Conception et optimisation de codes AL-FEC : les codes GLDPC-Staircase

Ferdaouss Mattoussi

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Ferdaouss Mattoussi

Thèse dirigée par **Dr. Claude Castelluccia**
et codirigée par **Dr. Vincent Roca** et **Dr. Bessem Sayadi**

préparée au sein d' **INRIA Rhônes-Alpes, équipes Privatics**
et de **l'École Doctorale : Mathématiques, Sciences et Technologies de l'Information, Informatique**

Conception et optimisation de codes AL-FEC : les codes GLDPC-Staircase

Thèse soutenue publiquement le **13 Février 2014**,
devant le jury composé de :

M. Pr Charly Poulliat
Professeur, INP-ENSEEIHT Toulouse, Rapporteur
M. Pr Jérôme Lacan
Professeur, ISAE Toulouse, Rapporteur
M. Gerard Faria
Directeur de la technologie et du conseil d'administration, Teamcast, Examinateur
M. Dr Claude Castelluccia
Directeur de recherche, Inria, Directeur de thèse
M. Dr Vincent Roca
Chargé de recherche, Inria, Co-Directeur de thèse
M. Dr Bessem Sayadi
Ingenieur de recherche, Alcatel Lucent Bell-Labs Paris, Co-Directeur de thèse

Abstract

This work is dedicated to the design, analysis and optimization of Application-Level Forward Erasure Correction (AL-FEC) codes. In particular, we explore a class of Generalized LDPC (GLDPC) codes, named GLDPC-Staircase codes, involving the LDPC-Staircase code (base code) as well as Reed-Solomon (RS) codes (outer codes).
In the first part of this thesis, we start by showing that RS codes having "quasi" Hankel matrix-based construction are the most suitable MDS codes to obtain the structure of GLDPC-Staircase codes. Then, we propose a new decoding type, so-called hybrid (IT/RS/ML) decoding, for these codes to achieve Maximum Likelihood (ML) correction capabilities with a lower complexity. To investigate the impact of the structure of GLDPC-Staircase codes on decoding, we propose another construction: they differ on the nature of generated LDPC repair symbols. Afterwards, to predict the capacity approaching GLDPC-Staircase codes, we derive an asymptotic analysis based on DE, EXIT functions, and area theorem. Eventually, based on finite length analysis and asymptotic analysis, we tune important internal parameters of GLDPC-Staircase codes to obtain the best configuration under hybrid (IT/RS/ML) decoding.
The second part of the thesis benchmarks GLDPC-Staircase codes in various situations. First, we show that these codes are asymptotically quite close to Shannon limit performance and achieve finite length excellent erasure correction capabilities very close to that of ideal MDS codes no matter the *objects size*: very small decoding overhead, low error floor, and steep waterfall region. Second, we show that these codes outperform Raptor codes, LDPC-Staircase codes, other construction of GLDPC codes, and have correction capabilities close to that of RaptorQ codes. Last but not least, we propose a general-methodology to address the problem of the impact of packet scheduling on GLDPC-Staircase codes for a large set of loss channels (with burst loss or not). This study shows the best packet scheduling. All the aforementioned results make GLDPC-Staircase codes an ubiquitous Application-Level FEC (AL-FEC) solution.

Résumé

Ce travail est consacré à la conception, l'analyse et l'optimisation des codes correcteurs d'effacements de niveau applicatif (AL-FEC). Nous nous intéressons à une famille des codes LDPC généralisés (GLDPC), nommés les codes GLDPC-Staircase, qui sont composés d'un code LDPC-Staircase (*code de base*) ainsi que des codes Reed-Solomon (RS) (*codes externes*).
Dans la première partie de cette thèse, nous commençons par montrer que les codes RS ayant une construction basée sur la matrice "quasi" Hankel sont les codes MDS les plus appropriés pour obtenir la structure des codes GLDPC-Staircase. Ensuite, nous proposons un nouveau type de décodage à ces codes, baptisé le décodage hybride (IT/RS/ML), pour atteindre les caspacités de correction du décodage par maximum de vraisemblance (ML) avec de faible complexité. Afin d'étudier l'impact de la structure des codes GLDPC-Staircase sur le décodage, nous proposons une autre construction : ils se diffèrent sur la nature des symboles de redondance LDPC générés. Puis, pour prédire le seuil de décodage et les écarts par rapport à la limite de Shannon des codes GLDPC-Staircase, nous élaborons une analyse asymptotique en fonction de la technique d'évolution de densité (DE), la technique EXIT (Extrinsic Information Transfer) et le théorème d'air. Finalement, en se basant sur l'analyse à taille finie et l'analyse asymptotique, nous ajustons les différentes paramètres internes de ces codes pour obtenir la configuration optimale sous le décodage hybride (IT/RS/ML).
La deuxième partie de la thèse est consacrée à l'étude des codes GLDPC-Staircase dans diverses situations. Tout d'abord, nous montrons que ces codes ont des performances asymptotiquement très proches de la limite de Shannon. En plus, à taille finie, ils permettent d'atteindre d'excellentes capacités de correction d'effacements (i.e., très proches de celles des codes MDS) peu importe la taille des objets : très faible overhead de décodage, faible plancher d'erreur, et une zone "waterfall" raide. Nous montrons aussi que les performances de ces codes sont trés proches des codes RaptorQ et surpassent celles des codes Raptor, les codes LDPC-Staircase, et un autre code GLDPC avec une construction différente. Finalement, nous proposons une méthodologie générale pour étudier l'impact de l'ordonnancement des paquets envoyés sur les performances des codes GLDPC-Staircase sur un grand nombre des canaux à effacements (avec des pertes en rafale ou pas). Cette étude montre le meilleur ordonnancement de paquets.
Tous les résultats mentionnés ci-dessus montrent que les codes GLDPC-Staircase peuvent considérés comme des codes AL-FEC universels.

Acknowledgements

I would like to start by thanking the members of the technical committee for their agree to evaluate this thesis and especially my reviewers; Professor Charly Poulliat and Professor Jerôme Lacan for their insightful suggestions, which improved this thesis in many ways.

I owe my deep gratitude to my supervisors Vincent Roca and Bessem Sayadi, for their guidance through the work on this thesis and for the opportunity of working on this subject. I would also like to thank my Prevatics INRIA Team for their help, good ambiances, and friendship, including Abdelberi, Gergelay, Jagdish, Ludovic and James.

Finally, I would like to express special thanks to my family. I thank my parents for their love, help and encouragement which made it possible for me to get a doctoral degree. No words are enough to show my appreciation for them. I also wish to thank my sisters and my brothers and I greatly appreciate their thoughtful support and encouragement.

This work was supported by the ANR-09-VERS-019-02 grant (ARSSO project) and by the INRIA - Alcatel Lucent Bell Labs joint laboratory.

List of contributions

International conference papers

1. "Complexity comparison of the use of Vandermonde versus Hankel matrices to build systematic mds Reed Solomon codes" [1]
   Ferdaouss Mattoussi, Vincent Roca, and Bessem Sayadi. in IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Turkey, June 2012.

2. "Optimization with EXIT functions of GLDPC Staircase codes for the BEC" [2]
   Ferdaouss Mattoussi, Valentin Savin, Vincent Roca, and Bessem Sayadi. in IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Turkey, June 2012.

3. "Design of small rate, close to ideal, GLDPC Staircase AL-FEC codes for the erasure channel" [3]
   Ferdaouss Mattoussi, Vincent Roca, and Bessem Sayadi. in IEEE Global Communications Conference (GLOBECOM 2012), USA, December 2012, acceptance rate 37.7%.

4. "Good coupling between LDPC-Staircase and Reed-Solomon for the design of GLDPC codes for the Erasure Channel", [4]
   Ferdaouss Mattoussi, Bessem Sayadi and Vincent Roca. in IEEE Wireless Communications and Networking Conference (WCNC), China, April 2013, acceptance rate 42%.

5. "Impacts of the packet scheduling on the performance of certain erasure codes: methodology and application to GLDPC Staircase",
   Ferdaouss Mattoussi, Vincent Roca, and Bessem Sayadi. **submitted** in IEEE International Conference on Communications (ICC), 2014

Journal paper (in preparation)

1. "Good GLDPC Staircase AL-FEC codes: design, configuration setting, performance",
   Ferdaouss Mattoussi, Vincent Roca, and Bessem Sayadi.

Patent

1. **Ferdaouss Mattoussi**, Bessem Sayadi and Vincent Roca, "Method for encoding with GLDPC codes based on Reed-Solomon codes", European Patent, 12172196.3, filled 28/12/2012, Alcatel-Lucent.

Software: Contributions to the library of erasure codecs (http://openfec.org)

- Design of the GLDPC staircase codec (100%),
- Generator matrix construction of the RS Hankel codec (100%).
- Participation to the design of the advanced LDPC-Staircase codes (2.5%),

Contents

1 Introduction **23**
- 1.1 Context and historical background . . . 23
- 1.2 Goal of this thesis . . . 26
- 1.3 Contributions of this thesis . . . 26
- 1.4 Thesis Outline . . . 28

2 Literature review **31**
- 2.1 Introduction . . . 31
- 2.2 Channel coding overview . . . 31
  - 2.2.1 Channel coding foundation . . . 31
  - 2.2.2 Communication system . . . 32
    - 2.2.2.1 Source . . . 32
    - 2.2.2.2 Erasure channels . . . 33
- 2.3 Erasure Forward Error Codes (FEC) codes . . . 35
- 2.4 Block FEC codes . . . 36
  - 2.4.1 Definition . . . 36
  - 2.4.2 Decoding algorithms . . . 37
  - 2.4.3 Low Density Parity Check (LDPC) codes . . . 38
    - 2.4.3.1 History . . . 38
    - 2.4.3.2 Description and Construction . . . 38
    - 2.4.3.3 Encoding LDPC codes . . . 40
    - 2.4.3.4 Decoding LDPC codes over an erasure channel . . . 41
    - 2.4.3.5 Asymptotic analysis tools for LDPC codes . . . 44
      - 2.4.3.5.a Introduction . . . 44
      - 2.4.3.5.b DE technique for LDPC codes over an erasure channel . . . 45
      - 2.4.3.5.c EXtrinsic Information Transfer (EXIT) technique for LDPC codes over an erasure channel . . . 46
    - 2.4.3.6 Performance affecting the structures of LDPC codes . . . 49
  - 2.4.4 Reed Solomon codes . . . 51
- 2.5 Rate-less erasure codes . . . 53
  - 2.5.1 Introduction . . . 53
  - 2.5.2 Luby Transform (LT) codes . . . 53
  - 2.5.3 Raptor codes . . . 54

I Design and optimization of GLDPC-Staircase codes 57

3 LDPC-Staircase codes and systematic Reed Solomon (RS) codes over erasure channels 59
3.1 Introduction . . . 59
3.2 LDPC-Staircase codes . . . 60
3.2.1 Parity check matrix construction . . . 60
3.2.2 Encoding . . . 61
3.2.3 Decoding . . . 61
3.2.4 Performance: correction capabilities and complexity . . . 62
3.3 Systematic RS codes . . . 62
3.3.1 Construction methods . . . 62
3.3.1.1 Construction based on Vandermonde matrix . . . 63
3.3.1.2 Construction based on Hankel matrix . . . 64
3.3.2 Code construction method complexity analysis . . . 65
3.3.2.1 Performance evaluation environment . . . 65
3.3.2.2 Generator matrix creation time . . . 66
3.3.2.3 Generator matrix creation complexity . . . 67
3.3.2.4 Impacts on the global encoding and decoding times . . . 68
3.4 Conclusions . . . 68

4 Generalized LDPC (GLDPC)-Staircase codes 71
4.1 Introduction . . . 71
4.2 Proposed GLDPC-Staircase coding schemes . . . 72
4.2.1 Design of GLDPC-Staircase codes . . . 72
4.2.2 Encoding of GLDPC-Staircase codes . . . 76
4.2.3 Decoding of GLDPC-Staircase codes . . . 77
4.2.3.1 (IT+RS) decoding . . . 77
4.2.3.2 Hybrid (IT/RS/ML) decoding . . . 79
4.3 Asymptotic analysis of GLDPC-Staircase codes under (IT+RS) and ML decoding . . . 80
4.3.1 Preliminaries . . . 80
4.3.2 Density Evolution . . . 81
4.3.2.1 Introduction . . . 81
4.3.2.2 DE equations of GLDPC-Staircase codes . . . 81
4.3.3 EXIT functions of GLDPC-Staircase codes . . . 85
4.3.3.1 Introduction . . . 85
4.3.3.2 (IT+RS) EXIT function: $h^{(IT+RS)}(\varepsilon)$ . . . 85
4.3.3.3 (IT+RS) EXIT curve . . . 85
4.3.3.4 Upper bound on the ML decoding threshold . . . 87
4.4 Conclusions . . . 89

5 Optimization of GLDPC-Staircase codes 91
5.1 Introduction . . . 91
5.2 Experimental conditions . . . 92
5.3 Best coding scheme for GLDPC-Staircase codes . . . 93
5.3.1 Asymptotic results . . . 93
5.3.2 Finite length results . . . 96
5.3.3 Conclusion of the analysis . . . 99
5.4 Tuning internal parameters of GLDPC-Staircase codes . . . 100
5.4.1 The extra-repair symbols distribution . . . 100
5.4.2 N1 parameter . . . 102
5.4.3 The base code rate r_L . . . 103
5.5 Conclusions . . . 105

II Performance evaluation in different use cases 107

6 Performance evaluation of GLDPC-Staircase codes over memory-less channel 109
6.1 Introduction . . . 109
6.2 Achieved performance . . . 110
6.2.1 (IT+RS) versus Hybrid (IT/RS/ML) decoding . . . 110
6.2.2 Hybrid (IT/RS/ML) decoding inefficiency ratio results . . . 110
6.2.3 Hybrid (IT/RS/ML) decoding failure probability results . . . 111
6.3 Comparison with other erasures correcting codes . . . 113
6.3.1 LDPC-Staircase codes . . . 113
6.3.2 Generalized Low Density Parity Check (GLDPC) codes designed in [5] . . . 114
6.3.3 Raptor codes . . . 115
6.4 Decoding complexity . . . 116
6.4.1 Experimental conditions . . . 116
6.4.2 Results . . . 116
6.5 Conclusions . . . 118

7 Impacts of the packet scheduling on the performance for broadcast/multicast services 121
7.1 Introduction . . . 121
7.2 Preliminaries . . . 122
7.3 Proposed methodology . . . 123
7.3.1 Channel Gilbert model . . . 123
7.3.2 Evaluation methodology . . . 124
7.4 Transmission scheduling considered . . . 126
7.5 Simulation results and discussion . . . 127
7.5.1 Simulation Setup . . . 127
7.5.2 Results W.R.T. the $\{p,q\}$ parameters . . . 128
7.5.3 Results W.R.T. the $\{PLR,ABL\}$ parameters . . . 130

7.6 Conclusions 131

III Conclusions and perspectives 135

8 Conclusions and Perspectives 137
8.1 Conclusions 137
8.1.1 Design of GLDPC-Staircase codes 137
8.1.2 Hybrid IT/RS/binary or non binary ML decoding of GLDPC-Staircase codes 138
8.1.3 Asymptotic analysis of GLDPC-Staircase codes through DE and EXIT methods 138
8.1.4 Evaluation of GLDPC-Staircase codes 139
8.2 Future works 140
8.2.1 GLDPC-Staircase Encoding/Decoding optimization 140
8.2.2 Extension of GLDPC-Staircase coding scheme over an Additive White Gaussian Noise (AWGN) channel 141

IV Appendices 143

A Performance evaluation metrics and tools for correcting codes 145
A.1 Performance evaluation metrics 145
A.1.1 Correction capabilities tools 145
A.1.1.1 Distance to the capacity Δ 145
A.1.1.2 Decoding failure probability curve 145
A.1.2 Decoding inefficiency ratio 147
A.1.3 Algorithmic complexity metrics 147
A.1.3.1 Theoretical complexity 148
A.1.3.2 Encoding/decoding throughput 148
A.1.3.3 Maximum memory consumption 148

List of Figures

2.1 Communication system model. 32
2.2 Erasure channel model. 33
2.3 $H(X)$, $H(Y)$, joint ($H(X,Y)$), and conditional entropies for a pair of correlated subsystems X,Y with mutual information $I(X;Y)$. 34
2.4 Bipartite graph of LDPC code (8,4). 39
2.5 Extrinsic information processing of variable node V. 47
2.6 Extrinsic information processing of check node C. 47
2.7 Example of a Cycle in the Tanner graph of an LDPC code. 50
2.8 Example of a stopping set in the Tanner graph of an LDPC code. 50
2.9 Bipartite graph of LT code with n = 8 and k = 6. 53

3.1 Bipartite graph of LDPC-Staircase code with $N_L = 10$, $K = 6$ and $N1 = 2$. 60
3.2 Systematic generator matrix creation times. 66
3.3 Comparison between (3k,k) Vandermonde and Hankel Reed-Solomon codes during encoding and decoding with symbols of size 4 bytes. . . . 69
3.4 Comparison between (255,k) Vandermonde and Hankel RS codes during encoding and decoding with symbols of size 4 bytes. 70

4.1 GLDPC-Staircase$(13,4)$ code, $e(m) = 2$ extra-repair symbols per generalized check node (i.e, regular distribution). 73
4.2 GLDPC-Staircase$(13,4)$ code with irregular distribution, $e(m) = \{3,1,2\}$ 75
4.3 Example of (IT+RS) decoding on the graph of GLDPC-Staircase code (scheme A). 79
4.4 The evolution, for scheme A, of the (IT+RS) decoding process for LDPC-Staircase with $r_L = 0.8$ and $N1 = 5$. r_G=$\frac{1}{2}$, $E = 3$ and $\varepsilon = 0.3$. Shannon limit=0.5, threshold $\varepsilon^{(IT+RS)}$=0.3443 84
4.5 Computation of EXIT function based on entropy of a GLDPC-Staircase code. 86
4.6 The (IT+RS) EXIT function $h^{(IT+RS)}(\varepsilon)$ for an ensemble of GLDPC-Staircase code with rate = $\frac{1}{3}$. 87
4.7 ML-threshold upper-bound computation using the (IT+RS) EXIT function $h^{(IT+RS)}(\varepsilon)$ for an ensemble of GLDPC-Staircase code with rate = $\frac{1}{3}$. 88

5.1 The evolution, for schemes A and B, of the (IT+RS) decoding process for GLDPC-Staircase with $r_L = 0.8$, $N1 = 5$, r_G=$\frac{1}{2}$, $E = 3$ and $\varepsilon = 0.32$. Shannon limit=0.5, threshold $\varepsilon^{(IT+RS)}$=0.3443 (scheme A) and threshold $\varepsilon^{(IT+RS)}$=0.2819 (scheme B). 94
5.2 Average performance under (IT+RS) decoding or ML decoding, with rate $\frac{1}{2}$, as a function of K. 98
5.3 Average performance under (IT+RS) decoding or ML decoding, with rate $\frac{1}{3}$, as a function of K. 99
5.4 Decoding failure probability under ML decoding, with rate $\frac{1}{2}$ and $K = 1000$, as a function of the number of received symbols. 100
5.5 Performance for irregular uniform distribution versus regular distribution of extra-repair symbols, with N1= 5, $r_L = \frac{2}{3}$, and r_G=$\frac{1}{2}$ 101
5.6 Performance as a function of base code rate r_L with N1= 5, r_G=$\frac{1}{3}$, and ML decoding. 104

6.1 Performance for (IT+RS) versus hybrid (IT/RS/ML) decoding schemes, with r_G=$\frac{1}{2}$. 110
6.2 Average decoding inefficiency ratio as a function of the object sizes (a) and code rates (b). 111
6.3 Average hybrid (IT/RS/ML) decoding failure probability as a function of the channel loss percentage experienced for GLDPC codes ($r_G = \frac{1}{2}$) with $K = 32$, $K = 256$, and $K = 1000$. 112
6.4 LDPC Staircase codes versus GLDPC-Staircase codes for rate=$\frac{1}{2}$ and N1=5. 114
6.5 ML decoding failure probability versus overhead for GLDPC and Raptor using three cases. 115
6.6 Throughput of hybrid (IT/RS/ML) decoding of GLDPC-Staircase codes with $r_G = \frac{1}{2}$, E=1, N1=5, Symbol size = 1024 bytes 117
6.7 Number of source symbols decoded with jump from NB-ML decoding to B-ML decoding for GLDPC Staircase codes 118
6.8 Throughput of GLDPC Staircase codes with rate $\frac{1}{2}$ and $E = 1$ 119

7.1 The various steps considered in the transmission chain. 122
7.2 Two states Markov loss model. 124
7.3 Impossible decoding area for $CR = (1/2, 1/3, 2/3)$ 126
7.4 Total number of (IT+RS) successful decodings W.R.T. the ABL, with Packet Loss Ratio (PLR) =33.333%, when $r_G = \frac{1}{3}$. 131
7.5 Total number of (IT+RS) successful decodings W.R.T. the ABL, with PLR =50% (top 3 figs.) and 66.66% (last fig.), when $r_G = \frac{1}{3}$. 132
7.6 Total number of ML successful decodings W.R.T. the Average Burst Length (ABL), with PLR =33.333%, and PLR =50%, when $r_G = \frac{1}{3}$. . . 132
7.7 Total number of ML successful decodings W.R.T. the ABL, with PLR =66.66%, when $r_G = \frac{1}{3}$. 133

A.1 Interest areas on a curve of block error probability according to the erasure rate (k = 1000, R = 2/3). 146

List of Tables

2.1 Decoding failure probability distribution and the required overhead for Raptor codes [6] 56
2.2 Decoding failure probability distribution and the required overhead for RaptorQ codes [6] 56

3.1 Generator matrix creation times and performance gains made possible by the Hankel approach. 67
3.2 Generator matrix creation complexity (number of XOR and table access (TA) operations) and performance gains made possible by the Hankel approach). 68

5.1 LDPC-Staircase DD for different values of N1 where $r_L = \frac{2}{3}$. 92
5.2 LDPC-Staircase DD for different values of r_L where N1=5. 93
5.3 Decoding threshold comparison between scheme A and B for different values of N1 for (IT+RS) decoding and ML decoding. GLDPC-Staircase codes with r_L=$\frac{2}{3}$ and regular distribution of extra-repair symbols. . . . 95
5.4 Decoding threshold comparison between (IT+RS) decoding and ML decoding, for r_G=$\frac{1}{2}$ (Shannon limit =0.5) and regular distribution of extra-repair symbols. 96
5.5 Decoding threshold comparison between (IT+RS) decoding and ML decoding, for r_G=$\frac{1}{2}$ (Shannon limit=0.5) and irregular uniform distribution of extra-repair symbols. 96
5.6 Decoding threshold comparison between (IT+RS) decoding and ML decoding for different values of r_G with $r_L = \frac{2}{3}$ and regular distribution of extra-repair symbols. 97
5.7 Decoding threshold comparison between (IT+RS) decoding and ML decoding for different values of r_G with $r_L = \frac{2}{3}$ and irregular uniform distribution of extra-repair symbols. 97
5.8 (ITerative (IT)+RS) decoding and Maximum Likelihood (ML) decoding gaps to capacity for irregular uniform distribution (Δ_2) versus regular distribution (Δ_1) of extra-repair symbols for GLDPC-Staircase codes with $N1 = 5$, and $r_L = \frac{2}{3}$. 102
5.9 (IT+RS) decoding and ML decoding gaps to capacity for irregular uniform distribution (Δ_2) versus regular distribution (Δ_1) of extra-repair symbols for GLDPC-Staircase codes with $N1 = 5$, and $r_G = \frac{1}{2}$. 102

5.10 Threshold decoding comparison between the two distributions of extra-repair symbols for different values of N1 and r_G with $r_L = \frac{2}{3}$. 103
5.11 Average decoding inefficiency ratio us a function of N1 and the decoding scheme (IT+RS versus ML), for $K = 1000$ and $r_L = \frac{2}{3}$. 104
5.12 Decoding threshold comparison between (IT+RS) decoding and ML decoding versus $N1$ for different values of r_L and r_G. 105
5.13 $\bar{\varepsilon}_{ML}$ of GLDPC-Staircase codes as a function of r_G 105

6.1 Average decoding failure probability of GLDPC-Staircase codes ($r_G = \frac{1}{2}$) under ML decoding for K=32 and K=1000 113
6.2 Comparison decoding threshold between GLDPC-Staircase codes and other GLDPC construction code for rate $\frac{1}{2}$. 115

7.1 The 22 packet schedulings considered in this work. 127
7.2 Total number of successful tests for scheduling category (2) under (IT+RS) decoding. 128
7.3 Total number of successful tests for scheduling category (3) under (IT+RS) decoding. 128
7.4 Total number of successful tests for scheduling categories (1) and (4) under (IT+RS) decoding. 128
7.5 Total number of successful tests for scheduling category (2) under ML decoding. 129
7.6 Total number of successful tests for scheduling category (3) under ML decoding. 129
7.7 Total number of successful tests for scheduling categories (1) and (4) under ML decoding. 130

List of acronyms

ABL Average Burst Length

AL-FEC Application Layer Forward Error Correction

AWGN Additive White Gaussian Noise

ARQ Automatic Repeat reQuest

BCH Bose Chaudhuri Hocquenghem

BP Belief Propagation

BEC Binary Erasure Channel

BSC Binary Symmetric Channel

BM Berlekamp Massey

CRC Cyclic Redundancy Control

DD Degree Distribution

DE Density evolution

DVB Digital Video Broadcasting

DVB-SH Digital Video Broadcasting-Satellite Handheld

DVB-H Digital Video Broadcasting-Handheld

ECC Error Correcting Codes

EXIT EXtrinsic Information Transfer

FEC Forward Error Codes

GLDPC Generalized Low Density Parity Check

GE Gaussian Elimination

HIHO Hard Input Hard Output

IETF the Internet Engineering Task Force

IT ITerative

IRA Irregular Repeat Accumulate

LDPC Low Density Parity Check

LDGM Low Density Generator Matrix

LT Luby Transform

MBMS Multimedia Broadcast/Multicast Service

MDS Maximum Distance Separable

MI Mutual Information

ML Maximum Likelihood

MPE-IFEC Multiple Protocol Encapsulation for Inter-Burst Forward Error Correction

MPE-FEC Multiple Protocol Encapsulation for Forward Error Correction

PEG progressive-edge-growth

PRNG Pseudo Random Number Generator

PLR Packet Loss Ratio

pdf probability density function

TCP Transmission Control Protocol

UDP User Datagram Protocol

UEP Unequal Erasure Protection

RA Repeat Accumulate

RS Reed Solomon

RSD Robust Soliton Distribution

SDD Soft Decision Decoding

SECDED Single Error Correcting Double Error Detecting

SISO Soft Input Soft Output

SNR Signal Noise Ratio

SPC Single Parity Check

SGE Structured Gaussian Elimination

3GPP 3rd Generation Partnership Project

3GPP MBMS 3GPP Multimedia Broadcast/Multicast Service

Chapter 1

Introduction

1.1 Context and historical background

The main objective of communications is the transmission of a message through a noisy channel so that the receiver can determine this message despite the distortions of the channel. In 1948, Claude Shannon laid down the foundations of channel coding [7]. He showed that we can efficiently and reliably transmit information (i.e, without any loss of information during transmission) based on a *channel code* and *a decoding algorithm*. Said differently, the noisy channel coding theorem asserts both that reliable communication at any rate up to the channel capacity is possible whereas it is impossible beyond the channel capacity.

Shannon's theorem only shows the existence of such a channel code and such a decoding algorithm but refrains from defining code construction and explaining how to decode the code efficiently.

When channel codes are used to correct errors occurring on coded information, in digital form, they are called Error Correcting Codes (ECC) or FEC codes. These codes add redundancy to the message that can be exploited to combat the imperfection introduced by the channel. In addition, ECCs are used to maintain data integrity across noisy channels and minimize interference in multi-user systems.
Since, Shannon's theory is not constructive, this triggered widespread research efforts for practical coding systems that communicate reliably over a noisy channel and approach the fundamental limit.
In modern transmission systems, the codes can be found at:

- Physical/MAC layers: Such codes work on binary channels (e.g., of symmetric type Binary Symmetric Channel (BSC) or AWGN) that inject errors, i.e. change values at the bit level;

- Link layer: Such codes work on erasure channels. For instance Multiple Protocol Encapsulation for Forward Error Correction (MPE-FEC) Digital Video Broadcasting-Handheld (DVB-H), or Multiple Protocol Encapsulation for Inter-Burst Forward Error Correction (MPE-IFEC) Digital Video Broadcasting-Satellite Handheld (DVB-SH);

- Transport or application layers: Such codes work on erasure channels where the losses affect packets. The channel is therefore a "packet erasure channel" where packets either arrive (with no error) or lost. In that case, these codes are often called **Application Layer Forward Error Correction (AL-FEC) codes**. AL-FEC codes are usually implemented in the upper layers (within or near the application).

These different codes in different protocol layers will then work in parallel, generally without direct interaction. This raises the problem of initialization of these codes, and in particular the considered question of where it is preferable putting redundancy. It is obvious that the answer to this question can not be done independently of the type of application (we will not face the same strategy in case of real-time streaming or file transfer) and the application area.
The application can be :

- File transfer: this application works either in push/pull mode (single transfer) or on demand mode (transfer loop in a carousel). This application is characterized by a total reliability, an incentive to use very large encoding blocks in order to optimize the protection, or very small code rates in cases where the same content is transmitted over long periods in order to minimize the probability of duplicated packet reception;

- Audio stream: this application is characterized by a small tolerance to losses, a high sensitivity to delay, small packets to reduce the audio encoding delay and small encoding blocks to limit the decoding delay;

- Video stream: this application is more tolerant to losses than audio stream, often features a hierarchical data structure that may be used for Unequal Erasure Protection (UEP), small to medium block sizes and low delay (it depends on the stream type).

The problem of packet loss can be solved by using an Automatic Repeat reQuest (ARQ) approach which consists, via a return channel, in requesting retransmission of missing packets (e.g., as in Transmission Control Protocol (TCP) in case of point to point communications). However, this solution can not be used in several situations, for instance with a content delivery system like IP Data-cast (IPDC) [8][9] in DVB-H, or Multimedia Broadcast/Multicast Service (MBMS) [10] in 3rd Generation Partnership Project (3GPP), or data broadcast to cars (e.g. [11]). First, the return channel may not exist (e.g., with a broadcast link) and therefore no repeat request mechanism can be used. Second, there is a scalability problem in terms of the number of receivers with multicast or broadcast scenarios [12] (each receiver undergoes a different loss scheme). Third, the retransmission involves an *unacceptable delay* for such application as real-time streaming, and long distance communications (e.g., interplanetary Internet [13]). Using a reliable multicast transmission protocol like ALC [14] (which can in fact offer either a fully or partially reliable delivery service, depending on the way it is used) along with the FLUTE [15] file delivery application, can turn out to be highly effective in this context [11] when associated to FEC codes. The great advantage of using FEC with multicast or broadcast transmissions is that the same repair packet can recover different lost packets at different receivers.
FEC codes play an important role in communication systems today, for instance in:

- **Wireless and Mobile Communications**: FEC techniques are widely used in mobile communication systems to cope with the perturbations caused by interference, noise, multi path fading, shadowing, or propagation loss.

- **Satellite Communications**: For digital satellite TV, Reed-Solomon (RS) codes have been used for a long time. However they are currently replaced by modern codes such as LDPC and Turbo codes (e.g., LDPC codes have been incorporated into the Digital Video Broadcasting (DVB)-S2 standard [16]).

- **Data Storage**: RS codes is widely used for error correction in storage systems such as CDs, DVDs and hard discs. Single Error Correcting Double Error Detecting (SECDED) codes, which correct single errors and detect double errors, are widely used in many data storage units with RS codes or Hamming codes.

During the 1990s, remarkable progress was made towards the Shannon limit, using FEC codes that are defined in terms of ***sparse random graphs***. This technique plays an important role in modern communications, especially in coding theory and practice. The well known category of sparse-graph codes is LDPC codes. These codes were conceived by Gallager in his doctoral dissertation in 1960 [17], and rediscovered thirty years later [18–23] after the invention of Turbo Codes [24]. LDPC codes have been intensively studied due to their excellent asymptotic performance (i.e., near-Shannon limit performance) under iterative Belief Propagation (BP) decoding with moderate decoding complexity over a wide range of communication channels [20, 22, 25–31]. Very good asymptotic performance in terms of decoding threshold does not necessarily correspond to a satisfying finite length performance. In fact, finite length LDPC codes with a degree distribution associated to a close-to-capacity decoding threshold, though typically characterized by good waterfall performance, are usually affected by high error floors [32–35]. This is one of the main reasons for investigating more powerful LDPC coding schemes. For instance, a generalization of these codes, called GLDPC codes, was suggested by Tanner in [36], for which subsets of the variable nodes obey a more complex constraint than a Single Parity Check (SPC) constraint. They are replaced with a generic linear block codes (n,k) referred to as sub-codes or component codes while the sparse graph representation is kept unchanged. More powerful decoders at the check nodes have been investigated by several researchers in recent years after the work of Boutros et al.[37] and Lentmaier and Zigangirov [38] where Bose Chaudhuri Hocquenghem (BCH) codes and Hamming codes were proposed as component codes respectively. Later several works, on several channels, have been carried out in order to afford very large minimum distance and exhibit performance approaching Shannon's limit. Each construction differs to other by modifying components codes [39, 40, 40–44], or/and the distribution of the structure of GLDPC codes [39] to offer a good compromise between waterfall performance and error floor under iterative decoding.
Over erasure channels, AL-FEC codes are now a key component of reliable multicast/broadcast transmission systems. They are the key building block of the FLUTE/ALC (RFC 3926) [15] reliable multicast transport protocol that is used to push any kind of files (e.g. multimedia) for instance over the wireless 3G/4G channels (e.g. they are part of the

3GPP Multimedia Broadcast/Multicast Service (3GPP MBMS), DVB-H/SH IPDatacasting, or ISDB-Tmm services). They are also the key building block of robust streaming protocols, like the FECFRAME (RFC 6363) [45] transport protocol, that is present in the above systems.
For instance, in the Reliable Multicast Transport (RMT) WG, which focuses on reliable file delivery, LDPC-Staircase AL-FEC code is standardized as RFC5170 [46] and Reed-Solomon AL-FEC code as RFC5510 [47]. These standards explain how to use them with FLUTE/ALC [15], to enable a reliable and scalable multicast/broadcast delivery of contents over unidirectional transport networks. In the FEC Framework WG, which focuses on real-time delivery services, LDPC-Staircase and Reed-Solomon AL-FEC codes are also standardized as RFC6816 [48] and RFC6865 [49]. In addition, since mid 2012, LDPC-Staircase code is the AL-FEC scheme being used in the Japanese ISDB-Tmm standard [50]. This code is the core AL-FEC technology, and it is used along with FLUTE/ALC to improve the reliability and efficiency of push video services.

