonal Latin Squares, Pairwise Balanced designs, including group divisible designs, balanced tournament designs, block-transitive designs, frequency designs and hypercubes, nested designs and quasi-symmetric designs...

Another interesting direction is the consideration of these designs over other composed finite groups, yielding to multi-dimensional MLGD codes, where the indeterminate is not only x, but many indeterminates are considered. Moreover, the exploration of the construction of OSMLD codes over non-Abelian groups is another important future direction. One interesting mathematical open problem in coding theory related to this topic is to find if there is

a duality and MacWilliams formula for codes over non-Abelian groups, and to discover a subclass of non-Abelian groups for which a duality and a MacWilliams formula exist.

In the construction of OSMLD codes, an important future direction includes the exploration of random construction techniques using constrained optimization techniques, where the construction problem of OSMLD codes can be formulated using an objective function, in addition to a set of constraints that represent the mathematical conditions for obtaining powerful OSMLD codes randomly, where the irregularity can be introduced, similarly to the PEG and ACE construction algorithms. This direction includes the introduction of the flexibility and the support of IR-HARQ retransmission techniques, by appropriately designing rate-compatible base graphs that incorporate the properties of OSMLD codes, rather that one parity-check matrix per code.

In the topic of decoding OSMLD codes, future research directions include the investigation of the finite precision of quantization functions, where other quantization functions can be explored for further performance improvement of the proposed QGD-MLGD algorithm. Additionally, the analysis of false converged codewords can be continued towards devising a neural networks based technique for the classification of false decoding decisions during iterations. The statistical analysis of these scenarios can pursued in order to achieve an optimal strategy for classifying these use cases, aiming to improve the performance of the proposed MLGD algorithms. As shown in Chapter 4, the proposed algorithms include extra parameters, namely the step-descent α, the overscaling factor β and the offset factor θ, where we have considered these parameters to be fixed after numerical optimization. An interesting future direction is to introduce dynamism in these parameters with respect to the decoding iterations as well as the channel conditions. One such possible technique is the use of artificial deep neural networks for training the proposed decoders for classifying the appropriate values of these parameters depending on the dynamic of the decoder as well as the channel conditions. This perspective includes the generalization to other transmission channels, namely fading channels with the combination with high-order modulation techniques, including channels for applications to wireless body area networks (802.15.6 standard) and taped delay line channels for 5G networks, that take into consideration the multipath propagation effect of the signals. Furthermore, the design of other quantization functions using deep learning techniques is an interesting direction that might lead to significant performance improvement of the QGD-MLGD algorithm, and to dynamically learn the quantization function with respect to various channels and codes. Another important future direction is to exploit other local optimization techniques as well as other variants of the Gradient-Descent

technique in order to improve the performance of the GD-MLGD algorithm.

As shown in Chapter 5, we have proposed an universal view of all majority-logic decoding algorithms as a gradient-descent maximization of an objective function. Therefore, it will be interesting to investigate how to devise an optimal objective function that represents the decoding problem of OSMLD codes. This improvement needs only mathematical investigations and knowledge instead of algorithmic considerations, and the computational complexity depends only on the first-order derivatives, which is an important feature of the proposed models. Another interesting direction is to investigate other local optimization techniques in order to solve those mathematical optimization problems.

Another interesting future direction is the application of the proposed decoding schemes to decode protograph LDPC codes already adopted in the 5G NR eMBB use cases, that was standardized by Qualcomm, presented in the paper of Tom Richardson [136] which details their performance as well as their design, such that this work was supported and standardized by Qualcomm Incorporated [2]. The main aim is to emphasize the benefits of the proposed approach in terms of satisfying robust performance at the cost of low computational complexity compared to the BP-SP algorithm.

Perspectives of our contribution that was not presented in this report, concerning the iterative decoding of TSMLD codes, include the generalization of the proposed algorithms TS-ITD and TS-QITD to MSMLD codes, as well as the performance evaluation of decoding other TSMLD codes derived from different construction techniques. Similarly to Chapter 4, a future direction is the optimization of the extra parameters, including the overscaling factor α, the attenuation factor λ and the offset parameter θ to different transmission scenarios. This can be achieved by introducing dynamism in these variables with respect to the channel conditions and the dynamic of the decoder during iterations. The use of artificial intelligence is a key challenge towards optimizing these parameters by suitably devising a classifier based on a large data-set of decoding use cases. The use of deep neural networks is a key solution to prevent the manual tuning of the parameters of the algorithm, by suitably learn the optimal values of these parameters for various SNRs. Moreover, a future direction is to emphasize the benefit of cyclic TSMLD codes along with the proposed decoding algorithms compared to BCH codes, currently used in various wireless communication standards. This is somehow connected to the fact that we have decoded the BCH code with parameters $(n,k) = (63,45)$ using the TS-(Q)ITD algorithms, and according to our knowledge, we have obtained the best performance ever presented in the literature for this code, except when it is decoded with the Ordered Statistic Decoder (OSD) which requires a significant complexity. Thus, decoding

BCH codes using multi-step majority-logic decoders is a key direction for achieving near optimal performance for this interesting class of codes.

Finally, future research directions on applications of OSMLD codes include the analysis of an overall comparison between various coding schemes compared to cyclic OSMLD codes, for the eventual use in future generation energy-constrained 5G NR mMTC data transmissions, as well as for the URLLC use cases. Due to the major benefits of regular OSMLD codes especially for short blocklengths, a future perspective consists of the construction of new base graphs of LDPC codes (PR-LDPC) suitable for short length data communications, by using constrained optimization and machine learning techniques, and possibly by the use of other combinatorial block designs. For DNA data storage systems, current and future investigations include the proposition of a framework for the simulation of the performances of DNA data storage schemes, including forward error correction, DNA modulation and DNA channels with the imposed constraints. Future directions include additionally the proposition of other coding schemes, for instance the design of new constrained error correcting codes that take into account the homopolymer and the GC content constraints, that align with the current evolution of sequencing and synthesizing technologies provided by Illumina, which is a leading company specialized in developing sequencing and array-based solutions for genetic research [1].

Appendix A

Difference-families and Skolem sequences

The following definitions of difference-families and Skolem sequences are taken from [27].

Definition A.1 (Difference-Families)**.** Let G be a group of order v. A collection $\{B_1, B_2, ..., B_t\}$ of k-subsets of G form a (v, b, r, k, λ) **difference family** if every non-identity element of G occurs λ times in $\Delta B_1 \cup ... \cup \Delta B_t$. The sets B_i are base blocks. A difference family having at least one short block is **partial**.

Example A.1. $B_1 = \{0, 1, 3, 5, 10, 11\}$, $B_2 = \{0, 6, 1, 7, 3, 9\}$, and $B_3 = \{0, 4, 8, 1, 5, 9\}$ form a $(12, 6, 5)$ difference family over $\mathbb{Z}_{12}$. The stabilizers of B_1, B_2, B_3 are $\{0\}$, $\{0, 6\}$, and $\{0, 4, 8\}$, respectively [27].

Definition A.2 (Skolem sequences)**.** A Skolem sequence of order n is a sequence $S = (s_1, s_2, ..., s_{2n})$ of $2n$ integers satisfying the conditions

1. for every $k \in \{1, 2, ..., n\}$ there exist exactly two elements $s_i, s_j \in S$ such that $s_i = s_j = k$, and
2. if $s_i = s_j = k$ with $i < j$, then $j - i = k$.

Skolem sequences are also written as collections of ordered pairs $\{(a_i, b_i) : 1 \leq i \leq n, b_i - a_i = i\}$ with $\bigcup_{i=1}^{n} \{a_i, b_i\} = \{1, 2, ..., 2n\}$.

Example A.2. A Skolem sequence of order 5 is defined by $S = (1, 1, 3, 4, 5, 3, 2, 4, 2, 5)$ or, equivalently, the collection $\{(1, 2), (7, 9), (3, 6), (4, 8), (5, 10)\}$.

Definition A.3 (Extended Skolem sequence)**.** An extended Skolem sequence of order n is a sequence $ES = (s_1, s_2, ...s_{2n+1})$ of $2n+1$ integers satisfying conditions 1 and 2 of the definition A.44 and

3. there is exactly one $s_i \in ES$ such that $s_i = 0$. The element $s_i = 0$ is the hook or zero of the sequence.

Definition A.4 (Hooked Skolem sequence)**.** A hooked Skolem sequence HS of order n is an extended Skolem sequence of order n with $s_{2n} = 0$.

Example A.3. A hooked Skolem sequence of order 6 that is also an extended Skolem sequence of order 6 is given by: $ES = (1,1,2,5,2,4,6,3,5,4,3,0,6)$.

Appendix B

A set of Cyclic OSMLD codes constructed using the OSMLD-GA algorithm

The tables below list a set of cyclic binary and non-binary OSMLD codes constructed with the OSMLD-GA algorithm. Although for each codelength n, many equivalent and non equivalent codes were constructed with various dimensions and minimum distances, we restrict our illustration to only one code with the higher minimum distance for each codelength n, so that we list a set of the constructed OSMLD codes in an increasing order of minimum distances with respect to the codelengths, and with coderates $r > 0.5$. For a fixed code length, we only present the code with higher minimum distance besides that there are other codes with higher coderates. Sometimes the minimum distance is not following the increasing order, just to represent some interesting codes with relatively high coderate, however we mainly focus on a representation following an increasing order of d_{min}. Note that a complete list of thousands of constructed cyclic OSMLD codes will be presented soon in an online database of our research team, along with their corresponding generator polynomials and orthogonal parity-check equations. Note that the set $\Omega_{E(x)}$ is sufficient to characterize an OSMLD (LDPC) code, along with its parity-check matrix and its generator polynomial.

Table B.1: A set of Cyclic binary OSMLD codes with code-lengths $7 \leq n \leq 1057$ constructed using the OSMLD-GA algorithm

n	k	d_{min}	J	$r = \frac{k}{n}$	$\Omega_{E(x)}$
7	3	4	3	0.43	$\{1\}$
15	7	5	4	0.47	$\{7\}$
21	11	6	5	0.52	$\{3,7\}$
31	15	6	5	0.48	$\{15\}$
63	37	9	8	0.59	$\{13,21\}$
73	45	10	9	0.62	$\{25\}$
93	61	11	10	0.66	$\{23\}$
105	55	11	10	0.52	$\{7,15,45\}$
217	153	16	15	0.70	$\{37\}$
255	175	17	16	0.69	$\{3,13\}$
273	191	18	17	0.70	$\{67,91,117\}$
315	213	19	18	0.68	$\{3,35\}$
345	173	20	19	0.50	$\{15,23,161\}$
357	227	20	19	0.64	$\{51,85,119,133\}$
465	337	21	20	0.72	$\{97\}$
	285	22	21	0.61	$\{45,105,155,217,225\}$
511	303	22	21	0.59	$\{21,27,219\}$
527	273	24	23	0.52	$\{17,31,119,255\}$
585	393	25	24	0.67	$\{29,43\}$
765	637	25	24	0.83	$\{127\}$
803	559	29	28	0.70	$\{33,73,275\}$
819	515	30	29	0.63	$\{41,91,195,273,351\}$
861	577	30	29	0.67	$\{41,49,369\}$
1057	813	34	33	0.77	$\{109,353,453\}$

Table B.2: A set of Cyclic binary OSMLD codes with code-lengths $1059 \leq n \leq 2001$ constructed using the OSMLD-GA algorithm

n	k	d_{min}	J	$r = \frac{k}{n}$	$\Omega_{E(x)}$
1023	781	33	32	0.76	$\{35,45,179,341\}$
1071	767	34	33	0.72	$\{29,255,459\}$
1143	749	35	34	0.66	$\{9,127,261,387,423\}$
1155	825	30	29	0.71	$\{105,143,231,495\}$
	665	35	34	0.58	$\{133,187,385\}$
1157	815	35	34	0.71	$\{39,89,429\}$
1197	763	31	30	0.64	$\{133,135,285\}$
	707	36	35	0.59	$\{181,399,437,513,589\}$
1209	817	35	34	0.68	$\{31,217,221\}$
1235	881	35	34	0.71	$\{57,65,247\}$
1271	695	36	35	0.55	$\{41,111,123,205\}$
1275	739	37	36	0.58	$\{105,119,475\}$
1285	737	37	36	0.57	$\{13,159,257\}$
1287	819	37	36	0.64	$\{117,275,319,429\}$
1305	721	37	36	0.55	$\{69,145,435\}$
1335	947	37	36	0.71	$\{89,195,267,285,445,623\}$
	831	38	37	0.62	$\{65,135,267\}$
1359	907	37	36	0.67	$\{135,151,153\}$
1365	813	37	36	0.60	$\{25,143,469\}$
	741	38	37	0.54	$\{37,65,195,309,637\}$
1395	983	38	37	0.70	$\{15,105,217,225\}$
1419	885	39	38	0.62	$\{121,129,143\}$
1457	739	39	38	0.51	$\{31,47,235,705\}$
1479	915	37	36	0.62	$\{29,51\}$
	753	39	38	0.51	$\{51,261,493\}$

1495	1073	39	38	0.72	{65,161,299,325}
1513	1079	39	38	0.71	{89,221,267,323}
1519	1095	40	39	0.72	{21,31,217}
1533	1053	37	36	0.69	{65,125}
1547	901	39	38	0.58	{129,221,273,663}
	815	40	39	0.53	{15,289,663}
1561	781	41	40	0.50	{21,669}
1581	1069	41	40	0.68	{181}
1615	1173	35	34	0.73	{85,399,703}
	833	39	38	0.52	{85,95,247,323}
1617	1191	41	40	0.74	{33,147,385,693}
1635	975	41	40	0.60	{95,327}
1645	827	42	41	0.50	{47,175,235,705}
1677	1229	41	40	0.73	{117,129,143}
1683	1251	41	40	0.74	{153,165,187,297,363}
1705	1065	41	40	0.62	{39,171}
1725	1041	43	42	0.60	{69,125}
1755	1453	43	42	0.83	{59,195}
1785	1225	41	40	0.69	{77,217,255,301,315,595,765}
	1157	42	41	0.65	{85,133,259,637,765,777}
	1055	43	42	0.59	{31,221,357,595}
1827	1197	43	42	0.65	{63,319,609,899}
	1113	44	43	0.61	{105,203,783,899}
1905	1521	43	42	0.80	{85,213}
	1099	45	44	0.58	{25,221,635}
1935	1409	43	42	0.73	{135,165,301,645}
	1255	45	44	0.65	{57,301,903}
1953	1537	41	40	0.79	{105,167}
	1251	45	44	0.64	{9,155,315,403,483,651}