The research in the area of AL-FEC codes is still very active and there are many open problems under study to obtain both good AL-FEC codes and good AL-FEC codec.

1.2 Goal of this thesis

This thesis focuses on AL-FEC codes, for the erasure channels. A good AL-FEC solution requires both ***good codes and good codecs***. Since AL-FEC codes work within or close to the application, we only consider ***software codecs*** (rather than hardware codecs). This AL-FEC solution must:

- be able to decode an object from a number of encoding symbol as small as possible no matter the object size (***very small overhead***);
- have ***high encoding and decoding speeds*** and ***low maximum memory consumption*** to be suited for terminals with limited computational resources, memory and energy (e.g., smart-phones);
- have ***large-block capability***, i.e., be able to encode a very large object directly, without being obliged to split it into several blocks that are encoded separately (e.g., this is important for file transfer use-cases);
- be characterized by ***small-rate capability***, i.e., be able to produce a large number of encoding symbols. It is a feature that is well suited to situations where the channel conditions can be worse than expected, or to fountain like content distribution applications.

1.3 Contributions of this thesis

A construction approach for GLDPC codes has recently been proposed for the erasure channel in [51]. This method uses LDPC-Staircase code as base code and Reed-Solomon (RS) codes as component codes. This construction allows ***each component code to produce a potentially large number of repair symbols in terms of RS codes (named* extra-repair symbols*) on demand***. The production of these extra-repair symbols allows

to extend the initial LDPC-Staircase code (base code) to a generalized LDPC-Staircase code and ***very small rates*** are easily achievable. However this work does not explain how to build these GLDPC codes and the performance are only evaluated under (IT+RS) decoding (***it remains a theoretical work***).
However GLDPC-Staircase codes are potentially a good candidate to obtain a good AL-FEC solution and we have spent our efforts on this solution.
More specifically our contributions are:

- ***We describe and evaluate low complexity construction for systematic RS codes over erasure channel:***
  The traditional method for building systematic RS codes based on Vandermonde matrices requires many manipulations over the Finite Field. We introduce another low complexity approach, based on "pure" or "quasi" Hankel matrix. This choice is of high importance for all situations where a software RS(n, k) codec needs to generate on the fly the generator matrix, with appropriate dimension and length values.

- ***We propose the design of GLDPC-Staircase codes using RS codes based on "quasi" Hankel matrix:***
  We detail how to use RS codes based on quasi Hankel matrix to achieve the desired features in GLDPC-Staircase codes.

- ***We detail an hybrid (IT/RS/ML) decoding for GLDPC-Staircase codes:***
  GLDPC-Staircase codes are principally decoded by an ITerative+Reed Solomon (IT+RS) decoding. In order to obtain the optimal ML correction capabilities with low decoding complexity, we propose the hybrid (IT/RS/ML) decoding scheme.

- ***We investigate the impacts of the GLDPC-Staircase structure on the decoding performance:***
  We compare two internal structures of these codes, that differ by the nature of the LDPC repair symbols: are they also RS repair symbols (scheme A) or not (scheme B)?
  These two schemes are two kinds of coupling between LDPC-Staircase and RS codes.

- ***We derive an asymptotic analysis of GLDPC-Staircase codes (scheme A and B) under (IT+RS) and ML decoding over erasure channel:***
  We derive the Density evolution (DE) equations for the two schemes following [52]. We extend the ideas of [53] to our case and give the EXIT functions of the two schemes under (IT+RS) and ML decoding. Both the upper bound on the ML and (IT+RS) decoding thresholds are computed for these codes.

- ***We discuss the configuration of GLDPC-Staircase codes for hybrid (IT/RS/ML) decoding:***
  These codes are characterized by an important internal parameters:

  – the extra-repair symbols distribution, which impacts the correction capabilities;

  - the LDPC-Staircase coding rate (for a given global code rate), which impacts the correction capabilities;
  - the source variable node degree, which impacts the correction capabilities and decoding complexity.

  Based on finite length and asymptotic analysis, we tune these parameters to obtain the best configuration of GLDPC-Staircase codes over hybrid decoding.

- ***We evaluate the performance of these codes in different use cases:***
  More specifically, we measure the decoding overhead, the decoding failure probabilities and the error floor. We provide some insights in terms of decoding complexity. Finally, we compare with other erasure correcting codes such as LDPC-Staircase codes, Raptor codes, RaptorQ codes, and other GLDPC codes with a degree distribution associated to a close-to-capacity decoding threshold [5, 41].

- ***We analyze the impacts of the packet scheduling on the performance:***
  We define a methodology to measure the impacts of packet scheduling on any AL-FEC code both under IT and/or ML decoding, for a large variety of bursty erasure channels characterized by a finite-state Markov chain. Then we apply this methodology on GLDPC-Staircase codes. Thanks to this analysis, we define several recommendations on how to best use these codes, which turns out to be of utmost practical importance.

1.4 Thesis Outline

The rest of the thesis is organized as follows:

- In Chapter 2 we introduce information theory, linear block codes, decoding algorithms and the DE and EXIT asymptotic analysis tools

- In Chapter 3 we recall the main characteristics of LDPC-Staircase codes and we introduce the best construction of systematic RS codes over erasure channel.

- In Chapter 4 we explain the practical construction of GLDPC-Staircase codes and introduce two coding schemes. We discuss encoding and decoding aspects, then we derive an asymptotic analysis of these codes over the erasure channel for (IT+RS) and ML decodings.

- In Chapter 5 we tune important internal parameters of GLDPC-Staircase codes in order to obtain good performance under hybrid (IT/RS/ML) decoding.

- In Chapter 6 we perform a finite length analysis over memory-less channel to show the achieved decoding overhead, the decoding failure probability and the error floor. We also compare with other erasure codes such as LDPC-Staircase codes, Raptor codes, RaptorQ, and other construction of GLDPC codes.

- In Chapter 7 we propose a methodology to determine the impact of packet scheduling on decoding performance of any AL-FEC codes. Then we apply this methodology to analyze the impacts of the packets scheduling on the GLDPC-Staircase performance for different loss channels.
- Finally we discuss the results achieved and propose some future works.

Chapter 2

Literature review

Contents

2.1 Introduction 31
2.2 Channel coding overview 31
2.2.1 Channel coding foundation 31
2.2.2 Communication system 32
2.3 Erasure FEC codes 35
2.4 Block FEC codes 36
2.4.1 Definition 36
2.4.2 Decoding algorithms 37
2.4.3 LDPC codes 38
2.4.4 Reed Solomon codes 51
2.5 Rate-less erasure codes 53
2.5.1 Introduction 53
2.5.2 LT codes 53
2.5.3 Raptor codes 54

2.1 Introduction

In this section we briefly recall some important notions that are used in this research. Therefore, we start with some fundamental notions of information theory. Then we introduce block FEC codes, erasure FEC codes, Rate-less erasure codes, and their fundamental techniques (construction, encoding and decoding methods).

2.2 Channel coding overview

2.2.1 Channel coding foundation

Claude E. Shannon invented the fields of channel coding, source coding, and information theory in 1948 [7]. Shannon proved the existence of the channel codes which ensure

reliable communication through a noisy channel, when the coding rate was below the so-called *capacity of the channel*.
Unfortunately, this proof is not constructive, and the design of efficient codes in terms of error correction and complexity is always a topical problem. Following Shannon's publication, several of very effective coding systems had been designed to closely approach Shannon's theoretical limit.
We note that when these codes are used to correct errors occurring on coded information, in digital form, they are called ECC or FEC codes.

2.2.2 Communication system

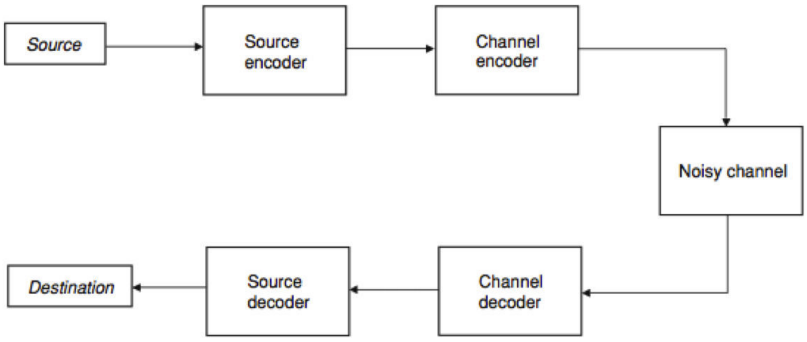


FIGURE 2.1: Communication system model.

Figure 2.1 gives a basic communication model, as related to information theory, which is composed basically on source, channel, source encoder/decoder, and channel encoder/decoder (FEC). Since our work focuses mainly on FEC codes, it is not of our interest to describe the source encoder/decoder in this Chapter. Some concepts relating to the source and channel are introduced in the following sections.

2.2.2.1 Source

The source is the data to be communicated, such as a computer file, a video sequence. It is generally regarded as a stream of random numbers governed by some probability distribution. Every source of data can be exactly quantified in terms of *entropy*.

Definition 2.1. *Entropy of random variable X is denoted by $H(X)$. It is a measure of the uncertainty in a random variable X. It quantifies the expected value of the information contained in a message. In other words, entropy is the average unpredictability in a random variable X, which is equivalent to its information content [54].*

- *For a binary source* X *having two outcomes occurring with probabilities p_s and $1-p_s$, the binary entropy function, denoted as either $H_2(X)$(indicating that it is the entropy of the source) or $H_2(p_s)$ (indicating that it is a function of the outcome probabilities) is (in bits)*

$$H_2(X) = H_2(p_s) = E[-log_2 P(X)] = -p_s log_2(p_s) - (1-p_s) log_2(1-p_s) \quad (2.1)$$

- *For a non binary source X having M outcomes $x_1, x_2, \ldots, x_M$, with probabilities $P(X = x_i) = p_i, i = 1, \ldots, M$, the entropy is (in bits)*

$$H(X) = E[-log_2 P(X)] = -\sum_{i=1}^{M} p_i log_2 p_i \tag{2.2}$$

Therefore, the entropy of a source is equal to the minimum number of bits required for encoding.

2.2.2.2 Erasure channels

A communication channel is characterized by an input set of symbols $X = \{x_1, x_2, \ldots, x_k\}$, an output set $Y = \{y_1, y_2, \ldots, y_n\}$ and a set of conditional probabilities $P(y_j/x_i)$. This conditional probability determines the relationship between the input x_i (transmitted symbol) and the output y_j (received symbol). The set of probabilities $P(y_j/x_i)$ is arranged into a matrix P_{ch} that characterizes the corresponding discrete channel: $P_{ij} = P(y_j/x_i)$. Each row in this matrix corresponds to an input, and each column corresponds to an output. The addition of all the values of a row is equal to *one*. Therefore, $\sum_{j=1}^{n} P_{ij}$ with $i = 1, 2, \ldots, k$.

The erasure channel causes only erasures in the transmitted codeword. Thus, the channel model will be described by the erasure probability, denoted by ε in this thesis. Therefore, the probability matrix P_{ch} is:

$$P_{ch} = \begin{pmatrix} 1-\varepsilon & \varepsilon & 0 \\ 0 & \varepsilon & 1-\varepsilon \end{pmatrix} \tag{2.3}$$

The channel capacity is given by $C_{ch}(\varepsilon) = 1 - \varepsilon$ and the stability function is given by $S_{ch}(\varepsilon) = \varepsilon$ [55, 56].

This channel was introduced by Elias in [57]. Following the type of the codeword i.e, it is a flow of bits or symbols (packets) we have two cases of channels:

- **The Binary Erasure Channel (BEC):**
  In this case the erasures affect the bits. Each bit of the codeword is correctly received with probability $1 - \varepsilon$ and is erased with probability ε. The channel model is given in Figure 2.2.

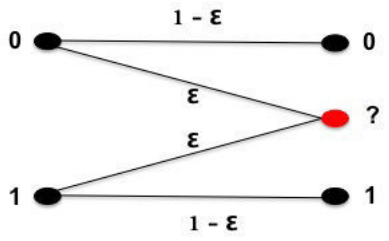


FIGURE 2.2: Erasure channel model.

- **Packet erasure channel:**
  In most applications, the erasures do not affect the bits but rather groups of bits, ranging from a few bytes to several gigabytes. Therefore, it is necessary to introduce a more general model of channel called *packet erasures channel* in which the symbols are words of several bits. Each packet of the codeword is correctly received with probability $1-\varepsilon$ and is erased with probability ε.
  On this channel, the packets are correctly received, or are completely erased. Most of the time the integrity of the packet is guaranteed by error detection systems (Cyclic Redundancy Control (CRC), TCP checksum or User Datagram Protocol (UDP) checksum, hash function). This channel is found in many practical applications such that communication networks, distribution of data via a carousel and distributed storage. The location of the erasures is known, and it remains only to correct them. It is possible to make a permutation of packets. Indeed, some codes can be sensitive to the loss burst. Therefore, the permutation of packets allows, viewpoint of code, transform these consecutive erasures to scattered erasures in the codeword, thereby improving the performance of the system.

After the transmission over the channel, the receiver tries to deduce what X has transmitted knowing the received value of Y. Thus, we talk about the Mutual Information (MI). As shown in Figure 2.3, the MI of two random variables X and Y is a quantity that measures the mutual dependence of the two random variables. It is denoted by $I(X;Y)$. The most common unit of measurement of MI is the bit, when logarithms to the base 2 are used.

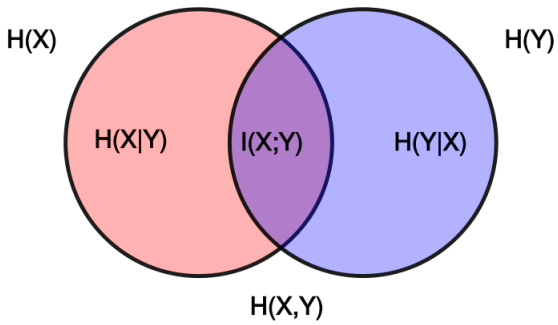


FIGURE 2.3: $H(X)$, $H(Y)$, joint ($H(X,Y)$), and conditional entropies for a pair of correlated subsystems X,Y with mutual information $I(X;Y)$.

This quantity can be expressed as follows [58]

$$\begin{aligned} I(X;Y) &= H(X)-H(X|Y) \\ &= H(Y)-H(Y|X) \\ &= H(X)+H(Y)-H(X,Y) \\ &= H(X,Y)-H(X|Y)-H(Y|X) \end{aligned} \tag{2.4}$$

$H(X)$ and $H(Y)$ are the marginal entropies, $H(X|Y)$ and $H(Y|X)$ are the conditional entropies, and $H(X,Y)$ is the joint entropy of X and Y [58].

Since the entropy, $H(X)$, is regarded as a measure of uncertainty about the random variable then [58]:

- $H(X|Y)$ is a measure of what Y does not say about X. This is *the amount of uncertainty remaining about X after Y is known* ;
- $I(X;Y) = H(X) - H(X|Y)$ is
  - the amount of uncertainty in X, minus the amount of uncertainty in X which remains after Y is known
  - the amount of uncertainty in X which is removed by knowing Y.

Therefore, MI is the amount of information (that is, reduction in uncertainty) that knowing either variable provides about the other [58].
Note that in the perfect channel (i.e., without error), we have $H(X|Y) = 0$. Indeed Y is completely determined by the value of X, so there is no uncertainty about this first. This means that the MI of the two variables is maximum and $I(X;Y) = H(X)$. In the worst case, the channel is so bad and the X and Y are completely independent, we have $H(X|Y) = H(X)$ and therefore the MI is minimal (i.e. $I(X;Y) = 0$). The channel input has no effect on the output of the channel, this is the worst possible case, and it is impossible to transmit information on this channel.
Based on the MI, the channel capacity (C_{ch}) is defined as follows:

$$C_{ch} = max_X\{I(X;Y)\} \tag{2.5}$$

Thus, in the case of perfect channel the capacity is maximum ($C_{ch} = max_X\{H(X)\}$), whereas in the worst case of the channel the capacity is zero ($C_{ch} = 0$).

2.3 Erasure FEC codes

The aim of (n, k) FEC code is as follows. The sender encodes, using FEC code, the k source symbols. They are sent on the channel and therefore subject to distortions, caused by the channel. The receiver uses the received codewords and (hopefully) recovers the original message after FEC decoding.

Considering an erasure channel with erasure probability ε, the n symbols are subject to losses. Therefore, $(\varepsilon * n)$ symbols will be lost and $(1-\varepsilon) * n$ symbols will be available at the receiver. If the number of received symbols is greater than or equal to k, then the decoding is (usually) possible, and the receiver reconstructs the original source symbols. In the opposite case, the decoding fails and the receiver will not have the totality of the source symbols.
Unlike error FEC codes, erasure FEC codes generally work within the *communication stack upper layers*. Unlike error codes that work on small amounts of data (a few thousand bits), erasure codes often work on data sizes ranging up to several megabytes. Erasure FEC codes are found mostly just below the application layer and are called ***AL-FEC codes***. The symbol sizes are usually several hundreds of bytes long and are assumed to be of the same size (padding may be required from time to time, e.g. for the last symbol of a file). An IP

datagram then contains one (sometime several) symbol.

The encoding of symbols with codes defined on the finite field $GF(2)$ (binary codes) is performed with XOR operations. These XOR operations can be performed at a bit level, in parallel for all the bits of all the symbols of the block. Alternatively, they can be performed at word level (e.g., 64 bits at a time on a 64-bit OS/machine), for all the words of the symbols of the block (which is faster). With Reed-Solomon codes on the finite field $GF(2^8)$ for instance, operations are performed at the element level, in that case 8 bit-elements, for all the elements of all the symbols [47].

2.4 Block FEC codes

In this section, we define a kind of FEC codes called block FEC codes and we describe briefly the features of some interested codes.

2.4.1 Definition

Block coding consists in associating a block of k source symbols to a codeword c of n encoding symbols where $n \geq k$. The difference $(n-k)$ is the redundancy introduced by the code. The knowledge of coding rules at the receiver allows to detect and correct errors under certain conditions. The ratio $\frac{k}{n}$ is called coding rate of the code.
They can also be capable of correcting data losses called erasures. The central of block codes, is the parity check matrix denoted by H, of size $m \times n$ where $m = n-k$. This matrix defines a linear system of m check (or parity) equations. This system of linear equations can be represented graphically with the *bipartite graph* of the code. This graph represents the connexions between nodes belonging to two different classes:

- Variable nodes: they correspond to the symbols of codeword $(v_j, j \in \{1,\ldots,n\})$, i.e, to the columns of H.
- Check nodes: they correspond to the parity equations $(c_i, i \in \{1,\ldots,m\})$, i.e, to the rows of H.

Each arc in the graph connects variable node v_j to the check node c_i and corresponds to a value "1" in the i^{th} row and the j^{th} column of H.
The encoding of linear block codes is based on the generator matrix. It allows to generate the m redundancy symbols using the k data symbols. Let $X = (x_1,\ldots,x_k)$ represents the data symbols. X is encoded to codeword $Y = (y_1,\ldots,y_n)$ by a simple multiplication with the generator matrix of code $C(n,k)$ as follows:

$$Y = X * G \tag{2.6}$$

When the data vector is in the encoded vector, therefore $C(n,k)$ is called a *systematic* code and its generator matrix contains the identity matrix as follows:

$$G_{sys} = [I_k | P_{k,n-k}]. \tag{2.7}$$

Since H and G matrices are orthogonal ($G.H^T = 0$), then we have:

$$H = [P^T_{n-k,k} | I_{n-k}]. \tag{2.8}$$

2.4.2 Decoding algorithms

Before explaining the decoding methods of the block codes, let us define a few additional important notions.

Definition 2.2. *The Hamming weight (w_H) of a sequence of symbols, in a given alphabet, is the number of non null symbols.*

Definition 2.3. *The Hamming distance (d_H) between two sequences of symbols $X1$ and $X2$ is the number of positions for which the corresponding symbols are different. In other words, the Hamming distance is the Hamming weight of the difference $X1$-$X2$.*

The application of the concept of the Hamming distance of a code, leads to the notion of the minimum of these distances, called the *minimum distance* of the code.

Definition 2.4. *Minimum distance (d_{min}) of a linear block code, denoted by d_{min}, is the smallest Hamming distance of all codewords. A code $C(n,k)$ with minimum distance d is denoted by $C(n,k,d)$ code.*

Based on the notion of the distance, the decoding of block codes has the objective of finding the sequence that has been most probably sent and closer to that received. The distance used is the *Hamming distance* resulting from the *Hamming weight* function.
The correction capabilities of a block code are highly dependent on several metrics such as the minimum distance. This minimum distance gives a bound on the correction capabilities: a code of minimum distance d_{min} will always be able to correct if,

$$d_{min} \geq 2t + 1 + e, \tag{2.9}$$

where e is the number of erasures and t is the number of errors.
This minimum distance has an upper bound that depends on the length (n) and dimension (k) of the code. The codes that achieve this upper bound are called *Maximum Distance Separable (MDS)* codes, and have optimal correction capabilities. Therefore, a linear block code is an MDS code iff:

$$d_{min} = n - k - 1, \tag{2.10}$$

and it reaches the Singleton bound [59].

Assume that we send a codeword X and we receive a channel output Y. The errors introduced by the channel avoids to determining which codeword was sent with absolute certainty. We can find the *most likely* codeword that was sent, in the sense that the *probability* that this codeword was sent given Y, is maximized. This means that list all the possible codewords and calculate the conditional probability for each of them. Then, from the list, find the codewords that give the *maximum likelihood* and return one of them. It may be erroneous, but it is the best. This decoder is called the *ML* decoder. However, this optimality has a cost in complexity (it takes a lot of time and operations), which can be quickly prohibitive as the code length increases. Therefore, the ML problem has been shown to be *NP-hard* for many classes of codes (e.g., general linear codes over F_q for any q). For the erasure channel, the ML decoding allows to solving a linear system with a Gaussian Elimination (GE) method.
Therefore, all the decoding approaches are proposed in order to get closer the optimal capacity of the ML decoding.

2.4.3 LDPC codes

2.4.3.1 History

Robert Gallager gave the birth of LDPC codes in his PhD thesis in 1960 [17]. These codes are the class of linear block codes at the heart of iterative-coding idea and they can be defined in terms of a sparse parity-check matrix. These codes are quite general and have been shown to exhibit very good performance under iterative Belief Propagation (BP) decoding (see section 2.4.3.4) with moderate decoding complexity over a wide range of communication channels. They have been firstly proposed in a *regular* way (see section 2.4.3.2). Shortly after the rediscovery of LDPC codes, a new type of LDPC codes has been introduced in [60] in order to obtain *irregular* LDPC codes (see section 2.4.3.2).
LDPC codes have been intensively studied during the last decade due to their excellent asymptotic performance (i.e., near-Shannon limit performance) over a wide range of channels under BP decoding [20, 22, 25–29]. Over BEC, it was shown that it is possible to design sequences of degree distributions, known as capacity-achieving sequences whose threshold converges to the channel capacity [31]. It also has been proved that irregular LDPC codes are able to asymptotically achieve the BEC capacity for any code rate [30, 31]. Whereas, it has shown that this very good asymptotic performance in terms of decoding threshold does not necessarily correspond to a satisfying finite length performance. In fact, finite length LDPC codes with a degree distribution associated to a close-to-capacity decoding threshold, though typically characterized by very good waterfall performance, are usually affected by high error floors [32–35]. When considering transmission on the BEC, the low weight codewords induce small stopping sets (see section 2.4.3.6), thus resulting in *high error floors* [61]. Therefore, the design good LDPC codes or codes combined with LDPC codes with excellent correction capabilities in finite length is the objective of many researchers. One of the features that makes LDPC codes very attractive is the possibility to design, for several transmission channels, the bipartite graph degree distribution for the variable and check nodes in order to obtain a decoding threshold which is extremely close to the channel capacity [25]. Then, the design of finite length LDPC codes mostly relies on the search for the best compromise between the two regions of the performance curve, by carefully constructing the bipartite graph. Among the several techniques that have been proposed to design good LDPC codes and they are quite effective, those based on finite geometries [20], on the progressive-edge-growth (PEG) construction [62], on the Irregular Repeat Accumulate (IRA) construction [63], on circulant permutation matrices [26], and on protographs [64]. These techniques, or their combinations, lead to codes with good properties (in terms, for instance, of girth of the bipartite graph and of the possibility to perform the encoding procedure efficiently).
LDPC codes have also formed the basis for several codes, such as LT codes (see section 2.5.2) and Raptor codes (see section 2.5.3), as well as countless variations of LDPC codes such as GLDPC codes [36], the LDPC codes based on protograph, and Quasi-cyclic LDPC codes.

2.4.3.2 Description and Construction

The name comes from the fact that H contains a small number of non-zero values in comparison to the amount of zero values (i.e., it is a *sparse matrix*). Since the LDPC codes

are linear block codes, there are two different possibilities to represent them. They can be described via parity check matrix H_{LDPC} or with a graphical representation called *bipartite graph*. Therefore, the associated bipartite graph of the following matrix is shown in Figure 2.4.

$$H_{LDPC} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \tag{2.11}$$

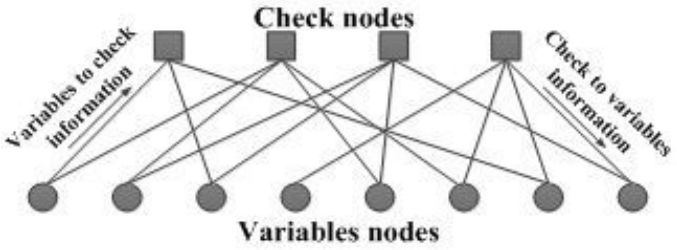


FIGURE 2.4: Bipartite graph of LDPC code (8,4).

Alon and Luby [65] made the first attempt to design an LDPC code capable of correcting erasures.
As mentioned previously, LDPC codes are divided into two sets as follows:

- Regular LDPC codes: in this case the parity check matrix H_{LDPC} contains a constant number d_c of "1" in each row and a constant number d_v of "1" in each column. Therefore, the variable nodes are of degree d_v and the check nodes are of degree d_c. These codes are denoted by (d_v, d_c) LDPC codes. The rate of LDPC codes can be determined by:

$$R = 1 - \frac{d_v}{d_c} \tag{2.12}$$

  Gallager has proposed to create a regular LDPC code. The construction consists of filling the sparse parity check matrix H_{LDPC} by randomly determining the positions of "1", with a fixed number of ones "1" per column and per row.

- Irregular LDPC codes: in this case H_{LDPC} is of low density but the numbers of "1" in each row or column are not constant. The irregularities of variable and check nodes are defined by polynomials. Let d_{vmax}, d_{cmax} be the maximum degrees of variable nodes and check nodes respectively. Let $\lambda_i(resp.\rho_i)$ is the fraction of edges connected to variable-nodes (resp. check-nodes) of degree i. $L_i(resp.R_i)$ represents the fraction of variable-nodes (resp. check nodes) of degree i. Any LDPC graph is specified by the sequences $(\lambda_1,\ldots,\lambda_{d_{vmax}})$, $(\rho_1,\ldots,\rho_{d_{cmax}})$, $(L_1,\ldots,L_{dvmax})$ and $(R_1,\ldots,R_{dcmax})$. Further, the polynomials are defined as follows:

$$\lambda(x) = \sum_{i=1}^{dvmax} \lambda_i.x^{i-1} \quad \text{and} \quad \rho(x) = \sum_{i=1}^{dcmax} \rho_i.x^{i-1}, \tag{2.13}$$

to be the edge-perspective Degree Distribution (DD) polynomials and from a node perspective, the DD polynomials are given by

$$L(x) = \sum_{i=1}^{dvmax} L_i.x^i \quad \text{and} \quad R(x) = \sum_{i=1}^{dcmax} R_i.x^i. \tag{2.14}$$

Therefore the design rate of the code will be,

$$R = 1 - \frac{\sum_i \rho_i / i}{\sum_j \lambda_j / j} \tag{2.15}$$

or equivalently

$$R = 1 - \frac{\int_0^1 \rho(x)\mathrm{d}x}{\int_0^1 \lambda(x)\mathrm{d}x}. \tag{2.16}$$

In general, irregular LDPC codes can significantly outperform regular LDPC codes [21]. All LDPC codes, which can approach the Shannon limit on different channels are irregular LDPC codes. The irregular LDPC code allows to improve the performance in the waterfall region and it will be better than regular LDPC codes. The drawback of irregular LDPC codes is that they present an error floor (e.g., over BEC is caused by *small stopping sets* [61], and over AWGN channel is caused by small trapping sets and by the minimum distance [33]). Therefore, several construction methods for irregular LDPC codes exist [66].

LDPC codes can also be classified, according to the construction method used for generating the corresponding sparse parity check matrix H, into:

- random LDPC codes, and
- structured LDPC codes.

In general, random LDPC codes show a slightly better correction capabilities than that of structured LDPC codes, but these latter codes can be designed so that they are much less complex to encode and decode than the former codes. The construction approach proposed by MacKay in [19, 67] is random, while other approaches include those based on finite field geometries, balanced incomplete block designs and cyclic or quasi-cyclic structures [20, 68]. The discovery of irregular LDPC codes influenced consideration of irregular structures for other codes defined on graphs such as irregular turbo codes [69] and IRA codes [63]. In [20, 70, 71], they show that structured LDPC codes which are constructed algebraically, in general, have much lower error-floor rather than unstructured random or pseudo-random LDPC codes constructed using computer-based methods or algorithms.

2.4.3.3 Encoding LDPC codes

The encoding of LDPC codes can be performed by two methods. The first method is to use the *generator matrix*. It consists in multiplying the data sequence by the generator matrix to get the sequence corresponding to the codeword. However, this approach has two drawbacks:

- the generator matrix construction (systematic or non systematic): LDPC codes are defined by their parity check matrix. Therefore, using the parity check matrix, the generator matrix must be calculated first and this transformation requires a GE method in general case. This can sometimes be done in advance and therefore has no impact on the complexity of the encoding step. However, when the code is generated on the fly, it is impossible to perform this pre-calculus.

- the complexity: the multiplication of the generator matrix (which is dense in general) with the data is very expensive. It requires *(O(n-k)n)* operations.

In practice, the generator matrix is not used for encoding with LDPC codes owing to the important length of codewords. Oenning and al [72] propose to construct directly a systematic sparse generator matrix in such way that the parity check matrix remains sparse. These codes are called Low Density Generator Matrix (LDGM) codes. Their performances are however mediocre [19] even if it is possible to optimize their construction [73] to reduce the error floor.

The second method is to use the *parity check matrix*. Richardson and al [74] proposed an approach to reduce the encoding complexity. This method transforms the parity check matrix H, which defines the code, into a parity check matrix with a nearly triangular structure on the lower right portion of the matrix using only rows and columns permutations. Then, we solve the linear system, of reduced size, represented by the following matrix equations:

$$c.H^T = 0, \tag{2.17}$$

and by a substitution step down (*backward substitution*) we can encode with a lower cost. Another approach is to consider the LDPC codes whose their parity check matrices are subdivided into two parts. More precisely, the right part is a staircase matrix (i.e., it is the part of redundant symbols, which has a *lower triangular structure*) while the left part is a *sparse matrix with no particular structure*. These codes are called Repeat Accumulate (RA) codes [75]. During encoding, each redundant symbol depends on the previous ones, so there is an effect of *accumulation*. Moreover, the data symbols are present in several equations which means they are *repeated*. The left part can be either regular (we obtain Regular Repeat Accumulate codes), or irregular (we obtain IRA codes). Follow the "stair" fashion, each repair symbol is produced by assigning its value to the sum of the previous symbols, which are presented on the associated row. The encoding can be done in linear time, provided these symbols are generated in their natural order. Section 3.2 provides a specific example of LDPC codes with lower triangular structure, called LDPC-Staircase codes [46] that we have consistently followed in our work.

2.4.3.4 Decoding LDPC codes over an erasure channel

The sparseness of the LDPC graph is the key feature that led to determine iterative decoding algorithms on graph whose complexity is manageable even for very long code lengths (up to thousands of bits). Therefore, the general decoding algorithm for LDPC codes is the BP decoding algorithm or the IT decoding algorithm. The reason for this name is that at each iteration of the algorithm, messages are passed from variable nodes to check nodes, and and vice versa.

The messages that are sent from the variable nodes to the check nodes are computed using the channel observed value and some of the messages passed from their neighboring check nodes. Whereas the messages that are sent from the check nodes to the variable nodes are computed using some of the messages passed from their connected variable nodes. An important aspect is that the message that is sent from a variable node V to a check node C must not take into account the message sent in the previous iteration from C to V. The same is true for messages passed from check nodes to variable nodes.

The LDPC codes have first been used for reliable transmission over the noise channels such as BSC and AWGN channel. Efforts were then concentrated on the improvement of LDPC codes and their decoders for these types of channels. Afterwards, new applications such as the internet and distributed storage bring the need to effectively address the loss of data. Therefore, LDPC codes and their decoders were quickly adapted to transmit on erasure channels. The IT algorithm over an erasure channel can be determined from the general case (over AWGN channel) as presented in [17, 76]. Therefore, it is sufficient to assume that the all-zero codeword was sent and the log-likelihood of the variable nodes (m_v), at iteration 0, is equal to:

$$m_v = \begin{cases} +\infty & \text{if the corresponding message bit is not erased} \\ 0 & \text{if the message bit is erased.} \end{cases} \tag{2.18}$$

Moreover, consulting the update equations for the messages, we see that if V is not erased, then the message passed from V to any of its incident check nodes is always $+\infty$. The update equations also imply that $m_{c\to v}$ is equal to:

$$m_{c\to v} = \begin{cases} +\infty & \text{iff all the variable nodes incident to } C \text{ except } V \text{ are not erased} \\ 0 & \text{in all other cases.} \end{cases} \tag{2.19}$$

If V is an erased variable node, then $m_{v\to c} = 0$. The message $m_{v\to c}$ is $+\infty$ if and only if there is some check node incident to V other than C which was sending a message $+\infty$ to V in the previous iteration.
Zyablov in [77] proposed the first algorithm of IT for the BEC. Because of the binary feature of the messages, the IT decoding for an (n, k) LDPC code can be described much easier, as shown the algorithm 1 [78].
The IT decoding can also be interpreted on the parity check matrix (equations system): on each iteration, if an equation of the system has one variable whose value is unknown, then it is assigned to the value of the corresponding constant member of this equation. After that, we inject this value in all the equations where this variable node is used. The decoding converges and stops when a stable state is achieved after a certain number of iterations.
The number of operations that this algorithm performs, is proportional to the number of edges in the graph. Hence, for sparse graphs, the algorithm runs in linear time in the block length (n) of the code. However, there is no guarantee that this algorithm can decode all variable nodes because it depends on the graph structure (see section 2.4.3.6). In other words, the IT decoding can not fully exploit the correction capabilities of the code. This sub-optimality is the price to pay for a linear complexity algorithmic.

An ML decoding is another possible decoding application. This decoding achieves the *optimal correction capabilities* at the cost of increased complexity [79, 80]. Over

Algorithm 1 IT decoding of LDPC codes over an erasure channel

```
1: Initialization                          ▷ Initialize the values of all the check nodes to zero
2: for c = 1 to m do
3:     m_c = 0
4: end for
5: Direct recovery                         ▷ update messages sent from nodes V to nodes C (m_{v→c})
6: for v = 1 to n do
7:     if node v is received then
8:         for c = 1 to m do
9:             m_c = m_c + m_vc
10:            delete e_vc
11:            deg^1(c) = deg(c) - 1
12:        end for
13:    end if
14: end for
15: Substitution recovery                  ▷ recover the erased nodes V if possible
16: for c = 1 to m do
17:    for v = 1 to n do
18:        if deg(c) = 1 and e_vc not delete then
19:            m_v = m_c
20:            delete e_vc
21:            go to 7
22:        end if
23:    end for
24: end for
25: if All erased variable nodes are recovered then
26:    Status OK
27: else
28:    Status Failure
29: end if
```

an erasure channel the ML decoding is equivalent to resolve a linear system using the GE method. Therefore, the decoding complexity becomes cubic (theoretical complexity). Recently, the ML decoding complexity was further reduced by using the Structured Gaussian Elimination (SGE) approach introduced simultaneously by LaMacchia/Odlyzko and Pomerance/Smith [81, 82].

In order to combine the advantages of both decoding types, an hybrid (IT/ML) decoding approach is proposed in [83–85]. When the receiver receives the symbols, the main operations of hybrid decoding are as follows:

- Step 1: the IT decoder triggers,
- Step 2: the ML decoder triggers if Step 1 fails (i.e., the IT decoder can not recover all the erased source symbols and no additional symbol is expected) on the simplified linear system by the IT decoding,

- Step 3: if ML fails to recover all the erased source symbols, the hybrid decoding fails, otherwise it succeeds.

This hybrid decoding provides excellent correction capabilities with a reduced complexity compared to the ML decoding [83–85]. Indeed, in the failure case at Step 1, the IT decoding allows to reconstruct a certain number of erased symbols that lead to reduce the system's size of the ML decoding that will be triggered. Moreover, the correction capabilities, obtained with the hybrid (IT/ML) decoding, are not impacted by the IT decoding and only depend on the ML decoding. Therefore, the IT decoding reduces the overall decoding complexity while the ML decoding achieves the superior capabilities of the correction.

2.4.3.5 Asymptotic analysis tools for LDPC codes

2.4.3.5.a Introduction

We present in this section two asymptotic analysis tools that are used in this thesis: DE and EXIT functions. These techniques provide an exact analysis for infinite-length codes, and an approximate analysis for finite-length codes. To perform these methods, it must assume that the code length $N \rightarrow \infty$, under which it may be assumed there are *no cycle* in the bipartite graph (the messages passed on the edges are statistically independent). From viewpoint of the complexity, the DE technique requires intense calculations and in some cases it becomes intractable, whereas the EXIT technique is fast and applicable to many iterative decoders.

- The DE technique has been proposed by Richardson and al. to understand the limits of performance of LDPC decoders [23]. It tracks the evolution of the *probability density function (pdf)* of the messages during the decoding process. We can notice that the DE technique is not specific to LDPC codes. It is a technique which can be adopted for other codes defined on graphs associated with an iterative decoding. However, it becomes intractable when the codes composition is complex (e.g., turbo codes).

- The EXIT technique, were first introduced by Ten Brink in [86–88] as a technique to analyze the convergence of an iterative decoding process of parallel concatenated component codes. It tracks only *one parameter per iteration* unless the pdf of the messages. For instance, one might track a statistic of the extrinsic-values based on their mean, variance, an error probability [89, 90], the Signal Noise Ratio (SNR) of the extrinsic messages [91, 92], a fidelity or a mutual information between messages and decoded bits [87]. A comparison of some of these metrics [93] proves that mutual information seems to be the most accurate and most robust statistic between them. This technique appeared as a handy tool to visualize the iterative decoding process in a graph, called *EXIT chart*, which shows the "bottlenecks" in the iterative decoding process. This technique is very popular, as it provides deep insight to the behavior of iterative decoders. It allows to analyze iterative decoders including codes with complicated constituent codes [86–88, 91, 92].

These tools introduce the idea of the *decoding threshold* of a code in order to show that this code performs well or not (the probability of error is non-negligible). This decoding

threshold is the only parameter that characterizes the performance of a code and is used to measure the gap to the channel capacity.
For a given channel, these methods are also used in the design of families of LDPC codes, since their performance can be predicted by the tracked message method, which is faster than by simulation.

In the following, we summarize relevant characteristics of the DE and EXIT tools over an erasure channel.

2.4.3.5.b DE technique for LDPC codes over an erasure channel

Over an erasure channel, the DE recursively computes the fraction of erased messages passed during the IT decoding.

Theorem 2.1. *[23] Consider a degree distribution pair (λ, ρ) of LDPC codes. Over an erasure channel with probability ε, the DE recursion can be written in closed form as follows:*

$$p^{l+1} = \varepsilon.\lambda(1-\rho(1-p^l)), \tag{2.20}$$

Let us more explain this theorem. The fraction of erased messages sent from check nodes to variable nodes (resp. from variable nodes to check nodes) at l iteration are denoted by q^l (resp. p^l).
At the first iteration $l = 0$, the initial variable-to-check messages which are equal to the received message with probability ε (i.e., $p^0 = \varepsilon$).
Then, we start with the check-to-variable messages in the $(l+1)^{th}$ iteration. According to the belief propagation algorithm, a check-to-variable message, emitted by a check node m of degree i along a particular edge, is not an erasure message iff the $(i-1)$ incoming messages from connected variable nodes are not erasures. Assuming that each incoming erasure message has a probability p^l and all the messages are independent, therefore the probability that the outgoing message is an erasure, is given by:

$$q^{l+1}(m) = 1-(1-p^l)^{i-1}. \tag{2.21}$$

Since the edge has probability ρ_i to be connected to a check node of degree i, it follows that the expected average erasure probability of a check-to-variable message in the $(l+1)^{th}$ iteration is equal to:

$$\begin{array}{rcl} q^{l+1} & = & \sum_{i=1}^{d_{vmax}} \rho_i (1-(1-p^l)^{i-1}) \\ & = & 1-\rho(1-p^l) \end{array} \tag{2.22}$$

Similarly, we consider the erasure probability of the variable-to-check messages in the $(l+1)^{th}$ iteration. Consider an edge e, which is connected to a variable node s of degree i. The outgoing variable-to-check message along this edge in the $(l+1)^{th}$ iteration is an erasure, if the received channel value of the associated variable node is an erasure and all the $(i-1)$ incoming messages are erasures. This with probability comes:

$$\begin{array}{rcl} p^{l+1}(s) & = & \varepsilon.(q^{l+1})^{i-1} \\ & = & \varepsilon.(1-\rho(1-p^l))^{i-1} \end{array} \tag{2.23}$$

Averaging over the edge degree distribution λ, we get the DE recursion as follows:

$$p^{l+1} = \varepsilon.\lambda(1-\rho(1-p^l)). \tag{2.24}$$

Given a degree distribution pair (λ,ρ) and a real number $\varepsilon \in [0,\ \varepsilon^{IT}]$ (ε^{IT} is the erasure threshold of the IT decoding for a code ensemble), the condition for convergence can hence be written as:

$$p^l > \varepsilon.\lambda(1-\rho(1-p^l)) \quad \text{for} \quad p^l \in (0,1]. \tag{2.25}$$

The IT decoding threshold, ε^{IT}, is defined in [22] as:

$$\varepsilon^{IT} = sup\{\varepsilon \in [0,1] : \varepsilon.\lambda(1-\rho(1-p^l)) < p^l\}. \tag{2.26}$$

Operationally, when $n \to +\infty$, if we transmit at $\varepsilon \leq \varepsilon^{IT}$, then all the bits can be recovered and if $\varepsilon > \varepsilon^{IT}$, then a fixed fraction of bits remains erased after IT decoding.

2.4.3.5.c EXIT technique for LDPC codes over an erasure channel

In this section, we present two methods to determine the EXIT functions of LDPC codes, which differ from the tracked parameter.

Method 1: EXIT functions based on mutual information
As mentioned above, instead of tracking the density of messages, this technique tracks the evolution of a single parameter iteration by iteration. In literature, the term "EXIT" is usually used when mutual information is the parameter whose evolution is tracked. Over an erasure channel, the EXIT functions turn out to be one minus the fraction of erasures being passed from one side of the graph to the other [94]. Therefore, the tracked parameter measures the decoder's success. This technique computes:

- The average extrinsic information coming out of the variable nodes, denoted by $I_{E,v}$, using the average *a prior* information going into the variable nodes denoted by $I_{A,v}$ and the erasure probability, ε, coming from channel as shown in Figure 2.5,

$$I_{E,v} = f(I_{A,v}, \varepsilon) \tag{2.27}$$

- The average extrinsic information coming out of the check nodes, denoted by $I_{E,c}$, using the average a *prior* information going into the check nodes denoted by $I_{A,c}$ as shown in Figure 2.6,

$$I_{E,c} = f(I_{A,v}) \tag{2.28}$$

Denote the extrinsic erasure probability that is exchanged between variable nodes and check nodes during the iterative decoding process by p. The value of p depends on the decoding iteration [95].
Consider a variable node v of degree d_v with $I_{A,v}$ = 1- p, the extrinsic information coming out is equal to [95]:

$$\begin{aligned} I_{E,v}(v) &= 1-\varepsilon.p^{d_v-1}. \\ &= 1-\varepsilon.(1-I_{A,v})^{d_v-1} \end{aligned} \tag{2.29}$$

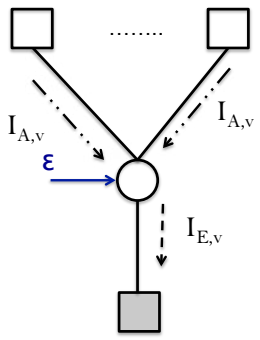


FIGURE 2.5: Extrinsic information processing of variable node V.

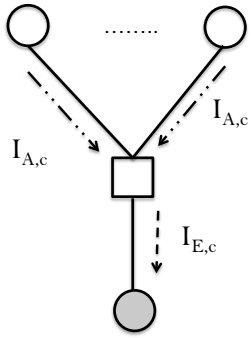


FIGURE 2.6: Extrinsic information processing of check node C.

The average extrinsic information for all the variable nodes is equal to:

$$\begin{aligned} I_{E,v} &= \sum_{i=1}^{d_{vmax}} \lambda_i (1 - \varepsilon.(1 - I_{A,v})^{i-1}) \\ &= 1 - \varepsilon.\lambda(1 - I_{A,v}) \end{aligned} \tag{2.30}$$

Consider a check node c of degree d_c with $I_{A,c}$ = 1- p, the extrinsic information coming out is equal to [95]:

$$\begin{aligned} I_{E,c}(c) &= (1-p)^{d_c-1}. \\ &= (I_{A,c})^{d_c-1} \end{aligned} \tag{2.31}$$

The average extrinsic information for all the check nodes is equal to:

$$\begin{aligned} I_{E,c} &= \sum_{i=1}^{d_{cmax}} \rho_j((I_{A,c})^{j-1}) \\ &= \rho(I_{A,c}) \end{aligned} \tag{2.32}$$

EXIT chart of an LDPC code is a plot of the EXIT function of the variable nodes, $I_{E,v}$, and the inverse of the EXIT function of the check nodes, $I_{E,c}^{-1}$.
The LDPC code can be seen as a turbo code where the check nodes are the first decoder and variable node are the second decoder where they exchange the messages during the iterative decoding process. Therefore, the output of the first decoder is the input of the second decoder and vice versa. Then we have,

$$I_{A,v} = I_{E,c} \quad \text{and} \quad I_{A,c} = I_{E,v} \tag{2.33}$$

Therefore using equations (2.30), (2.32) and (2.33) we obtain at $(l+1)^{th}$ iteration,

$$I_{E,v}^{l+1} = 1 - \varepsilon.\lambda(1-\rho(I_{E,v}^{l})) \tag{2.34}$$

We can notice from equations (2.34) and (2.24) that, over an erasure channel, the EXIT technique is equivalent to DE technique.

Method 2: EXIT functions based on the entropy
For binary LDPC codes, a slightly different definition of the EXIT curve has been introduced by Méasson in [96], where the EXIT curve is associated with the sparse graph system rather than with component codes. Roughly speaking, the EXIT curve gives the fraction of erased bits "contained" in the extrinsic information produced by the decoding algorithm, assuming that the code length tends to infinity. It is based on the computation of the extrinsic probability in terms of *entropy*. This EXIT curve can be defined for any decoding algorithm (e.g. IT or ML decoding), and it relates to the asymptotic performance of an ensemble of codes under the considered decoding. Obviously, in case of IT decoding, there is a tight relation between the EXIT curve and the DE equations derived in section 2.4.3.5.b.

This technique determines the asymptotic average (on all variable nodes) extrinsic erasure probability at the output of an IT decoding, denoted by $h_{LDPC}^{IT}(\varepsilon)$. This IT EXIT function is given by,

$$h_{LDPC}^{IT}(\varepsilon) = \frac{1}{n}\sum_{i=1}^{n} h_i^{IT}(\varepsilon), \quad \text{with} \quad n \to +\infty. \tag{2.35}$$

where $h_i^{IT}(\varepsilon)$ is the extrinsic erasure probability of symbol "i" after IT decoding [96].
The IT EXIT is computed using DE equations, for a fixed ε, as follows:

1. Let $p^{\infty} = \lim_{l \to +\infty} p^l$ (The value of p^{∞} can be determined by recursively computing p^l until it reaches a fixed (limit) value)
2. Take q^{∞}, where $p^l \xrightarrow[l \to +\infty]{} p^{\infty}$
3. Compute

$$h_{LDPC}^{IT}(\varepsilon) = \sum_{i=1}^{d_v} L_i q^{\infty,i} = L(q^{\infty}), \quad \text{since L is a finite-degree polynomial.} \tag{2.36}$$

Therefore, the IT EXIT curve is given in parametric form by [53]:

$$h_{LDPC}^{IT}(\varepsilon) : \begin{cases} 0 & \text{when} \quad \varepsilon \in [0\ \varepsilon^{IT}] \\ L(q^{\infty}) & \text{when} \quad \varepsilon \in]\varepsilon^{IT}\ 1] \end{cases} \tag{2.37}$$

After plotting h_{LDPC}^{IT} vs ε, the IT decoding threshold, denoted by ε^{IT}, represents the maximum value of ε where the h_{LDPC}^{IT} jumps from 0 to another value different from 0.

The exact computation of the EXIT function for the ML decoding is a difficult task [96]. Therefore, there is another application of EXIT functions: they can be used to connect the performance of a code under IT decoding to that under ML decoding using the area theorem [96]. The area theorem determines the relationship between the area under the ML EXIT function (in terms of entropy) after an ML decoding, denoted by $h_{LDPC}^{ML}(\varepsilon)$, and the asymptotic rate (R) of the LDPC code as following:

$$\int_0^1 h_{LDPC}^{ML}(\varepsilon) \cdot d\varepsilon = R. \tag{2.38}$$

We have,

$$\int_0^{\varepsilon^{ML}} h_{LDPC}^{ML}(\varepsilon) \cdot d\varepsilon = 0, \tag{2.39}$$

therefore equation (2.38) is equal to,

$$\int_{\varepsilon^{ML}}^1 h_{LDPC}^{ML}(\varepsilon) \cdot d\varepsilon = R, \tag{2.40}$$

where ε^{ML} is the threshold of ML decoding.
The sub-optimality of the IT decoding implies,

$$h_{LDPC}^{ML}(\varepsilon) \leq h_{LDPC}^{IT}(\varepsilon). \tag{2.41}$$

Therefore, the equation (2.40) becomes equal to [96],

$$\int_{\bar{\varepsilon}^{ML}}^1 h_{LDPC}^{IT}(\varepsilon) \cdot d\varepsilon = R \tag{2.42}$$

From equation (2.42), we can derive the upper bound of the ML decoding threshold, denoted by $\bar{\varepsilon}^{ML}$. This upper bound is the value of ε where the area under the IT EXIT curve (h_{LDPC}^{IT}), is equal to the LDPC code rate.

Remark 2.4.1. *[53] This upper bound is conjectured to be tight in a quite general settings, especially for binary codes defined by quasi regular graphs (e.g., LDPC-Staircase codes).*

In chapter 4, we extend method 2 to determine the EXIT functions, the (IT+RS) decoding threshold, and the upper bound of the ML decoding threshold for GLDPC-Staircase codes.

2.4.3.6 Performance affecting the structures of LDPC codes

The LDPC codes perform very well asymptotically with IT decoding. However, in finite length, most of them have a common severe weakness known as *error-floor* [97]. For the erasure channel, the error-floor is mostly caused by an undesirable structure, known as a *stopping set* [61] of the code's graphical representation, on which decoding is done. In other words, the effectiveness of IT decoding on graphs with sub-graphs depends primarily on the associated *cycles* and on how they are clustered to form *stopping sets*, which were introduced in [61].

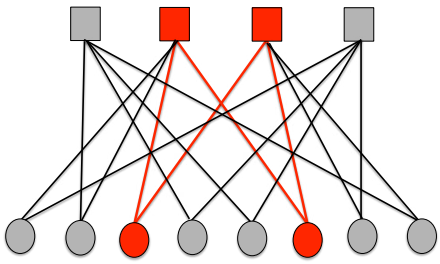


FIGURE 2.7: Example of a Cycle in the Tanner graph of an LDPC code.

Definition 2.5. *A cycle in a bipartite graph is a sequence of connected vertices which start and end at the same vertex in the graph, and which contains other vertices no more than once, as shown in Figure 2.7. The length of a cycle is the number of edges it contains, and the girth of a graph is the size of its smallest cycle.*

Girths in the tanner graphs of LDPC codes prevent the IT algorithm from converging to the true message information [19, 98–105]. Further, the cycles (especially short cycles) touch the independence of the extrinsic information exchanged between the variable and check nodes of the bipartite graph in the IT decoding [19, 98]. Several works proposed approaches to determine, reduce or/and remove the shortest cycles (of length 4, 6, 8, etc.) that degrade the performance of LDPC codes [106]. The degrading effect of short-length cycles decreases as the code length increases and is strongly reduced if the code length is large (i.e., the graph became very *sparse*).

Definition 2.6. *A stopping set S_a is a subset "a" of variable nodes, for which the induced sub-graph in the code graph contains no check nodes of degree one (i.e, every check node neighbor of this set is connected to this set at least twice). It contains one or several interconnected cycles. The stopping number of a stopping set is equal to the smallest number of uncorrected erasures (cannot be corrected under iterative decoding). Figure 2.8 shows a stopping set of size 4.*

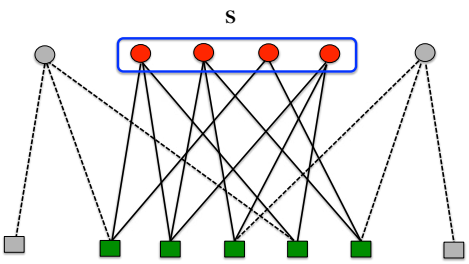


FIGURE 2.8: Example of a stopping set in the Tanner graph of an LDPC code.

We note that the symbols of stopping set can not be rebuilt because the adjacent check nodes can not determine their values. In fact, each check node has at least two erased symbols in their neighborhood, while a reconstruction is only possible if one symbol is erased.
In low erasure probability, the performance of LDPC codes are limited by the small

stopping sets, which affect on the error floor. A detailed description of the connection between stopping sets and both the bit and frame error-probability can be found in [61]. A great deal of research effort has been expended to design LDPC graphs free of the sub-graphs that induce the problem of failure decoding. Some of them try to identify all the possible stopping sets (sizes and configurations) to determine their influences on the point of onset of the error-floor of LDPC codes [33, 107–109], whereas others propose algorithms to avoid these phenomena [110, 111].

2.4.4 Reed Solomon codes

Irving Reed and Gus Solomon, in 1960, gave the foundation of the non-binary cyclic FEC codes that are called RS codes [112]. These codes are defined on Galois field of size q, denoted by $GF(q)$. In practice, this class are defined over $GF(q=2^b)$ where b is any positive integer having a value greater than 2. Each element of this finite field have a binary representation in the form of a vector with elements over $GF(2)$. The length of the RS codes is limited by the size of the used finite field. Therefore, RS (n, k, d_{min}) codes exist for all n and k for which

$$0 < k < n < 2^b + 2. \tag{2.43}$$

Theoretically, regardless of the desired length, we can always choose the finite field's size large enough to build such an RS code. However, in practice, it is preferable to keep the smallest finite field's size, due to the cost of operations which increases rapidly with size. Therefore, 2^4, 2^8 and sometimes 2^{16} are always considered. In summary, an RS (n, k, d_{min}) code over $GF(2^b)$ has the following parameters:

- Length: $n = 2^b - 1$,
- Number of redundancy symbols: $m = n - k = 2t$,
- Dimension: $k = 2^b - 2t - 1$,
- Error correcting capability: $t = \frac{n-k}{2}$,
- Minimum distance: $d_{min} = 2t + 1$,
- Erasure correcting capability: $\rho = d_{min} - 1 = n - k$.

An extended RS code can be made up with $n = 2^b$ or $n = 2^b + 1$, but not any further [113]. These codes achieve the largest possible code minimum distance between codewords. Said differently, they are MDS codes and meet the Singleton bound with equality as shown in equation (2.10). An RS (n, k, d_{min}) code is generated by the following polynomial,

$$\begin{aligned} g(x) &= (x-a^1)(X-a^2)\dots(X-a^{n-k}) \\ &= (x-a^1)(X-a^2)\dots(X-a^{2t}) \\ &= g_0 + g_1 x + g_2 x^2 + \dots + g_{2t} x^{2t}, \end{aligned} \tag{2.44}$$

where a is a primitive element of $GF(q)$ that induces $a^{q-1} = 1$ and the coefficients g_i belong to $GF(q)$.

The minimal polynomials of these codes have the form $\phi_i(x) = x - a^i$. Each element a^i is root of the minimal polynomial $x - a^i$ so that $x - a^i$ is a factor of $x^n - 1$. Therefore, $g(x)$ is also a factor of $x^n - 1$ and hence it is the generator polynomial of a cyclic code with elements taken from $GF(2^b)$ [114].
In addition, an RS code can be defined as the set of code polynomials $c(x)$ over $GF(q)$ of degree $deg(c(x)) \leq n-1$ that have $a, a^2, \ldots, a^{n-k}$ as their roots [115]. Therefore $c(x) \in$ RS code iff

$$c(a) = c(a^2) = \cdots = c(a^{2t}) = 0, \text{where deg(c(x))} \leq n-1. \tag{2.45}$$

This is because:

- $g(x)$ has $(a, a^2, \ldots, a^{(n-k)})$ as its roots,
- Any code polynomial is generated by multiplying a given message polynomial by g(x), and
- If $c(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1} \in$ RS code then, for $1 \leq i \leq n-k$,

$$c(a^i) = c_0 + c_1 a^i + \cdots + c_{n-1}(a^i)^{n-1} = c_0 + c_1 a^i + \cdots + c_{n-1}(a^{n-1})^i = 0. \tag{2.46}$$

This leads to determine the parity check matrix, H, of the RS codes as follows:

$$H = \begin{bmatrix} 1 & a^1 & a^2 & \ldots & a^{(n-1)} \\ 1 & a^2 & (a^2)^2 & \ldots & (a^2)^{n-1} \\ 1 & a^3 & (a^3)^2 & \ldots & (a^3)^{n-1} \\ 1 & \vdots & \vdots & \ldots & \vdots \\ 1 & a^{n-k} & (a^{n-k})^2 & \ldots & (a^{n-k})^{n-1} \end{bmatrix} \tag{2.47}$$

In this matrix any set of $n-k$ or fewer columns is linearly independent [114].

The encoding and the different decoding methods use the generator polynomial to generate the redundancy symbols and to detect/correct the erroneous symbols respectively. In addition, for these codes, errors up to half the minimum distance can be efficiently corrected which requires no more than $2t$ redundancy symbols, using algorithms such as the Berlekamp Massey (BM) algorithm [116] or the Euclid algorithm [116]. The error-correction and erasure-correction capabilities of these codes can be expressed as follows:

$$2\alpha + \beta < d_{min} < n-k, \tag{2.48}$$

where α is the number of the error patterns of the symbol that can be corrected and β is the number of the erasure patterns of the symbol that can be corrected.
As mentioned previously that ML decoding is the optimal one for the linear block FEC codes but for these codes, Guruswami and Vardy [117] have shown that it is *NP-hard*.

2.5 Rate-less erasure codes

2.5.1 Introduction

In the late 1990s, a class of erasure FEC codes so-called fountain codes was invented, for the packet-based-file multicast application, to generate a continuous flow of transmitted data packets [118, 119]. Thus, these codes has been added at the higher network layers. On the opposite of the previously codes, these codes produce from a finite set of data symbols an *infinite set of redundancy*. They are known as *rate-less erasure* codes and they have the property that a potentially unlimited sequence of encoding symbols can be generated from a given set of data symbols. The original data symbols can be completely recovered from any subset of the encoding symbols of size equal to or only slightly larger than the number of data symbols. Said differently, fountain codes are able to reliably recover the original k data symbols from (1 + ε)*k received code symbols, where ε is small [120].

2.5.2 LT codes

Michael Luby invented the first class of non-systematic rate-less erasure codes, called LT [121]. They are fountain codes and are characterized by a linear sparse parity check matrix similar to that of LDPC codes as shown in Figure 2.9. LT codes become very efficient as the code dimension grows. Each redundancy symbol is obtained by adding d

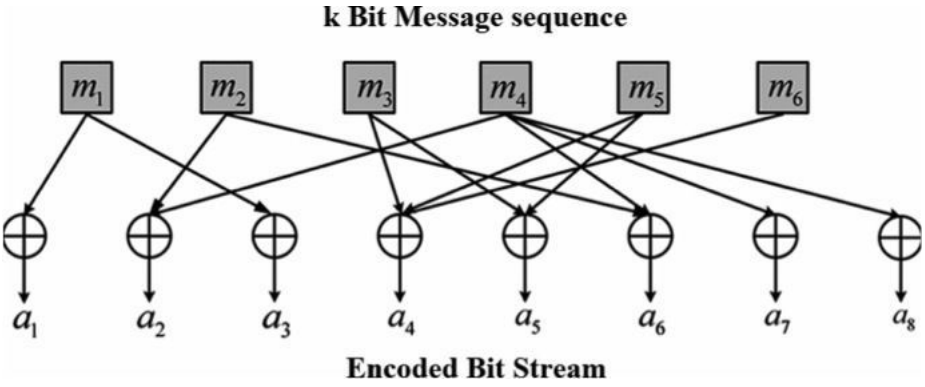


FIGURE 2.9: Bipartite graph of LT code with n = 8 and k = 6.

data symbols randomly selected where d, the redundancy symbol degree, depends on a particular distribution. In [121], Luby proposes the use of the Robust Soliton Distribution (RSD) as degree distribution. Then the average degree is $log(k)$, the encoding complexity is of $O(log(k))$ operations per symbol. Thanks to this distribution, LT codes can decode on average from $k + log^2(k)/\sqrt{(k)}$ symbols using an iterative decoder which then has a cost of $O(klog(k))$.
Furthermore, the redundancy symbols can be generated as needed and sent over the erasure channel until a sufficient number has arrived at the decoder in order to recover the data. The advantages of these codes are:

- Asymptotically optimal, regardless of the channel erasure model considered when they are decoded with an IT decoder,

- They are simultaneously near optimal for every erasure channel and they are very efficient as the data length grows [121].

Whereas the drawbacks are:

- Non-systematic codes,
- Did not have linear decoding properties. The amount of time needed to decode was more than linear than the size of the data being decoded [122],
- Poor performance for small k [51].

2.5.3 Raptor codes

Introduction: Shokrollahi proposed the Raptor codes which are build-on LT codes to overcome their disadvantages [122]. He solved the problems of LT codes by adding an outer erasure code to ease the recovery process and speed up decoding. Therefore, Raptor codes are composed of two codes in series,

- Pre-code: this is a linear block code with high rate to encode the k data symbols to l intermediate symbols. It performs the inverse LT encoding to make the Raptor code systematic.
  In [122], several types of pre-codes are considered: LDPC codes, Hamming codes. In the case of RFC5053 [123], the precode is composed on an LDPC code, a Hamming code and an inverse LT code.
- LT code: does not exactly follows the construction given by Luby [121]. Indeed, the degree distribution isn't the RSD. To reduce the encoding and decoding complexities, the average degree of the nodes of LT code is reduced to a low value and not dependent on k. This will obviously degrade the performance of the code, in particular the $k + log^2(k)/\sqrt{(k)}$ encoded symbols will not be sufficient on average to complete the decoding. But thanks to the pre-code, which is an erasures code, a partial decoding of LT code will be sufficient to complete the decoding of the Raptor code.

They have been the subject of several patents and standards within the Internet Engineering Task Force (IETF) RFC5053 [123], 3GPP MBMS standard for broadcast file delivery and streaming services [10], DVB-H IPDC standard for delivering IP services over DVB networks, and DVB-IPTV for delivering commercial TV services over an IP network [124].
Recently in 2011, Luby proposed a newer member in Raptor codes family kwon as RaptorQ codes in IETF RFC6330 [125]. These codes provide a superior flexibility, are most powerful, support for larger data block sizes, have small overhead and better coding efficiency than the older Raptor codes [125]. RaptorQ code introduces certain design selections that ensure superior performance compared with that of Raptor code [126]. The major difference between the two standardized codes is that Raptor code operates over Galois field GF(2), while the RaptorQ code uses symbol operations over GF(256) instead of over GF(2). Moreover, RaptorQ codes can encode up to 56,403 source symbols (up to 16, 777,216 encoding symbols) in contrast to 8,192 for the Raptor codes (up to 65,384 encoding symbols). This provides a higher flexibility to RaptorQ codes.

Encoding and decoding: The encoding of Raptor codes is performed in two steps. First the pre-code encodes the k source symbols to generate the l intermediate symbols. Then, the LT code produces the Raptor encoded symbols using the l intermediate symbols. Raptor codes can produce as many redundancy symbols as need, but unlike LT codes, Raptor codes can be made *systematic*. The encoding process of RaptorQ code is mostly identical with that of Raptor code [125].
To recover the k data symbols using the encoded symbols, the Raptor decoding performs in two steps; first the LT decoding, then the pre-code decoding. Shokrollahi proposed in [122] to decode the Raptor codes with an IT decoder as LT codes. However, the correction capabilities of these codes, when decoded with an IT algorithm, degrade rapidly as their dimensions decreases. The ML decoding is considered relevant for codes whose dimension is of the order of few thousand symbols, whereas for dimensions of the order of several tens of thousands of symbols, the IT decoding is preferred [122]. In RFC5053, Luby proposed an efficient ML decoding algorithm to reduce the number of XOR. This algorithm considers that the dimensions are under $k_{max} = 8192$. It is performed into two steps: first it decodes the LT code using a patented variant of GE [127], and then it deduces the source symbols from the intermediate symbols with a matrix multiplication. In RFC6330, RaptorQ code represents the latest and the best performing Raptor FEC. The decoding of these codes is based on ML decoding also but it is more better than of RFC5053.