1971	1195	45	44	0.61	$\{87,143,219,657\}$
1995	1243	46	45	0.62	$\{19,75,171,855\}$

Table B.3: A set of Cyclic binary OSMLD codes with code-lengths $2003 \leq n \leq 4001$ constructed using the OSMLD-GA algorithm

n	k	d_{min}	J	$r = \frac{k}{n}$	$\Omega_{E(x)}$
2047	1167	45	44	0.57	$\{169,245,699,759\}$
2049	1365	45	44	0.67	$\{13,87\}$
2093	1229	46	45	0.59	$\{39,437\}$
2139	1101	47	46	0.51	$\{115,207,253,345,465,759\}$
2145	1439	47	46	0.67	$\{187,195,325,451,715\}$
2193	1567	47	46	0.71	$\{17,187,387,731,817\}$
2261	1239	49	48	0.55	$\{57,323,527,969\}$
2289	1623	48	47	0.71	$\{109,245,763,981\}$
	1195	49	48	0.52	$\{53,109,327,981\}$
2325	1325	49	48	0.57	$\{75,155,175,397,525,1085\}$
2331	1423	49	48	0.61	$\{203,259,333,999\}$
2359	1369	49	48	0.58	$\{119,131,337,1011\}$
2365	1421	49	48	0.60	$\{129,209\}$
2387	1327	49	48	0.56	$\{407,497,511,819,1023\}$
2317	1215	50	49	0.52	$\{47,51,423,799,893\}$
2415	1995	50	49	0.83	$\{105,161,253,345,525,575,805\}$
2475	1405	51	50	0.57	$\{25,171\}$
2555	1291	52	51	0.51	$\{219,255,365,375,855,915\}$
2665	1637	53	52	0.61	$\{13,41,195\}$
2667	1511	53	52	0.57	$\{15,381,497,609,651\}$
2691	1733	53	52	0.64	$\{115,195,299,667\}$
2759	1737	54	53	0.63	$\{31,267,279,403,445,979,1335\}$
2821	1733	54	53	0.61	$\{31,91,195,273,403,637,1001,1209\}$
2829	1897	54	53	0.67	$\{205,253,615\}$
2907	2085	55	54	0.72	$\{119,425,513,969,1083\}$

2937	2135	55	54	0.73	$\{33,89,267,363,445,979\}$
3003	1737	56	55	0.58	$\{33,253,273,429,455,715,1001\}$
3045	1817	56	55	0.60	$\{145,175,377,435,725\}$
3135	2191	57	56	0.70	$\{11,551\}$
3213	2021	57	56	0.63	$\{63,255,381,527\}$
3255	2135	58	57	0.66	$\{225,245,483,651,735,1395\}$
3465	1993	60	59	0.58	$\{11,99,315,363,495,525\}$
3471	2873	60	59	0.83	$\{39,89,195,623,1157,1287\}$
3575	1959	61	60	0.55	$\{39,55,385,605,715\}$
3627	2475	61	60	0.68	$\{391\}$
3641	2141	61	60	0.59	$\{191,549\}$
3795	2539	63	62	0.67	$\{55,69,667\}$
3843	2241	63	62	0.58	$\{55,69,667\}$
3861	2315	63	62	0.60	$\{117,143,231,351,957\}$
3895	2585	63	62	0.66	$\{95,133,205,779\}$
3927	2311	64	63	0.59	$\{55,357,595,693,1463,1683\}$
3937	2507	64	63	0.64	$\{31,201,651,961,1953\}$
3999	2519	64	63	0.63	$\{43,129,403,473,645,651,903\}$

Table B.4: A set of Cyclic binary OSMLD codes with code-lengths $4003 \leq n \leq 6001$ constructed using the OSMLD-GA algorithm

n	k	d_{min}	J	$r = \frac{k}{n}$	$\Omega_{E(x)}$
4005	3303	64	63	0.82	$\{75,89,267,445,801,1485,1869\}$
4017	2013	64	63	0.50	$\{117,721\}$
4029	2029	64	63	0.50	$\{51,79,237,1501\}$
4059	3459	61	60	0.85	$\{317\}$
4085	2833	65	64	0.69	$\{57,215,285,817\}$
4095	2263	65	64	0.55	$\{145,157,831,895,1403,1911\}$
4097	2497	65	64	0.61	$\{241,299,333,723\}$
4123	2513	65	64	0.61	$\{19,31,217,399,589,1463\}$
4161	3431	66	65	0.82	$\{1,285,307,357,1387\}$
4277	2171	66	65	0.51	$\{141,455,517,611,893,1833\}$
4305	3305	67	66	0.67	$\{181,1025\}$
4329	2335	67	66	0.54	$\{177,481,925,1073\}$
4365	2531	67	66	0.58	$\{299,485,679\}$
4371	2923	67	66	0.67	$\{31,235,1081\}$
4433	2277	68	67	0.51	$\{117,195,341,533,923,1001,1105\}$
4445	2707	68	67	0.61	$\{35,175,345,385,889,945,1085,2205\}$
4485	2537	68	67	0.57	$\{195,221,437\}$
4515	3203	68	67	0.71	$\{105,129,217,301,903,1505,1935\}$
4539	3797	68	67	0.84	$\{51,89,255,445,979,1513,1683,1691\}$
4575	2627	69	68	0.57	$\{103,305,915\}$
4599	2491	69	68	0.54	$\{207,225,237,949,1143,1533,2263,2295\}$
4641	2745	69	68	0.59	$\{31,51,153,867,1729\}$
4743	2967	70	69	0.62	$\{153,155,387\}$
4781	3545	70	69	0.74	$\{135,683\}$
4845	2831	71	70	0.58	$\{19,85,285,323,1045,1653,1729,2109\}$

4879	3473	71	70	0.71	$\{119,357,533,697,2091\}$
4977	2673	72	71	0.54	$\{9,395,711,869,1659,1817,2133,2449\}$
4991	2819	72	71	0.56	$\{31,161,345,621,2139\}$
5037	3277	72	71	0.65	$\{207,219,345,365,759,897,1679\}$
5073	4259	72	71	0.84	$\{57,89,445,513,741,1691\}$
5115	2817	73	72	0.55	$\{145,215,307,415,625,835,1705\}$
5187	3363	73	72	0.64	$\{29,155\}$
5225	4865	73	72	0.93	$\{95,275,285,475,1045\}$
5285	3365	73	72	0.64	$\{23,151\}$
5313	3507	74	73	0.66	$\{99,231,483,1155,1265,1771\}$
5369	3055	74	73	0.57	$\{91,531,2301\}$
5439	3205	75	74	0.59	$\{7,111,259,777,1295,1813\}$
5565	3107	76	75	0.56	$\{133,689,795,1325,1855\}$
5673	3225	76	75	0.57	$\{7,183,549,2013\}$
5715	4563	71	70	0.80	$\{115,951\}$
5735	3473	77	76	0.61	$\{279,407,925,1295,2035,2775\}$

Table B.5: A set of Cyclic binary OSMLD codes with code-lengths $6003 \leq n \leq 8001$ constructed using the OSMLD-GA algorithm

n	k	d_{min}	J	$r = \frac{k}{n}$	$\Omega_{E(x)}$
6027	4757	76	75	0.79	$\{123,147,287,539,1435,2009\}$
	3121	78	77	0.52	$\{123,343,287,369,1435,2583\}$
6035	3397	79	78	0.56	$\{85,213,595\}$
6039	3597	79	78	0.60	$\{219,671,915,2013\}$
6045	4485	73	72	0.75	$\{73,341\}$
	3829	79	78	0.63	$\{139,403,1495,2821\}$
6095	4717	75	74	0.77	$\{23,265,1325\}$
6105	3663	79	78	0.60	$\{37,297,703,2035\}$
6215	4105	79	78	0.66	$\{55,113,339,565\}$
6273	3871	79	78	0.62	$\{525,615,697,779\}$
6279	3203	80	79	0.51	$\{115,575,621,759,897,1365,2093,2231,2691\}$
6321	3643	81	80	0.58	$\{363,387,539,2709\}$
6327	3439	81	80	0.54	$\{11,379,703,2109\}$
6355	3475	81	80	0.55	$\{57,257,1075,2355\}$
6405	3737	81	80	0.58	$\{17,61,427,1281\}$
6425	5721	81	80	0.89	$\{557\}$
6435	3599	81	80	0.56	$\{297,341,583,979,1131,1353\}$
6495	3751	81	80	0.58	$\{33,433,3031\}$
6555	5091	82	81	0.78	$\{95,115,437,575,1311,1425,3059\}$
6643	3627	82	81	0.55	$\{135,195,247,325,377,2431\}$
6665	4255	82	81	0.64	$\{155,215,465,473,1085,1333,1505,2365\}$
6765	3703	83	82	0.55	$\{233,401,497,1133,2255\}$
6825	6121	61	60	0.90	$\{661\}$
6851	4243	82	81	0.62	$\{85,403,663,1209\}$
6853	4115	83	82	0.60	$\{253,385,623,693,979,1463,2937\}$

6923	4609	84	83	0.67	{43,473,989,1127}
6975	5745	67	66	0.82	{7,775}
	3831	85	84	0.55	{185,687,3255}
6993	4005	85	84	0.57	{43,267,1221,1443}
7011	4907	85	84	0.70	{171,399,533,779,1025,2337}
7077	5301	85	84	0.75	{97,575}
7085	4153	85	84	0.59	{67,503,1199}
7105	5321	74	73	0.75	{145,203,245,609}
7119	5259	81	80	0.74	{21,113,339,1695,1743,2599}
	4431	85	84	0.62	{815}
7161	5241	61	60	0.73	{235,737}
	4121	86	85	0.58	{119,259,441,1029,1057,1221,1225,1267}
7353	5993	81	80	0.81	{19,129,645,2451}
	4887	87	86	0.66	{215,399,513,645,817,1197,2451}
7493	5371	80	79	0.72	{127,295,1121,3717}
	4915	87	86	0.66	{127,295,885,2537,3717}
7565	5155	81	80	0.68	{89,255,425,623,935,1335,1513,2805,3293}
	4171	88	87	0.55	{85,255,445,765,801,1615,2581,2805,3293}
7659	4185	89	88	0.55	{23,333,555,851,1665,2553}
7667	4427	89	88	0.58	{357,451,943,1547}
7749	5223	85	84	0.69	{465,861,943}
	4533	89	88	0.58	{287,441,567,1107,1353,1599,2829,3321,3813}
7755	6375	85	84	0.82	{165,235,517,705,825,1175,3619}
7905	5601	81	80	0.71	{211,267}
	4725	90	89	0.60	{49,93,255,391,527,1581,3441}
7975	6485	83	82	0.81	{29,275,493,725,1595}

Table B.6: A set of Cyclic binary OSMLD codes with code-lengths $8003 \leq n \leq 10001$ constructed using the OSMLD-GA algorithm

n	k	d_{min}	J	$r = \frac{k}{n}$	$\Omega_{E(x)}$
8001	6209	85	84	0.78	$\{229,475\}$
	4633	90	89	0.58	$\{229,423,609,635,1651\}$
8029	4429	90	89	0.55	$\{111,259,341,481,555,3441\}$
8099	4731	91	90	0.58	$\{247,801,949,1691\}$
8103	4539	91	90	0.56	$\{43,63,999,1887\}$
8151	4781	91	90	0.59	$\{55,209,247,627,741,1235\}$
8191	4407	92	91	0.54	$\{395,245,675,1749,1773,1909,3035\}$
8211	5907	91	90	0.72	$\{187,759\}$
	5209	92	91	0.63	$\{153,391,759,805,1173,1449,1955,3519\}$
8215	6275	87	86	0.76	$\{155,477,1643,1855,3975\}$
	4727	92	91	0.58	$\{265,583,589,1325,1643,1855\}$
8415	4571	93	92	0.54	$\{153,209,495,561,1045,1683,3135\}$
8463	6079	91	90	0.72	$\{65,255\}$
	4961	93	92	0.59	$\{117,279,589,793,1209,1829,2015,2821\}$
8547	5421	93	92	0.63	$\{37,165,777,1221,1295,3663\}$
8635	5095	93	92	0.59	$\{77,157,471\}$
8645	4741	94	93	0.55	$\{133,173,323,325,1235,1615\}$
8733	5781	93	92	0.66	$\{123,781,861,2911\}$
8745	5095	95	94	0.58	$\{121,583,1007,1325,1749,4081\}$
8763	5577	94	93	0.64	$\{161,165,2921\}$
	4541	95	94	0.52	$\{127,207,345,529,759,1311,1495,1863,2921,3243\}$
8789	5587	95	94	0.64	$\{141,187,517,935\}$
8835	5241	95	94	0.59	$\{437,513,527,775\}$
8845	6785	93	92	0.77	$\{29,305,1769\}$
8855	7567	92	91	0.85	$\{161,253,385,483,805,1771,1925,3795\}$