Correction capabilities: According to a Raptor code presentation of Shokrollahi, the ML decoding of a code with dimension $k = 1024$ provides an overhead $= 1\%$ and frame error probability $P_{frame\_error} < 10^{-3}$. These codes provides, under ML decoding, the same correction capabilities of *Generalized Repeat accumulate* (GRA) for all parameters considered [84].
Tables 2.1 and 2.2 summarize the required decoding overhead and the decoding failure probability for the standardized Raptor codes and RaptorQ codes respectively, given from 3GPP [6]. In theses tables, P(O=i) denotes the probability that decoding is not successful with an overhead equals to “i” symbols.
Table 2.1 shows that RFC5053 Raptor code achieves the 0,8692167 decoding failure probability rate with zero reception overhead when applied to source blocks with 1,024 source symbols and rate 1/3. Additionally, these tables reveal that RaptorQ codes achieve overhead better than Raptor codes. Indeed, while RaptorQ code requires only two additional symbols to succeed a practically zero decoding failure probability, Raptor code requires to receive more than 9 additional symbols. Based on this, we can say that RaptorQ code almost perfectly emulates an ideal fountain FEC code.

Code (k,n) / Probability	(32, 128)	(256,1024)	(1024, 3072)
P(O=0)	0,7803401	0,8701107	0,8692167
P(O=1)	0,5052031	0,640379	0,640757
P(O=2)	0,2935873	0,4133323	0,4156782
P(O=3)	0,1609069	0,2443177	0,2471652
P(O=4)	0,0857378	0,136446	0,1391596
P(O=5)	0,0449816	0,0737557	0,0756274
P(O=6)	0,0234534	0,039154	0,0402792
P(O=7)	0,0122182	0,0206101	0,0211817
P(O=8)	0,0063783	0,0107332	0,0110751
P(O=9)	0,0033121	0,0055941	0,0057861

TABLE 2.1: Decoding failure probability distribution and the required overhead for Raptor codes [6]

Code (k,n) / Probability	(32, 128)	(256,1024)	(1024, 3072)
P(O=0)	0,0049447	0,0049238	0,0048933
P(O=1)	2,6E-05	2,34E-05	2,4E-05
P(O=2)	2E-07	1E-07	2E-07
P(O=3)	0	0	0
P(O=4)	0	0	0

TABLE 2.2: Decoding failure probability distribution and the required overhead for RaptorQ codes [6]

Part I

Design and optimization of GLDPC-Staircase codes

Chapter 3

LDPC-Staircase codes and systematic RS codes over erasure channels

Contents

3.1 Introduction 59

3.2 LDPC-Staircase codes 60

3.2.1 Parity check matrix construction 60

3.2.2 Encoding 61

3.2.3 Decoding 61

3.2.4 Performance: correction capabilities and complexity 62

3.3 Systematic RS codes 62

3.3.1 Construction methods 62

3.3.2 Code construction method complexity analysis 65

3.4 Conclusions 68

3.1 Introduction

This chapter presents an introduction of two erasure codes that are the basis of the work presented in this thesis, so-called LDPC-Staircase codes and systematic RS codes.

LDPC-Staircase codes whose the construction of their parity check matrices is very simple as described in RFC5170 [46], are a sub-class of RA codes class [75]. In this work, we follow the construction of RFC5170 where in addition to length, size, and seed (used to initialize the Pseudo Random Number Generator (PRNG)), LDPC-Staircase codes are determined by the parameter $N1$ also called left degree. This parameter defines the degree of source variable nodes. We describe the concepts of these codes in section 3.2.

For several use cases, systematic and MDS codes are rather attractive [113]. First of all, because they achieve the maximum possible minimum distance for given length and dimension (optimal correction capabilities); but also because they are systematic, i.e. data is a part of the encoded data. In this Chapter, we determine the best construction of

systematic RS codes with low complexity in terms of the generator matrix creation. In section 3.3, we give more details of this construction.

3.2 LDPC-Staircase codes

3.2.1 Parity check matrix construction

Let H_L be the binary parity-check matrix of the LDPC-Staircase code, of size $M_L = N_L - K$ rows and N_L columns. An example is given in matrix (3.1) for $K = 6$, $N_L = 10$ and $N1 = 2$.

$$H_L = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & \underline{\mathbf{1}} & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & \mathbf{1} & \underline{\mathbf{1}} & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & \mathbf{1} & \underline{\mathbf{1}} & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & \mathbf{1} & \underline{\mathbf{1}} \end{bmatrix} \tag{3.1}$$

This matrix is divided into two parts and has the form $(H_1|H_2)$. The $M_L \times K$ left-hand side part, H_1, defines the emplacements of source symbols in equations (rows). It is created in a fully regular way, in order to have constant column and row degrees. More precisely, each column of H_1 is of degree $N1$, which is an input parameter during the LDPC-Staircase code creation [46]. The $M_L \times M_L$ right-hand side part, H_2, defines in which equations the repair symbols are involved and features a staircase (i.e. double diagonal) structure.

For an LDPC-Staircase code with rate r_L, each row m of H_1 is of degree,

$$d_{r_{H_1}} = \frac{N_1}{\frac{1}{r_L} - 1}, \tag{3.2}$$

and due to the staircase structure of H_2, a row m of H_L is of degree:

$$d_r(m) = \begin{cases} d_{r_{H_1}} + 1 & \text{if} \quad m = 1 \\ d_{r_{H_1}} + 2 & \text{if} \quad m > 1. \end{cases} \tag{3.3}$$

Figure 3.1 provides the associated bipartite graph. Since H_2 has a uniform distribution

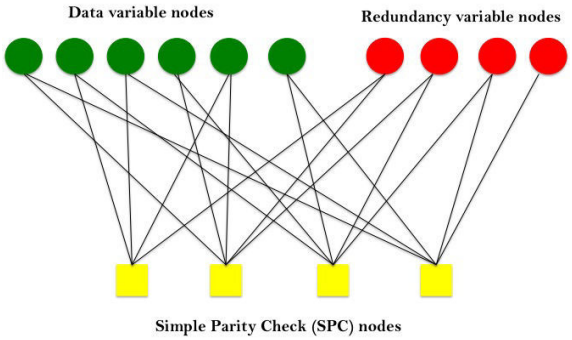


FIGURE 3.1: Bipartite graph of LDPC-Staircase code with $N_L = 10$, $K = 6$ and $N1 = 2$.

(with the exception of the first row and the last column) and H_1 is built on a regular distribution using the algorithm defined in [46], we consider LDPC-Staircase codes as quasi-regular structured LDPC codes.

3.2.2 Encoding

The encoding of LDPC-Staircase codes is based on the parity check matrix H_L. The repair symbols (i.e., redundancy symbols) are computed as usual, by "following the stairs" of the H_L matrix. Thus, each repair symbol is equal to the sum of all source symbols in the associated row, plus the previous repair symbol (with the exception of the first matrix row).

Let $S = (s_1, s_2, \cdots, s_K)$ be the source symbols and $P = (p_1, \cdots, p_m, \cdots, p_{M_L})$ be the repair symbols. For each row m in H_L, p_m is the XOR sum of LDPC source symbols $x = (x_1, \cdots, x_{K_m})$, where x is a subset of S that correspond to a "1" coefficients in row m of H_L, plus the LDPC repair symbol p_{m-1} if $m > 1$.

Example 3.1. *Let $S = (1,0,1,0,1,0)$ and the LDPC-Staircase code of Figure 3.1. Therefore, the repair symbols are generated as follows:*

- $P_1 = s_2 \oplus s_3 \oplus s_5 = 0$
- $P_2 = P_1 \oplus s_1 \oplus s_4 \oplus s_5 = 0$
- $P_3 = P_2 \oplus s_2 \oplus s_4 \oplus s_6 = 0$
- $P_4 = P_3 \oplus s_1 \oplus s_3 \oplus s_6 = 0$

With this encoding method, each generated repair symbol requires $N1.\frac{K}{N_L - K}$ XOR operations. The total number of operations required for producing $M_L = N_L - K$ repair symbols is therefore [51]:

$$nb\_tot\_op = N1.K \tag{3.4}$$

One can notice that, the complexity of the LDPC-Staircase encoder is linear in the dimension of the code and the speed encoding will be high (thanks to the structure inherited from RA codes).

3.2.3 Decoding

LDPC-Staircase codes can be decoded basically by an iterative method following algorithm 1 (see section 2.4.3.4). Since, H_L defines a set of linear equations whose variables are the source symbols and repair symbols, therefore the decoding method consists in recovering recursively the erased variables based on the n-k linear equations [46]. This means that, if one of these equations has only one remaining unknown variable, then its value is that of the constant term of the equation (i.e., sum of all known variable values in the equation). Then, we replace this variable by its value in all the remaining linear equations and reiterate.

As mentioned in section 2.4.3.4, with combining the advantages of IT and ML decoding to achieve the optimal correction capabilities with low decoding complexity compared to ML decoding, an hybrid decoding approach is investigated for LDPC codes [83–85]. This hybrid decoding starts by using the IT decoding above and finishes with ML decoding

(based on GE method). Therefore, the IT decoding can often decode, otherwise it reduces the size of the system that will be solved by the ML decoding. This approach was applied to the LDPC-Staircase codes [51].

3.2.4 Performance: correction capabilities and complexity

The effectiveness of the IT decoding is related to the density of the parity check matrix of LDPC codes. Therefore, to improve the performance of LDPC-Staircase codes under IT decoding and reduce the decoding complexity, it has been proposed in [128] to set the value of the parameter N1 = 3.
Thanks to the hybrid (IT/ML) decoding, LDPC-Staircase codes obtain ML correction capabilities with low decoding complexity. Applying the hybrid (IT/ML) decoding on these codes requires the tuning of the N1 parameter. Experimental results show that N1 = 5 is a good compromise between correction capabilities degradation of IT decoding, correction capabilities improvement of ML decoding, and the growth of the decoding complexity [83]. Recently, in the context of the 3GPP-eMBMS, N1=7 has been shown as a good compromise [129]. Hence, depending on the target use-case, the codec user will favor the adequate N1 value. This makes the LDPC-Staircase code more flexible.
Despite their simple construction, these codes achieve, under hybrid (IT/ML) decoding, correction capabilities close to ideal codes. But this is not always true for any code size; if the performance are great for large code size, they are of poor quality for small sizes. Moreover, the performance of these codes can approach those of Raptor codes whose their construction is far more complex. For large blocks, if Raptor codes are better, the difference between them is very low, since the LDPC-Staircase codes converge rapidly and extremely close to the optimal [51, 52].
Even though the complexity of hybrid (IT/ML) decoding increases with N1, LDPC-Staircase codes are on average at least on order of magnitude faster than the RS codec [52].

3.3 Systematic RS codes

3.3.1 Construction methods

RS codes are MDS erasure codes that can be put into a systematic form. Therefore their systematic generator matrix, G, can be written as:

$$G = [I_k | A_{k,n-k}], \tag{3.5}$$

where I_k is the identity matrix of order k, and $A_{k,n-k}$ represents the repair matrix of order $k \times (n-k)$ that does not contain any singular sub-matrix.
Said differently, any square sub-matrix (formed from any i rows and any i columns with $i \in \{1, \cdots, min\{k, n-k\}\}$) of A is non singular [113, 130]. Therefore the construction of a systematic RS generator matrix with the MDS property is equivalent to finding an appropriate A matrix.
In order to build this A matrix, it is of common practice to use Vandermonde matrices. Lacan et al. in [131] introduce a construction method for systematic MDS erasure codes, based on two Vandermonde matrices, which is the usual approach. However this approach

is relatively costly as we will demonstrate. This is not an issue if the code is fixed, i.e., if its code dimension (k) and length (n) parameters are fixed and known in advance, since A can be pre-calculated in that case. For instance, when dealing with codes for the physical layer, the code is fixed and the codec is implemented in hardware. But this assumption of a fixed code is no longer valid, in general, when dealing with the AL-FEC codes where the codec is usually a software component. In that case the $\{n,k\}$ parameters are dynamically determined, when there is a need to instantiate an AL-FEC encoder or decoder. For such applications that generate RS codes on demand, dynamically, there is a clear interest in reducing the generator matrix (G) creation complexity. This is also our case with GLDPC-Staircase codes construction [3].

In this section we first detail how Vandermonde matrices are used to that purpose, then we detail an alternative solution based on Hankel matrices [132]. To the best of our knowledge, this solution is the first reference to this alternative way of designing systematic RS codes since their introduction in 1985. We detail in this section how this is possible, we explain what complexity gains are expected during the systematic generator matrix creation stage, and finally we give an account of experiments carried out in order to assess the practical gains.

3.3.1.1 Construction based on Vandermonde matrix

Definition 3.1. *A Vandermonde matrix $V(q,q)$ is defined by one vector of q distinct elements $(a_1,\cdots,a_q)$ over $GF(q)$, i.e. such that each a_i, $1 \leq i \leq q$, is an element of $GF(q)$. A representation of the Vandermonde matrix, $V = (a_i^{j-1})_{i,j}$ is:*

$$V(q,q) = \begin{bmatrix} 1 & a_1 & \cdots & a_1^{q-1} \\ 1 & a_2 & \cdots & a_2^{q-1} \\ \vdots & \vdots & & \vdots \\ 1 & a_q & \cdots & a_q^{q-1} \end{bmatrix}. \tag{3.6}$$

Vandermonde matrices defined over $GF(q)$ can contain singular square sub-matrices [130, 131], and an upper bound of the number of singular sub-matrices is given in [133, 134]. Consequently these matrices cannot be directly used to design MDS systematic codes over $GF(q)$.
However, [130, 131] introduced a method to use these matrices in order to build a systematic MDS generator matrix $G(k,n)$ (i.e, systematic MDS RS generator matrix). This simplest solution consists in considering two matrices: the first one is the $V(k,k)$ matrix formed by the first k columns of the second matrix $V(k,n)$, defined as matrix (3.6). Then we invert the first matrix and multiply this inverse by $V(k,n)$. Clearly, the product $V(k,k)^{-1} * V(k,n)$ contains the identity matrix I_k on its first k columns, meaning that the first k encoding elements are equal to source elements. Besides, the resulting code features the MDS property and the resulting systematic RS generator matrix based on Vandermonde matrix is equal to:

$$\begin{aligned} G_V &= V(k,k)^{-1} * V(k,n) \\ &= [I_k | V(k,k)^{-1} * V(k,n-k)] \\ &= [I_k | A(k,n-k)] \end{aligned} \tag{3.7}$$

This construction can be optimized by taking the representative vector of Vandermonde matrix $(a_1, \cdots, a_k)$ equals to $(1, a, a^2, ..., a^{k-1})$ where a is an element of $GF(q)$ with order k [130]. Since the Vandermonde matrix is now defined by one element only (instead of q distinct elements), this matrix is denoted by $V(a)$ and has the following form:

$$V(a) = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & (a^1)^1 & (a^1)^2 & \cdots & (a^1)^{n-1} \\ 1 & (a^2)^1 & (a^2)^2 & \cdots & (a^2)^{n-1} \\ 1 & (a^3)^1 & (a^3)^2 & \cdots & (a^3)^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & (a^{k-1})^1 & (a^{k-1})^2 & \cdots & (a^{k-1})^{n-1} \end{bmatrix} \tag{3.8}$$

This choice has two objectives:

- The inversion of $V(a)$ is easier since $(V(a))^{-1} = \frac{1}{k} \times V(a^{-1})$ [130];
- The matrix-vector multiplication between $V(a)$ and $V(a^{-1})$ can be performed very efficiently [135].

3.3.1.2 Construction based on Hankel matrix

Let us now focus on an alternative way of building systematic MDS generator matrix using a Hankel matrix, a solution that has never been studied (as far as we can tell) since their introduction in [132].

Definition 3.2. *Hankel matrices are square matrices whose values are constant along the ascending diagonals.*

The construction method of systematic Hankel-RS codes is based on the creation of the maximal triangular array, B_q, defined over $GF(q)$. The coefficients of B_q are constant along diagonals in a Hankel matrix fashion:

$$B_q = \begin{matrix} b_1 & b_2 & b_3 & \cdots & b_{q-2} & b_{q-1} \\ b_2 & b_3 & . & \cdots & b_{q-1} & \\ b_3 & . & . & \cdots & & \\ . & . & . & & & \\ . & . & b_{q-1} & & & \\ b_{q-2} & b_{q-1} & & & & \\ b_{q-1} & & & & & \end{matrix} \tag{3.9}$$

This triangular array has a "pure" Hankel matrix. The coefficients of the triangular array are equal to: $b_i = \frac{1}{1-y^i}$, where $1 \leq i \leq q-1$, y is an arbitrary primitive element of $GF(q)$ and y^i is computed over $GF(q)$. By using these coefficients, B_q has the property that every square sub-matrix is non-singular [132].
Next step consists in extracting from the upper triangle of B_q a rectangular sub-matrix A of size $k \times (n-k)$ (this is always feasible since $k \leq n < q$). This matrix has of course the desired property that any square sub-matrix is non-singular. Therefore $A(k, n-k)$ can then

be used to construct a generator matrix of a systematic *RS* code.

The following triangular array, T_q, is a particular case:

$$T_q = \begin{array}{ccccccccc} 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 \\ 1 & b_1 & b_2 & b_3 & \cdots & b_{q-3} & b_{q-2} & \\ 1 & b_2 & b_3 & . & \cdots & b_{q-2} & & \\ 1 & . & . & \cdots & & & & \\ . & . & . & \cdots & & & & \\ . & . & b_{q-2} & & & & & \\ 1 & b_{q-2} & & & & & & \\ 1 & & & & & & & \end{array} \tag{3.10}$$

T_q has a "quasi" Hankel matrix form (instead of "pure" form as with B_q). It is built by removing b_{q-1} to B_q and adding an additional first row and column full of "1" entries. One can check that T_q does not contain any singular square sub-matrix as well [132]. Therefore one can extract an appropriate $A(k,n-k)$ matrix from the upper triangle of T_q as well in order to build the generator matrix of a systematic *RS* code.
This second version has the nice property that the first repair symbol is also equal to the direct XOR sum of the source symbols. This is a key point of our GLDPC-Staircase codes [3].

We therefore have two methods to build a generator matrix $G = [I_k|A_{k,n-k}]$ of a systematic MDS RS code: by extracting $A(k,n-k)$ from B_q ("pure" Hankel matrix form) or from T_q ("quasi" Hankel matrix form). With these techniques, the systematic generator matrix is directly obtained by a trivial concatenating operation, instead of having to invert a matrix and then perform matrix-vector multiplications as is the case with the Vandermonde solution. This is a major benefit when the code needs to be produced on-the-fly.

3.3.2 Code construction method complexity analysis

3.3.2.1 Performance evaluation environment

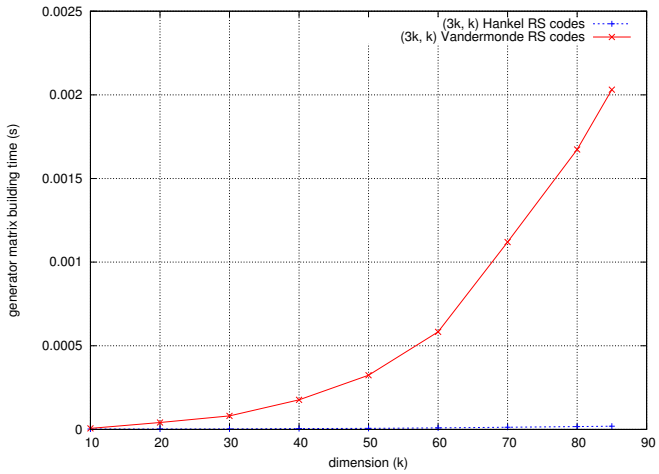
Let us now evaluate the generator matrix creation complexity, both in terms of the creation time and the number of elementary XOR operations.
To that purpose, we are using a C language RS software codec for Vandermonde matrices, derived from L. Rizzo's well known codec [136]. This codec has been optimized and is freely distributed at: `http://openfec.org/`. We also derived a codec for RS based on Hankel matrices that only differs during the generator matrix creation stage. The remaining of the two codecs, as well as the testing application, are strictly identical which warrants fair comparisons. In both cases we focus on $GF(2^8)$ codes since this is the most practical solution for software codecs: finite field elements are aligned on byte boundaries for easy transfers to/from memory buffers, and the finite field operations pre-computed tables have a small size that fits well in CPU caches and RAM. All the initialization, encoding and decoding speeds are evaluated on a MacBookPro laptop, featuring a 2.4 GHz Intel Core i5 CPU, and running MacOS 10.6.7.

Note that we only consider the "pure" Hankel matrix variant, B_q. Using the "quasi" Hankel matrix variant, T_q, would slightly simplify the encoding (and perhaps decoding)

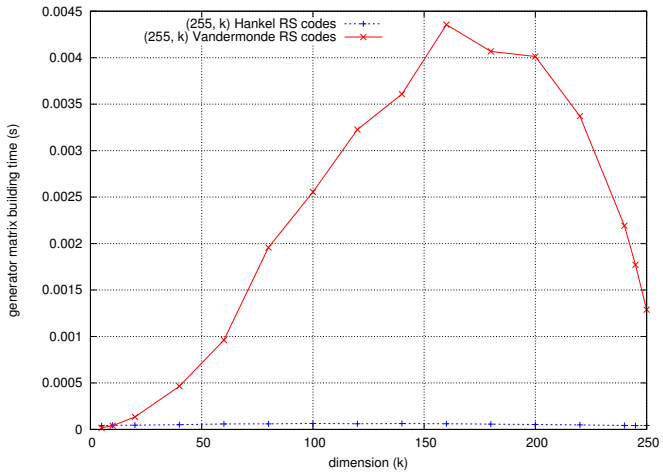
steps, the first repair symbol being equal to a simple XOR sum of the source symbols.

3.3.2.2 Generator matrix creation time

As explained in section 3.3.1.1, with the Vandermonde approach, the complexity of building G corresponds to the complexity of inverting $V_{k,k}$ and multiplying $V_{k,k}^{-1}$ by $V_{k,n-k}$. On the opposite, with the Hankel approach, once the B_q (or T_q with "quasi" Hankel variant) coefficients are computed, a simple concatenation is sufficient.
In order to appreciate the practical consequences, we have measured these times.



(a) RS(3k,k) codes.



(b) RS(255,k) codes.

FIGURE 3.2: Systematic generator matrix creation times.

Figure 3.2 illustrates the major gains permitted by the use of Hankel matrices, in terms of creation times. If the systematic generator matrix creation times depend on $n-k$ and k with RS codes based on Vandermonde matrices as k increases, the progression is linear with their Hankel equivalent.

This is confirmed in Table 3.1 that evaluates the processing time gains made possible

TABLE 3.1: Generator matrix creation times and performance gains made possible by the Hankel approach.

	(30,10)	(250,50)	(250,100)	(250,125)
Vandermonde	0.007 ms	0.620 ms	2.577 ms	3.143 ms
Hankel	0.002 ms	0.011 ms	0.020 ms	0.020 ms
Ratio	3.5	56.36	128.85	157.15

by the use of the Hankel approach: the gains vary between 3.5 and 157.15.

3.3.2.3 Generator matrix creation complexity

Let us now consider the number of operations required to build a systematic generator matrix of RS codes.
With Vandermonde matrices, the creation of the systematic generator matrix requires finite field additions and multiplications over $GF(q)$ in order to perform matrix inversion and multiplication. Finite field additions consist in XORing the two values. Finite field multiplications are more complex, requiring in general a log table lookup, an addition operation and an exponentiation table lookup to determine the result. However, with $GF(2^8)$, multiplications can be pre-calculated and the result stored in a table of size 255×255. This is a common optimization with software codecs and this is how the initial RS codec was implemented. With this optimization, multiplying two elements of $GF(2^8)$ consists in accessing the right element of this pre-calculated table. Since this is a simple operation, the complexity evaluation only considers the number of XOR operations and ignores the number of multiplications. As a result, for an $RS(n,k)$ code with a Vandermonde base matrix, the systematic generator matrix creation consists of:

- The initialization of $V(a)$ which requires $k(n-1)$ read accesses to pre-calculated tables;
- The $(k \times k)$ matrix inversion, which requires $2k(k-1)+(k-1)+\frac{(k-1)(k-2)}{2}$ XOR operations, and $0.5(k-1)(k-2)+2k(k-1)+2k^2$ read accesses to pre-calculated tables;
- The $(k \times k)$ - $(k \times n-k)$ matrix multiplication, which requires $k(n-k)(k-1)$ XOR operations, and $k^2(n-k)$ read accesses to pre-calculated tables.

With Hankel matrices, the creation of the systematic generator matrix essentially consists in calculating the $b_i = \frac{1}{1-y^i}$ coefficients, with $1 \leq i \leq n$. Since the same values are used along the diagonals, we only calculate the n coefficients of the first line, which requires n XOR operations. In addition, the processing requires $2n+k(n-k)$ read accesses to pre-calculated tables.
We see that the Hankel approach outperforms the Vandermonde approach both in terms of the number of XOR operations and read accesses to tables.

Table 3.2 shows the statistics obtained, counting the actual number of operations in the two codecs. The results confirm the benefit of the Hankel matrix approach, with speedups similar to that achieved in Table 3.1 when measuring time. The small difference is due to

TABLE 3.2: Generator matrix creation complexity (number of XOR and table access (TA) operations) and performance gains made possible by the Hankel approach).

	(30,10)	(250,50)	(250,100)	(250,125)
Vandermonde	2,225 XOR	506,125 XOR	1,524,750 XOR	1,991,875 XOR
	2,706 TA	523,526 TA	1,569,551 TA	2,054,126 TA
Total # oper.	4,931 op.	1,029,651 op.	3,094,301 op.	4,046,001 op.
Hankel	30 XOR	250 XOR	250 XOR	250 XOR
	260 TA	10,500 TA	15,500 TA	16,125 TA
Total # oper.	290 op.	10,750 op.	15,750 op.	16,375 op.
Ratio	17.0	95.8	196.5	247.1

the simplification performed when counting operations: we do not include write operations, loop control operations, function call overheads, modulo calculations, and we assign the same cost factor to XOR and table access operations.
A more detailed complexity analysis is feasible, but we consider that the accuracy achieved is sufficient to give an account of the observed behavior.

3.3.2.4 Impacts on the global encoding and decoding times

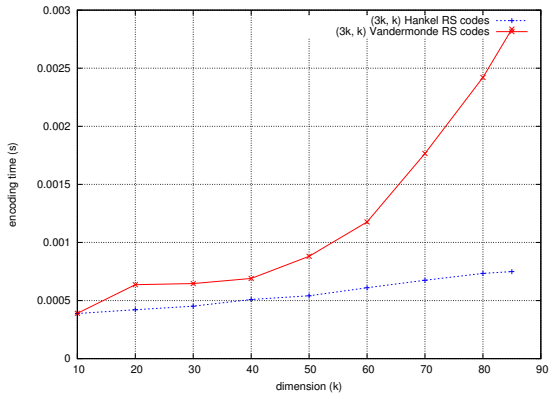
Let us now try to answer another question: what are the impacts of G creation on the total encoding or decoding times ?

Due to our assumptions, the systematic generator matrix creation time is included in the encoding (resp. decoding) times. Figures 3.3 and 3.4 show the relative gains possible made by the use of Hankel matrices on the encoding and decoding times, for several RS codes. These tests have been carried out with symbols of size 4 bytes each, i.e. the same operations are performed on each byte of the symbol (since erasures take place at the symbol level in case of RS codes for the erasure channel, see [47]).
We see a clear gain in using Hankel matrices, even if this gain depends on the actual code dimension and length being used. However it should be noted that a symbol size of 4 bytes is rather small in case of AL-FEC codes. Symbols, that form the payload of UDP/IP datagrams, are more often on the order of a few hundreds of bytes. In that case, even that the encoding/decoding times increase since the manipulations on symbols increase, Hankel approach still also outperforms Vandermonde approach.

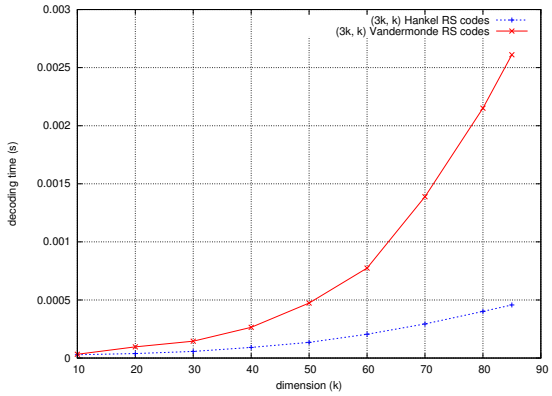
Both theoretical and experimental results show that systematic RS codes construction based on Hankel matrix provides an order of magnitude simpler than the Vandermonde approach. Additionally, the results of Figures 3.3 and 3.4 should therefore be regarded as upper bounds of the gains made possible by the use of Hankel matrices during encoding and decoding with many AL-FEC codes. Therefore, we select this construction method to design the GLDPC-Staircase codes.

3.4 Conclusions

In this chapter, first we recalled the characteristics of LDPC-Staircase codes as specified in RFC5170 [46] and their obtained performance. Despite their simplicity of construction



(a) Encoding time

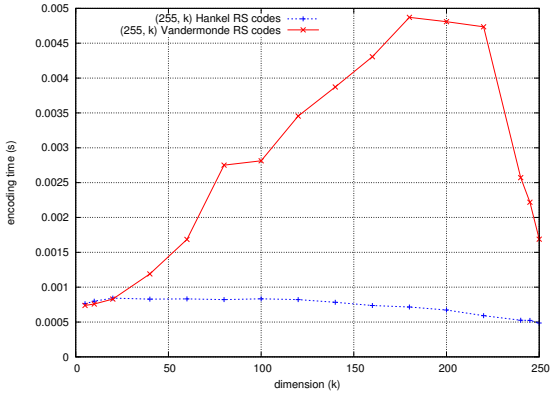


(b) Decoding time

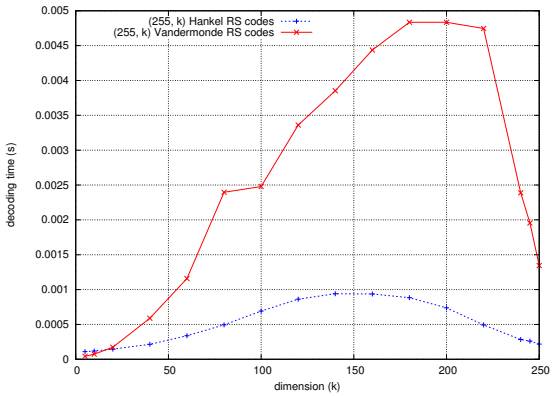
FIGURE 3.3: Comparison between (3k,k) Vandermonde and Hankel Reed-Solomon codes during encoding and decoding with symbols of size 4 bytes.

inherited from RA codes, LDPC-Staircase codes achieve good correction capabilities when they are decoded with an hybrid (IT/ML) decoder. This is made possible by the control of the N1 parameter, which determines the degree of source symbols nodes. This parameter is chosen as a compromise between the correction capabilities of ML decoding, the correction capabilities of IT decoding and the complexity of decoding. Additionally, LDPC-Staircase codes provide high encoding/decoding speeds that are an order of magnitude higher than those of RS codes.

Then, in addition to the traditional method for building systematic RS codes, based on Vandermonde matrices, we have introduced another approach, based on Hankel matrices. We proved, both theoretically and experimentally, that the code construction time and the number of operations performed to build the target RS code are largely in favor of the Hankel approach. The systematic generator matrix is produced immediately, instead of having to invert a matrix and multiplying this inverted matrix with another one as the case of Vandermonde approach. This result is of high importance for all situations where a software RS(n, k) codec needs to generate on the fly an RS code with appropriate dimension and length values.



(a) Encoding time



(b) Decoding time

FIGURE 3.4: Comparison between (255,k) Vandermonde and Hankel RS codes during encoding and decoding with symbols of size 4 bytes.

Chapter 4

Generalized LDPC (GLDPC)-Staircase codes

Contents

4.1 Introduction 71

4.2 Proposed GLDPC-Staircase coding schemes 72

4.2.1 Design of GLDPC-Staircase codes 72

4.2.2 Encoding of GLDPC-Staircase codes 76

4.2.3 Decoding of GLDPC-Staircase codes 77

4.3 Asymptotic analysis of GLDPC-Staircase codes under (IT+RS) and ML decoding 80

4.3.1 Preliminaries 80

4.3.2 Density Evolution 81

4.3.3 EXIT functions of GLDPC-Staircase codes 85

4.4 Conclusions 89

4.1 Introduction

A GLDPC is a low density parity check code in which the constraint nodes of the code graph are block codes rather than SPCs, in order to have more powerful decoders in these nodes [36].

The notion of the structure of GLDPC-Staircase codes is presented in [52] where an LDPC-Staircase code and RS MDS codes are chosen as base code and outer codes respectively. We note that the construction of these codes differs from the construction of GLDPC codes proposed by Tanner [36] and their successive varieties [37–39, 39–44]. In addition, GLDPC-Staircase scheme has the property that *on each check node of LDPC-Staircase code, the generated repair symbol is one of the repair symbols of the associated RS MDS code*.

To obtain these codes in practice the authors said: "*this can be obtained by multiplying the RS generator matrix on the left by* ***an appropriate invertible matrix****, which will change the*

way in which the encoding is done, but not the properties of the MDS code". However, the exact structure of this matrix remains unknown as no construction method has been provided. Trying to design such matrices is challenging as we can not to keep the MDS properties of the resulted RS codes. Therefore, design of these codes remains an open question.
In this Chapter, we explain the design of GLDPC-Staircase codes in section 4.2. Firstly, a solution to design these codes in practice is given. The basic idea of our solution is to use the ***systematic "quasi" Hankel-RS codes*** presented in section 3.3.1.2. Then, in order to investigate the impact of the structure of these GLDPC-Staircase codes on the decoding performance, we propose another scheme where the generated LDPC repair symbol is a source symbol viewpoint RS code. Afterwards, to recover the erased symbols, in addition to the (IT+RS) decoding method, we propose a new decoding approach called hybrid (IT/RS/ML) decoding.
Then in the second part of this Chapter, we present an asymptotic analysis of GLDPC-Staircase codes in terms of DE and EXIT functions in section 4.3 to analyze the decoding convergence and the gap to the Shannon limit under (IT+RS) and ML[1] decoding. More precisely, we derive the DE equations following the method presented in [52] to evaluate the evolution of the erasure probability during (IT+RS) decoding. We adapt the approach of Méasson [96] to derive an upper bound of the ML decoding threshold of GLDPC-Staircase codes.

4.2 Proposed GLDPC-Staircase coding schemes

In this section, we explain our solution to design GLDPC-Staircase codes and we present another GLDPC-Staircase construction structure. These constructions differ on the nature of the generated LDPC repair symbols. Henceforth, GLDPC-Staircase codes are denoted by scheme A when the generated LDPC repair symbols are RS repair symbols, whereas are denoted by scheme B when the generated LDPC repair symbols belong the set of RS source symbols.

4.2.1 Design of GLDPC-Staircase codes

GLDPC-Staircase codes, studied in this work, are constructed from:

- **LDPC-Staircase code** : It is defined as a *base code* with length N_L and dimension K ($M_L = N_L - K$). Based on its parity check matrix H_L, each row of this matrix defines the connection between the source symbols and the produced LDPC repair symbols viewpoint SPC. In other words, in our case, this row defines the connection between the RS repair symbols and the LDPC symbols (source, repair) viewpoint RS code. Consequently, each check node of this code is represented as a powerful check node, which is called *generalized check node* in GLDPC-Staircase code.

- **RS codes**: They are defined as *outer codes* (components codes). Each RS code is associated to each check node m (i.e, row in H_L) of the base code to generate RS

[1]Hybrid decoding give the same performance as ML decoding but with low complexity. Therefore in the second part of this Chapter we only refer the hybrid decoding performance by ML decoding performance

repair symbols; $e(m)$ extra-repair symbols (and one LDPC repair symbol if we use scheme A). These RS repair symbols are generated by RS(n_m, k_m) encoding over $GF(2^b)$ with $0 \leq e(m) \leq E$ and $m = 1, ..., M_L$. Here E, k_m, and n_m are respectively the maximum number of extra-repair symbols per generalized check node, the RS code dimension and length for the generalized check node m.

These GLDPC-Staircase (N_G, K) codes can be represented by a bipartite graph as shown in Figure 4.1, where N_G is the number of variable nodes (i.e., code length), K is the

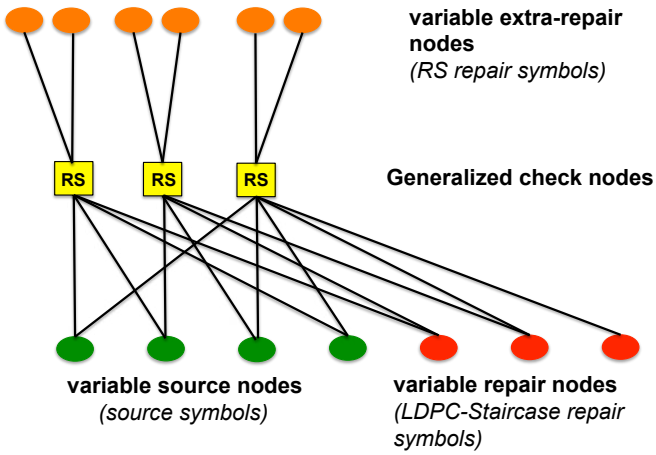


FIGURE 4.1: GLDPC-Staircase(13,4) code, $e(m) = 2$ extra-repair symbols per generalized check node (i.e, regular distribution).

number of source variable nodes (i.e, code dimension). This graph is composed of two sets of nodes with the following meaning:

- **generalized check node**: it corresponds to an RS code (powerful check node).
- **variable nodes** are broken into three categories:
  1. the source symbols,
  2. the LDPC repair symbols generated by the LDPC-Staircase code (and RS codes in scheme A) which are reliant symbols,
  3. the extra-repair symbols: RS repair symbols generated by RS codes which are independent symbols. Their associated variable nodes are all of degree 1.

Let us define the two GLDPC-Staircase codes variants. The difference between this coding schemes is in the definition of n_m and k_m:

- Scheme A
  For row $m > 1$, the various source symbols (i.e., from the user point of view) that are involved in this row plus the *previous repair symbol* are considered as source symbols from the RS point of view. The generated LDPC repair symbol on this row plus the $e(m)$ extra-repair symbols are considered as repair symbols from the RS point of view *(i.e., the new LDPC repair symbol is an RS repair symbol)*. For $m = 1$

the only difference is the fact there is no previous repair symbol (it's the beginning of the staircase). So:

$$n_m = k_m + 1 + e(m) \quad \text{and} \quad k_m = d_r(m) - 1 \quad \text{(no matter the row)}, \tag{4.1}$$

where $d_r(m)$ is given in equation (3.3).
To obtain the property of this scheme, we propose a construction of RS codes based on "quasi" Hankel matrix. As we showed in section 3.3, the resulted RS generator matrix (G) of these codes has the following form:

$$G = \begin{bmatrix} 1 & 0 & . & 0 & \mathbf{1} & 1 & 1 & . & 1 & 1 \\ 0 & 1 & . & . & \mathbf{1} & b_1 & b_2 & . & . & b_{n_m-k_m} \\ 0 & . & . & . & \mathbf{1} & b_2 & . & . & . & b_{n_m-k_m+1} \\ . & . & . & . & \bullet & . & . & . & . & . \\ . & . & . & . & \bullet & . & . & . & . & . \\ . & . & . & 0 & \bullet & . & . & . & . & b_{n_m-2} \\ 0 & . & 0 & 1 & \mathbf{1} & . & . & . & b_{n_m-2} & b_{n_m-1} \end{bmatrix} \tag{4.2}$$

For GLDPC-Staircase codes, this choice has the advantage that the repair symbol generated by the row m, p_m, can be considered indifferently as an LDPC-Staircase symbol or RS repair symbol thanks to the column of "1", in the matrix G, associated to the first generated repair symbol.

- Scheme B
  For each row m, the various source symbols (i.e., from the user point of view) that are involved in this row plus the *LDPC repair symbols* are considered as source symbols from the RS point of view. The $e(m)$ extra-repair symbols are considered as repair symbols from the RS point of view. So:

$$n_m = k_m + e(m) \quad \text{and} \quad k_m = d_r(m) \quad \text{(no matter the row)}. \tag{4.3}$$

  For this scheme, any MDS RS code can be used (e.g., RS based on Hankel matrix or Vandermonde matrix).

In other words, scheme *A* has the property that on each check node of the base code the repair symbol generated by the LDPC code is also an RS repair symbol. On the opposite, with scheme *B* for each check node of the base code the two repair symbols are RS source symbols.

For a fixed code rate r_L of LDPC Staircase code (N_L, K), the code rate of the GLDPC-Staircase code is given by:

$$\begin{aligned} r_G &= \frac{K}{N_L + M_L \bar{f}} \\ &= \frac{r_L}{1 + (1 - r_L)\bar{f}}. \end{aligned} \tag{4.4}$$

Where $\bar{f}$ is the average number of extra-repair symbols per generalized check node:

$$\bar{f} = \sum_{e=0}^{E} (f_e . e). \tag{4.5}$$

and f_e denotes the fraction of generalized check nodes with e extra-repair symbols:

$$f_e = \frac{card\{m = 1 \ldots M_L \mid e(m) = e\}}{M_L}, \tag{4.6}$$

We can consider the following two distributions of extra-repair symbols on the various generalized check nodes:

- Regular distribution: $f_e = 0$ for $e \in \{0, 1, \ldots, E-1\}$ and $f_E = 1$. Thus each generalized check node, m, has the same number $e(m) = E$ of extra-repair symbols and the rate of the extended code (GLDPC-Staircase code) is equal to:

$$r_G = \frac{r_L}{1 + (1 - r_L) * E} \tag{4.7}$$

  Figure 4.1 represents the GLDPC-Staircase code with a regular distribution of extra-repair symbols per generalized check nodes.

- Irregular distribution: All the generalized check nodes have a different number of extra-repair symbols. We denote the number of extra-repair symbols per generalized check node m, by $e(m)$. In Figure 4.2, we show the irregular distribution of extra-repair symbols per generalized check nodes of GLDPC-Staircase code.

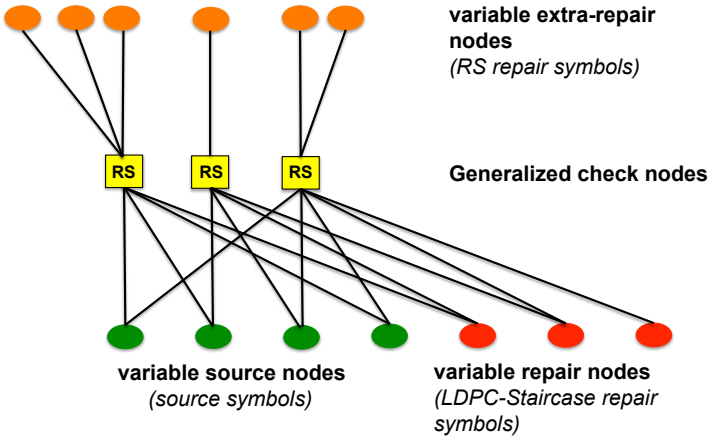


FIGURE 4.2: GLDPC-Staircase$(13,4)$ code with irregular distribution, $e(m) = \{3,1,2\}$

  There is an optimal irregular distribution which provides the best correction capabilities among several other irregular distributions. In [52], they show that it exists an irregular uniform distribution of extra-repair symbols which achieves performance close to the optimal irregular distribution. This irregular uniform distribution allows to allocate the extra-repair symbols with $\bar{f} = \frac{E}{2}$ and $f_e = \frac{1}{E+1}$ for $e \in \{0, 1, \ldots, E\}$.

Therefore throughout this thesis we consider only the regular distribution and the irregular uniform distribution. We will discuss, in section 5.4.1, the impact of these two distribution types on the decoding correction capabilities of GLDPC-Staircase codes.

4.2. PROPOSED GLDPC-STAIRCASE CODING SCHEMES

4.2.2 Encoding of GLDPC-Staircase codes

Two types of repair symbols are produced during encoding:

- M_L LDPC-Staircase repair symbols, $(p_1, \cdots, p_{M_L})$, and
- $M_L\bar{f}$ extra-repair symbols, $\left(\left(e_{1,1}, \cdots, e_{1,e(1)}\right), \ldots, \left(e_{M_L,1}, \ldots, e_{M_L,e(M_L)}\right)\right)$.

Let $S = (s_1, s_2, \cdots, s_K)$ be the source symbols. The $(p_1, \cdots, p_{M_L})$ repair symbols are computed following the "stairs" of the H_L matrix. Moreover, for each row m in H_L, p_m is the XOR sum of LDPC source symbols $x = (x_1, \cdots, x_{K_m})$, where x is a subset of S that corresponds to a "1" coefficients in row m of H_L, (and the LDPC repair symbol p_{m-1} if $m \neq 1$).
For each row m, the $e(m)$ extra-repair symbols are computed by multiplying the k_m LDPC symbols by the systematic generator matrix G_m of RS (n_m, k_m) associated to this row. As mentioned in section 4.2.1, for scheme A the k_m symbols are defined by x plus p_{m-1} (if $m \neq 1$). For scheme B, they are defined by x plus all LDPC repair symbols.
We note that the distribution types of extra-repair symbols have no impact on the encoding method.

Example 4.1. *Consider the GLDPC-Staircase code (with scheme A) defined by the bipartite graph, which is presented in Figure 4.1. This code has:*

- $N_G = 13$, $K = 4$ *and* $e(m) = 2$ *extra-repair symbols per generalized check node (i.e, regular distribution),*
- H_L*: the parity check matrix, shown in (4.8), of its base code.*

We also provide in (4.8) the codes dimensions and codes lengths of RS codes that are associated to generalized check nodes (i.e., rows of H_L*).*

$$H_L = \begin{array}{c} \begin{matrix} S_1 & S_2 & S_3 & S_4 & P_1 & P_2 & P_3 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \end{matrix} \\ \left[\begin{array}{cccc|ccc} 1 & 1 & 0 & 1 & \boldsymbol{1} & 0 & 0 \\ 0 & 1 & 1 & 0 & \boldsymbol{1} & \boldsymbol{1} & 0 \\ 1 & 0 & 0 & 1 & 0 & \boldsymbol{1} & \boldsymbol{1} \end{array}\right] \end{array} \begin{array}{l} RS_1 = RS(6,3) \\ RS_2 = RS(6,3) \\ RS_3 = RS(6,3) \end{array} \tag{4.8}$$

In this case (regular distribution), all the symbols are generated by the same RS code where its generator matrix G_{rs} *has the following form:*

$$G_{rs} = \begin{array}{c} \begin{matrix} & & & G_{rs}^4 & G_{rs}^5 & G_{rs}^6 \\ & & & \downarrow & \downarrow & \downarrow \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & b_1 & b_2 \\ 0 & 0 & 1 & 1 & b_2 & b_2 \end{bmatrix} \end{array} \tag{4.9}$$

Let us consider $S = (S_1, S_2, S_3, S_4)$*, therefore we can summarize the encoding steps as follow:*

- *First row (generalized check node) produces (P_1, $e_{1,1}$ and $e_{1,2}$) using $x = (S_1, S_2, S_4)$*
  $P_1 = G_{rs}^4 \times (S_1, S_2, S_4)$
  $e_{1,1} = G_{rs}^5 \times (S_1, S_2, S_4)$
  $e_{1,2} = G_{rs}^6 \times (S_1, S_2, S_4)$

- *Second row (generalized check node) produces (P_2, $e_{2,1}$ and $e_{2,2}$) using $x = (S_2, S_3)$*
  $P_2 = G_{rs}^4 \times (S_2, S_3, P_1)$
  $e_{2,1} = G_{rs}^5 \times (S_2, S_3, P_1)$
  $e_{2,2} = G_{rs}^6 \times (S_2, S_3, S_1)$

- *Third row (generalized check node) produces (P_3, $e_{3,1}$ and $e_{3,2}$) using $x = (S_1, S_4)$*
  $P_3 = G_{rs}^4 \times (S_1, S_4, P_2)$
  $e_{3,1} = G_{rs}^5 \times (S_1, S_4, P_2)$
  $e_{3,2} = G_{rs}^6 \times (S_1, S_4, P_2)$

An advantage of these two schemes is the fact that extra-repair symbols can be produced incrementally, on demand, rather than all at once (unlike LDPC-Staircase repair symbols for example). Their number can also be rather high since it is only limited by the finite field size, usually GF(2^8). Said differently, GLDPC-Staircase codes can easily and dynamically be turned into small rate codes.

4.2.3 Decoding of GLDPC-Staircase codes

To recover the erased source symbols, we can apply to GLDPC-Staircase codes (scheme *A* or *B*), two types of decoders: (IT+RS) decoding or Hybrid (IT/RS/ML) decoding. Let us consider GLDPC-Staircase codes with length N_G, dimension K and the base codes with length N_L and dimension K.

4.2.3.1 (IT+RS) decoding

The (IT+RS) decoding consists of a joint use of IT and RS decoders.

- IT decoder over the LDPC-Staircase graph: extra-repair symbols are ignored at this step. This solution features a linear complexity with sub-optimal erasure recovery capabilities;

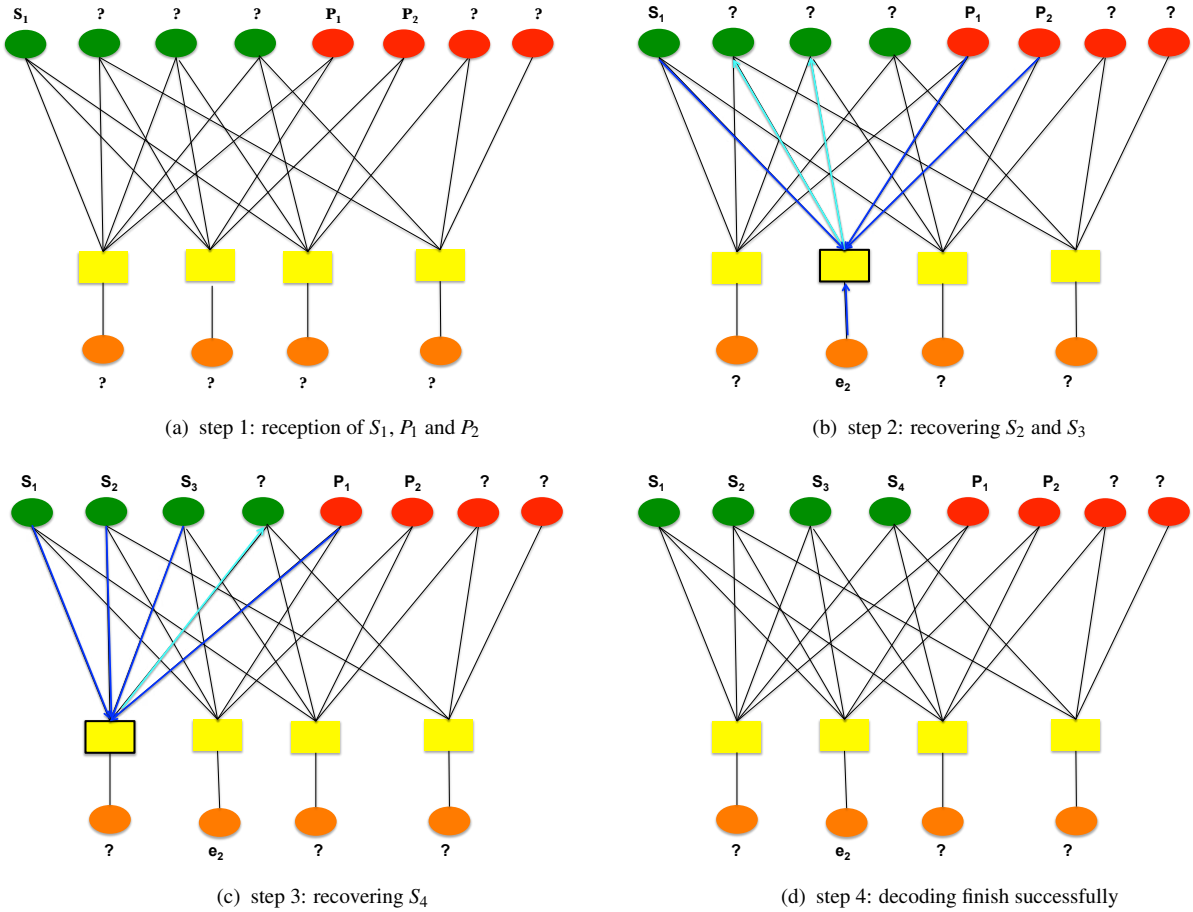
- RS decoder for a given generalized check node: this is a classic RS decoding that takes into account the three types of symbols. It has a higher complexity but is *MDS*;

These two decoders work together to recover the erased symbols.
The (IT+RS) decoding tries to determine the erased source symbols from the first received symbol until it reaches the maximum number of received symbols (denoted by N_b).
In Figure 4.3, we show a simple example which gives the progress of various steps of this decoding algorithm on GLDPC-Staircase code (scheme *A*) with N_G=12 , K= 4 , N1=3, and $r_L = \frac{1}{2}$. We assume that the received symbols are $\{S_1, P_1, P_2, P_3, e_1, e_2\}$ (loss percentage=50%) and the order of receiving these symbols is $\{S_1, P_1, P_2, e_2, e_1, P_3\}$ (chosen randomly). After receiving the first four symbols (S_1, P_1,P_2 and e_2) the decoder triggers with RS decoding on the second generalized check node. This node is associated with RS (6, 4) code which produces the erased symbols (S_2, S_3) in step 2. After that, these

Algorithm 2 (IT+RS) decoder: decoding with new received symbol

```
for symb = 1 to N_b do
    Function: Decode with new symbol "symb"
    Check the identity of "symb" and decode
    if 1 ≤ symb ≤ N_L then                                  ▷ symb is an LDPC symbol
        Select IT decoding with received symbol
        if IT decoding possible then
            Recover symbols
            New_symb ← each recovered symbol
            Decode with new symbol "symb=New_symb"
        else                    ▷ Check if with symbol symb we can do RS decoding
            Select RS decoding
            if RS decoding possible then
                Recover symbols                                  ▷ LDPC symbols
                New_symb ← each recovered symbol
                Decode with new symbol "symb=New_symb"
            end if
        end if
    else                                        ▷ symb is an extra-repair symbol
        Select RS decoding
        if RS decoding possible then
            Recover symbols                                      ▷ LDPC symbols
            New_symb ← each recovered symbol
            Decode with new symbol "symb=New_symb"
        end if
    end if
    if all erased source symbols are recovered then        ▷ Check the end of decoding
        Status OK
    else
        Continue
    end if
    End function
end for
if all erased source symbols are recovered then            ▷ Check the end of decoding
    Status OK
else
    Status Failure
end if
```

recovered symbols trigger the SPC decoding on the first generalized check node, which allows to decode S_4 in step 3. Then, 2 steps achieve the decoding with success because all the erased source symbols are recovered.
We give in algorithm 2, the steps of (IT+RS) decoding algorithm processing symbol per symbol where *"symb"* means the new received or recovered symbol. Therefore with this



(a) step 1: reception of S_1, P_1 and P_2

(b) step 2: recovering S_2 and S_3

(c) step 3: recovering S_4

(d) step 4: decoding finish successfully

FIGURE 4.3: Example of (IT+RS) decoding on the graph of GLDPC-Staircase code (scheme *A*).

algorithm, it is not necessary to use all the received symbols to recover the erased source symbols.
We note that if the reception of a symbol can trigger two decoding types (RS decoding or SPC decoding) on a generalized check node to recover one erased symbol, our algorithm chooses the SPC decoding to reduce the decoding complexity.
In addition, the two coding schemes A and B use the same (IT+RS) decoding algorithm.

4.2.3.2 Hybrid (IT/RS/ML) decoding

We propose a new type of decoding to determine the erased symbols called hybrid (IT/RS/ML) decoding. This decoding is inspired by the hybrid decoding approach proposed for LDPC codes [83, 84].
Hybrid (IT/RS/ML) decoding consists of a joint use of IT, RS and (binary/Non binary) ML decoding to achieve the performance of ML decoding but with a lower complexity.
The process of the hybrid (IT/RS/ML) decoding is as follows. It starts with (IT+RS) decoding as shown in algorithm 2 to try decoding all the erased source symbols. If the decoder can succeed with this decoding type, then the codeword is marked as decoded

and the hybrid decoding succeeds. If the (IT+RS) decoding fails to recover the codeword (with no additional received symbols), then the receiver switches to the ML decoding. ML decoding will attempt to solve the simplified linear system, which is produced by the (IT+RS) decoding.
In the ML decoding step, we use the following decoder types:

- Binary ML decoder: extra-repair symbols are ignored at this step and we only consider the binary ML decoding. Yet this solution features a quadratic complexity in terms of the number of `XOR` operations between symbols, it allows to reach the maximum correction capabilities even when ignoring extra-repair symbols;
- Non binary ML decoder: this solution also features a quadratic complexity but operations are now significantly more complex (performed on $GF(2^b)$) than simply `XORing` two symbols. However it allows reaching the maximum correction capabilities of the code. It is equivalent to the ML decoding over the full system which is hopefully simplified by the previous decoders.

Let us now detail the operations of this decoding. The first step of ML decoding (i.e., in our case, based on the GE method) is to solve the binary system simplified during the (IT+RS) decoding procedure. This system is a subset of the original system that is only composed with the binary equations of H_L.
If the receiver cannot solve the simplified system with the binary ML decoder, then the receiver switches to the non binary ML decoder. This decoder is based on GE method too, but it considers the global linear system. This global linear system is composed by the previous binary linear system plus equations that define the relations between the received extra-repair symbols, the source symbols, and the repair LDPC symbols using the H_L matrix and the RS generator matrices of the received extra-repair symbols. Finally, if the resolution completes successfully, all the erased source symbols are marked as decoded, otherwise the ML decoding fails and the erased source symbols remain unknown. This causes the failure of the hybrid decoding. The hybrid (IT/RS/ML) decoding algorithm is presented in Algorithm 3.
We note that the two schemes use the same algorithm of hybrid decoding whereas the difference between them is localized in the global linear system. More precisely, the two schemes have not the same non binary sub-system, which is composed from equations associated to the received extra-repair symbols. Therefore, scheme B has this sub-system more dense than that of scheme A.

4.3 Asymptotic analysis of GLDPC-Staircase codes under (IT+RS) and ML decoding

4.3.1 Preliminaries

In the sequel, we denote by d_{vmax} and d_{cmax} the maximum variable and check node degrees respectively in the bipartite (Tanner) graph associated with the LDPC-Staircase code. Following [23], we define the *edge-perspective* DD polynomials by $(\lambda(x), \rho(x))$ and the *node perspective* DD polynomials by $(L(x), R(x))$ given in equations (2.13) and (2.14)

Algorithm 3 Hybrid decoder of GLDPC-Staircase codes

```
(IT+RS) decoder: decoding with new received symbol
if all erased source symbols are recovered then                ▷ Check the end of decoding
    Status OK
else                                                           ▷ Trigger the ML decoding
    Do binary ML decoding                                      ▷ Using the simplified H_L
    if all erased source symbols are recovered then            ▷ Check the end of decoding
        Status OK
    else
        Do non binary ML decoding ▷ Using the simplified H_L and the extra-repair equations
        if all erased source symbols are recovered then        ▷ Check the end of decoding
            Status OK
        else
            Status Failure
        end if
    end if
end if
```

respectively.
Given a GLDPC-Staircase code, DD pair (λ, ρ) are defined by the underlying LDPC-Staircase code, defined by the bottom graph of Figure 4.1 (that is, not containing the extra-repair nodes). Assume that transmission takes place over an erasure channel with parameter ε. We denote by $\mathscr{E}(\lambda, \rho, f_e)$ the ensemble of GLDPC-Staircase with D.D pair (λ, ρ), and with f_e the fraction of generalized check nodes with e extra-repair symbols as presented in equation (4.6).

4.3.2 Density Evolution

4.3.2.1 Introduction

Over erasure channels, DE becomes one-dimensional, and it allows to analysis and even to construct capacity-achieving codes [94]. It works by recursively tracking the erasure probability messages passed around the edges of the graph during IT decoding. Roughly speaking, this means that it recursively computes the *fraction of erased messages passed during the IT decoding*. Using this technique, the decoding threshold of codes is defined as the supremum value of ε (that is, the worst channel condition) that allows transmission with an arbitrary small error probability assuming N goes to infinity [23].
Let us determine the DE equations of GLDPC-Staircase codes.

4.3.2.2 DE equations of GLDPC-Staircase codes

Assume that an arbitrary GLDPC-Staircase code from $\mathscr{E}(\lambda, \rho, f_e)$, with length N_G goes to infinity. We are interested in the erasure probability of messages exchanges by the (IT+RS) decoding along the messages of the LDPC-Staircase code using extra-repair variable nodes. We denote by:

- P_ℓ, the probability of an LDPC symbol (source or repair) node sending an erasure at iteration ℓ to the connected generalized check nodes. Clearly, P_0 is equal to the

channel erasure probability ε.

- Q_ℓ, the probability of a generalized check node sending an erasure (to an LDPC symbol-node) at iteration ℓ.

The calculus of these probabilities depends on the coding scheme used to design the GLDPC-Staircase code (scheme A or B). Next, we give more details for each case. At iteration ℓ, the LDPC symbols are erased with probability P_ℓ, while extra-repair symbols are always erased with probability ε (the channel erasure probability).

Scheme A: The first repair symbol generated by any RS code is one of the repair symbols of the LDPC-Staircase code.
Consider a generalized check node c connected to symbol-nodes $(v_1,\ldots,v_d,e_{1,c},\ldots,e_{e(c),c})$ where v_i denotes an LDPC (source or repair) symbol node and $e_{i,c}$ denotes the i^{th} extra-repair symbol node. Since c corresponds to an RS code, it can recover the value of an LDPC symbol node, say v_1, if and only if the number of erasures among the other symbol-nodes $(v_2,\ldots,e_{e(c),c})$ is less than or equal to $e(c)$.
It follows that the probability of a generalized check node c recovering the value of an LDPC symbol at iteration $\ell+1$, denoted by $\bar{Q}_{\ell+1,A}(d,e(c))$, is given by:

$$\bar{Q}_{\ell+1,A}(d,e(c)) = \sum_{\substack{0\leq i<d, 0\leq j\leq e(c)\\ i+j\leq e(c)}} \binom{d-1}{i} P_{\ell,A}^i (1-P_{\ell,A})^{d-1-i} \binom{e(c)}{j} \varepsilon^j (1-\varepsilon)^{e(c)-j} \tag{4.10}$$

Hence, the probability of a generalized check node c sending an erasure to an LDPC symbol at iteration $\ell+1$ is $(1-\bar{Q}_{\ell+1,A}(d,e(c)))$. Averaging over all possible values of d and $e(c)$, we get:

$$Q_{\ell+1,A} = 1 - \sum_{d=1}^{d_{cmax}} \rho_d \sum_{e=0}^{E} f_e \bar{Q}_{\ell+1,A}(d,e) \tag{4.11}$$

Scheme B: All the LDPC-Staircase repair symbols are source symbols for the RS codes. Consider a constraint node c connected to symbol-nodes $(v_1,\ldots,v_d,e_{1,c},\ldots,e_{e(c),c})$ where v_i denotes an LDPC (source or repair) symbol node and $e_{i,c}$ denotes the i^{th} extra-repair symbol node. The node c corresponds both to a parity check constraint between LDPC symbol nodes $(v_1,\ldots,v_d)$ and to an RS linear constraint between all the symbol-nodes $(v_1,\ldots,v_d,e_{1,c},\ldots,e_{e(c),c})$.
Thus, c can recover the value of an LDPC symbol node, say v_1, if and only if one of the following (disjoint conditions) holds:

(1) there are no erased symbols among $v_2,\ldots,v_d$ *(i.e, LDPC decoding)*;

(2) there is at least one erased symbol among $v_2,\ldots,v_d$, but the number of erasures among all the symbol-nodes $(v_1,\ldots,v_d,e_{1,c},\ldots,e_{e(c),c})$ is less than or equal to $e(c)-1$.

The second condition is also equivalent to the following one:

($2'$) the number of erased symbols among $v_2, \ldots, v_d$ is equal to i and the number of erased symbols among $e_{1,c}, \ldots, e_{e(c),c}$ is equal to j, with $1 \leq i \leq \min(d-1, e(c)-1)$ and $0 \leq j \leq e(c)-1-i$.