	6223	95	94	0.70	{165,605,805,1265,1771,1925}
8925	7133	89	88	0.80	{435,637,795}
	4663	95	94	0.52	{119,265,525,665,875,2065,2205,2975,3045}
8979	5945	95	94	0.66	{123,369,615,1107,1353,1599,1679,2091,2993,3075}
8995	5039	96	95	0.56	{119,245,259,771,1071,1285,1393}
9009	5195	95	94	0.58	{77,385,663,819,1529,3025,3289}
9021	5403	96	95	0.60	{97,403,679,873,1455,2037,3007,3201,4365}
9051	6209	96	95	0.67	{147,483,1293,2155}
9135	6287	91	90	0.69	{31,3335}
	5379	97	96	0.59	{29,105,147,725,1595,1827,1885,3335}
9165	5865	97	96	0.64	{195,517,611,799,975,1833,1927,3055,4277}
9225	7777	61	60	0.84	{1123}
	6377	81	80	0.69	{17,75}
	5295	87	86	0.57	{39,441,849,981,1025}
	5221	93	92	0.57	{231,405,453,699,1435}
	4893	97	96	0.53	{15,177,519,621,1435,4305}
9265	5473	97	96	0.59	{327,379,2289,4033}
9271	7507	73	72	0.81	{169,1143}
9331	7433	91	90	0.80	{43,217,387,645,651,1333,1519}
	5195	98	97	0.56	{301,387,473,713,1161,3311}
9405	7601	81	80	0.81	{19,495,3135}
	6593	97	96	0.70	{165,551,935}
9465	5855	97	96	0.62	{15,285,631,3155}
9513	6241	97	96	0.65	{71,281,867,1661}
	5049	99	98	0.53	{63,569,1017,1449,3171,3177,4681}
9555	8851	85	84	0.93	{661}
	6771	97	96	0.71	{41,1001}
9597	7677	97	96	0.80	{105,457,1371,2285,3199,4113}
9605	6357	97	96	0.66	{221,791,1921,2007}
9657	5339	99	98	0.55	{87,333,555,1073}

9709	5205	100	99	0.54	$\{521,1165,1607,2237,2269,4845\}$
9765	7285	96	95	0.75	$\{137,425,4725\}$
	5725	99	98	0.59	$\{375,403,693,713,837,2263\}$
	4987	100	99	0.51	$\{35,335,945,1829,1953,2205,2415,4185\}$
9815	5715	100	99	0.58	$\{69,453,715,755\}$
9821	6887	100	99	0.70	$\{61,161,1403,4209\}$
9911	6307	101	100	0.64	$\{187,477,1749\}$
9933	5817	101	100	0.59	$\{301,385,473,693,1045,1617\}$
9945	5405	101	100	0.54	$\{39,73,427,459,975,1131,1677,3705\}$

Table B.7: A set of Cyclic binary OSMLD codes with code-lengths $n \geq 10001$ constructed using the OSMLD-GA algorithm

n	k	d_{min}	J	$r = \frac{k}{n}$	$\Omega_{E(x)}$
10005	6101	101	100	0.61	$\{161, 253, 377\}$
10019	7367	101	100	0.74	$\{129, 215, 233, 699, 1631\}$
10095	6055	101	100	0.60	$\{99, 683, 4711\}$
10137	6769	94	93	0.67	$\{31, 109, 327, 545, 763, 1853, 2289, 3379, 4905\}$
	6053	99	98	0.60	$\{545, 763, 1085, 1199, 1635, 1853, 2289, 3379, 3597, 4905\}$
	5753	102	101	0.57	$\{327, 763, 837, 981, 1199, 1635, 1853, 2507, 3597, 4905\}$
10143	7041	102	101	0.69	$\{207, 511, 1771, 3381, 3703\}$
10305	5875	103	102	0.57	$\{123, 229, 1603, 3435\}$
10335	6273	103	102	0.60	$\{117, 477, 583, 689, 2067, 3445, 3551, 4823\}$
10413	7451	100	99	0.72	$\{117, 801, 1053, 1157, 1287, 1513, 2223, 2225, 3471, 3861\}$
10419	7761	100	99	0.74	$\{207, 453, 483, 851, 2265, 2553, 3473\}$
10341	8785	103	102	0.84	$\{171, 183, 915, 1159\}$
10485	7715	103	102	0.74	$\{45, 233, 699, 1165, 1305, 1631, 2097, 3495, 4893\}$
10509	6959	104	103	0.66	$\{113, 155, 565, 791, 961, 1017, 3503, 3729, 5085\}$
10791	7979	105	104	0.74	$\{33, 109, 327, 981, 1199, 1635, 3597\}$
11023	6705	106	105	0.61	$\{73, 267, 803, 1679, 2701\}$
11295	7177	107	106	0.64	$\{201, 1255\}$
11303	9191	85	84	0.81	$\{31, 623\}$
11315	7495	96	95	0.66	$\{219, 455, 657, 1825, 2555\}$
	6895	107	106	0.61	$\{73, 365, 595, 1095, 1395, 1705, 2263, 2635\}$
11385	8907	95	94	0.78	$\{165, 253, 345, 759, 825, 1265, 1725, 2277, 5313\}$
	8305	105	104	0.73	$\{429, 437\}$
	8223	107	106	0.72	$\{115, 495, 621, 1771, 2001, 2475, 3795\}$
11417	8411	107	106	0.74	$\{147, 233, 245, 699, 1631, 4893\}$
11557	8233	109	108	0.71	$\{69, 2041, 4953\}$
11985	7721	110	109	0.64	$\{329, 425, 893, 1275, 1363, 4089, 4277\}$

12075	9415	107	106	0.78	$\{275,483,1127\}$
	7953	111	110	0.66	$\{375,483,575,1265,1955,2415,2625,2875,4025,5635\}$
12087	7247	111	110	0.60	$\{237,869,901\}$
12369	7461	112	111	0.60	$\{57,361,651,899,931,1271,4123\}$
12441	9393	113	112	0.75	$\{319,435,429,1885,4147\}$
12465	9117	113	112	0.73	$\{435,831,1385,2493,4155,5817\}$
12533	9871	113	112	0.79	$\{151,1245,2905\}$
13053	8567	115	114	0.66	$\{171,229,1145,4351\}$
16513	14325	130	129	0.87	$\{1,89,947,965,1699,2737,7077\}$
65793	59231	258	257	0.90	$\{1,1205,1309,1395,1749,2209,3713,4427,6931,9399,9553,11529,21931\}$

Table B.8: A set of Cyclic non-binary OSMLD codes over $GF(4)$ constructed using the NB-OSMLD-GA algorithm (Construction B)

n	k	d_{min}	J	$r=\frac{k}{n}$	$\Omega_{E(x)}$
7	3	4	3	0.43	$\{3\}$
15	7	5	4	0.47	$\{1,2\}$
21	11	6	5	0.52	$\{3,7,14\}$
31	15	6	5	0.48	$\{15\}$
63	37	9	8	0.59	$\{5,10,21,42\}$
73	45	10	9	0.62	$\{25\}$
85	37	9	8	0.44	$\{9,18\}$
93	61	11	10	0.66	$\{5,10\}$
105	55	11	10	0.52	$\{7,15,14,45\}$
119	59	12	11	0.50	$\{7,14,51\}$
155	79	10	11	0.51	$\{5,31,62\}$
217	153	16	15	0.70	$\{37\}$
231	119	16	15	0.52	$\{7,14,77,99,154\}$
273	191	18	17	0.70	$\{39,91,97,101,182\}$
315	203	19	18	0.64	$\{65,107,130,151\}$
345	173	20	19	0.50	$\{15,23,46,161,253\}$
399	213	21	20	0.53	$\{133,143,187,266\}$
451	251	21	20	0.56	$\{65,69\}$
465	337	21	20	0.72	$\{53,106\}$
511	291	22	21	0.57	$\{45,79,219\}$
527	273	24	23	0.52	$\{31,62,85,187,255\}$
589	303	24	23	0.51	$\{31,62,209\}$
595	403	25	24	0.68	$\{23,46\}$
635	403	26	25	0.63	$\{15,35,127,145,254\}$
665	361	26	25	0.54	$\{95,133,155,225,266\}$

713	367	27	26	0.51	{31,69,115,253}
803	559	29	28	0.70	{55,73,146,187}
819	531	25	24	0.65	{37,74,191,331}
861	577	30	29	0.67	{123,161,205,301,410}
889	637	25	24	0.72	{51,127}
	587	28	27	0.66	{127,133,217,329,381}
945	777	31	30	0.82	{5,10,21,42}
993	511	31	30	0.51	{49,59}
1143	609	35	34	0.53	{63,127,243,254,261,423}
1157	815	35	34	0.70	{39,89,178,429}
1241	629	36	35	0.51	{17,51,187,219,438}
	883	35	34	0.71	{73,146,187,219,221,438}
1333	749	35	34	0.56	{31,62,129,215,473,645}
1335	703	36	35	0.53	{15,25,50,623,979}
1359	757	37	36	0.56	{99,151,302,333}
1387	811	37	36	0.58	{139,278,321,563}
1395	1019	33	32	0.73	{147,217,341,294}
1457	739	39	38	0.51	{47,155,235,705}
1495	1073	39	38	0.72	{65,253,299,325,506,598}
	743	40	39	0.50	{65,161,207,253,299,506,598}
1533	905	40	39	0.59	{21,93,175,219,245}
1561	781	41	40	0.50	{21,669}
1581	1069	41	40	0.68	{43,86}
1651	1051	41	40	0.64	{65,127,143,254,559,819}
1661	1231	41	40	0.74	{11,151,302,407}
1679	1207	41	40	0.72	{69,73,365,575}
1691	1211	41	40	0.72	{19,89,178,209}
1705	1065	41	40	0.62	{69,138,139,278}
1755	1459	37	36	0.83	{29,58}

1785	1401	33	32	0.78	{139,278,301,602}
1963	1471	43	42	0.75	{91,151,302,481}
2007	1007	46	45	0.50	{171,223,446,669,1338}
2015	1207	45	44	0.60	{31,39,62,78,217,279}
2093	1229	46	45	0.59	{143,253,506}
2107	1127	46	45	0.53	{85,170,903}
2139	1091	47	46	0.51	{69,93,207,253,299,345,529,1058}
2201	1281	41	40	0.58	{317,781}
2263	1815	46	45	0.80	{229}
	1335	47	46	0.59	{219,279,341,403,511,775}
2313	1929	49	48	0.83	{191,271}
2325	1349	46	45	0.58	{57,114,229,353,525}
2331	1525	43	42	0.62	{159,259,318,518}
	1463	46	45	0.63	{63,126,333,481,962}
2359	1477	49	48	0.63	{13,179,337,1011}
2387	1807	49	48	0.76	{11,191,382,1023}
2397	1215	50	49	0.51	{141,235,255,282,329,517,799,1034,1598}
2415	1727	46	45	0.71	{15,253,506}
	1217	50	49	0.50	{21,42,805,1035,1610}
2457	2073	37	36	0.84	{31,62}
	1401	49	48	0.57	{15,30,59,118}
2607	1307	52	51	0.50	{99,395,553,869,1738}
2635	1867	41	40	0.71	{71,142}
	1607	51	50	0.61	{259,383,935,1275}
2667	2163	49	48	0.81	{95,127,190,254}
2691	1733	53	52	0.64	{23,46,161,195,273,299,322,598}
2695	2117	47	46	0.79	{55,231,245,462,490,1155}
2759	2003	53	52	0.73	{31,89,267,341,445,623,979,1335}
	1495	54	53	0.54	{89,155,279,445,589,623,1335}

2829	1897	54	53	0.67	{23,41,46,82,123}
2869	2191	49	48	0.76	{133,151,302,323}
2937	1671	55	54	0.57	{143,165,267,286,429,534}
2945	2195	53	52	0.74	{155,310,437,551,589,665,1045,1178}
3005	1801	55	54	0.60	{105,305,601,1202}
3115	2073	56	55	0.67	{75,89,178,445,623,1246,1335}
3139	1927	56	55	0.61	{43,73,146,219,438,473,731}
3311	2371	59	58	0.72	{11,22,301,473,602,1419}
3379	1893	57	56	0.56	{109,279,341,545,763,1199}
3381	2531	57	56	0.75	{147,207,245,343,1127,2254}
3395	2021	55	54	0.60	{39,78,485,1455}
3441	1887	59	58	0.55	{111,341,403,555,1147,1221,1665,2294}
3451	2387	56	55	0.69	{119,238,377,551,1479}
3627	2475	61	60	0.68	{173,346}
3655	2375	57	56	0.65	{319,627}
3675	3047	42	41	0.83	{225,343,539}
3683	2115	57	56	0.57	{275,311,355,550}
3689	2101	60	59	0.57	{187,341,561,682,1309}
3843	2241	63	62	0.58	{115,230,1281,2562}
3965	2017	61	60	0.51	{101,167}
3995	2605	63	62	0.65	{85,141,282,425,705,1410}
4165	3105	58	57	0.75	{85,119,238,483,581}
	2513	62	61	0.60	{85,343,371,567,686,1813,2009}
4301	2875	63	62	0.67	{187,529,667,935}
4305	3157	64	63	0.73	{43,86,1845}
4361	3103	66	65	0.71	{49,267,441,539,1617}
4371	2651	64	63	0.61	{31,62,141,987,1457,2115,2914}
4487	2243	68	67	0.50	{63,91,641}
4495	2765	68	67	0.62	{31,62,145,203,406,899,1595,1798,2175}