It follows that the probability of a generalized check node c recovering the value of an LDPC symbol at iteration $\ell+1$, denoted by $\bar{Q}_{\ell+1,B}(d, e(c))$, is given by:

$$\begin{aligned}\bar{Q}_{\ell+1,B}(d, e(c)) = (1-P_{\ell,B})^{d-1} + \sum_{i=1}^{\min(d-1,e(c)-1)} \sum_{j=0}^{e(c)-1-i} \\ \binom{d-1}{i} P_{\ell,B}^{i} (1-P_{\ell,B})^{d-1-i} \binom{e(c)}{j} \varepsilon^{j} (1-\varepsilon)^{e(c)-j}.\end{aligned} \tag{4.12}$$

Averaging over all possible values of d and $e(c)$, we get:

$$Q_{\ell+1,B} = 1 - \sum_{d=1}^{d_{cmax}} \rho_d \sum_{e=0}^{E} f_e \bar{Q}_{\ell+1,B}(d, e) \tag{4.13}$$

Remark 4.3.1. *For both schemes with regular distribution of extra-repair symbols, all the generalized check nodes have E extra-repair symbols, the equations (*4.11*) and (*4.13*) are reduced to:*

$$Q_{\ell+1} = 1 - \sum_{d=1}^{d_{cmax}} \rho_d \bar{Q}_{\ell+1}(d) \tag{4.14}$$

Conversely, for both schemes, an LDPC symbol node v of degree d, connected to generalized check nodes $c_1, \ldots, c_d$, sends an erasure to c_1 iff it was erased by the channel, and it received erased messages from all generalized check nodes $c_2, \ldots, c_d$. Since this happens with probability $\varepsilon \cdot Q_{\ell+1}^{d-1}$, and averaging over all possible degrees d, we get:

$$P_{\ell+1} = \varepsilon \sum_{d=1}^{d_{vmax}} \lambda_d Q_{\ell+1}^{d-1} = \varepsilon \lambda(Q_{\ell+1}) \tag{4.15}$$

For both schemes, using equations (4.10 or 4.12), (4.11 or 4.13), and (4.15) we can determine a recursive relation between P_ℓ and $P_{\ell+1}$, with $P_0 = \varepsilon$.
The decoder can recover from a fraction of ε erased symbols iff $\lim_{\ell \to +\infty} P_l = 0$. This means that, when $l \to +\infty$, the (IT+RS) decoding succeeds if the DE recursion converges to zero. Then, the (IT+RS) decoding threshold of an GLDPC-Staircase code over an erasure channel is defined as the supremum value of ε such that the DE recursion converges to zero. Therefore, the (IT+RS) decoding threshold can be computed by:

$$\varepsilon^{(\text{IT+RS})}(\mathscr{E}) = \max\{\varepsilon \mid \lim_{\ell \to +\infty} P_l = 0\}. \tag{4.16}$$

If we transmit at $\varepsilon \leq \varepsilon^{(IT+RS)}$, then all the erased LDPC symbols can be recovered. But if we transmit at $\varepsilon > \varepsilon^{(IT+RS)}$, then some or all the erased LDPC symbols remain erased

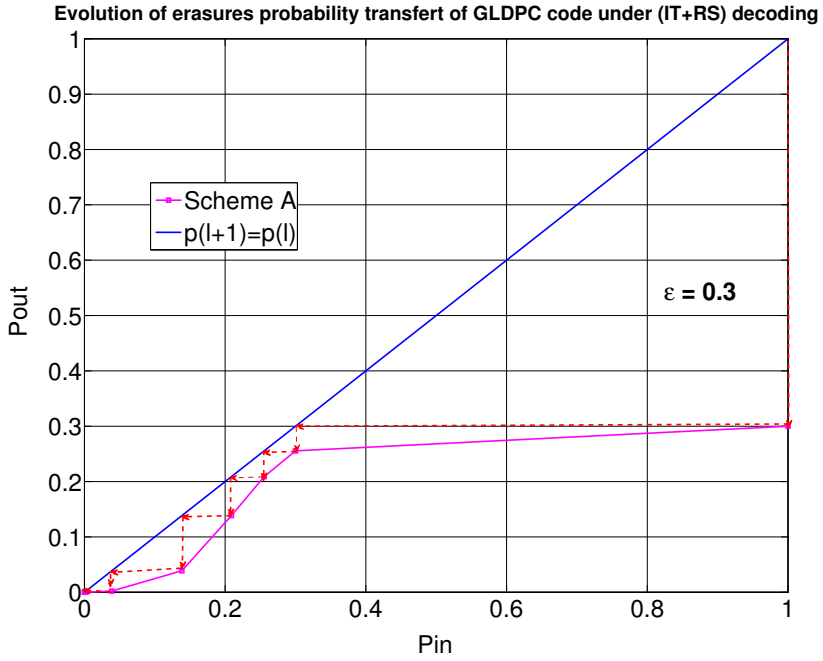


FIGURE 4.4: The evolution, for scheme A, of the (IT+RS) decoding process for LDPC-Staircase with $r_L = 0.8$ and $N1 = 5$. r_G=$\frac{1}{2}$, $E = 3$ and $\varepsilon = 0.3$. Shannon limit=0.5, threshold $\varepsilon^{(IT+RS)}$=0.3443

after the decoding ends.
Additionally, using the DE recursion equation, we can plot the evolution of the (IT+RS) decoding process of an GLDPC-Staircase code for an erasure channel probability ε by tracing $P_{\ell+1} = f(P_\ell)$ with $l \to +\infty$ as shown in the following example.

Example 4.2.

Let us consider a GLDPC-Staircase (scheme A) code with the following parameters:

- Rate: $r_G = \frac{1}{2}$
- Base code: $r_L = 0.8$, N1=5

$$DD : \begin{cases} \lambda(x) = 0.0909.x^1 + 0.9091.x^4, \rho(x) = x^{21} \\ L(x) = 0.2.x^2 + 0.8.x^5, R(x) = x^{22} \end{cases} \tag{4.17}$$

- E=3 (regular distribution of extra-repair symbols).

Figure 4.4 provides the evolution of erasure probability during the (IT+RS) decoding of GLDPC-Staircase at $\varepsilon = 0.3$. The initial fraction of erasure messages emitted by the LDPC variable nodes is $P_0 = 1$. After an iteration (at the next output of the LDPC variable nodes) this fraction has evolved to $P_1 = 0.3$. After second full iteration, i.e., at the output of the LDPC variable nodes, we see an erasure fraction of $P_2 = 0.2555$. This process continues in the same fashion for each subsequent iteration, corresponding graphically to a staircase function which is bounded above by $P_{\ell+1} = P_\ell$ and below by P_{out}.

4.3.3 EXIT functions of GLDPC-Staircase codes

4.3.3.1 Introduction

EXIT technique is a tool for predicting the convergence behavior of iterative processors for a variety of communication problems [86]. Over erasure channel, to visualize the convergence of iterative systems, rather than mutual information, the entropy information can be used (i.e., one minus mutual information). It is natural to use entropy in the setting of the erasure channel since the parameter ε itself represents the channel entropy.
We focus in our work on EXIT based on entropy (see section 2.4.3.5.c) to evaluate the performance of GLDPC-Staircase codes under (IT+RS) and ML decoding. Therefore, we extend the method presented in section 2.4.3.5.c. These EXIT functions are based on DE equations derived in section 4.3.2.
The EXIT technique defined in this section relates to the asymptotic performance of the ensemble $\mathscr{E}(\lambda, \rho, f_e)$ under the decoding.

4.3.3.2 (IT+RS) EXIT function: $h^{(IT+RS)}(\varepsilon)$

The (IT+RS) EXIT function of GLDPC-Staircase code is denoted by $h^{(IT+RS)}(\varepsilon)$. It corresponds to running an (IT+RS) decoder on a very large LDPC-Staircase graph that is connected to the extra-repair variable nodes at ε until the decoder reaches a fixed point. This fixed point defines the stability of erasure probability improvement during decoding iterations. The extrinsic erasure probability of the LDPC-Staircase symbols at this fixed point gives the (IT+RS) EXIT function.
Therefore, consider an $\mathscr{E}(\lambda, \rho, f_e)$, the EXIT function of the GLDPC-Staircase codes under (IT+RS) decoding, over erasure channel (ε), is equal to the following equation:

$$h^{(IT+RS)}(\varepsilon) = \frac{1}{N_L} \sum_{i=1}^{N_L} h_i^{(IT+RS)}(\varepsilon) \tag{4.18}$$

where, $h_i^{(IT+RS)}$ is the extrinsic (IT+RS) erasure probability of LDPC-Staircase symbol "i" as shown in Figure 4.5. $h^{(IT+RS)}(\varepsilon)$ is the asymptotic (average on all the LDPC variable nodes, $N_L \rightarrow +\infty$) extrinsic erasure probability at the output of an (IT+RS) decoding. This function value can be easily computed using the DE equations of GLDPC-Staircase codes. After an infinite number of iterations of the DE recursion (equation (4.15)), the (IT+RS) decoder reaches a fixed point (i.e, $P_{\ell+1} = P_\ell$, $\ell \rightarrow +\infty$).
Hence we can also write:

$$h^{(IT+RS)}(\varepsilon) = L(Q_{+\infty}) \tag{4.19}$$

where $Q_{+\infty}$ is Q_ℓ, derived from the DE equations of GLDPC-Staircase codes in section 4.3.2, when the number of iterations goes to infinity.
Next, we present how can we visualize the evolution of extrinsic erasure probability during (IT+RS) decoding in a graph called EXIT curve.

4.3.3.3 (IT+RS) EXIT curve

The (IT+RS) EXIT curve of the GLDPC-Staircase code under (IT+RS) decoding can be derived, in terms of extrinsic erasure probability (at the output of the decoder) as a

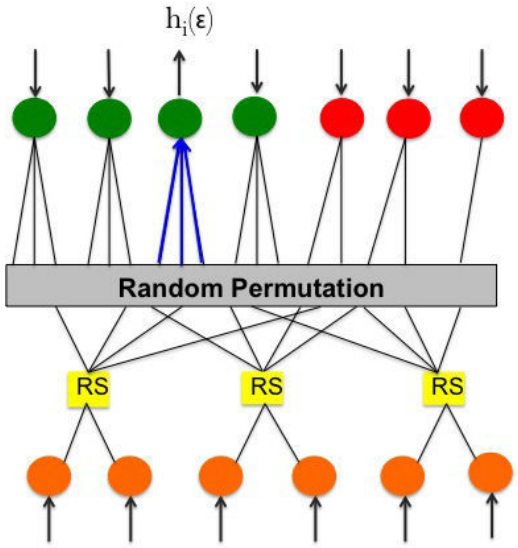


FIGURE 4.5: Computation of EXIT function based on entropy of a GLDPC-Staircase code.

function of the *a prior* erasure probability (input of the decoder, ε).
Therefore, the asymptotic (IT+RS) EXIT curve, denoted by $h^{(IT+RS)}$, is given in a parametric form by,

$$h^{(IT+RS)}(\varepsilon) = \begin{cases} 0 & \text{if} \quad \varepsilon \in [0\ \varepsilon^{(IT+RS)}] \\ L(Q_{+\infty}) & \text{if} \quad \varepsilon \in]\varepsilon^{(IT+RS)}\ 1] \end{cases} \tag{4.20}$$

Summarizing, the (IT+RS) EXIT curve is the trace of $h^{(IT+RS)}(\varepsilon)$ equation for ε starting from $\varepsilon = \varepsilon^{(IT+RS)}$ until $\varepsilon = 1$. In other hand, it is zero up to the (IT+RS) decoding threshold $\varepsilon^{(IT+RS)}$. It then jumps to a non-zero value and also continues smoothly until it reaches one at $\varepsilon = 1$. Therefore, by using this curve, $\varepsilon^{(IT+RS)}$ is given by the value of ε where $h^{(IT+RS)}(\varepsilon)$ drops down to zero.

Example 4.3. *Given a GLDPC-Staircase code with rate $r_G = \frac{1}{3}$, 2 extra-repair symbols per generalized check nodes (regular distribution) and base code with the following parameters:*

- *r_L=0.6, N1=5*
- *DD:*

$$\begin{cases} \lambda(x) = 0.2105x^1 + 0.7895x^4, \rho(x) = x^9, \\ L(x) = 0.4x^2 + 0.6x^5, R(x) = x^{10} \end{cases} \tag{4.21}$$

The (IT+RS) EXIT function $h^{(IT+RS)}(\varepsilon)$ is depicted in Figure 4.6. The (IT+RS) decoding threshold, $\varepsilon^{(IT+RS)}$, is given by the point where $h^{(IT+RS)}(\varepsilon)$ drops down to zero. This gives $\varepsilon^{(IT+RS)}$ =0.5376. It can be seen that $h^{(IT+RS)}(\varepsilon)$ = 0 for values $\varepsilon \leq \varepsilon^{(IT+RS)}$, then it jumps to a non-zero value and continue to increase until it reaches a value of 1 for $\varepsilon = 1$.

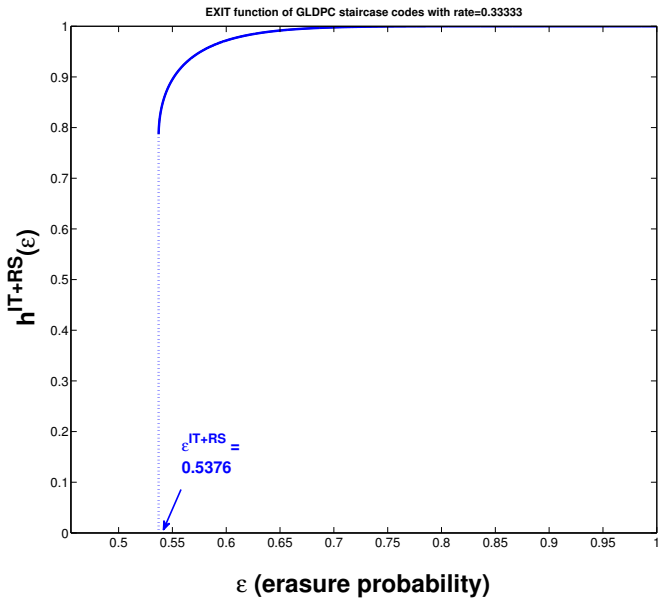


FIGURE 4.6: The (IT+RS) EXIT function $h^{(IT+RS)}(\varepsilon)$ for an ensemble of GLDPC-Staircase code with rate = $\frac{1}{3}$.

4.3.3.4 Upper bound on the ML decoding threshold

As for the (IT+RS) decoding, the EXIT curve of the ML decoding is also defined in terms of extrinsic erasure probability based on entropy. Precisely, in the limit of infinite code length, for a given channel erasure probability ε, $h^{ML}(\varepsilon)$ is the probability of a symbol node being erased after ML decoding, assuming that the received value (if any) of this particular symbol has not been submitted to the decoder. The asymptotic, average on all the LDPC variable nodes, extrinsic erasure probability at the output of an ML decoding (ML EXIT function) is obtained by,

$$h^{ML}(\varepsilon) = \frac{1}{N_L} \sum_{i=1}^{N_L} h_i^{ML}(\varepsilon) \tag{4.22}$$

where, $h_i^{ML}(\varepsilon)$ is the extrinsic erasure probability of LDPC symbol "i" after ML decoding as shown in Figure 4.5.
As mentioned in section 2.4.3.5.c, the exact computation of the EXIT function for the ML decoding is a difficult task [96]. However, using the area theorem [53, 96], we get

$$\int_{\varepsilon^{ML}}^{1} h^{ML}(\varepsilon) = r_G, \tag{4.23}$$

where r_G is the designed coding rate of the given ensemble of GLDPC-Staircase codes. Moreover, since the (IT+RS) decoding is suboptimal with respect to the ML decoding, we have $h^{IT+RS}(\varepsilon) \geq h^{ML}(\varepsilon)$. Hence, if for some $\bar{\varepsilon}^{ML}$

$$\int_{\bar{\varepsilon}^{ML}}^{1} h^{(IT+RS)}(\varepsilon) = r_G, \tag{4.24}$$

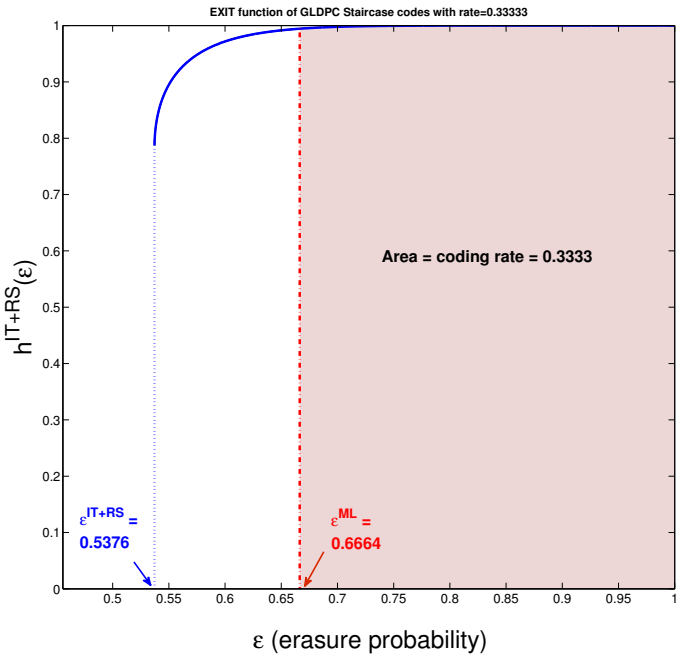


FIGURE 4.7: ML-threshold upper-bound computation using the (IT+RS) EXIT function $h^{(IT+RS)}(\varepsilon)$ for an ensemble of GLDPC-Staircase code with rate = $\frac{1}{3}$.

we necessarily have $\bar{\varepsilon}^{ML} \geq \varepsilon^{ML}$. This gives an upper bound on the ML threshold, which is easily computed using $h^{(IT+RS)}$.

The ML EXIT curve of the GLDPC-Staircase codes, $h^{ML}(\varepsilon)$, can be constructed in the following manner:

- Step 1: Plot the (IT+RS) EXIT curve as parametrized in equation (4.20).
- Step 2: Determine the $\bar{\varepsilon}^{ML}$ by integrate backwards from the right end of the curve where $\varepsilon = 1$. The integration process stops at $\bar{\varepsilon}^{ML}$ where it assure equation (4.24). This gives the upper bound $\bar{\varepsilon}^{ML}$ of the GLDPC-Staircase codes.
- Step 3: The ML EXIT curve is now the curve which is zero at the left of the upper bound on the ML decoding threshold and equals to the (IT+RS) EXIT curve to the right of this decoding threshold (i.e., the (IT+RS) EXIT and the ML EXIT curves coincide above $\bar{\varepsilon}^{ML}$).

Remark 4.3.2. *This upper bound is conjectured to be tight because the GLDPC-Staircase codes are based on LDPC-Staircase codes, which are binary codes and defined by quasi-regular graphs.*

Example 4.4. *Consider the same code of the example 4.3.*

Figure 4.7 shows the (IT+RS) EXIT curve ($h^{(IT+RS)}(\varepsilon)$) and the integral bound on ε^{ML} for GLDPC-Staircase code with the same distributions of the Figure 4.6.
The (IT+RS) decoding threshold value is $\varepsilon^{(IT+RS)} = 0.5376$. The ML decoding threshold

upper-bound is the unique point $\bar{\varepsilon}^{ML} \in [\varepsilon^{(IT+RS)}\ 1]$ such that the red area below the (IT+RS) EXIT curve, delimited by $\varepsilon = \bar{\varepsilon}^{ML}$ at the left and by ε =1 at the right, is equal to the GLDPC code rate, $r_G = 1/3$. In this case, we obtain $\bar{\varepsilon}^{ML}$ = 0.6664.

4.4 Conclusions

In this Chapter, we presented two coding schemes to design GLDPC-Staircase codes. These codes are designed by extending the LDPC-Staircase code (base code) to generalized LDPC-Staircase codes using RS codes (outer codes). The difference in the resulting GLDPC-Staircase codes lies in the repair symbol generated by each generalized check node being (scheme A) or not (scheme B) an RS repair symbol. Therefore, to design scheme A in practice, we proposed to use RS codes based on "quasi" Hankel matrix. Then, we presented a new type of decoder for these codes on the erasure channel called hybrid (IT/RS/ML) decoding. This decoding is able to obtain the correction capabilities of ML decoding with reduced complexity thanks to a preliminary use of (IT+RS) decoding. Additionally, these codes are characterized by three important internal parameters: extra-repair symbols distribution, base code rate, and the N1 parameter of the base code. This leads to obtain flexible GLDPC codes, since these parameters can be adapted according to the receiver conditions.
In the second part of this Chapter, an asymptotic analysis through the DE and EXIT functions methods of these codes for an erasure channel has been proposed to predict the decoding behavior and the decoding convergence. Applying these two techniques, we will investigate under different situations these codes in Chapter 5 and Chapter 6.

Chapter 5

Optimization of GLDPC-Staircase codes

Contents

5.1 Introduction 91

5.2 Experimental conditions 92

5.3 Best coding scheme for GLDPC-Staircase codes 93

5.3.1 Asymptotic results 93

5.3.2 Finite length results 96

5.3.3 Conclusion of the analysis 99

5.4 Tuning internal parameters of GLDPC-Staircase codes 100

5.4.1 The extra-repair symbols distribution 100

5.4.2 N1 parameter 102

5.4.3 The base code rate r_L 103

5.5 Conclusions 105

5.1 Introduction

The main goal of Chapter 4 is to explain the design of the GLDPC-Staircase codes based on RS codes over erasure channels. These codes can be viewed as an extension of LDPC-Staircase code (base code) into generalized LDPC-Staircase code using RS codes. GLDPC-Staircase codes can be constructed using two structures which differ in the type of the generated LDPC repair symbols that are either RS repair symbols or not, as follows:

- Scheme A has the property that on each generalized check node, the repair symbol generated by the LDPC code is also an *RS repair symbol*.
- On the opposite, with scheme B the generated LDPC repair symbol, on each generalized check node, is an *RS source symbol*.

Therefore in this Chapter we start by studying the impacts of the property that *the generated LDPC repair symbols are at the same time RS repair symbols*, on the decoding behavior (i.e, compare scheme A and scheme B).
Then, as noted in Chapter 4, the configuration of these codes depends on the important internal parameters, namely:

- the extra-repair symbols distribution across the H_L rows: regular distribution or irregular uniform distribution,
- the $N1$ parameter of the base code: degree of source variable nodes in H_L,
- the base code rate r_L.

Hence, the best configuration of these parameters for hybrid (IT/RS/ML) decoding will be investigated.
To gauge the correction capabilities of decoding, we use the asymptotic analysis based on DE and EXIT techniques presented in Chapter 4, as well as the finite length analysis.

5.2 Experimental conditions

For the finite length analysis, we have developed a GLDPC-Staircase codec based on RS codes under (IT+RS) and ML decoding methods, in C language, using the Open-FEC.org project (htpp://openfec.org).
In this Chapter, all experiments are carried out by considering a memory-less erasure channel along with a transmission scheme where all the source and repair symbols are sent in a fully random order. This has the benefit to make the performance results independent of the loss model[1] and the target channel loss rate is the only parameter that needs to be considered. Different LDPC-Staircase matrices are used (more precisely we change the PRNG seed used to create the matrix).
Then the results, averaged over the tests obtained by varying LDPC-Staircase matrix, show the average behavior of GLDPC-Staircase codes.
In the sequel, we evaluate the finite length performance based on metrics given in appendix A.

For the asymptotic analysis, we use commonly the following DD of LDPC-Staircase codes as presented in Table 5.1 and Table 5.2. The calculus of these degree distributions is

$N1$	edge DD $(\lambda(x),\rho(x))$	node DD $(L(x),R(x))$
3	$\lambda(x)=0.25.x^1+0.75.x^2$, $\rho(x)=x^7$	$L(x)=0.3333.x^2+0.6667.x^3$, $R(x)=x^8$
4	$\lambda(x)=0.2.x^1+0.8.x^3$, $\rho(x)=x^9$	$L(x)=0.3333.x^2+0.6667.x^4$, $R(x)=x^{10}$
5	$\lambda(x)=0.1666.x^1+0.8333.x^4$, $\rho(x)=x^{11}$	$L(x)=0.3333.x^2+0.6667.x^5$, $R(x)=x^{12}$
6	$\lambda(x)=0.1429.x^1+0.8571.x^5$, $\rho(x)=x^{13}$	$L(x)=0.3333.x^2+0.6667.x^6$, $R(x)=x^{14}$

TABLE 5.1: LDPC-Staircase DD for different values of N1 where $r_L=\frac{2}{3}$.

[1] It is equivalent to the order of packets loss.

r_L	edge DD $(\lambda(x), \rho(x))$	node DD $(L(x), R(x))$
$\frac{1}{2}$	$\lambda(x) = 0.2857.x^1 + 0.7143.x^4$, $\rho(x) = x^6$	$L(x) = 0.5000.x^2 + 0.5000.x^5$, R(x)=$x^7$
$\frac{2}{3}$	$\lambda(x) = 0.1666.x^1 + 0.8333.x^4, \rho(x) = x^{11}$	$L(x) = 0.3333.x^2 + 0.6667.x^5, R(x) = x^{12}$
0.75	$\lambda(x) = 0.1176.x^1 + 0.8824.x^4$, $\rho(x) = x^{16}$	$L(x) = 0.2500.x^2 + 0.75.x^5$, R(x) =$x^{17}$
0.8	$\lambda(x) = 0.0909.x^1 + 0.9091.x^4, \rho(x) = x^{21}$	$L(x) = 0.2.x^2 + 0.8.x^5, R(x) = x^{22}$

TABLE 5.2: LDPC-Staircase DD for different values of r_L where N1=5.

based on the parameters N1, $d_{r_{H_1}}$ (see equation (3.2)), and the structure of LDPC-Staircase codes.
For an irregular uniform distribution of extra-repair symbols, we use the notation $f(\%) = [f_0\ f_1\ f_2\ \ldots f_e]$ to define the fractions of generalized check nodes with e extra-repair symbols. For example, $f(\%) = [25\ 50\ 25]$ means that we have 25% of generalized check nodes have **0** extra-repair symbols, 50% of generalized check nodes have **1** extra-repair symbols, and 25% of generalized check nodes have **2** extra-repair symbols.
We note that hybrid (IT/RS/ML) decoding and ML decoding have the same correction capabilities but they are different at decoding complexity level. Thus, we mention "ML decoding" to refer the correction capabilities obtained by hybrid decoding.

5.3 Best coding scheme for GLDPC-Staircase codes

Throughout this section we investigate the impacts of the property given by scheme A on decoding performance in different configurations of GLDPC-Staircase codes. For this reason, the study allows to determine the best for the hybrid (IT/RS/ML) decoding through a comparison between scheme A and scheme B.

5.3.1 Asymptotic results

Let's consider a base code with distribution defined in Table 5.1 for $r_L = 0.8$ and N1=5. We use the DE equations proposed in section 4.3.2 to plot in Figure 5.1 the evolution of the erasure probability transfer on the graph of GLDPC-Staircase code with r_G=$\frac{1}{2}$ and E=3 (regular distribution) for scheme A ($P_{\ell+1,A} = f(P_{\ell,A})$) and scheme B ($P_{\ell+1,B} = f(P_{\ell,B})$). These curves represent the value of the erasure probability on all the LDPC symbols during the propagation of the erasure probability between generalized check nodes and variable nodes of the GLDPC-Staircase tanner graph where ε) equals to 0.32. This figure shows that the initial fraction of erasure messages emitted by the LDPC variable nodes is P_l = 1 in schemes *A* and *B*. After an iteration (at the next output of the LDPC variable nodes) this fraction has evolved to P_{l+1} = 0.32 for the two schemes.
After second full iteration, i.e., at the output of the LDPC variable nodes, we see that an erasure fraction of scheme *A* is equal to *P* = 0.2889 whereas it is equal to *P* = 0.3117 for scheme *B*. This difference explains that the erasure probability in scheme *A* decreases more quickly than scheme *B* (i.e the correction of the erasure in scheme *A* is better than scheme *B*). After that the process of the transfer continues in the same fashion for each subsequent iteration.
The figure also shows that the process of DE for scheme *B* is stuck at value > 0 (P=0.3094)

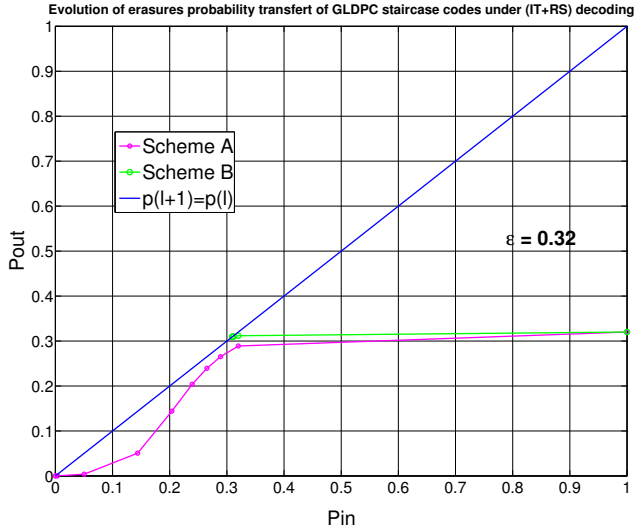


FIGURE 5.1: The evolution, for schemes A and B, of the (IT+RS) decoding process for GLDPC-Staircase with $r_L = 0.8$, $N1 = 5$, r_G=$\frac{1}{2}$, $E = 3$ and $\varepsilon = 0.32$. Shannon limit=0.5, threshold $\varepsilon^{(IT+RS)}$=0.3443 (scheme A) and threshold $\varepsilon^{(IT+RS)}$=0.2819 (scheme B).

while for scheme A the process finishes with $P = 0$. This means that under (IT+RS) decoding and at $\varepsilon = 0.32$ the GLDPC-Staircase codes converge (i.e can recover all the erased LDPC symbols) only with scheme A.

We continue the comparison between the two schemes in terms of *decoding threshold* using the EXIT analysis presented in section 4.3.3. This analysis allows us to compute the (IT+RS) decoding threshold ($\varepsilon^{(IT+RS)}$) and the upper bound on the ML decoding threshold ($\bar{\varepsilon}^{\mathrm{ML}}$). We note that DE also allows to determine the decoding threshold, but it requires several calculations.
Table 5.3 provides the comparison in terms of $\varepsilon^{(IT+RS)}$ and $\bar{\varepsilon}^{\mathrm{ML}}$ between scheme A and scheme B (with regular distribution and $r_L = \frac{2}{3}$) for two global code rates ($\frac{1}{2}$ and $\frac{2}{5}$) and different values of N1. This table reveals that for different values of N1, scheme A outperforms scheme B under (IT+RS) decoding. Therefore, the property that the generated LDPC repair symbols are RS repair symbols helps to get closer to channel capacity limit. Whereas under ML decoding, below of N1=5, scheme B is preferable and beyond this value, this property has no a great significant impact. This is explained as follows. In practice the efficiency of the ML decoder over BEC is related to the densification of its linear system. Therefore, low value of N1 implies a sparse binary linear system of LDPC-Staircase codes which causes degradation on the ML decoding results [51]. Whereas, as mentioned in section 4.2.3.2, the linear system of GLDPC-Staircase codes is composed of a binary sub-system (composed from LDPC-Staircase equations) and a non binary sub-system (composed from extra-repair equations) which is somewhat more dense with scheme B than scheme A. Therefore, this difference has an impact on performance of global system when the binary sub-system is sparse; otherwise it is vanished. We will see next, that N1=5 is the best value for the hybrid decoding type, therefore we prefer scheme A in this case.

In the previous analysis, we fixed the base code rate and the distribution type of extra-repair symbols to study only the impact of the property of scheme A when varying N1. Let us now see the impact of this property when we vary r_L (i.e, vary E) and the distribution of extra-repair symbols with N1 fixed to 5. Table 5.4 provides the comparison in terms

r_G	Decoding threshold	N1	scheme A	scheme B
$\frac{1}{2}$ (E=1)	$\varepsilon^{(IT+RS)}$	3	0.4571	0.2905
		4	0.4269	0.2867
		5	0.3943	0.2709
		6	0.3646	0.2536
	$\bar{\varepsilon}^{\mathrm{ML}}$	3	0.4900	0.4985
		4	0.4976	0.4996
		5	0.4993	0.4997
		6	0.4998	0.4999
$\frac{2}{5}$ (E=2)	$\varepsilon^{(IT+RS)}$	3	0.5532	0.4041
		4	0.5136	0.3878
		5	0.4744	0.3639
		6	0.4392	0.3401
	$\bar{\varepsilon}^{\mathrm{ML}}$	3	0.5946	0.5994
		4	0.5990	0.5998
		5	0.5998	0.5999
		6	0.5999	0.5999

TABLE 5.3: Decoding threshold comparison between scheme A and B for different values of N1 for (IT+RS) decoding and ML decoding. GLDPC-Staircase codes with r_L=$\frac{2}{3}$ and regular distribution of extra-repair symbols.

of $\varepsilon^{(IT+RS)}$ and $\bar{\varepsilon}^{\mathrm{ML}}$ between scheme A and scheme B (with $r_G = \frac{1}{2}$) for different values of r_L (i.e vary E) using a regular distribution of extra-repair symbols. This table proves that, for different values of E > 0, the (IT+RS) decoding threshold of scheme A is higher than that of scheme B. On the opposite, the ML decoding thresholds of the two schemes, for different values of E, are almost equivalent. Additionally, for an irregular uniform distribution of extra-repair symbols, Table 5.5 also shows that, for different values of f (distribution of extra-repair repair symbols), scheme A is better than scheme B for (IT+RS) decoding and both achieve the same ML decoding thresholds.

Let us move to see the results when varying the rate of GLDPC-Staircase code. Table 5.6 provides the comparison in terms of $\varepsilon^{(IT+RS)}$ and $\bar{\varepsilon}^{\mathrm{ML}}$ between scheme A and scheme B for different values of r_G with $r_L = \frac{2}{3}$, $N1 = 5$ and regular distribution. This table reveals that for different rates of GLDPC-Staircase code, scheme A outperforms scheme B under (IT+RS) decoding and both have the same behavior under ML decoding. For an irregular uniform distribution, in Table 5.7, we provide a comparison between decoding thresholds of the two schemes with $r_L = \frac{2}{3}$ for different values of f (distribu-

Decoding threshold	E	r_L	scheme A	scheme B
$\varepsilon^{(IT+RS)}$	0	$\frac{1}{2}$	0.4380	0.4380
	1	$\frac{2}{3}$	0.3943	0.2709
	2	0.75	0.3647	0.2773
	3	0.8	0.3443	0.2819
$\bar{\varepsilon}^{\mathrm{ML}}$	0	$\frac{1}{2}$	0.4946	0.4946
	1	$\frac{2}{3}$	0.4993	0.4997
	2	0.75	0.4999	0.4999
	3	0.8	0.4999	0.4999

TABLE 5.4: Decoding threshold comparison between (IT+RS) decoding and ML decoding, for r_G=$\frac{1}{2}$ (Shannon limit =0.5) and regular distribution of extra-repair symbols.