4623	2357	69	68	0.51	$\{11, 22, 1541, 3082\}$
4669	3239	68	67	0.69	$\{87, 161, 322, 667, 2001\}$
4681	2685	66	65	0.57	$\{233, 273, 459, 701, 2265\}$
4717	2945	64	63	0.62	$\{89, 178, 689\}$
4743	3351	70	69	0.71	$\{69, 138, 341, 459, 682\}$
4781	3545	70	69	0.74	$\{9, 18, 683\}$
4893	3079	68	67	0.63	$\{63, 105, 1165, 2097, 2330\}$
4991	2545	72	71	0.51	$\{161, 207, 217, 299, 345, 483, 1771\}$
5083	3581	71	70	0.70	$\{115, 221, 230, 1105, 1265, 2139\}$
5313	3777	70	69	0.71	$\{33, 69, 138, 1265, 2530\}$
5369	3055	74	73	0.57	$\{91, 182, 767, 1121, 2242\}$
5429	3393	72	71	0.62	$\{89, 178, 549\}$
5487	3285	74	73	0.60	$\{93, 186, 295, 590, 2655\}$
5499	3675	71	70	0.67	$\{39, 47, 78, 94, 235, 470\}$
5607	3203	76	75	0.57	$\{89, 178, 207, 261, 801\}$
5621	3061	76	75	0.54	$\{55, 219, 438, 451, 473, 605, 1925\}$
5635	3991	74	73	0.71	$\{105, 345, 483, 966, 1127, 2254, 2415\}$
5735	4005	76	75	0.70	$\{155, 185, 310, 333, 555, 666, 1147, 1295, 2294\}$
5889	4029	76	75	0.68	$\{125, 139, 429\}$

Table B.9: A set of Cyclic non-binary OSMLD codes over $GF(8)$ constructed using the NB-OSMLD-GA algorithm (Construction B)

n	k	d_{min}	J	$r=\frac{k}{n}$	$\Omega_{E(x)}$
15	7	5	4	0.47	$\{7\}$
31	15	6	5	0.48	$\{15\}$
93	61	11	10	0.66	$\{23\}$
127	63	8	7	0.50	$\{63\}$
155	79	10	9	0.51	$\{25,31\}$
195	113	15	14	0.58	$\{11,22,44,65\}$
217	153	16	15	0.70	$\{15,23,46\}$
231	119	16	15	0.52	$\{33,35,66,77,132\}$
255	175	17	16	0.69	$\{31,39\}$
279	167	17	16	0.60	$\{31,45,62,99,124\}$
315	191	17	16	0.61	$\{31,47,62,63\}$
341	205	16	15	0.60	$\{11,15\}$
345	173	20	19	0.50	$\{15,23,161\}$
357	185	20	19	0.52	$\{7,35,51,102,204\}$
381	255	17	16	0.67	$\{17,127\}$
387	231	21	20	0.60	$\{9,43,86,172\}$
451	251	21	20	0.56	$\{155\}$
465	337	21	20	0.72	$\{97\}$
	271	23	22	0.58	$\{31,75,93,165,217\}$
527	273	24	23	0.52	$\{51,93,119,187\}$
589	303	24	23	0.51	$\{31,62,124,209\}$
595	403	25	24	0.68	$\{3,6,12\}$
615	371	25	24	0.60	$\{123,127\}$
623	333	26	25	0.53	$\{21,231,267,445,534\}$
635	403	26	25	0.63	$\{5,127,135,235\}$

645	405	25	24	0.63	{43,105,215,301}
651	491	26	25	0.75	{99,161,198,291}
713	367	28	27	0.51	{23,31,155}
715	497	27	26	0.70	{11,22,44,65,143}
765	429	29	28	0.56	{17,34,68,135,381}
825	517	29	28	0.63	{19,165,385}
833	437	30	29	0.52	{49,51,85,102}
889	697	29	28	0.78	{11,22,44,49}
935	697	31	30	0.75	{33,85,143,187}
1035	549	33	32	0.53	{15,69,115,230,460}
1105	657	33	32	0.59	{39,93,186,213}
1143	589	35	34	0.52	{45,127,129,254,508,567}
1155	715	33	32	0.62	{99,198,203,396}
1157	815	35	34	0.70	{13,89,143,178,356}
1161	753	33	32	0.65	{27,43,86,172}
1271	735	36	35	0.58	{205,215,287,615}
1275	723	37	36	0.57	{25,115,119}
1285	705	37	36	0.55	{31,193,257}
1305	721	37	36	0.55	{111,145,290,435,580}
1335	947	37	36	0.71	{89,195,267,285,445,623}
	703	38	37	0.53	{65,89,495}
1395	933	37	36	0.67	{85,170,279,340,465}
1395	1003	38	37	0.72	{27,31,62,124,495}
1419	1029	39	38	0.73	{99,121,129}
1479	915	37	36	0.62	{51,145}
1495	1073	39	38	0.72	{65,69,138,276,299,325}
1513	1079	39	38	0.71	{89,221,267,323}
1581	1069	41	40	0.68	{181}
1615	1173	35	34	0.73	{19,85,170,340,551}

1651	1051	41	40	0.64	{91,127,169,254,508,559,819}
1677	1229	41	40	0.73	{39,91,129,258,516}
1695	903	41	40	0.53	{57,113,339,791}
1705	1065	41	40	0.62	{53,411}
1725	1041	43	42	0.60	{25,69}
1755	1459	37	36	0.83	{209,251,418}
	1035	41	40	0.59	{35,70,117,140}
1885	1221	41	40	0.65	{13,87,174,348}
1905	1521	43	42	0.80	{57,355}
	1179	45	44	0.62	{93,185,635}
1985	1013	45	44	0.51	{211}
1995	1219	41	40	0.61	{133,169,289,571}
2007	1007	46	45	0.50	{27,223,446,669,892}
2047	1167	45	44	0.57	{19,51,181,199}
2049	1365	45	44	0.67	{9,173}
2091	1235	45	44	0.59	{119,205,451,779}
2115	1065	46	45	0.50	{45,235,329,423,470,611,940,1222}
2139	1563	45	44	0.73	{69,93,465,529,713,759}
	1463	47	46	0.67	{187,215,357,473,731}
2277	1317	49	48	0.58	{69,99,207,253,495,506,1012}
2325	1385	49	48	0.60	{143,155,271,465}
2337	1517	41	40	0.65	{123,133,246,492,779}
2365	1421	49	48	0.60	{93,429}
2403	1631	43	42	0.68	{89,117,178,356,801}
2451	1795	49	48	0.73	{95,129,171,258,516,817}
2465	1513	49	48	0.61	{51,203,435,493}
2475	1405	51	50	0.57	{63,125,175,250}
2499	1315	51	50	0.53	{85,147,170,221}
2635	1867	41	40	0.71	{467}

	1607	51	50	0.61	{33,85,255}
2731	1535	53	52	0.56	{101,217}
2755	2049	51	52	0.74	{145,209,290,551,580}
2759	2003	53	52	0.73	{89,267,403,445,589,623,979,1335}
2871	2313	55	54	0.81	{87,99,319,435,638,1276}
2937	2135	55	54	0.73	{89,267,429,445,627,979}
2945	2535	48	47	0.86	{155,209,310,589,620,1425}
	1617	54	53	0.55	{155,171,310,475,620,1045,1425}
2967	2229	51	50	0.75	{129,437,483,645}
3005	1801	55	54	0.60	{15,315,601}
3135	2035	53	52	0.65	{165,330,361,475,660,1463}
3195	1605	58	57	0.50	{45,71,142,284,355,639,710,1420}
3309	2205	59	58	0.67	{559}
3335	2497	55	54	0.75	{145,253,667,725}
3381	1825	61	60	0.77	{49,69,138,147,276}
3519	2837	60	59	0.81	{153,255,437,575,1127,1173}
3553	2363	59	58	0.66	{187,374,437,748}
3565	1825	61	60	0.51	{31,155,805}
3627	2475	61	60	0.68	{215,275,430}
3655	2199	61	60	0.60	{43,119,387,559,645}
3689	2265	61	60	0.61	{133,187,357,374,748}
3765	2713	61	60	0.72	{15,251,1255,1757}
3795	2853	63	62	0.75	{165,207,575,759,825,1265,1771}
3813	2083	63	62	0.55	{35,205,333,369,615,1271}
3937	2073	64	63	0.53	{31,193,403,651,1705}
3995	2011	64	63	0.50	{47,85,141,611,705,987}
3999	3291	60	59	0.82	{93,279,645,651,989,1333}
3999	2409	64	63	0.60	{215,301,341,387,651,903,1419}
4085	2901	61	60	0.71	{95,215,247,430,860}

	2453	65	64	0.60	$\{133,301,602,989\}$
4191	2691	63	62	0.64	$\{127,627,635,715,759,1551,1815\}$
	2193	65	64	0.52	$\{381,429,635,891,957,1397,1419,1551,1815\}$
4257	3065	65	64	0.72	$\{33,43,86,99,172,473,946,1892\}$
4371	2433	67	66	0.56	$\{47,93,465,1551,2115\}$
4495	2765	68	67	0.62	$\{93,145,261,899,1595,2175\}$
4495	3393	63	62	0.75	$\{145,341,435,725,899,1015,1595,2175\}$
4539	3065	68	67	0.68	$\{89,221,255,267,445,979,1513\}$
4669	3293	65	64	0.71	$\{87,161,174,348,667,1334,2668\}$
4715	3145	67	66	0.67	$\{205,483,345,943,1025\}$
4785	3149	69	68	0.66	$\{145,319,363,435,725,1595,2233\}$
4859	2829	71	70	0.58	$\{113,183,379\}$
4895	4153	70	69	0.85	$\{363,445,715,979\}$
5035	3869	71	70	0.77	$\{19,265,530,1060\}$
5313	3591	74	73	0.68	$\{77,115,161,230,231,299\}$
5365	4065	69	68	0.76	$\{145,259,290,435,1073\}$
5763	4119	73	72	0.71	$\{51,323,339,1017\}$
5763	3653	75	74	0.63	$\{113,255,1017,1411,1921\}$
5911	3953	71	70	0.67	$\{161,257,299,1035,1285\}$
5945	3473	73	72	0.58	$\{41,87,261,1189\}$

Table B.10: A set of Cyclic non-binary OSMLD codes over $GF(16)$ constructed using the NB-OSMLD-GA algorithm (Construction B)

n	k	d_{min}	J	$r=\frac{k}{n}$	$\Omega_{E(x)}$
7	3	4	3	0.43	$\{3\}$
21	11	6	5	0.52	$\{3,7,14\}$
31	15	6	5	0.48	$\{15\}$
63	37	9	8	0.59	$\{13,21,26,42\}$
73	45	10	9	0.62	$\{25\}$
93	61	11	10	0.66	$\{5,10\}$
217	153	16	15	0.70	$\{37\}$
315	187	13	12	0.59	$\{67,71,134,193\}$
341	205	16	15	0.60	$\{39,55,78\}$
381	255	17	16	0.67	$\{71,127,142,254\}$
399	213	21	20	0.53	$\{133,143,187,256\}$
465	337	21	20	0.72	$\{7,14,28,56\}$
483	249	23	22	0.52	$\{69,105,115,161,230,322\}$
511	303	22	21	0.59	$\{1,63,73\}$
623	333	26	25	0.53	$\{89,91,133\}$
651	491	26	25	0.75	$\{33,35,70\}$
651	443	27	26	0.68	$\{31,35,49,62,70,98\}$
713	367	28	27	0.51	$\{31,155,345\}$
803	559	29	28	0.70	$\{73,121,146,143\}$
837	487	29	28	0.58	$\{31,62,207,414\}$
889	697	29	28	0.78	$\{7,97\}$
993	511	31	30	0.51	$\{83,166\}$
1333	809	30	29	0.61	$\{31,62,215,473,645\}$
	749	35	34	0.56	$\{43,215,217,279,301,645\}$
1359	849	37	36	0.62	$\{17,34,151,302\}$

1387	811	37	36	0.58	{115,173,195,230}
1397	955	32	31	0.68	{33,55,127,254,297}
1533	819	39	38	0.53	{59,118,369,511,513,1022}
	797	40	39	0.52	{171,219,273,281,562}
1617	1191	41	40	0.74	{33,77,147,154,231,294}
1659	869	42	41	0.52	{93,553,1106}
1661	1031	41	40	0.62	{85,151,170,302}
1691	1211	41	40	0.72	{89,178,247,361}
1869	1381	40	39	0.74	{165,445,890}
	1289	43	42	0.69	{27,89,178,801}
1953	1441	31	30	0.74	{73,83}
	1139	42	41	0.58	{45,217,225,434,441}
2047	1123	45	44	0.55	{9,19,185,205}
2117	1531	47	46	0.72	{73,87,146,292,584,725}
2139	1307	40	39	0.61	{155,217,345,713,759,1035,1426}
	1143	45	44	0.53	{69,93,115,230,465,713,759,1426}
2263	1815	46	45	0.80	{229}
2277	1317	49	48	0.58	{99,207,253,345,414,483,495,506}
2359	1345	49	48	0.57	{47,337,395,1011}
2387	1687	49	48	0.71	{31,62,341,363}
2581	1871	51	50	0.72	{89,145,178,261,356,712}
2709	1813	49	48	0.67	{61,122,559,1118}
	1729	52	51	0.64	{129,145,258,290,1161}
2759	2003	53	52	0.73	{89,93,267,445,623,979,1023,1335}
	1737	54	53	0.63	{89,93,155,267,403,445,1335}
2869	2191	49	48	0.76	{95,151,302,323}
	1897	54	53	0.67	{23,41,46,82,92,115,123}
2937	1705	55	54	0.58	{89,99,178,363,429,1089}
2967	1979	51	50	0.67	{215,253,301,483,506,621}