Decoding threshold	$f(\%)$	r_L	scheme A	scheme B
$\varepsilon^{(IT+RS)}$	[0]	$\frac{1}{2}$	0.4380	0.4380
	[25 50 25]	$\frac{2}{3}$	0.4064	0.3029
	[0 50 25 0 25]	0.75	0.3874	0.3089
	[5 10 15 20 50]	0.8	0.3589	0.3007
$\bar{\varepsilon}^{\mathrm{ML}}$	[0]	$\frac{1}{2}$	0.4946	0.4946
	[25 50 25]	$\frac{2}{3}$	0.4993	0.4997
	[0 50 25 0 25]	0.75	0.4998	0.4998
	[5 10 15 20 50]	0.8	0.4998	0.4998

TABLE 5.5: Decoding threshold comparison between (IT+RS) decoding and ML decoding, for r_G=$\frac{1}{2}$ (Shannon limit=0.5) and irregular uniform distribution of extra-repair symbols.

tion of extra-repair symbols). This table also shows the same results for regular distribution.

Therefore, for all configurations of GLDPC-Staircase codes, the structure of scheme A resists to the channel loss more than scheme B under (IT+RS) and both have the same behavior under ML decoding with dense system.

5.3.2 Finite length results

This section aims to give additional claims on the impact of the property of scheme A in terms of decoding inefficiency ratio, decoding overhead, decoding failure probability and error floor. All results are determined using N1 = 5 and a regular distribution of extra-repair symbols.
Figures 5.2(b) and 5.3(b) provide the average (over $1,000$ different codes) decoding inefficiency ratio of both schemes under ML decoding for two different code rates ($\frac{1}{2}$ and $\frac{1}{3}$). They show that no matter the dimension, K, both schemes perform the same, with results quite close to that of MDS codes (characterized by an decoding inefficiency ratio always equals to 1). This means that for small and large object size, the property of scheme

Decoding threshold	r_G	E	scheme A	scheme B	Shannon limit
$\varepsilon^{(IT+RS)}$	$\frac{2}{3}$	E=0	0.2709	0.2709	0.3333
	$\frac{1}{2}$	E=1	0.3943	0.2709	0.5
	$\frac{2}{5}$	E=2	0.4744	0.3639	0.6
	$\frac{1}{3}$	E=3	0.5332	0.4394	0.6667
$\bar{\varepsilon}^{\mathrm{ML}}$	$\frac{2}{3}$	E=0	0.3301	0.3301	0.3333
	$\frac{1}{2}$	E=1	0.4993	0.4997	0.5
	$\frac{2}{5}$	E=2	0.5998	0.5999	0.6
	$\frac{1}{3}$	E=3	0.6666	0.6666	0.6667

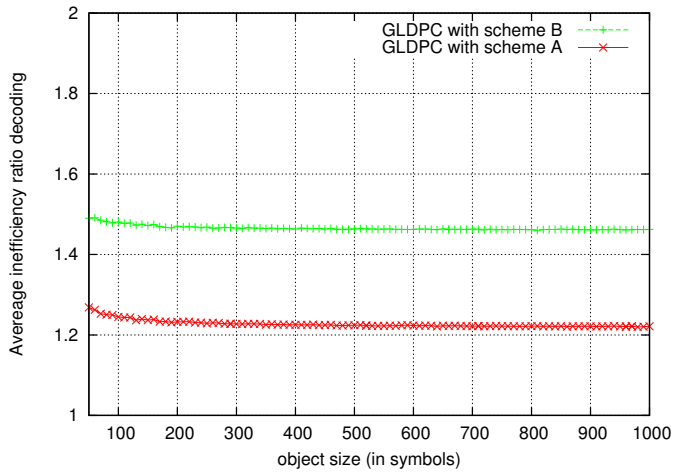
TABLE 5.6: Decoding threshold comparison between (IT+RS) decoding and ML decoding for different values of r_G with $r_L = \frac{2}{3}$ and regular distribution of extra-repair symbols.

Decoding threshold	r_G	$f(\%)$	scheme A	scheme B	Shannon limit
$\varepsilon^{(IT+RS)}$	$\frac{2}{3}$	[0]	0.2709	0.2709	0.3333
	$\frac{1}{2}$	[25 50 25]	0.4064	0.3029	0.5
	$\frac{2}{5}$	[0 25 50 25]	0.4819	0.3765	0.6
	$\frac{1}{3}$	[0 25 25 10 15 15 10]	0.5598	0.4768	0.6667
$\bar{\varepsilon}^{\mathrm{ML}}$	$\frac{2}{3}$	[0]	0.3301	0.3301	0.3333
	$\frac{1}{2}$	[25 50 25]	0.4993	0.4997	0.5
	$\frac{2}{5}$	[0 25 50 25]	0.5997	0.5999	0.6
	$\frac{1}{3}$	[0 25 25 10 15 15 10]	0.6665	0.6666	0.6667

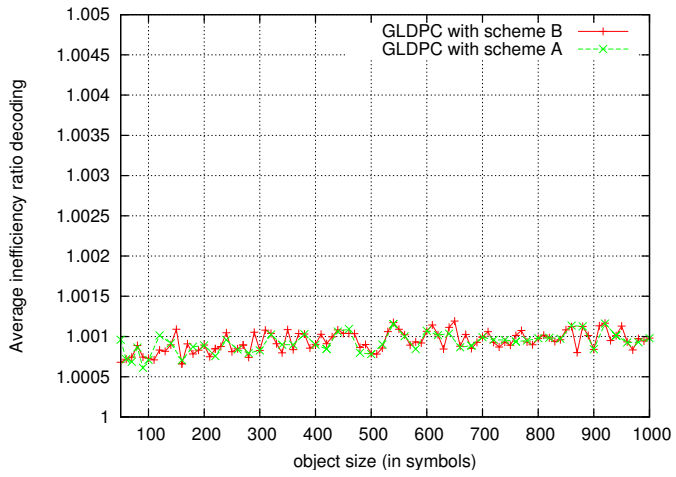
TABLE 5.7: Decoding threshold comparison between (IT+RS) decoding and ML decoding for different values of r_G with $r_L = \frac{2}{3}$ and irregular uniform distribution of extra-repair symbols.

A has no impact on the ML decoding inefficiency ratio. These results hold for the two considered code rates. Figures 5.2(a) and 5.3(a) do the same in case of (IT+RS) decoding only. They show that scheme A exhibits the lowest average decoding inefficiency ratio in all cases. This is made possible by a higher number of RS repair symbols (i.e, increase of the minimum distance) for scheme A, which mechanically increases the success probability of decoding an erased symbol on each generalized check nodes. The increase of the RS repair symbols also avoids stopping sets associated to short cycles that stuck (IT+RS) decoding. This means that scheme A is more efficient on (IT+RS) decoding than scheme B.

In order to go further to see the error floor and overhead achieved by each schemes under ML decoding, we analyze the ML decoding failure probability. In Figure 5.4, we plot the ML decoding failure probability versus channel loss percentage (in Figure 5.4(a)) and versus number of received symbols (in Figure 5.4(b)). To that purpose we choose $r_G = \frac{1}{2}$ (E=1), K=1000, and 10^6 tested codes. In Figure 5.4(a), the black vertical line corresponds to ideal, MDS code, for which the decoding failure is equal to 0 as long as the experienced



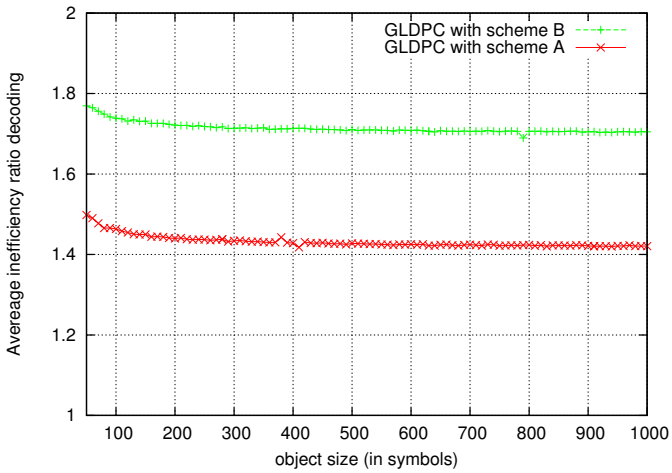
(a) (IT+RS) decoding, $r_G = \frac{1}{2}$ $(E = 1)$



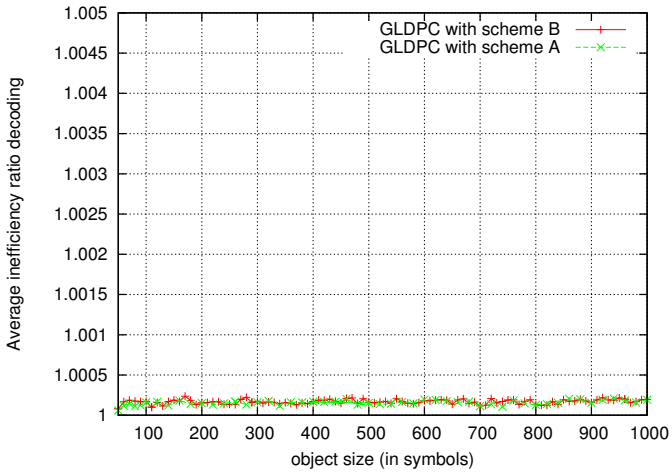
(b) ML decoding, $r_G = \frac{1}{2}$ $(E = 1)$

FIGURE 5.2: Average performance under (IT+RS) decoding or ML decoding, with rate $\frac{1}{2}$, as a function of K.

loss rate is strictly inferior to 50% for $r_G = \frac{1}{2}$. This figure confirms that for two schemes, the GLDPC-Staircase codes have a very small decoding failure probability, with no visible error floor above 10^{-5}. The little difference between the two curves is readable at the foot where we test several codes (i.e at 49.45% scheme A has $4.16.10^{-6}$ as decoding failure probability whereas scheme B has $5.45.10^{-6}$). Figure 5.4(b) gives additional details of the behavior of the two schemes using ML decoding. This figure confirms that scheme B has almost same decoding overhead as scheme A (i.e, with *6 symbols added to K*, scheme A has $6.93.10^{-6}$ decoding failure probability while $7.27.10^{-6}$ with scheme B for channel erasure probability equals to 49.6%). Also, the two schemes achieve $\simeq 5.10^{-2}$ decoding failure probability with overhead equals to 2. Therefore, both schemes have a very small decoding overhead, close to that of MDS codes.



(a) (IT+RS) decoding, $r_G = \frac{1}{3}$ ($E = 3$)

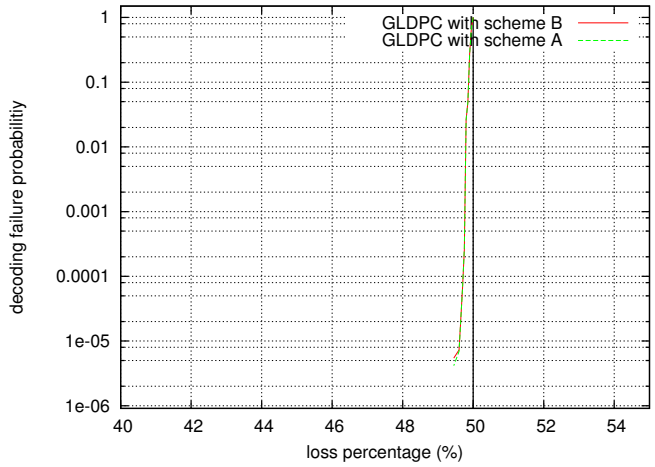


(b) ML decoding, $r_G = \frac{1}{3}$ ($E = 3$)

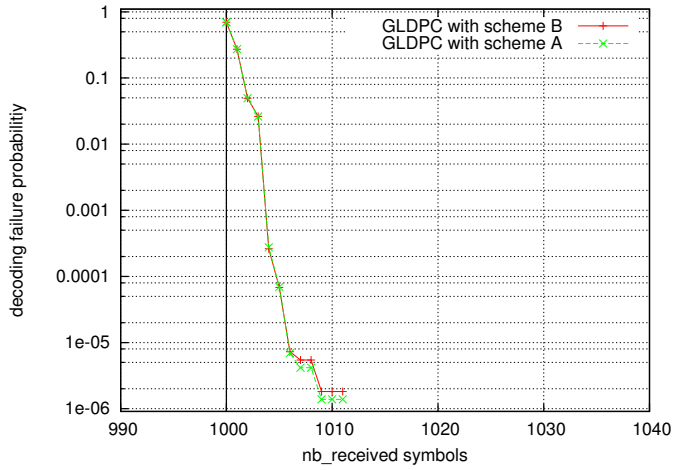
FIGURE 5.3: Average performance under (IT+RS) decoding or ML decoding, with rate $\frac{1}{3}$, as a function of K.

5.3.3 Conclusion of the analysis

The asymptotic analysis and finite length analysis confirms that all results prove that scheme A is globally the best solution: it significantly performs better than scheme B with an (IT+RS) decoding and leads similar performance to scheme B with an ML decoding with dense system. Thus to design a GLDPC-Staircase codes, with hybrid (IT/RS/ML) decoding, we must choose scheme A. Therefore, the rest of this document will only consider scheme A.



(a)



(b)

FIGURE 5.4: Decoding failure probability under ML decoding, with rate $\frac{1}{2}$ and $K = 1000$, as a function of the number of received symbols.

5.4 Tuning internal parameters of GLDPC-Staircase codes

In this section, we analyze the impact of three configuration parameters of GLDPC-Staircase codes on the erasure recovery performance in order to obtain the best configuration over hybrid (IT/RS/ML) decoding.

5.4.1 The extra-repair symbols distribution

As shown in section 4.2.1, we can distribute the extra-repair symbols on the generalized check nodes in two ways: regular, or irregular uniform distribution. In [52], based on asymptotic results, it is shown that these codes with (IT+RS) decoding perform the best under an irregular uniform distribution rule. However, in our work, we consider also the ML decoding scheme and the situation is completely different. Therefore we test these

two distributions to determine the best on each decoding type.
Figure 5.5 provides the average decoding inefficiency ratio (i.e, average of 1000 GLDPC codes with r_G=$\frac{1}{2}$) of GLDPC-Staircase codes, for different object sizes. It shows that the regular distribution performs significantly better under ML decoding, both with small and large objects.

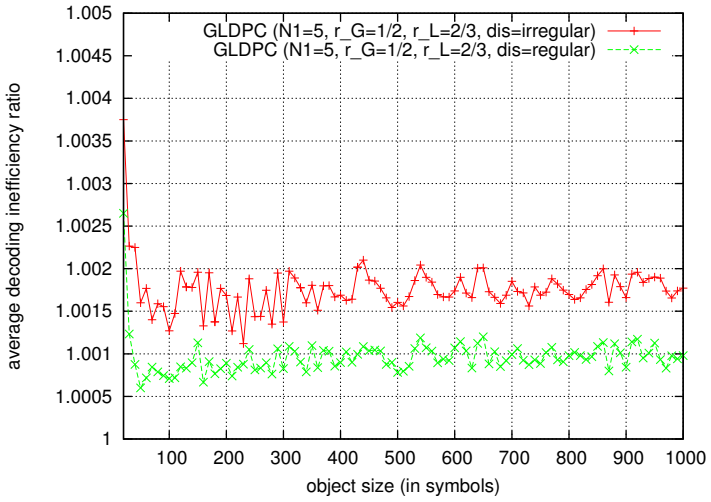


FIGURE 5.5: Performance for irregular uniform distribution versus regular distribution of extra-repair symbols, with N1= 5, $r_L = \frac{2}{3}$, and r_G=$\frac{1}{2}$

Based on asymptotic results, we give the gaps to capacity of GLDPC-Staircase codes with irregular uniform distribution and regular distribution of extra-repair symbols *for different code rates r_G with N1=5 and $r_L = \frac{2}{3}$* in Table 5.8. In addition, we provide in Table 5.9 the gaps to capacity of the two distributions *for different values of r_L with fixed global code rate $r_G = \frac{1}{2}$ and N1=5*.
The gap to capacity (Δ) is computed using the following equation:

$$\Delta = 1 - r_G - \varepsilon_{th} \quad \text{(with } \varepsilon_{th} \text{ is the decoding threshold).} \tag{5.1}$$

These tables show that, for different values of global code rate and base code rate, under (IT+RS) decoding, the GLDPC-Staircase codes produce higher gap to capacity with a regular distribution rather than with an irregular uniform distribution. While, under ML decoding, the regular distribution allows to have GLDPC-Staircase codes very close to the channel capacity more than the irregular uniform distribution. This confirms the finite length analysis.
Let us see the case where N1 vary. Table 5.10 provides the decoding thresholds of the two decoding types for $r_L = \frac{2}{3}$ for the two distributions into different values of global code rates. This table also shows that ML decoding favors the regular distribution for different values of N1 (more advantage with low value of N1). Therefore, for different values of N1, r_L and r_G the regular distribution is accorded to the ML decoding whereas irregular uniform distribution is suitable for (IT+RS) decoding.

As our objective is to focus on the hybrid decoding to achieve the ML performance, therefore in the remaining of this work we only focus on the regular distribution, where there are exactly $e(c) = E$ extra-repair symbols per generalized check node c.

Decoding type	r_G	E	Δ_1	$f(\%)$	Δ_2
(IT+RS)	$\frac{2}{3}$	E=0	0.06243	[0]	0.06243
	$\frac{1}{2}$	E=1	0.1057	[25 50 25]	0.0936
	$\frac{2}{5}$	E=2	0.1256	[0 25 50 25]	0.1181
	$\frac{1}{3}$	E=3	0.13346	[0 25 25 10 15 15 10]	0.1068
	$\frac{1}{3,5}$	E=4	0.1350	[0 25 0 0 0 75]	0.1167
	$\frac{1}{5}$	E=7	0.1268	[0 0 0 0 0 0 10 80 10]	0.1262
ML	$\frac{2}{3}$	E=0	0.00323	[0]	0.00323
	$\frac{1}{2}$	E=1	0.0007	[25 50 25]	0.0007
	$\frac{2}{5}$	E=2	0.0002	[0 25 50 25]	0.0003
	$\frac{1}{3}$	E=3	0.0001	[0 25 25 10 15 15 10]	0.0002
	$\frac{1}{3,5}$	E=4	0.000081	[0 25 0 0 0 75]	0.0000857
	$\frac{1}{5}$	E=7	0.00007	[0 0 0 0 0 0 10 80 10]	0.00008

TABLE 5.8: (IT+RS) decoding and ML decoding gaps to capacity for irregular uniform distribution (Δ_2) versus regular distribution (Δ_1) of extra-repair symbols for GLDPC-Staircase codes with $N1 = 5$, and $r_L = \frac{2}{3}$.

Decoding type	r_L	E	Δ_1	$f(\%)$	Δ_2
(IT+RS)	$\frac{1}{2}$	0	0.062	[0]	0.062
	$\frac{2}{3}$	1	0.1057	[25 50 25]	0.0936
	0.75	2	0.1353	[0 50 25 0 25]	0.1126
	0.8	3	0.1557	[5 10 15 20 50]	0.1411
ML	$\frac{1}{2}$	0	0.0054	[0]	0.0054
	$\frac{2}{3}$	1	0.0007	[25 50 25]	0.0007
	0.75	2	0.0001	[0 50 25 0 25]	0.0002
	0.8	3	0.0001	[5 10 15 20 50]	0.0002

TABLE 5.9: (IT+RS) decoding and ML decoding gaps to capacity for irregular uniform distribution (Δ_2) versus regular distribution (Δ_1) of extra-repair symbols for GLDPC-Staircase codes with $N1 = 5$, and $r_G = \frac{1}{2}$.

5.4.2 N1 parameter

Let us now adjust the second internal parameter of the GLDPC-Staircase codes, N1, which represents the degree of the source variable nodes of the base code matrix H_L.

The increase of N1 parameter causes the "densification" of H_L and maybe the decrease of the smallest stopping sets size. It is well known that the effectiveness of IT decoding over erasure channel is related to the sparseness of the LDPC graph. In addition, the correction capabilities of LDPC codes is limited by the size of smallest stopping sets (see section 2.4.3.6).

Since, in our case we have IT and RS decoding working together, let's see if the RS decoding helps IT decoding to prevent the negative impact of the densification of the graph on the decoding (i.e., to see if the densification also causes the degradation of (IT+RS) decoding performance). Therefore, in Table 5.11 we provide the average decoding inefficiency ratio (i.e, average of 1000 GLDPC-Staircase codes for each r_G) for different r_G versus N1 under (IT+RS) decoding with $K = 1000$ and $r_L = \frac{2}{3}$. This table shows that,

Distribution type	Decoding threshold	$N1$	$r_G = \frac{1}{2}$	$r_G = \frac{2}{5}$	$r_G = \frac{1}{3}$
regular distribution	$\varepsilon^{(IT+RS)}$	3	0.4571	0.5532	0.6184
		4	0.4269	0.5136	0.5753
		5	0.3943	0.4744	0.5332
		6	0.3646	0.4392	0.4955
	$\bar{\varepsilon}^{\mathrm{ML}}$	3	0.49	0.5946	0.6633
		4	0.4976	0.5990	0.6661
		5	0.4993	0.5998	0.6666
		6	0.4998	0.5999	0.6666
irregular uniform distribution	$\varepsilon^{(IT+RS)}$	3	0.4605	0.5563	0.6287
		4	0.4365	0.5196	0.5959
		5	0.4064	0.4819	0.5598
		6	0.3779	0.4476	0.5257
	$\bar{\varepsilon}^{\mathrm{ML}}$	3	0.4863	0.5935	0.6600
		4	0.4966	0.5987	0.6654
		5	0.4993	0.5997	0.6666
		6	0.4997	0.5998	0.6666

TABLE 5.10: Threshold decoding comparison between the two distributions of extra-repair symbols for different values of N1 and r_G with $r_L = \frac{2}{3}$.

for different r_G and under (IT+RS) decoding, increasing N1 induces an increase in the average decoding inefficiency ratio. This means that, the extra-repair symbols which are used to cope with the problem of small stopping sets in the base code graph doesn't succeed. Moreover, increasing E (i.e, reducing the GLDPC code rate) does not solve the problem. Therefore, the increase of N1 leads to the deterioration of the ability to correct with (IT+RS) decoding. This table also shows that, for different GLDPC code rates, the behavior of ML decoding is totally different than the one observed for (IT+RS) decoding. We give the same remarks using the Table 5.12 where we compute the threshold of (IT+RS) decoding and ML decoding for different value of $N1$. Additionally, the decoding complexity depends on the number of XOR operations on the graph of IT decoding, whereas it depends on the size and the density of the linear system to be solved of ML decoding [51]. Then, the increases of N1 has a negative impact on the IT and ML decoding complexity.
Therefore, with ML decoding, with respect to the low decoding complexity, the most significant performance gains are obtained by switching from $N1 = 3$ to 5. Above this value, the performance only improves slightly. This value N1=5 also limits the performance degradation of (IT+RS) decoding compared to values $N1 > 5$. N1 = 5 is the best value that can be used by the hybrid (IT/RS/ML) decoding.

5.4.3 The base code rate r_L

Let us consider a GLDPC-Staircase code rate r_G. Several values of the base code rate r_L, or equivalently of E, enable to achieve the global code rate r_G (see equation (4.7)).

r_G	N1	(IT+RS) decoding	ML decoding
$\frac{2}{3}$ (E=0)	3	1.06669	1.04225
	5	1.09682	1.00636
	6	1.12217	1.00396
	7	1.14624	1.00250
$\frac{1}{2}$ (E=1)	3	1.10487	1.01627
	5	1.22160	1.00097
	6	1.27825	1.00040
	7	1.32857	1.00009
$\frac{1}{3}$ (E=3)	3	1.17963	1.00668
	5	1.42080	1.00019
	6	1.53045	1.00007
	7	1.62660	1.00001

TABLE 5.11: Average decoding inefficiency ratio us a function of N1 and the decoding scheme (IT+RS versus ML), for $K = 1000$ and $r_L = \frac{2}{3}$.

However choosing a value impacts the achieved performance.
In Figure 5.6, we plot the average ML decoding inefficiency ratio (i.e, average of 1000 GLDPC codes with r_G=$\frac{1}{3}$) for different object size. This figure shows that increasing r_L rate (i.e, increasing the number E), the average decoding inefficiency ratio quickly approaches 1 (i.e., ideal code) as $E = 3$ (i.e, $r_L = \frac{2}{3}$), even for very small code dimensions. We also apply the techniques developed in Chapter 4 to adjust E, by computing the upper

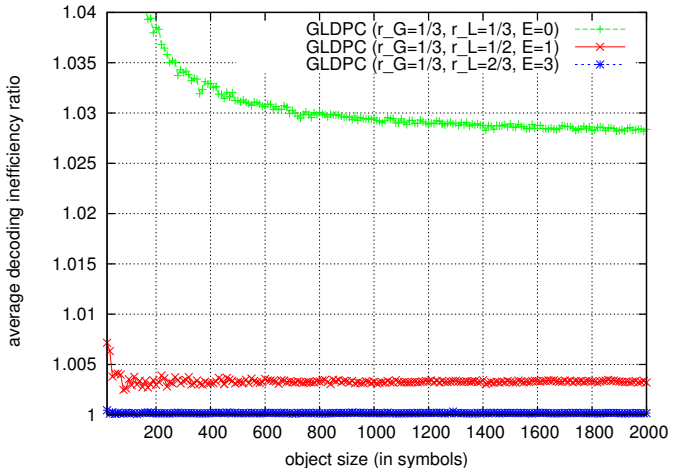


FIGURE 5.6: Performance as a function of base code rate r_L with N1= 5, r_G=$\frac{1}{3}$, and ML decoding.

bound on the ML decoding threshold for several values of E. These results are summarized in Table 5.13 and compared to the Shannon capacity limit (δ_{sh}). We notice that increasing E (or equivalently increasing the LDPC code rate) quickly increases the upper bound on the ML decoding threshold, until it reaches a stable value very close to the Shannon limit

r_L	Decoding threshold	$N1$	E=2	E=3
$r_L = \frac{2}{3}$	$\varepsilon^{(IT+RS)}$	3	0.5532	0.6184
		4	0.5136	0.5753
		5	0.4744	0.5332
		6	0.4392	0.4955
	$\bar{\varepsilon}^{\mathrm{ML}}$	3	0.5946	0.6633
		4	0.5990	0.6661
		5	0.5998	0.6666
		6	0.5999	0.6666
$r_L = \frac{1}{2}$	$\varepsilon^{(IT+RS)}$	3	0.7236	0.7769
		4	0.7002	0.7530
		5	0.66698	0.7232
		6	0.6387	0.6927
	$\bar{\varepsilon}^{\mathrm{ML}}$	3	0.7403	0.7933
		4	0.7476	0.7985
		5	0.7493	0.7996
		6	0.7498	0.7999

TABLE 5.12: Decoding threshold comparison between (IT+RS) decoding and ML decoding versus $N1$ for different values of r_L and r_G.

δ_{sh}. Depending on r_G this stable value is obtained with $E = 1$, 2 or 3. Therefore, a small number of extra-repair symbols per generalized check node is sufficient to get extremely close to the channel capacity. These results are identical to the finite length performance

r_G	**E = 0**	**E= 1**	**E= 2**	**E= 3**	**E= 4**	**E=5**	$\delta_{\mathbf{sh}}$
$1/3.5$	0.7054	0.7124	0.7138	0.7141	0.7142	0.7142	0.7142
$1/3$	0.6634	0.6652	0.6664	0.6665	0.6666	0.6666	0.6667
$1/2$	0.4946	0.4993	0.4999	0.4999	0.4999	0.4999	0.5000
$2/3$	0.3301	0.3330	0.3333	0.3333	0.3333	0.3333	0.3333
$3/4$	0.2484	0.2498	0.2499	0.2499	0.2499	0.2499	0.2500
$9/10$	0.0991	0.0999	0.0999	0.0999	0.0999	0.0999	0.1000

TABLE 5.13: $\bar{\varepsilon}_{ML}$ of GLDPC-Staircase codes as a function of r_G

results.
Since r_G=$\{\frac{1}{2}, \frac{1}{3}\}$ are commonly in our use-cases, therefore we choose to set the base code rate to $r_L = 2/3$.

5.5 Conclusions

In this Chapter, through an asymptotic and finite length analysis, we showed that obtaining the generated LDPC repair symbol as RS repair symbol on each generalized

check nodes avoids the stuck of the (IT+RS) decoding. However, this structure had no a great impact on the linear system which used in the ML decoding. This means that scheme A outperforms scheme B under (IT+RS) decoding and both achieve almost the same ML performance. Thus to design GLDPC-Staircase codes under hybrid (IT/RS/ML) decoding we must choose scheme A.
Then, in order to tune and optimize the configuration of these codes, important internal parameters such as the extra-repair distribution, the base code rate and the source variable degree denoted by N1, have been investigated and fixed. Based both on asymptotic and finite length analysis we concluded that:

- The regular distribution of extra-repair symbols per generalized check nodes has been chosen for the hybrid (IT/RS/ML) decoding.
- For a fixed GLDPC-Staircase code rate, the increase of the base code rate (i.e, increase the number of extra-repair symbols per generalized check nodes) induces the quick approaching of decoding overhead to a zero value, even for very small code dimensions. Moreover, it causes the increase of the upper bound on the ML decoding threshold until it reaches a stable value (no significant improvement beyond this value) quite close to the Shannon limit. This stable position is achieved with a small number of extra-repair symbols per generalized check node. Therefore small number is sufficient to get extremely quite close to the channel capacity.
- With ML decoding, the most significant performance gains are obtained by switching from N1 = 3 to 5 because the improvement of the correction capabilities is related to the densification of the resolved system. Above this value the performance improves only slightly (no great significant impact of N1). Since the improvement of the correction capabilities is related to the sparseness of the used graph, N1= 5 also limits the performance degradation of (IT+RS) decoding compared to values N1 $>$ 5. Additionally, the increase of N1 causes the increases of the decoding complexity in both decoding kinds. Therefore, we considered the value N1=5 as the best compromise between correction capabilities of ML decoding, correction capabilities of (IT+RS) decoding, and the decoding complexity. We also chose this value for the hybrid (IT/RS/ML) decoding.

Part II

Performance evaluation in different use cases

Chapter 6

Performance evaluation of GLDPC-Staircase codes over memory-less channel

Contents

6.1 Introduction 109

6.2 Achieved performance 110

6.2.1 (IT+RS) versus Hybrid (IT/RS/ML) decoding 110

6.2.2 Hybrid (IT/RS/ML) decoding inefficiency ratio results 110

6.2.3 Hybrid (IT/RS/ML) decoding failure probability results 111

6.3 Comparison with other erasures correcting codes 113

6.3.1 LDPC-Staircase codes 113

6.3.2 GLDPC codes designed in [5] 114

6.3.3 Raptor codes 115

6.4 Decoding complexity 116

6.4.1 Experimental conditions 116

6.4.2 Results 116

6.5 Conclusions 118

6.1 Introduction

In Chapter 5, we tuned the configuration of GLDPC-Staircase codes. We concluded that scheme A, N1=5, and small number of extra-repair symbols distributed regularly on generalized check nodes (*we chose* $r_L = \frac{2}{3}$ *for* $r_G = \{\frac{1}{2}$*(E=1) and* $\frac{1}{3}$*(E=3)}*) were the most appropriate values for hybrid (IT/RS/ML) decoding. Therefore, in this Chapter, let us investigate with more details the performance achieved by these codes in various situations. We start by analyzing the average decoding inefficiency ratio, the decoding

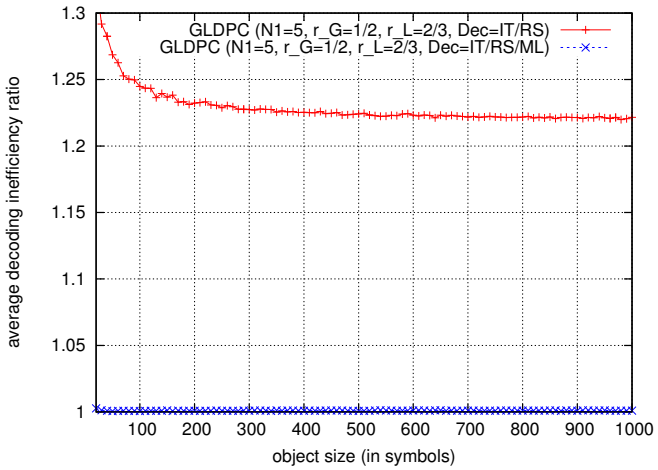


FIGURE 6.1: Performance for (IT+RS) versus hybrid (IT/RS/ML) decoding schemes, with r_G=$\frac{1}{2}$.

failure probability, and the achieved decoding overhead of the hybrid decoder. Then, we give a comparison between these codes and other erasure correcting codes. Finally, we show the hybrid decoding complexity.
In this study we use an asymptotic analysis as well as a finite length analysis using the same conditions mentioned in section 5.2.

6.2 Achieved performance

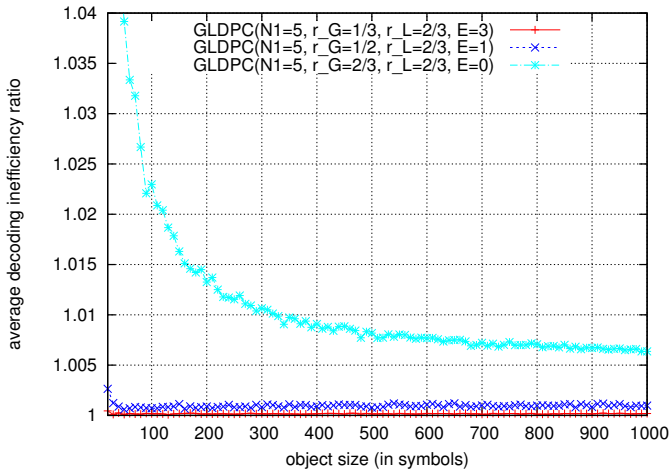
6.2.1 (IT+RS) versus Hybrid (IT/RS/ML) decoding

Let us quantify the erasure recovery performance gains made possible by the use of ML decoding. Figure 6.1 shows the average decoding inefficiency ratio (i.e, average of 1000 GLDPC codes with $r_G = \frac{1}{2}$) versus various object sizes. This figure confirms the major performance gains made possible by the use of ML decoding with GLDPC-Staircase codes for all object sizes (e.g, gain equal to 22% for K=1000).
More remarkable, the performance are excellent with hybrid (IT/RS/ML) decoding decoding even for very small code dimensions.