	1505	54	53	0.51	$\{129,207,414,437,483,529,621\}$
3059	1787	52	51	0.58	$\{23,46,209\}$
3069	2557	31	30	0.83	$\{379,431\}$
3069	1927	41	40	0.63	$\{17,34,297,693\}$
3139	1927	56	55	0.61	$\{43,73,146,219,438,559,1075\}$
3171	1867	55	54	0.59	$\{99,151,195,302,327,453\}$
3311	2371	59	58	0.72	$\{297,301,451,473,602,1419\}$
3473	2671	53	52	0.77	$\{151,161,755,851\}$
3479	1739	60	59	0.50	$\{49,213,1491\}$
3707	2605	53	52	0.70	$\{77,337,451,674\}$
3827	2251	59	58	0.59	$\{43,559,623,801,817,1419\}$
3933	2439	59	58	0.62	$\{161,171,322,391,713,855\}$
	2195	61	60	0.56	$\{69,115,138,230,285,399,1311,2622\}$
3937	2087	64	63	0.53	$\{63,155,279,899,1457\}$
4347	3071	63	62	0.71	$\{23,46,189,297\}$
4557	3221	58	57	0.71	$\{155,310,315\}$
4623	2357	69	68	0.51	$\{11,22,1541,3082\}$
4669	3239	68	67	0.69	$\{29,161,322,644,667,1288,2001\}$
4781	3545	70	69	0.74	$\{89,178,2049\}$
4893	3319	67	66	0.68	$\{35,70,1165,1631,2330,3262\}$
	2619	71	70	0.53	$\{233,357,466,567,1165,2330\}$
4991	3551	72	71	0.71	$\{155,759,805,851,2139\}$
5037	3783	72	71	0.75	$\{365,511,621,759,943,1095,1679,1886,3358\}$
5251	3281	70	69	0.62	$\{89,178,295\}$
5313	3809	70	69	0.72	$\{23,46,165,253,506\}$
5451	2705	75	74	0.50	$\{207,237,395,553,1817,3634\}$
5901	3089	76	75	0.52	$\{175,245,843,1967,3934\}$
5963	3531	78	77	0.59	$\{119,238,335\}$

Table B.11: A set of Cyclic non-binary OSMLD codes over $GF(32)$ constructed using the NB-OSMLD-GA algorithm (Construction B)

n	k	d_{min}	J	$r=\frac{k}{n}$	$\Omega_{E(x)}$
7	3	4	3	0.43	$\{3\}$
15	7	5	4	0.47	$\{7\}$
21	11	6	5	0.52	$\{7,9\}$
63	37	9	8	0.59	$\{11,21\}$
73	45	10	9	0.62	$\{25\}$
119	59	12	11	0.50	$\{7,17\}$
195	113	15	14	0.58	$\{1,65\}$
315	195	19	18	0.62	$\{5,45,63,105,135\}$
357	227	20	19	0.64	$\{17,35,119,153\}$
387	231	21	20	0.60	$\{43,63\}$
399	213	21	20	0.53	$\{37,133\}$
483	249	23	22	0.52	$\{21,23,161,207\}$
585	377	25	24	0.64	$\{61,201\}$
657	451	27	26	0.69	$\{9,73,81,219\}$
663	381	27	26	0.57	$\{1,221\}$
765	637	25	24	0.83	$\{127\}$
	421	29	28	0.55	$\{85,117,255,261,357\}$
819	471	28	27	0.57	$\{35,101,351\}$
889	587	28	27	0.66	$\{7,91,127,381,385\}$
945	657	31	30	0.70	$\{9,155\}$
1035	549	33	32	0.53	$\{15,115,483\}$
1071	783	31	30	0.73	$\{11,85\}$
	645	34	33	0.60	$\{63,153,221,255,357,399\}$
1143	729	35	34	0.64	$\{45,127,165,495\}$
1197	767	36	35	0.64	$\{19,171,205,209,399\}$

1235	881	35	34	0.71	$\{19,65,247\}$
1285	833	37	36	0.65	$\{37,109,257\}$
1365	839	34	33	0.61	$\{77,195,223,273,455\}$
	773	37	36	0.56	$\{3,109,125\}$
1449	969	35	34	0.67	$\{21,23,529\}$
1479	915	37	36	0.62	$\{51,551\}$
1533	1125	37	36	0.73	$\{79,257\}$
	881	40	39	0.57	$\{47,219,255,381\}$
1547	863	40	39	0.56	$\{75,289,663\}$
1635	975	41	40	0.60	$\{35,327\}$
1645	827	42	41	0.50	$\{35,47,235,705\}$
1665	929	41	40	0.56	$\{31,777\}$
1677	937	41	40	0.56	$\{129,143,301,559\}$
1691	1211	41	40	0.72	$\{19,89,209\}$
1725	1041	43	42	0.60	$\{25,69,138,207,414,621\}$
1755	1459	37	36	0.83	$\{217\}$
1785	1053	42	41	0.59	$\{49,77,85,119,255,357,609\}$
1869	1227	42	41	0.66	$\{21,63,89,399,623\}$
1935	1409	43	42	0.73	$\{43,135,165,645\}$
	1255	45	44	0.65	$\{21,43,129\}$
1971	1213	43	42	0.62	$\{177,219,227\}$
1985	1013	45	44	0.51	$\{211\}$
2049	1301	45	44	0.63	$\{13,149\}$
2205	1537	46	45	0.70	$\{45,49,357\}$
2261	1547	46	45	0.68	$\{57,119,969\}$
2289	1491	45	44	0.65	$\{33,109,763\}$
2289	1335	48	47	0.58	$\{31,327,545,763\}$
2313	1929	49	48	0.83	$\{283\}$
2331	1369	49	48	0.59	$\{69,333,999,1147\}$

2359	1495	43	42	0.63	$\{35,99\}$
2457	2073	37	36	0.84	$\{355\}$
2555	1459	50	49	0.57	$\{5,65,155,175,205,511\}$
2691	1755	49	48	0.65	$\{115,117,585,667,897\}$
2709	1723	53	52	0.64	$\{63,105,301,473,559,645\}$
2731	1535	53	52	0.56	$\{81,225\}$
2737	1925	53	52	0.70	$\{69,119,391,595,1173\}$
2755	2049	51	50	0.74	$\{19,145,551\}$
2967	2187	53	52	0.74	$\{23,69,129,645,989\}$
3033	2101	51	50	0.69	$\{81,337,1011,1017\}$
3045	1907	56	55	0.63	$\{35,87,145,1015,1305,1421\}$
3051	2181	47	46	0.71	$\{113,135\}$
3115	2271	50	49	0.73	$\{65,89,623\}$
3139	1927	56	55	0.73	$\{73,387,511,731,1075\}$
3213	1969	57	56	0.61	$\{119,255,321,765,1071\}$
3213	2701	43	42	0.84	$\{21,221\}$
3277	1989	57	56	0.61	$\{111,479\}$
3309	2205	59	58	0.67	$\{559\}$
3471	2873	60	59	0.83	$\{39,89,117,623,741,1157\}$
3381	2163	58	57	0.64	$\{69,147,161,483,735,1127,1449\}$
3451	2441	59	58	0.71	$\{119,319,493,1479\}$
3655	2225	59	58	0.61	$\{215,221,595,1247\}$
3723	2529	62	61	0.68	$\{153,187,219,561,657,1275\}$
3913	2831	61	60	0.72	$\{169,301,559,1677\}$
	2279	63	62	0.58	$\{91,129,387,731,817\}$
3915	2457	63	62	0.63	$\{45,145,261,435,1305,1827\}$
4029	2029	64	63	0.50	$\{153,395,711,869\}$
4085	2833	65	64	0.69	$\{95,215,741,817\}$
4095	2291	64	63	0.56	$\{85,327,489,595,607,1755\}$

4199	2737	63	62	0.65	{95,221,247,323}
4251	2497	63	62	0.59	{109,359,763,1417}
4263	3225	61	60	0.76	{87,147,609,1015,1421}
	2763	64	63	0.64	{147,261,609,1015,1421,1827}
4347	2499	67	66	0.57	{189,207,713,759,897,945,1035,1449}
	2793	64	63	0.64	{115,189,207,345,483,621,945,1449}
4361	3103	66	65	0.71	{245,267,441,539,637}
4365	2535	67	66	0.58	{63,291,485,873,2037}
4403	2445	67	66	0.55	{37,85,629,1887}
4515	2939	64	63	0.65	{133,175,387,903,1505,1935}
4515	2661	65	64	0.59	{105,217,735,1075,1505}
4539	2757	68	67	0.61	{89,255,267,445,561,801,969,1513}
4599	2773	69	68	0.60	{371,407,485,803,1533,1679}
4655	2797	68	67	0.60	{95,105,525,665,931,1995}
4669	3239	68	67	0.69	{161,319,667,2001}
4845	2739	71	70	0.57	{85,361,513,817,855,969,1197,1729}
4945	3449	69	68	0.70	{161,215,805,989,1075}
5031	2793	71	70	0.56	{129,215,301,429,559,903,1677}
5035	3869	71	70	0.77	{265,361}
5037	3495	70	69	0.69	{207,219,253,897,1095,1679,1725}
5117	3625	70	69	0.71	{43,459,2193}
5655	3197	75	74	0.57	{117,319,377,493,609,1885,2639}
5565	3981	68	67	0.72	{53,105,2385}
	3327	75	74	0.60	{399,795,901,2385,2597}

Table B.12: A set of Cyclic non-binary OSMLD codes over $GF(64)$ constructed using the NB-OSMLD-GA algorithm (Construction B)

n	k	d_{min}	J	$r=\frac{k}{n}$	$\Omega_{E(x)}$
93	61	11	10	0.66	$\{5,10\}$
217	153	16	15	0.70	$\{27,54,108\}$
341	205	16	15	0.60	$\{79,141,165\}$
345	173	20	19	0.50	$\{15,23,46,161,253\}$
381	255	17	16	0.67	$\{7,14,127,254\}$
465	337	21	20	0.72	$\{23,46\}$
635	403	26	25	0.63	$\{55,75,127,254,255\}$
713	367	28	27	0.51	$\{31,155,253\}$
775	439	26	25	0.57	$\{33,66,125\}$
979	593	22	21	0.61	$\{89,99,178\}$
1173	723	31	30	0.62	$\{51,253,255,506\}$
1271	695	36	35	0.55	$\{53,106,287,451,615\}$
1271	771	31	30	0.61	$\{123,143,286,615\}$
1333	809	30	29	0.61	$\{31,62,129,215,301\}$
	749	35	34	0.56	$\{31,62,129,215,473,645\}$
1335	703	38	37	0.53	$\{45,55,110,623,979\}$
1397	955	32	31	0.68	$\{127,143,254,297,517\}$
1457	739	39	38	0.51	$\{155,235,329,517\}$
1513	1079	39	38	0.71	$\{85,89,153,178,267,534\}$
1581	1069	41	40	0.68	$\{35,70\}$
1705	1065	41	40	0.62	$\{73,121,143,146\}$
1725	1037	43	42	0.60	$\{23,46,125,175\}$
1905	1113	43	42	0.58	$\{77,154,165,225\}$
1985	1013	45	44	0.51	$\{17,34\}$
2139	1427	33	32	0.67	$\{31,62,391,782\}$

	1079	47	46	0.50	{253,299,391,465,483,529,782,1058}
2225	1607	47	46	0.72	{75,89,178,445,825,890}
2263	1815	46	45	0.80	{227,235,470}
2607	1307	52	51	0.50	{79,99,158,198,396,869,1738}
2635	2043	46	45	0.78	{63,126,1275}
	1977	51	50	0.75	{59,85,118,1275}
2697	1471	51	50	0.55	{155,203,261,310,406,899,957,1798}
	2199	46	45	0.82	{87,93,186,493,899,986,1798}
2759	2003	53	52	0.73	{31,89,267,341,445,623,979,1335}
	1737	54	53	0.63	{89,155,341,445,589,623,1335}
2829	1897	54	53	0.67	{23,46,205,287,615}
2967	1977	53	52	0.67	{43,69,86,138,437,529,989,1978}
3005	1801	55	54	0.60	{125,145,601,1202}
3175	1739	56	55	0.55	{105,127,185,254,575}
3191	1815	56	55	0.57	{357}
3503	2293	44	43	0.65	{93,186,339,565,791}
3565	2381	58	57	0.67	{115,155,345,529,667,775,805}
	1963	60	59	0.55	{31,62,115,805,1725}
3655	2375	57	56	0.65	{81,162}
3827	2063	62	61	0.54	{215,267,534,623,801,817,1419}
3937	2739	57	56	0.70	{155,179,837,1457}
	2485	62	61	0.63	{93,181,279,889,1457}
3995	2965	63	62	0.74	{47,85,94,425,1363,2679}
	2003	64	63	0.50	{85,235,423,470,611,705,846,893,1363,1410,2679}
4301	2875	63	62	0.67	{187,207,299,935}
4405	2369	60	59	0.54	{325,881,1762}
4495	3077	48	47	0.68	{341,435,682,899,1015,1798,2175}
	2547	64	63	0.57	{87,145,174,217,434,435,725}
4715	1809	67	66	0.60	{69,138,205,207,414,943,1025,1886}
4867	2937	68	67	0.60	{157,279,465,1099,2355}

4895	2569	70	69	0.52	$\{55,143,286,445,890,979,1958\}$
4929	2479	70	69	0.50	$\{53,106,341,589,795,1643,3286\}$
4945	3449	69	68	0.70	$\{115,215,230,897,989,1075,1081,1978\}$
	3155	70	69	0.64	$\{115,215,230,473,301\}$
4991	3231	71	70	0.65	$\{69,105,138,276\}$
5251	3281	70	69	0.62	$\{89,178,1947\}$
5429	3393	72	71	0.62	$\{89,178,356,445,671,712,890\}$
5451	2749	75	74	0.50	$\{79,158,207,237,414,828,1817,3634\}$
5969	3907	75	74	0.65	$\{127,235,635,987,1081,1457\}$

Appendix C

Error Probability Bound of the WOSMLGD algorithm

In this appendix, we will present the analytical error probability of the Weighted One-Step Majority-Logic Decoder (WOSMLGD) algorithm, already presented in Chapter 4, Algorithm 4.1. This analytical formulation is based on the paper published in [163].

For an uncoded transmission over an AWGN channel with a BPSK modulation, the bit error probability p is given by:

$$p = Q(\sqrt{2E_s/N_0}) \tag{C.1}$$

where $Q(.)$ is the complementary error function, which takes the following form:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} \exp(-\frac{1}{2}t^2)dt \tag{C.2}$$

By assuming that a OSMLD code has J orthogonal equations on each symbol digit, the estimation of each orthogonal estimator is erroneous when the number of errors occurring in the symbols that participate in it is odd. Then by considering that each estimator has J symbols participating in it, the error probability p^* of an estimator is given by:

$$p^* = \sum_{i=0}^{[(J-1)/2]} \begin{pmatrix} J \\ 2i+1 \end{pmatrix} p^{2i+1}(1-p)^{J-(2i+1)} \tag{C.3}$$

The decision of each estimator is treated as if it came from independent antipodal signals. The required energy E_s^* for each signal is given by solving the following equation:

$$p^* = Q(\sqrt{2E_s^*/N_0}) \tag{C.4}$$

Thus, the optimum decision rule for all the estimators is equivalent to a binary antipodal signal of energy:

$$E = E_s + JE_s^* \tag{C.5}$$

Therefore, the expression of the bit error probability P_b of the WOSMLGD algorithm takes the following form:

$$P_b = Q(\sqrt{2(E_s + JE_s^*)/N_0}) \tag{C.6}$$

The block error probability P_e of the WOSMLGD algorithm can be approximated by:

$$P_e \approx \frac{n}{d_{min}} P_b \tag{C.7}$$

where n is the code length and d_{min} is its minimum distance.

Nomenclature

3GPP	3rd Generation Partnership Project
4G LTE	Fourth Generation Long Term Evolution
5G NR	Fifth Generation New Radio
ACE	Approximate Cycle Extrinsic message degree
Adagrad	Adaptive Gradient
ALMM	Augmented Lagrange Method of Multipliers
APP	A Posteriori Probability
AR	Augmented Reality
AWGN	Additive White Gaussian Noise
BCH	Bose-Chaudhuri-Hocquenghem
BER	Bit Error rate
BG	Base Graph
BIBD	Balanced Incomplete Block Design
BLER	Block Error Rate
BP-MS	Belief-Propagation Min-Sum
BP-SP	Belief Propagation Sum-Product
CDMA	Code Division Multiple Access

CN	Check Node
CRC	Cyclic Redundancy Check
CRC-SCL	CRC-aided Successive Cancellation List
cV2X	Cellular Vehicle-to-Everything
DS	Difference-Set
DSC	Difference-Set Code
ECC	Error Correcting Codes
EG	Euclidean Geometry
eMBB	Enhanced Mobile Broadband
GD-MLGD	Gradient-Descent Majority-Logic Decoding
GDBF	Gradient-Descent Bit-Flipping
GSM	Global System for Mobile Communications
HARQ	Hybrid Automatic Repeat Request
HDPC	High-Density Parity-Check
HIHO	Hard-Input Hard-Output
HRBI-MLGD	Hard Reliability-Based Majority-Logic Decoder
HSDPA	High Speed Download Packet Access
HSPA	High Speed Packet Access
IDA	Iterative Decoding Algorithm
IoT	Internet of Things
IR-HARQ	Incremental Redundancy Hybrid Automatic Repeat Request
ISRBI-MLGD	Improved Soft Reliability-Based Iterative Majority-Logic Decoder
LDPC	Low-Density Parity-Check

LLR	Log-Likelihood Ratio
LP	Linear Programming
MC-M2M	Mission-Critical Machine-to-Machine
MDPC	Moderate-Density Parity-Check
MGD	Momentum Gradient-Descent
MGD	Momentum Gradient-Descent
MIMO	Multiple Input Multiple Output
MLGD	Majority-Logic Decodable
mMTC	Massive Machine-Type Communications
MS	Mattson-Solomon
MSMLD	Multi-Step Majority-Logic Decodable
MTD	Multi-Threshold Decoder
NAG	Nesterov Accelerated Gradient
NGDBF	Noisy Gradient-Descent Bit-Flipping
OFDM	Orthogonal Frequency Divsion Multiplexing
OSD	Ordered Statistic Decoding
OSMLD	One-Step Majority-Logic Decodable
OSMLGD	One-Step Majority-Logic Decoder
PBDs	Pairwise Block Designs
PBIBDs	Partially Balanced Incomplete Block Designs
PEG	Progressive-Edge-Growth
PG	Projective Geometry
QAM	Quadrature Amplitude Modulation

QGD-MLGD	Quantized Gradient-Descent Majority-Logic Decoding
QPSK	Quadrature Phase Shift-Key
RC	Row-Column
RM	Reed-Muller
SC-FDMA	Single Carrier Frequency Division Multiple Access
SDN	Software-Defined Network
SIHO	Soft-Input Hard-Output
SNR	Signal to Noise Ratio
SOQC	Self-Orthogonal Quasi-Cyclic
SRBI-MLGD	Soft Reliability-Based Iterative Majority-Logic Decoder
STS	Steiner Triple System
t-CWP	t-bounded Cosets Weight Problem
TDD	Time Division Duplex
UMTS	Universal Mobile Telecommunications System
URLLC	Ultra Reliable Low Latency Communications
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VN	Variable Node
VR	Virtual Reality
WOSMLGD	Weighted One-Step Majority-Logic Decoder

Bibliography

[1] Genetic sequencing and synthesizing technologies. https://www.illumina.com/.

[2] Qualcomm incorporated. https://www.qualcomm.com/.

[3] Vision for latency reduction in future wireless communications. https://www.accelercomm.com/.

[4] 3GPP TS 38.212. Nr; multiplexing and channel coding. *3rd Generation Partnership Project; Technical Specification Group Radio Access Network*, 2017.

[5] I. Adjudeanu, J-Y. Chouinard, and P. Fortier. On the correlation between error weights and syndrome weights for belief propagation decoding of ldpc codes. In *2009 11th Canadian Workshop on Information Theory*, pages 36–41. IEEE, 2009.

[6] B. Ammar, B. Honary, Y. Kou, J. Xu, and S. Lin. Construction of low-density parity-check codes based on balanced incomplete block designs. *IEEE Transactions on Information Theory*, 50(6):1257–1269, 2004.

[7] G. Ananda, E. Walsh, K. D. Jacob, M. Krasilnikova, K. A. Eckert, F. Chiaromonte, and K. D. Makova. Distinct mutational behaviors differentiate short tandem repeats from microsatellites in the human genome. *Genome biology and evolution*, 5(3):606–620, 2012.

[8] 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, 2009.

[9] S. Barman, X. Liu, S. C. Draper, and B. Recht. Decomposition methods for large scale lp decoding. *IEEE Transactions on Information Theory*, 59(12):7870–7886, 2013.

[10] M. Belkasmi, M. Lahmer, and F. Ayoub. Iterative threshold decoding of product codes constructed from majority logic decodable codes. In *2006 2nd International Conference on Information and Communication Technologies*, volume 2, pages 2376–2381. IEEE, 2006.

[11] M. Belkasmi, M. Lahmer, and M. Benchrifa. Iterative threshold decoding of parallel concatenated block codes. In *4th International Symposium on Turbo Codes and Related Topics; 6th International ITG-Conference on Source and Channel Coding*, pages 1–4. VDE, 2006.

[12] E. Berlekamp, R. McEliece, and H. Van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[13] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo-codes. In *Proceedings of ICC'93-IEEE International Conference on Communications*, volume 2, pages 1064–1070. IEEE, 1993.

[14] D. P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[15] A. Bhowmick and S. Lovett. The list decoding radius for reed–muller codes over small fields. *IEEE Transactions on Information Theory*, 64(6):4382–4391, 2018.

[16] M. Blawat, K. Gaedke, I. Hütter, X-M. Chen, B. Turczyk, S. Inverso, B. W. Pruitt, and G. M. Church. Forward error correction for dna data storage. *Procedia Computer Science*, 80:1011–1022, 2016.

[17] C. Bockelmann, N. Pratas, H. Nikopour, K. Au, T. Svensson, C. Stefanovic, P. Popovski, and A. Dekorsy. Massive machine-type communications in 5g: Physical and mac-layer solutions. *IEEE Communications Magazine*, 54(9):59–65, 2016.

[18] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss. A dna-based archival storage system. *ACM SIGOPS Operating Systems Review*, 50(2):637–649, 2016.

[19] P. Camion. A proof of some properties of reed-muller codes by means of the normal basis theorem. *Combinatorial mathematics and its applications, University of North Carolina at Chapel Hill*, 1969.

[20] T. C-Y. Chang and Y. T. Su. Dynamic weighted bit-flipping decoding algorithms for ldpc codes. *IEEE Transactions on Communications*, 63(11):3950–3963, 2015.

[21] C. L. Chen. On majority-logic decoding of finite geometry codes. *IEEE Transactions on Information Theory*, 17(3):332–336, 1971.

[22] M. Chertkov and M. G. Stepanov. An efficient pseudocodeword search algorithm for linear programming decoding of ldpc codes. *IEEE Transactions on Information Theory*, 54(4):1514–1520, 2008.

[23] S. Chowla and H. J. Ryser. Combinatorial problems. *Canadian Journal of Mathematics*, 2:93–99, 1950.

[24] S-Y. Chung, G. D. Forney, T. Richardson, R. L. Urbanke, et al. On the design of low-density parity-check codes within 0.0045 db of the shannon limit. *IEEE Communications letters*, 5(2):58–60, 2001.

[25] G. M. Church, Y. Gao, and S. Kosuri. Next-generation digital information storage in dna. *Science*, page 1226355, 2012.

[26] G. C. Clark Jr. and J. B. Cain. *Error-correction coding for digital communications*. Springer Science and Business Media, 2013.

[27] C. J. Colbourn. *CRC handbook of combinatorial designs*. CRC press, 2010.

[28] R. Cramer, C. Xing, and C. Yuan. Efficient multi-point local decoding of reed-muller codes via interleaved codex. *IEEE Transactions on Information Theory*, 66(1):263–272, 2019.

[29] L. Dai, B. Wang, Y. Yuan, S. Han, I. Chih-Lin, and Z. Wang. Non-orthogonal multiple access for 5g: solutions, challenges, opportunities, and future research trends. *IEEE Communications Magazine*, 53(9):74–81, 2015.

[30] M. C. Davey and D. J. C. MacKay. Low density parity check codes over gf (q). In *1998 Information Theory Workshop (Cat. No. 98EX131)*, pages 70–71. IEEE, 1998.

[31] C. Di, D. Proietti, I. E. Telatar, T. Richardson, and R. L. Urbanke. Finite-length analysis of low-density parity-check codes on the binary erasure channel. *IEEE Transactions on Information theory*, 48(6):1570–1579, 2002.

[32] C. Ding, C. Li, and Y. Xia. Another generalisation of the binary reed–muller codes and its applications. *Finite Fields and Their Applications*, 53:144–174, 2018.

[33] D. Divsalar, S. Dolinar, and C. Jones. Construction of protograph ldpc codes with linear minimum distance. In *2006 IEEE International Symposium on Information Theory*, pages 664–668. IEEE, 2006.

[34] D. Divsalar, S. Dolinar, C. R. Jones, and K. Andrews. Capacity-approaching protograph codes. *IEEE Journal on Selected Areas in Communications*, 27(6):876–888, 2009.

[35] K. Drakakis. A review of the available construction methods for golomb rulers. *Adv. in Math. of Comm.*, 3(3):235–250, 2009.

[36] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[37] G. Durisi, T. Koch, J. Östman, Y. Polyanskiy, and W. Yang. Short-packet communications over multiple-antenna rayleigh-fading channels. *IEEE Transactions on Communications*, 64(2):618–629, 2015.

[38] O. El Mouaatamid, M. Lahmer, and M. Belkasmi. Construction and decoding of osmld codes derived from unital and oval designs. In *2018 International Conference on Advanced Communication Technologies and Networking (CommNet)*, pages 1–7. IEEE, 2018.

[39] O. El Mouaatamid, M. Lahmer, M. Belkasmi, Z. M'rabet, and A. Yatribi. One-step majority-logic decodable codes derived from oval designs. In *2017 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pages 1–6. IEEE, 2017.

[40] Y. Erlich and D. Zielinski. Capacity-approaching dna storage. *bioRxiv*, page 074237, 2016.

[41] Y. Erlich and D. Zielinski. Dna fountain enables a robust and efficient storage architecture. *Science*, 355(6328):950–954, 2017.

[42] N. Eroshenko, S. Kosuri, A. H. Marblestone, N. Conway, and G. M. Church. Gene assembly from chip-synthesized oligonucleotides. *Current protocols in chemical biology*, 4(1):1–17, 2012.

[43] B. C. Faircloth and T. C. Glenn. Not all sequence tags are created equal: designing and validating sequence identification tags robust to indels. *PloS one*, 7(8):e42543, 2012.

[44] J. Feldman. *Decoding error-correcting codes via linear programming*. PhD thesis, Massachusetts Institute of Technology, 2003.

[45] J. Feldman, M. J. Wainwright, and D. R. Karger. Using linear programming to decode binary linear codes. *IEEE Transactions on Information Theory*, 51(3):954–972, 2005.

[46] W. Feller. *An introduction to probability theory and its applications*, volume 1. Wiley, New York, 1968.

[47] R. A. Fisher. An examination of the different possible solutions of a problem in incomplete blocks. *Annals of Eugenics*, 10(1):52–75, 1940.

[48] R. Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1):21–28, 1962.

[49] J-M. Goethals and P. Delsarte. On a class of majority-logic decodable cyclic codes. *IEEE Transactions on Information Theory*, 14(2):182–188, 1968.

[50] M. J. E. Golay. Notes on digital coding. *Proc. IEEE*, 37:657, 1949.

[51] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney. Towards practical, high-capacity, low-maintenance information storage in synthesized dna. *Nature*, 494(7435):77, 2013.

[52] R. L. Graham and J. MacWilliams. On the number of information symbols in difference-set cyclic codes. *Bell System Technical Journal*, 45(7):1057–1070, 1966.

[53] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark. Robust chemical preservation of digital information on dna in silica with error-correcting codes. *Angewandte Chemie International Edition*, 54(8):2552–2555, 2015.

[54] F. Guo and L. Hanzo. Reliability ratio based weighted bit-flipping decoding for low-density parity-check codes. *Electronics Letters*, 40(21):1356–1358, 2004.

[55] V. Guruswami, L. Jin, and C. Xing. Efficiently list-decodable punctured reed-muller codes. *IEEE Transactions on Information Theory*, 63(7):4317–4324, 2017.

[56] J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE Transactions on information theory*, 42(2):429–445, 1996.

[57] Marshall Hall. *Combinatorial Theory (2nd Ed.)*. John Wiley & Sons, Inc., USA, 1998.

[58] R. W. Hamming. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160, 1950.

[59] S. A. Hashemi, N. Doan, M. Mondelli, and W. J. Gross. Decoding reed-muller and polar codes by successive factor graph permutations. In *2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC)*, pages 1–5. IEEE, 2018.

[60] P. Hauck, M. Huber, J. Bertram, D. Brauchle, and S. Ziesche. Efficient majority-logic decoding of short-length reed-muller codes at information positions. *IEEE transactions on communications*, 61(3):930–938, 2013.

[61] D. Heider and A. Barnekow. Dna-based watermarks using the dna-crypt algorithm. *BMC bioinformatics*, 8(1):176, 2007.

[62] M. R. Hestenes. Multiplier and gradient methods. *Journal of optimization theory and applications*, 4(5):303–320, 1969.

[63] J. H. Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[64] R. Horan, C. Tjhai, M. Tomlinson, M. Ambroze, and M. Ahmed. Idempotents, mattson-solomon polynomials and binary ldpc codes. *IEE Proceedings-Communications*, 153(2):256–262, 2006.

[65] F. Hu. *Opportunities in 5G networks: A research and development perspective*. CRC press, 2016.

[66] X-Y. Hu, E. Eleftheriou, and D-M. Arnold. Progressive edge-growth tanner graphs. In *GLOBECOM'01. IEEE Global Telecommunications Conference*, volume 2, pages 995–1001. IEEE, 2001.

[67] X-Y. Hu, E. Eleftheriou, and D-M. Arnold. Regular and irregular progressive edge-growth tanner graphs. *IEEE Transactions on Information Theory*, 51(1):386–398, 2005.

[68] Q. Huang, Q. Diao, S. Lin, and K. Abdel-Ghaffar. Cyclic and quasi-cyclic ldpc codes on constrained parity-check matrices and their trapping sets. *IEEE transactions on information theory*, 58(5):2648–2671, 2012.

[69] Q. Huang, J. Kang, L. Zhang, S. Lin, and K. Abdel-Ghaffar. Two reliability-based iterative majority-logic decoding algorithms for ldpc codes. *IEEE Transactions on communications*, 57(12):3597–3606, 2009.

[70] J. R. H. Hutton. *LDPC codes from semipartial geometries*. PhD thesis, University of Sussex, 2011.

[71] S. Jain, F. Farnoud, M. Schwartz, and J. Bruck. Duplication-correcting codes for data storage in the dna of living organisms. In *Information Theory (ISIT), 2016 IEEE International Symposium on*, pages 1028–1032. IEEE, 2016.

[72] M. Jiang, C. Zhao, Z. Shi, and Y. Chen. An improvement on the modified weighted bit flipping decoding algorithm for ldpc codes. *IEEE Communications Letters*, 9(9):814–816, 2005.

[73] M. Jiang, C. Zhao, L. Zhang, and E. Xu. Adaptive offset min-sum algorithm for low-density parity check codes. *IEEE communications letters*, 10(6):483–485, 2006.

[74] X. Jiang and M. H. Lee. Large girth non-binary ldpc codes based on finite fields and euclidean geometries. *IEEE Signal Processing Letters*, 16(6):521–524, 2009.

[75] S. Johnson and S. R. Weller. High-rate ldpc codes from unital designs. In *GLOBECOM'03. IEEE Global Telecommunications Conference (IEEE Cat. No. 03CH37489)*, volume 4, pages 2036–2040. IEEE, 2003.

[76] S. Johnson and S. R. Weller. Codes for iterative decoding from partial geometries. *IEEE Transactions on Communications*, 52(2):236–243, 2004.

[77] J. Jung and I-C. Park. Multi-bit flipping decoding of ldpc codes for nand storage systems. *IEEE Communications Letters*, 21(5):979–982, 2017.

[78] T. Kasami and S. Lin. On majority-logic decoding for duals of primitive polynomial codes. *IEEE Transactions on Information Theory*, 17(3):322–331, 1971.

[79] T. Kasami, S. Lin, and W. W. Peterson. Some results on cyclic codes which are invariant under the affine group and their applications. *Information and Control*, 11(5-6):475–496, 1967.

[80] T. Kasami, S. Lin, and W. W. Peterson. New generalizations of the reed-muller codes–i: Primitive codes. *IEEE Transactions on Information Theory*, 14(2):189–199, 1968.

[81] N. Kashyap and A. Vardy. Stopping sets in codes from designs. In *IEEE International Symposium on Information Theory, 2003. Proceedings.*, page 122. IEEE, 2003.

[82] A. Khajehnejad, A. G. Dimakis, B. Hassibi, B. Vigoda, and W. Bradley. Reweighted lp decoding for ldpc codes. *IEEE transactions on information theory*, 58(9):5972–5984, 2012.

[83] V. D. Kolesnik. Probabilistic decoding of majority codes. *Problemy Peredachi Informatsii*, 7(3):3–12, 1971.

[84] V. D. Kolesnik and E. T. Mironchikov. Cyclic reed–muller codes and their decoding. *Problemy Peredachi Informatsii*, 4(4):20–25, 1968.

[85] S. Kosuri, N. Eroshenko, E. M. LeProust, M. Super, J. Way, J. B. Li, and G. M. Church. Scalable gene synthesis by selective amplification of dna pools from high-fidelity microchips. *Nature biotechnology*, 28(12):1295, 2010.

[86] Y. Kou, S. Lin, and M. P. C. Fossorier. Low-density parity-check codes based on finite geometries: a rediscovery and new results. *IEEE Transactions on Information theory*, 47(7):2711–2736, 2001.

[87] S. Kudekar, S. Kumar, M. Mondelli, H. D. Pfister, E. ŞaşoÇ§lu, and R. L. Urbanke. Reed–muller codes achieve capacity on erasure channels. *IEEE Transactions on information theory*, 63(7):4298–4316, 2017.

[88] M. Lahmer, M. Belkasmi, and F. Ayoub. Iterative threshold decoding of one step majority logic decodable block codes. In *2007 IEEE International Symposium on Signal Processing and Information Technology*, pages 668–673. IEEE, 2007.

[89] S. Landner and O. Milenkovic. Algorithmic and combinatorial analysis of trapping sets in structured ldpc codes. In *2005 International Conference on Wireless Networks, Communications and Mobile Computing*, volume 1, pages 630–635. IEEE, 2005.

[90] G. Li and G. Feng. Improved parallel weighted bit-flipping decoding algorithm for ldpc codes. *IET communications*, 3(1):91–99, 2009.

[91] H. Li, B. Bai, X. Mu, J. Zhang, and H. Xu. Algebra-assisted construction of quasi-cyclic ldpc codes for 5g new radio. *IEEE Access*, 6:50229–50244, 2018.

[92] J. Li, S. Lin, K. Abdel-Ghaffar, D. J. Costello Jr, and W. E. Ryan. *LDPC code designs, constructions, and unification*. Cambridge University Press, 2016.

[93] J. Li, K. Liu, S. Lin, and K. Abdel-Ghaffar. Algebraic quasi-cyclic ldpc codes: Construction, low error-floor, large girth and a reduced-complexity decoding scheme. *IEEE Transactions on communications*, 62(8):2626–2637, 2014.

[94] R. Liebler. Implementing gradient descent decoding. *The Michigan Mathematical Journal*, 58(1):285–291, 2009.

[95] D. Limbachiya, V. Dhameliya, M. Khakhar, and M. K. Gupta. On optimal family of codes for archival dna storage. *arXiv preprint arXiv:1501.07133*, 2015.

[96] S. Lin. Multifold euclidean geometry codes. *IEEE Transactions on Information Theory*, 19(4):537–548, 1973.

[97] S. Lin and D. J. Costello. *Error control coding*. Pearson Education India, 2001.

[98] S. Lin and D. J. Costello. Error control coding. *The second international edition, Prentice-Hall*, pages 704–712, 2004.

[99] B. Liu, J. Gao, G. Dou, and W. Tao. Weighted symbol-flipping decoding for nonbinary ldpc codes. In *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, volume 1, pages 223–226. IEEE, 2010.

[100] B. Liu, J. Gao, W. Tao, and G. Dou. Weighted symbol-flipping decoding algorithm for nonbinary ldpc codes with flipping patterns. *Journal of Systems Engineering and Electronics*, 22(5):848–855, 2011.

[101] X. Liu and S. C. Draper. Admm lp decoding of non-binary ldpc codes in $gf(2^m)$. *IEEE Transactions on Information Theory*, 62(6):2985–3010, 2016.

[102] X. Liu and S. C. Draper. The admm penalized decoder for ldpc codes. *IEEE Transactions on Information Theory*, 62(6):2966–2984, 2016.

[103] Y-H. Liu, X-L. Niu, and M-L. Zhang. Multi-threshold bit flipping algorithm for decoding structured ldpc codes. *IEEE Communications Letters*, 19(2):127–130, 2014.

[104] G. Liva and M. Chiani. Protograph ldpc codes design based on exit analysis. In *IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference*, pages 3250–3254. IEEE, 2007.

[105] R. Lucas, M. Bossert, and M. Breitbach. On iterative soft-decision decoding of linear binary block codes and product codes. *IEEE Journal on selected areas in communications*, 16(2):276–296, 1998.

[106] R. Lucas, M. P. C. Fossorier, Y. Kou, and S. Lin. Iterative decoding of one-step majority logic decodable codes based on belief propagation. *IEEE Transactions on Communications*, 48(6):931–937, 2000.

[107] D. J. C. MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[108] D. J. C. MacKay and M. C. Davey. Evaluation of gallager codes for short block length and high rate applications. In *Codes, Systems, and Graphical Models*, pages 113–130. Springer, 2001.

[109] D. J. C. MacKay and R. M. Neal. Near shannon limit performance of low density parity check codes. *Electronics letters*, 32(18):1645–1646, 1996.

[110] F. J. MacWilliams and H. B. Mann. On the p-rank of the design matrix of a difference set. *Information and Control*, 12(5):474–488, 1968.

[111] F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes*, volume 16. Elsevier, 1977.

[112] J. F. MacWilliams. A table of primitive binary idempotents of odd length n, 7 {leq} n {leq} 511. *IEEE Transactions on Information Theory*, 25(1):118–121, 1979.

[113] J. L. Massey. Threshold decoding. 1963.

[114] R. G. Maunder. A vision for 5g channel coding. *AccelerComm White Paper*, 2016.

[115] M. Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[116] T. Mittelholzer. Construction of steiner systems and high-rate ldpc codes. *IEEE Trans. Inform. Theory*, 2000.

[117] T. K. Moon. *Error correction coding: mathematical methods and algorithms*. John Wiley & Sons, 2005.

[118] D. E. Muller. Application of boolean algebra to switching circuit design and to error detection. *Transactions of the IRE professional group on electronic computers*, (3):6–12, 1954.

[119] Y. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/k$\hat{2}$). In *Doklady an ussr*, volume 269, pages 543–547, 1983.

[120] T. M. N. Ngatched, A. S. Alfa, and J. Cai. An improvement on the soft reliability-based iterative majority-logic decoding algorithm for ldpc codes. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–5. IEEE, 2010.

[121] D. V. Nguyen and B. Vasic. Two-bit bit flipping algorithms for ldpc codes and collective error correction. *IEEE Transactions on Communications*, 62(4):1153–1163, 2014.

[122] D. V. Nguyen, B. Vasić, and M. W. Marcellin. Two-bit bit flipping decoding of ldpc codes. In *2011 IEEE International Symposium on Information Theory Proceedings*, pages 1995–1999. IEEE, 2011.

[123] T. T. B. Nguyen, T. Nguyen Tan, and H. Lee. Efficient qc-ldpc encoder for 5g new radio. *Electronics*, 8(6):668, 2019.

[124] K. Niu and K. Chen. Crc-aided decoding of polar codes. *IEEE Communications Letters*, 16(10):1668–1671, 2012.

[125] J. Oh and J. Ha. A two-bit weighted bit-flipping decoding algorithm for ldpc codes. *IEEE Communications Letters*, 22(5):874–877, 2018.

[126] A. Orlitsky, R. Urbanke, K. Viswanathan, and J. Zhang. Stopping sets and the girth of tanner graphs. In *Proceedings IEEE International Symposium on Information Theory,*, page 2. IEEE, 2002.

[127] V. Pepe. Ldpc codes from the hermitian curve. *Designs, Codes and Cryptography*, 42(3):303–315, 2007.

[128] W. W. Peterson and E.J. Weldon. *Error-correcting codes*. MIT press, 1972.

[129] M. J. D. Powell. A method for nonlinear constraints in minimization problems. *Optimization*, pages 283–298, 1969.

[130] G. K. Prayogo, R. Putra, A. H. Prasetyo, and M. Suryanegara. Evaluation of ldpc code and polar code coding scheme in 5g technology–massive machine type communication. In *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 170–174. IEEE, 2018.

[131] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[132] M. B. Quintana, M. A. B. Trenard, I. Márquez-Corbella, and E. Martínez-Moro. An algebraic view to gradient descent decoding. In *2010 IEEE Information Theory Workshop*, pages 1–4. IEEE, 2010.

[133] I. S. Reed. A class of multiple-error-correcting codes and the decoding scheme. Technical report, MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 1953.

[134] D. Reinsel, J. Gantz, and J. Rydning. Data age 2025: The evolution of data to life-critical. *Don't Focus on Big Data*, 2017.

[135] T. Richardson. Error floors of ldpc codes. In *Proceedings of the annual Allerton conference on communication control and computing*, volume 41, pages 1426–1435. The University; 1998, 2003.

[136] T. Richardson and S. Kudekar. Design of low-density parity check codes for 5g new radio. *IEEE Communications Magazine*, 56(3):28–34, 2018.

[137] T. Richardson, M. A. Shokrollahi, and R. L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE transactions on information theory*, 47(2):619–637, 2001.

[138] T. Richardson and R. Urbanke. Multi-edge type ldpc codes. In *Workshop honoring Prof. Bob McEliece on his 60th birthday, California Institute of Technology, Pasadena, California*, pages 24–25, 2002.

[139] K. Rkizat, S. Nouh, M. Lahmer, and M. Belkasmi. New quasi-cyclic majority logic codes constructed from disjoint difference sets by genetic algorithm. In *First International Conference on Real Time Intelligent Systems*, pages 168–177. Springer, 2017.

[140] M. G. Ross, C. Russ, M. Costello, A. Hollinger, N. J. Lennon, R. Hegarty, C. Nusbaum, and D. B. Jaffe. Characterizing and measuring bias in sequence data. *Genome biology*, 14(5):R51, 2013.

[141] S. Ruder. An overview of gradient descent optimization algorithms [eb/ol]. 2017.

[142] L. Rudolph. A class of majority logic decodable codes. *IEEE Transactions on Information Theory*, 13(2):305–307, 1967.

[143] L. Rudolph. Threshold decoding of cyclic codes. *IEEE Transactions on Information Theory*, 15(3):414–418, 1969.

[144] L. Rudolph and C. Hartmann. Decoding by sequential code reduction. *IEEE Transactions on Information Theory*, 19(4):549–555, 1973.

[145] L. D. Rudolph. *Geometric configurations and majority logic decodable codes*. PhD thesis, MEE-University of Oklahoma, 1964.

[146] W. Ryan and S. Lin. *Channel codes: classical and modern*. Cambridge university press, 2009.

[147] H. J. Ryser. The existence of symmetric block designs. *Journal of Combinatorial Theory, Series A*, 32(1):103–105, 1982.

[148] E. Santi, C. Hager, and H. D. Pfister. Decoding reed-muller codes using minimum-weight parity checks. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 1296–1300. IEEE, 2018.

[149] R. Saptharishi, A. Shpilka, and B. L. Volk. Efficiently decoding reed-muller codes from random errors. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 227–235, 2016.

[150] O. Sberlo and A. Shpilka. On the performance of reed-muller codes with respect to random errors and erasures. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1357–1376. SIAM, 2020.

[151] F. Schaich and T. Wild. Waveform contenders for 5g ofdm vs. fbmc vs. ufmc. In *2014 6th international symposium on communications, control and signal processing (ISCCSP)*, pages 457–460. IEEE, 2014.

[152] J. J. Schwartz, C. Lee, and J. Shendure. Accurate gene synthesis with tag-directed retrieval of sequence-verified dna molecules. *Nature methods*, 9(9):913, 2012.

[153] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[154] M. Shirvanimoghaddam, M. S. Mohammadi, R. Abbas, A. Minja, C. Yue, B. Matuz, G. Han, Z. Lin, W. Liu, Y. Li, et al. Short block-length codes for ultra-reliable low latency communications. *IEEE Communications Magazine*, 57(2):130–137, 2018.

[155] J. Singer. A theorem in finite projective geometry and some applications to number theory. *Transactions of the American Mathematical Society*, 43(3):377–385, 1938.

[156] K. J. C. Smith. Majority decodable codes derived from finite geometries. *Institute of Statistics Minieo Series, University of North Carolina*, 561, 1967.

[157] K. J. C. Smith. *An application of incomplete block designs to the construction of error-correcting codes*, volume 42. Citeseer, 1968.

[158] Y. Sun, H. Chen, X. Li, L. Luo, and T. Qin. Reliability-based iterative proportionality-logic decoding of ldpc codes with adaptive decision. *Journal of Communications and Networks*, 17(3):213–220, 2015.

[159] G. Sundararajan, C. Winstead, and E. Boutillon. Noisy gradient descent bit-flip decoding for ldpc codes. *IEEE Transactions on Communications*, 62(10):3385–3400, 2014.

[160] R. Sutton. Two problems with back propagation and other steepest descent learning procedures for networks. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 1986*, pages 823–832, 1986.

[161] M. Sybis, K. Wesolowski, K. Jayasinghe, V. Venkatasubramanian, and V. Vukadinovic. Channel coding for ultra-reliable low-latency communication in 5g systems. In *2016 IEEE 84th vehicular technology conference (VTC-Fall)*, pages 1–5. IEEE, 2016.

[162] M. H. Taghavi N. and P. H. Siegel. Adaptive methods for linear programming decoding. *IEEE Transactions on Information Theory*, 54(12):5396–5410, 2008.

[163] H. Tanaka, K. Furusawa, and S. Kaneku. A novel approach to soft decision decoding of threshold decodable codes. *IEEE Transactions on Information Theory*, 26(2):244–246, 1980.

[164] H. Tang, J. Xu, S. Lin, and K. Abdel-Ghaffar. Codes on finite geometries. *IEEE Transactions on Information Theory*, 51(2):572–596, 2005.

[165] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on information theory*, 27(5):533–547, 1981.

[166] J. Thorpe. Low-density parity-check (ldpc) codes constructed from protographs. *IPN progress report*, 42(154):42–154, 2003.

[167] T. Tian, C. Jones, J. D. Villasenor, and R. D. Wesel. Construction of irregular ldpc codes with low error floors. In *IEEE International Conference on Communications, 2003. ICC'03.*, volume 5, pages 3125–3129. IEEE, 2003.

[168] C. Tjhai, M. Tomlinson, M. Ambroze, and M. Ahmed. Cyclotomic idempotent-based binary cyclic codes. *Electronics Letters*, 41(6):341–343, 2005.

[169] C. Tjhai, M. Tomlinson, R. Horan, M. Ahmed, and M. Ambroze. Gf (2m) low-density parity-check codes derived from cyclotomic cosets. In *4th International Symposium on Turbo Codes & Related Topics; 6th International ITG-Conference on Source and Channel Coding*, pages 1–6. VDE, 2006.

[170] M. Tomlinson, C. Jung Tjhai, M. Ambroze, and M. Ahmed. Binary cyclic difference set codes derived from idempotents based on cyclotomic cosets. *submitted to IEEE Transations on information theory*, 2004.

[171] R. Townsend and E. Weldon. Self-orthogonal quasi-cyclic codes. *IEEE Transactions on Information Theory*, 13(2):183–195, 1967.

[172] Y-L. Ueng, C-Y. Wang, and M-R. Li. An efficient combined bit-flipping and stochastic ldpc decoder using improved probability tracers. *IEEE Transactions on Signal Processing*, 65(20):5368–5380, 2017.

[173] M. A. Ullah and H. Ogiwara. Performance improvement of multi-stage threshold decoding with difference register. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 94(6):1449–1457, 2011.

[174] M. A. Ullah, K. Okada, and H. Ogiwara. Multi-stage threshold decoding for self-orthogonal convolutional codes. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 93(11):1932–1941, 2010.

[175] A. Vardy. The intractability of computing the minimum distance of a code. *IEEE Transactions on Information Theory*, 43(6):1757–1766, 1997.

[176] B. Vasic. Combinatorial constructions of low-density parity check codes for iterative decoding. In *Proceedings IEEE International Symposium on Information Theory,*, page 312. IEEE, 2002.

[177] B. Vasic. High-rate low-density parity check codes based on anti-pasch affine geometries. In *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No. 02CH37333)*, volume 3, pages 1332–1336. IEEE, 2002.

[178] B. Vasić, S. K. Chilappagari, D. V. Nguyen, and S. K. Planjery. Trapping set ontology. In *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1–7. IEEE, 2009.

[179] B. Vasic, E. M. Kurtas, and A. V. Kuznetsov. Ldpc codes based on mutually orthogonal latin rectangles and their application in perpendicular magnetic recording. *IEEE transactions on magnetics*, 38(5):2346–2348, 2002.

[180] B. Vasic and O. Milenkovic. Combinatorial constructions of low-density parity-check codes for iterative decoding. *IEEE Transactions on information theory*, 50(6):1156–1176, 2004.

[181] B. Vasic, K. Pedagani, and M. Ivkovic. High-rate girth-eight low-density parity-check codes on rectangular integer lattices. *IEEE Transactions on Communications*, 52(8):1248–1252, 2004.

[182] P. O. Vontobel and R. Koetter. On the relationship between linear programming decoding and min-sum algorithm decoding. In *Proc. ISITA 2004*. Citeseer.

[183] P. O. Vontobel and R. Koetter. On low-complexity linear-programming decoding of ldpc codes. *European transactions on telecommunications*, 18(5):509–517, 2007.

[184] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi. Gradient descent bit flipping algorithms for decoding ldpc codes. In *2008 International Symposium on Information Theory and Its Applications*, pages 1–6. IEEE, 2008.

[185] E. J. Weldon. Difference-set cyclic codes. *The Bell System Technical Journal*, 45(7):1045–1055, 1966.

[186] S. R. Weller and S. J. Johnson. Regular low-density parity-check codes from oval designs. *European Transactions on Telecommunications*, 14(5):399–409, 2003.

[187] X. Wu, M. Jiang, C. Zhao, and X. You. Fast weighted bit-flipping decoding of finite-geometry ldpc codes. In *2006 IEEE Information Theory Workshop-ITW'06 Chengdu*, pages 132–134. IEEE, 2006.

[188] X. Wu, C. Zhao, and X. You. Parallel weighted bit-flipping decoding. *IEEE Communications letters*, 11(8):671–673, 2007.

[189] Hengzhou X., Zhongyang Y., Dan F., and Hai Z. New construction of partial geometries based on group divisible designs and their associated ldpc codes. *Physical Communication*, 39:100970, 2020.

[190] H. Xiao and A. H. Banihashemi. Improved progressive-edge-growth (peg) construction of irregular ldpc codes. *IEEE Communications Letters*, 8(12):715–717, 2004.

[191] R. B. Yale. Error correcting codes and linear recurring sequences. *Report MIT Lincoln Laboratory, Lexington, MA*, pages 34–77, 1958.

[192] K. Yamaguchi, H. Iizuka, E. Nomura, and H. Imai. Variable threshold soft decision decoding. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72(9):65–74, 1989.

[193] T-Y. Yang and H. Chen. Modified majority logic decoding of reed–muller codes using factor graphs. *IET Communications*, 12(7):759–764, 2018.

[194] A. Yatribi, M. Belkasmi, and F. Ayoub. Gradient-descent decoding of one-step majority-logic decodable codes. *Physical Communication*, 39:100999, 2020.

[195] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[196] L. Zeng, L. Lan, Y. Y. Tai, B. Zhou, S. Lin, and K. Abdel-Ghaffar. Construction of nonbinary cyclic, quasi-cyclic and regular ldpc codes: A finite geometry approach. *IEEE transactions on Communications*, 56(3):378–387, 2008.

[197] H. Zhang and José M. F. Moura. The design of structured regular ldpc codes with large girth. In *GLOBECOM'03. IEEE Global Telecommunications Conference (IEEE Cat. No. 03CH37489)*, volume 7, pages 4022–4027. IEEE, 2003.

[198] J. Zhang and M. P. C. Fossorier. A modified weighted bit-flipping decoding of low-density parity-check codes. *IEEE Communications Letters*, 8(3):165–167, 2004.

[199] L. Zhang, N. Liu, Z. Pan, and X. You. Tabu-list noisy gradient descent bit flipping decoding of ldpc codes. In *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–5. IEEE, 2019.

[200] Q. Zhang, A. Liu, X. Pan, and K. Pan. Crc code design for list decoding of polar codes. *IEEE Communications Letters*, 21(6):1229–1232, 2017.

[201] X. Zhang, L. Chen, J. Qiu, and J. Abdoli. On the waveform for 5g. *IEEE Communications Magazine*, 54(11):74–80, 2016.

[202] X. Zhang and P. H. Siegel. Adaptive cut generation algorithm for improved linear programming decoding of binary linear codes. *IEEE Transactions on Information Theory*, 58(10):6581–6594, 2012.

[203] X. Zhang and P. H. Siegel. Efficient iterative lp decoding of ldpc codes with alternating direction method of multipliers. In *2013 IEEE International Symposium on Information Theory*, pages 1501–1505. IEEE, 2013.

[204] C. Zhi and J. Fan. On optimal self-orthogonal quasi-cyclic codes. In *IEEE International Conference on Communications*, pages 1256–1260, 1990.

[205] B. Zhou, J. Kang, S. W. Song, S. Lin, K. Abdel-Ghaffar, and M. Xu. Construction of non-binary quasi-cyclic ldpc codes by arrays and array dispersions. *IEEE Transactions on Communications*, 57(6):1652–1662, 2009.

[206] N. Zierler. On a variation of the first-order reed-muller codes. *Lincoln Laboratories of Massachusetts Institute of Technology, Lexington, Mass*, 1958.

[207] V. Zolotarev, G. Ovechkin, D. Satybaldina, N. Tashatov, A. Adamova, and V. Mishin. Efficiency multithreshold decoders for self-orthogonal block codes for optical channels. *International Journal of Circuits, Systems and Signal Processing*, 8:487–495, 2014.