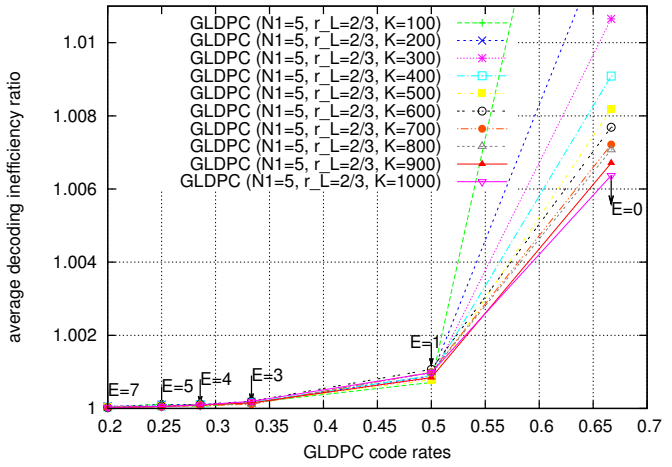
6.2.2 Hybrid (IT/RS/ML) decoding inefficiency ratio results

In Figure 6.2(a), we plot the average hybrid (IT/RS/ML) decoding inefficiency ratio (i.e, average of 1000 GLDPC codes), as a function of the object size, with various curves corresponding to the various code rates (i.e, r_G=$\frac{1}{3}$ ($E = 3$), r_G=$\frac{1}{2}$ ($E = 1$), and r_G=$\frac{2}{3}$ ($E = 0$)), and we do the opposite in Figure 6.2(b).

We see in both figures that the GLDPC-Staircase codes with $E = 1$ or $E = 3$ exhibit exceptional erasure recovery capabilities, even for tiny objects. On the opposite, codes with $E = 0$ corresponding to the base codes have performance level significantly smaller for small object sizes. Hence, the addition of extra-repair symbols makes the correction capabilities of GLDPC-Staircase codes under hybrid decoding close to that of ideal, MDS codes, both for tiny or large objects.



(a)



(b)

FIGURE 6.2: Average decoding inefficiency ratio as a function of the object sizes (a) and code rates (b).

These tests also show that GLDPC-Staircase codes can easily achieve very small code rates while keeping exceptionally high erasure recovery performance.

6.2.3 Hybrid (IT/RS/ML) decoding failure probability results

We continue the analysis with decoding failure probability, which enables us to more carefully analyze the GLDPC-Staircase codes behavior as a function of the number of received symbols and loss percentage. Figure 6.3 shows the average results of 10^7 GLDPC codes with $r_G = \frac{1}{2}$ for $K = 1000$, $K = 256$, and $K = 32$. The black vertical line corresponds to ideal, MDS code, for which the decoding failure is equal to 0 as long as the experienced loss rate is strictly inferior to 50%. This figure confirms that GLDPC-Staircase codes are close to ideal, MDS codes, with no visible error floor above 10^{-5} decoding failure

probability. This is obvious with objects of size $K = 1000$ and $K = 256$, it remains almost true with the $K = 32$ case (i.e, error floor started from 8.10^{-6} with 42% of loss for K=32, bellow $5.33.10^{-6}$ with 49.45% of loss for K=1000). Table 6.1 gives additional

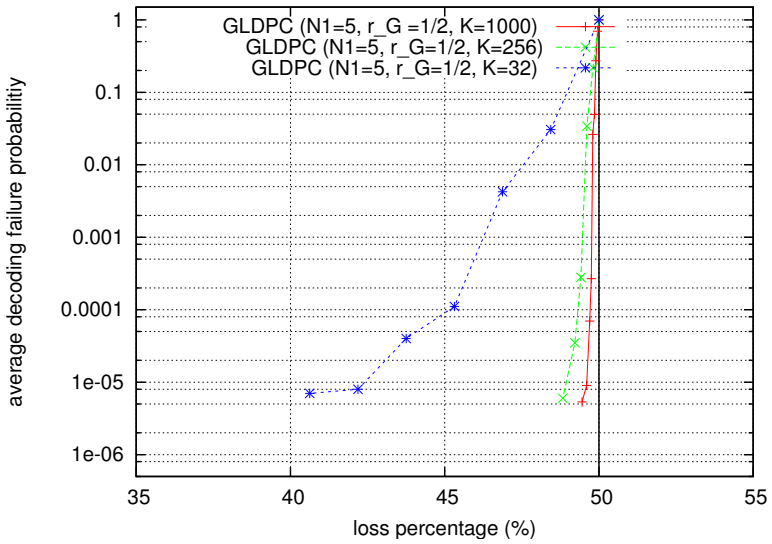


FIGURE 6.3: Average hybrid (IT/RS/ML) decoding failure probability as a function of the channel loss percentage experienced for GLDPC codes ($r_G = \frac{1}{2}$) with $K = 32$, $K = 256$, and $K = 1000$.

details where we tested 6.10^6 and 2.10^6 GLDPC codes ($r_G = \frac{1}{2}$) for K=1000 and K=32 respectively. It provides the average decoding failure probability as a function of the overhead (the number of received symbols above K).
It shows that, for large or small object sizes, a few symbols above K is sufficient for decoding to succeed with a high probability. For instance, with K=1000 received symbols, 1819473 among 6.10^6 GLDPC codes can recover all the erased symbols (i.e, 69.67 % as decoding failure probability) and with two received symbols more than K, 5703206 among 6.10^6 GLDPC codes can decode all the erased symbols (i.e, 4.9 % as decoding failure probability). With $K = 32$ received symbols, 1938832 among 2.10^6 GLDPC codes can recover all the erased symbols (i.e, with decoding failure probability equals to 3.0584%) and with two received symbols more than K, 1999778 among 2.10^6 GLDPC codes can recover all the erased symbols (i.e, 0.011% as decoding failure probability).
The previous different results prove that GLDPC-Staircase codes have excellent correction capabilities, very close to MDS codes, over erasure channel. This led us to make a comparison, in next section, with other erasure correcting codes such as LDPC-Staircase codes, Raptor codes (RFC5053 [123]) and other constructions of GLDPC codes that have recently been proposed in [5].

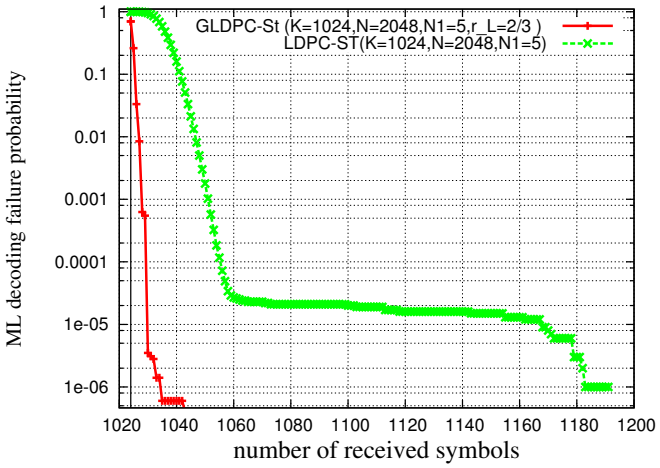
K	reception overhead	average decoding failure prob.
32	0	0.0305
	1	$4.2 * 10^{-3}$
	2	$1.1 * 10^{-4}$
	3	$4 * 10^{-5}$
	4	$8 * 10^{-6}$
	5	$7 * 10^{-6}$
	6	$2 * 10^{-6}$
256	0	0.22
	1	0.0351
	2	$1.18 * 10^{-3}$
	3	$7.39 * 10^{-4}$
	4	$4.97 * 10^{-4}$
	5	$3.35 * 10^{-4}$
	6	$1.37 * 10^{-4}$
1000	0	0.6967
	1	0.2725
	2	0.0494
	3	0.0262
	4	$2.68 * 10^{-4}$
	5	$6.96 * 10^{-5}$
	6	$9 * 10^{-6}$

TABLE 6.1: Average decoding failure probability of GLDPC-Staircase codes ($r_G = \frac{1}{2}$) under ML decoding for K=32 and K=1000

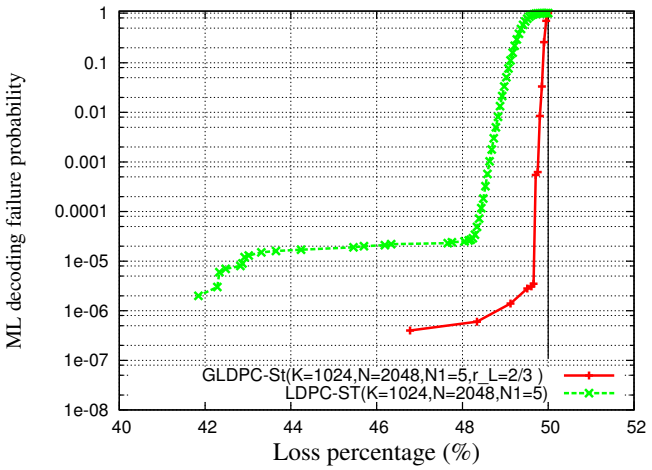
6.3 Comparison with other erasures correcting codes

6.3.1 LDPC-Staircase codes

Let us start by comparing the corrections capabilities of GLDPC-Staircase codes with those of LDPC-Staircase codes using the decoding failure probability metric (c.f A.1.1.2). We plot ML decoding failure probability versus loss channel percentage in Figure 6.4(b) and versus the number of received symbols in Figure 6.4(a) for the LDPC-Staircase codes and GLDPC-Staircase codes with rate equals to $\frac{1}{2}$.
Figure 6.4(b) shows that the GLDPC-Staircase codes are close to ideal with a very steep slope in the “waterfall" area compared to LDPC-Staircase codes. In addition, no error floor appears above 10^{-6} decoding failure probability, which is a very good performance, whereas for LDPC-Staircase codes the error floor started from 2.10^{-5} decoding failure probability.
In Figure 6.4(a) we observe that, for GLDPC-Staircase codes, the reception of a few additional symbols, in addition to K symbols, allows to reach quickly the decoding failure probability below 10^{-6}. To obtain ML decoding failure probability equals to 10^{-4}, it must receive, in addition to $K = 1024$, 28 symbols with LDPC-Staircase codes, against 4



(a)



(b)

FIGURE 6.4: LDPC Staircase codes versus GLDPC-Staircase codes for rate=$\frac{1}{2}$ and N1=5.

symbols with GLDPC-Staircase codes, a difference of twenty four symbols.

Therefore, GLDPC-Staircase codes provided a gain with low error floor and a very steep slope compared to LDPC-Staircase codes.

6.3.2 GLDPC codes designed in [5]

In this section, we compare the correction capabilities of our GLDPC code construction with another GLDPC code construction as defined by Tanner [36]. This GLDPC code construction is characterized by an optimal distribution (capacity-approaching) based on hybrid check nodes structure, which is composed of SPC and (31,21) linear block codes (BCH codes with $d_{min} = 5$) [5].
For our codes we use the selected configuration for code rate $\frac{1}{2}$ such as a regular distribution of extra-repair symbols and a base code with $N1 = 5$, $r_L = \frac{2}{3}$ and DD as shown in Table 5.1).

Code type	Decoding threshold
GLDPC-Staircase, Hankel-RS	0.4993
GLDPC, BCH	0.49671

TABLE 6.2: Comparison decoding threshold between GLDPC-Staircase codes and other GLDPC construction code for rate $\frac{1}{2}$.

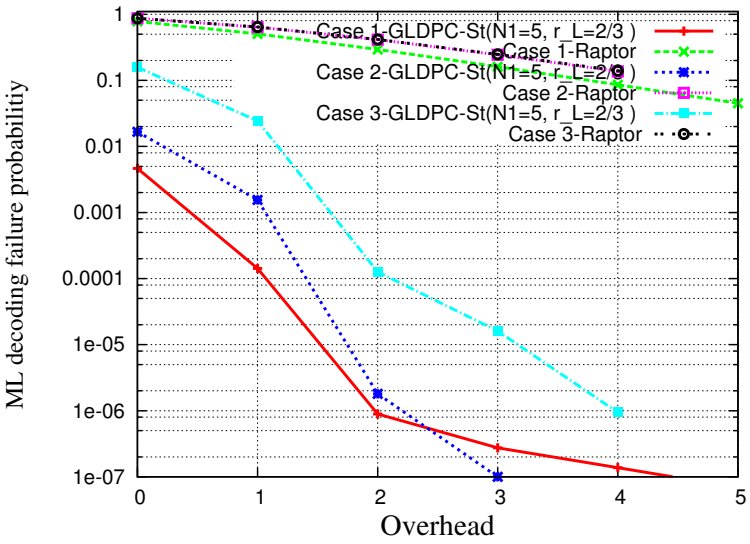


FIGURE 6.5: ML decoding failure probability versus overhead for GLDPC and Raptor using three cases.

Table 6.2 provides a comparison in terms of decoding threshold for rate $\frac{1}{2}$. This table shows that our construction method for GLDPC codes performs the best.

6.3.3 Raptor codes

Finally, we compare the achieved performance of GLDPC-Staircase codes with Raptor codes (RFC5053 [123]) in three cases:

- Case 1: K=32, N=128,
- Case 2: K=256, N=1024,
- Case 3: K=1024, N=3072.

In Figure 6.5, we plot the ML decoding failure probability versus decoding overhead for those three cases. For Raptor codes, we used the results provided by Qualcomm for 3GPP in [6]. This figure shows that GLDPC-Staircase codes outperform Raptor codes, no matter the object size. In fact, for 1/3-rate (Case 3), GLDPC-Staircase codes with 2 symbols more than K can achieve a decoding failure probability equal to $1,2.10^{-4}$ whereas Raptor codes feature a decoding failure probability equal to $0,4156782$.

Using the results that are given in section 2.5.3, the correction capabilities of our codes are close to those of RaptorQ codes (e.g., with case 3, the 10^{-7} of decoding failure probability is achieved with 2 symbols (overhead) for RaptorQ codes whereas our codes require 4 symbols).

6.4 Decoding complexity

We'll look at the decoding algorithmic complexity of GLDPC-Staircase codes decoded by an hybrid (IT/RS/ML) decoder. We note that in this section we give a preliminary study on the decoding complexity of these codes.

6.4.1 Experimental conditions

The decoding complexity of the hybrid (IT/RS/ML) decoding is performed using the method specified in the library "*Openfec.org*" based on the throughput decoding metric (see in Appendix A section A.1.3.2). We used a C language binary LDPC-Staircase software codec and an Hankel-RS software codec over $GF(2^8)$. We conducted tests on a GNU/Linux system, using kernel 2.6.27.41/64bits with Intel(R) Xeon(R) CPU E5410@2.33GHz processor.
Following the recommendations of Chapter 5, we use the parameters N1 = 5, $r_L = 2/3$ and regular distribution of extra-repair symbols. We consider an object of 1000 symbols with size equals to 1024 bytes. We use GF(2^8) in all our results.

6.4.2 Results

As mentioned in section 4.2.3.2, this decoding starts with the (IT+RS) decoder and then triggers the ML (binary and/or non-binary) decoder when the former decoder fails. In Figure 6.6, we plot the throughput decoding for global rate $\frac{1}{2}$ in function of loss channel percentage. This Figure shows what we said above. We see that when loss channel percentage is low (until 35%), in most cases, the (IT+RS) decoding is sufficient to recover the erased source symbols with speed in the order of 700 Mb/s. Then, as the channel loss percentage approaches the theoretical limit (50%), the speed decreases until it stabilizes around 50 Mb/s. This is due to the frequent use of ML decoding. We notice that (IT+RS) decoding is fast, but ML decoding remains costly. Therefore, optimizing this aspect is a key requirement.

Proposed optimization

In the ML decoding step, recovering source and repair LDPC symbols using binary ML decoding requires only XOR operations. However, the use of the extra-repair symbols adds finite field operations. This operation consists of multiplying symbols with associated coefficients over $GF(2^8)$ and then adding symbols. Finite field addition consists in XORing the two values. Finite field multiplications are more complex, requiring in general a log table lookup, an addition operation and an exponentiation table lookup to determine the result. With $GF(2^8)$, multiplications can be pre-calculated and the result is stored in a table of size 255×255. This is a common optimization with software codecs. With this optimization, multiplying two elements of $GF(2^8)$ consists in accessing the right element

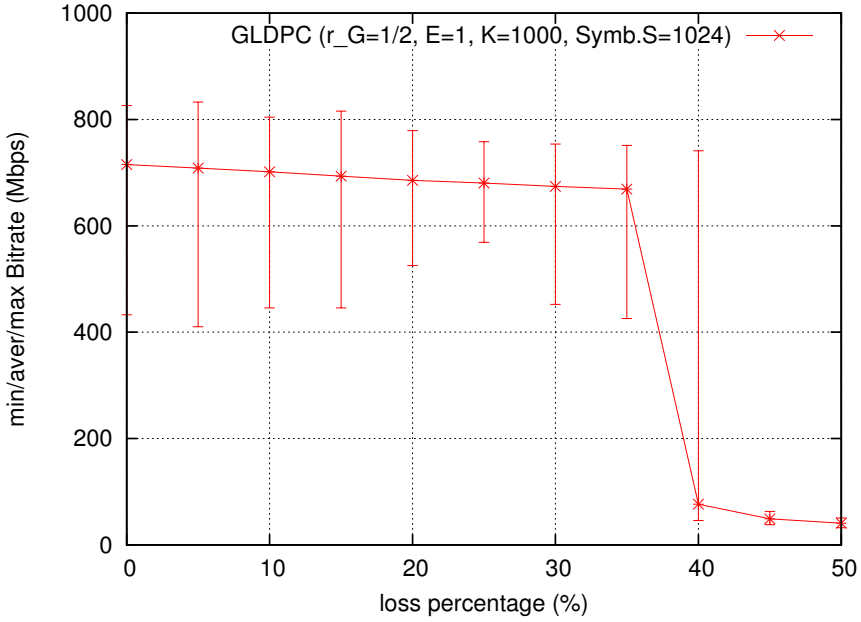


FIGURE 6.6: Throughput of hybrid (IT/RS/ML) decoding of GLDPC-Staircase codes with $r_G = \frac{1}{2}$, E=1, N1=5, Symbol size = 1024 bytes

Algorithm 4 Optimized algorithm

```
(IT+RS) decoding fails over H_L and it results H_s
if B-ML decoding fails on H_s and it results H_m then
    if NB-ML decoding on H_t can done then
        Step1: Start with NB-ML decoding on H_t until reach the col_jump
        Step2: Trigger the B-ML decoding on H_m from col_jump
        Status OK
    else
        Status Failure
    end if
else
    Status OK
end if
```

of this pre-calculated table. Therefore, the operation of multiplication and then addition becomes an operation of accessing appropriate table and then addition.
In order to reduce the impact of this operation on the decoding complexity, we propose an optimization in ML decoding step. This optimization consists of reducing the number of symbols decoded on the finite field in the non binary ML (NB-ML) decoding step of algorithm 3 when the (IT+RS) decoding and binary ML (B-ML) decoding fail. Algorithm 4 summarizes the optimization. We can more explain this algorithm as follows. After the failure of (IT+RS) decoding, we obtain a simplified matrix denoted by H_s. Let's denote H_t the global matrix used to perform the NB-ML decoding. H_m represents the resultant matrix after the failure of the B-ML decoding step on H_s. The B-ML decoding fails if H_s

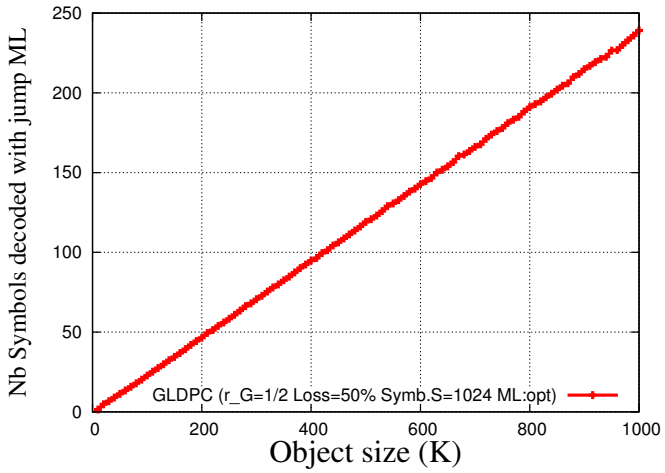


FIGURE 6.7: Number of source symbols decoded with jump from NB-ML decoding to B-ML decoding for GLDPC Staircase codes

has a "quasi triangulation" form. Therefore, the steps of the triangulation method stop at a column denoted by *col_jump*.
The proposed optimization takes advantage of the form of H_m. In other manner, we jump in the stage 2 of the NB-ML decoding (i.e, Backward substitution) from H_t to H_m when we achieve the column *col_jump* to complete the decoding of the erased source symbols with binary way.
In Figure 6.7, we plot, for 50% of channel loss percentage, the number of source symbols (averaged value on 1000 tests) that have been decoded by jumping from H_t to H_m versus object size for r_G=$\frac{1}{2}$ and E=1. This Figure shows that for instance for $K = 1000$ we have 239.0686 source symbols decoded with B-ML decoding instead of NB-ML decoding.
Looking at the throughput of the hybrid (IT/RS/ML) decoding, in Figure 6.8 we plot the throughput for rate $\frac{1}{2}$ in function of channel loss percentage for three object sizes. In this Figure, from 30% of channel loss percentage, the ML decoder is commonly used. Using the optimization mentioned later, we can see in Figure 6.8 a little improvement (not very significant) for different object sizes.

6.5 Conclusions

We evaluated the performance gains made possible by the use of the ML decoding type. We showed that these codes exhibit exceptional erasure recovery capabilities, even for tiny objects, on the opposite of the LDPC staircase codes, which have performance level significantly smaller for small object sizes. Hence, the addition of extra-repair symbols makes the correction capabilities of GLDPC-Staircase codes under ML decoding close to that of ideal codes no matter the object size. For most cases, for large or small object sizes, a few symbols above K is sufficient for success decoding with a high probability.
These codes outperform the Raptor codes for different rates. For example, for 1/3-rate, GLDPC-Staircase codes with 2 symbols more than K can achieve a decoding failure probability almost equals to 10^{-4} whereas raptor codes presents 0, 4156782 as decoding

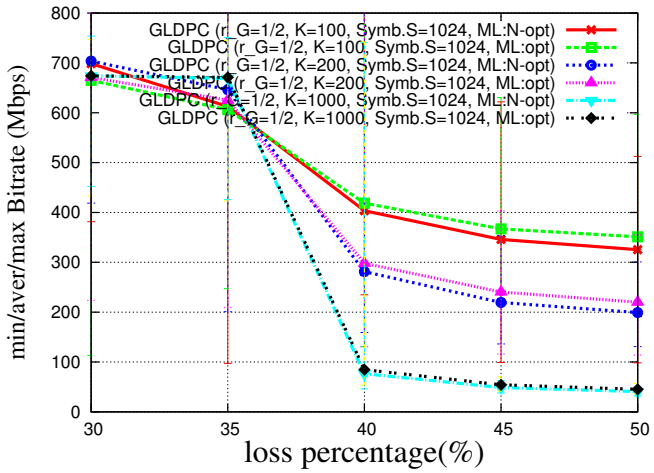


FIGURE 6.8: Throughput of GLDPC Staircase codes with rate $\frac{1}{2}$ and $E = 1$

failure probability. They have also correction capabilities close to that of RaptorQ codes. Moreover, by doing a comparison with other construction of GLDPC codes, GLDPC-Staircase codes give the highest decoding threshold.

Chapter 7

Impacts of the packet scheduling on the performance for broadcast/multicast services

Contents

7.1 Introduction 121
7.2 Preliminaries 122
7.3 Proposed methodology 123
7.3.1 Channel Gilbert model 123
7.3.2 Evaluation methodology 124
7.4 Transmission scheduling considered 126
7.5 Simulation results and discussion 127
7.5.1 Simulation Setup 127
7.5.2 Results W.R.T. the $\{p,q\}$ parameters 128
7.5.3 Results W.R.T. the $\{PLR,ABL\}$ parameters 130
7.6 Conclusions 131

7.1 Introduction

Content broadcasting over wireless networks heavily relies on AL-FEC codes to improve transmission robustness in front of channel impairments. Because they operate in the higher layers of the protocol stack, on an erasure channel, they benefit from a lot of flexibility. In particular, since streaming applications and bulk transfer applications have different constraints, it justifies that different packet schedules be used.
Some of the AL-FEC codes naturally do not depend on the packet scheduling. This is the case of ideal, MDS codes, like RS codes, that perform identically in terms of erasure recovery performance no matter the subset of packets received, as long as a minimum number of k (code dimension) packets among n (code length) encoding packets are available. This

is also the case of codes that by construction do not show any dependency over the set of packets received, because each encoding packet is produced in the same manner from a set of intermediary symbols: Raptor [123] and RaptorQ [125] have this property (called "strongly systematic" by the authors).
Non systematic AL-FEC codes (or codes used in a non-systematic manner) can have a low dependency on the packet scheduling because the question of source versus repair packets available at a receiver becomes meaningless. On the opposite, certain codes creates strong relationships between repair packets: e.g. with LDPC-Staircase codes (more generally repeat-accumulate codes), each repair packet is the sum of the previous repair packet with some additional source packets. In that case the subset of packets received does potentially impact the code performance.
Additionally, the type of AL-FEC decoding performed: on the fly IT decoding, as packets arrive, does not behave the same as ML decoding. In practice, there is a clear incentive in using IT decoding as long as possible, since this is a very low complexity decoder. ML decoding is on the opposite significantly more costly, even if the use of SGE, carefully implemented, can significantly help reducing the gap between IT and ML performance [129]. In this work we therefore consider both decoders, knowing that a practical codec will probably start with IT decoding and switch to ML decoding if all the packets have been received whereas the IT decoder failed to finish decoding. A good packet scheduling for a given AL-FEC code must therefore perform well both under IT and ML decoding.
Hence the new question: what are the impacts of the packet scheduling on GLDPC-Staircase AL-FEC performance?

In this chapter we address the problem of packet scheduling and distribution loss on the GLDPC-Staircase codes. To do that, first we introduce a methodology to measure the impacts of packet scheduling on AL-FEC codes, both under IT and ML decoding, for a large set of channels (with burst or not). Then, we apply this methodology on GLDPC-Staircase codes to determine the best packet scheduling for broadcast/multicast services.

7.2 Preliminaries

Figure 7.1 illustrates the problem: we see the sender that encodes the source object and chooses a packet scheduling. This packet flow is subject to erasures while traveling to a receiver (several thousands of such receivers can exist). This latter collects a subset of the encoding packets sent and (hopefully) recover the original object after FEC decoding. However, there are two key questions:

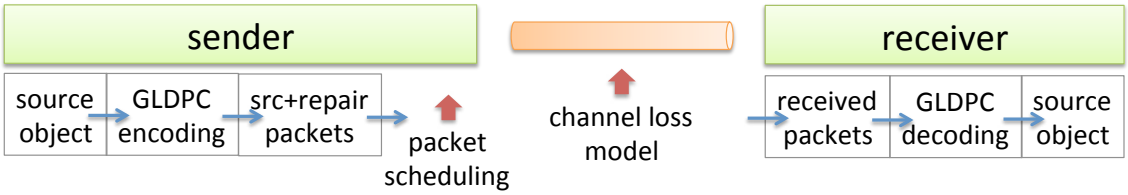


FIGURE 7.1: The various steps considered in the transmission chain.

1. How is the performance of a given type of AL-FEC codes impacted by the packet transmission schedule, i.e. the order in which packets are sent over the channel ?

2. How is it impacted by the packet loss distribution observed by a receiver (e.g., with a systematic code, it directly impacts the repartition between source and repair received packets)? For instance, does the channel seen by a particular receiver behave as a memory-less channel, with erasures uniformly spread, or does it feature erasure bursts of variable length ?

Since the use-case considers broadcast/multicast services, two different receivers can observe completely different channels: e.g., a receiver may move rapidly, which significantly impacts the short/long term fading properties of its channel, whereas the other one may be fixed with line-of-sight conditions.
It is therefore essential that the AL-FEC performance analysis considers a large set of channels since the packet scheduling is necessarily fixed, whereas there are as many channels as receivers.

Next we detail the methodology that we applied to carry out experiments.

7.3 Proposed methodology

We first discuss the well known Gilbert model, then we explain the proposed methodology.

7.3.1 Channel Gilbert model

For a given channel (packet loss distribution), finding an appropriate error model is a complex task especially with wireless networks where it is difficult to take into account all parameters (e.g. channel fading, reflections, refractions, diffractions, Doppler effects). Yet, in this thesis, we are only interested in a packet loss model (rather than a bit error model) that provides a high level of abstraction of the channel parameters.
To describe the long-term network packet loss (i.e. temporal memory and correlation in packet losses process for communication networks), Markov model is defined as the most suited model. The work in [9] proves that the finite-state Markov chain is more accurate than the Bernoulli model to characterize the bursty packet losses. Many other works such as [137, 138] have shown that Markov models are good approximations of the actual packet loss processes for wireless channels. Gilbert first proposed a two-state Markov model in his studies to characterize the bursty losses. It is a simplified loss model that it is widely used in the literature [8, 9, 139, 140].
We have seen in Section 7.2 that we need to test a given packet scheduling across a very large set of channels, since the packet scheduling is necessarily fixed (chosen by the sender) whereas there are as many channels as receivers, with features that will also dynamically vary. In order to consider a wide variety of channels in our subsequent analysis, we also use a two-state Markov model (called Gilbert model) [141]. In this model, the channel switches between the *error state* and the *error-free state*. When the channel is in the error state, the transmitted packet is always lost, while in the error-free state the packet is always

correctly received. Let state 1 and state 2 respectively denote the error-free and the error states.
As shown in Figure 7.2, the parameter p is the transition probability from state 1 to state 2, and q denotes the probability of the opposite transition. These two parameters are independent each other with $p+q \leq 1$. From the definition, the stationary probability for state 1 and 2, denoted by π_1 and π_2, can be computed as follows [8]:

$$\pi_1 = \frac{q}{p+q} \quad \text{and} \quad \pi_2 = \frac{p}{p+q}. \tag{7.1}$$

In the literature two alternative parameters are often used to characterize the gilbert model: the PLR and ABL. The relationship between these parameters are [142]:

$$PLR = \pi_1 = \frac{p}{p+q} \quad \text{and} \quad ABL = \frac{1}{q}. \tag{7.2}$$

Thus, given PLR and ABL, the gilbert model is determined. Increasing ABL while keeping PLR constant implies the decrease of q and p parameters proportionally. The gilbert model

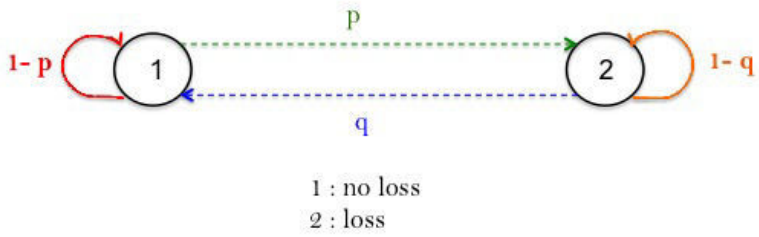


FIGURE 7.2: Two states Markov loss model.

also covers some specific loss behaviors that we want to emphasize:

- No loss: this perfect channel corresponds to $p = 0$.
- Independent and Identically Distributed (IID) losses: this memory-less channel corresponds to $q = 1 - p$. It is also known as a Bernoulli model.

7.3.2 Evaluation methodology

We are aware that the gilbert model has some shortcomings in error modeling accuracy [143][9]. We believe that it is however sufficient for our work since it already covers a very large set of loss behaviors (different values of loss pattern (p, q)). This means that in order to test a given packet scheduling with a large number of channels, we vary the $\{p,q\}$ parameters. More precisely, for each transmission scheduling scheme, we consider 21 values between 0 and 1 for either p or q values (i.e. every 0.05)[1]. This results in a 21×21 grid, i.e. 441 different channels. Since simulations follow a stochastic approach, for each of these channels (i.e. $\{p,q\}$ pair), we perform 1000 tests, for a total of 441000

[1] Of course a smaller granularity can be chosen for more detailed results.

tests. Some of these tests are successful, i.e. enable a receiver to decode and recover the original object, while others fail to recover the object.

The best packet scheduling for a given AL-FEC code is therefore the one that features the highest total number of successful tests over all the channels considered.

As explained above, the Gilbert model can also be totally determined by the $\{PLR, ABL\}$ parameters. Therefore it is possible to carry out the same tests by varying the PLR and ABL parameters between some predefined bounds and to proceed in the same way, by counting the number of successful decodings for each of the 1000 tests carried out for a given channel. The results are equivalent, but with a different presentation that is perhaps easier to interpret (the $\{PLR, ABL\}$ parameters are more intuitive) than the $\{p, q\}$ parameters. In our case we found easier to perform all the simulations by working on the $\{p, q\}$ parameters, storing all the results in an SQL database, and doing the appropriate extractions to compute the $\{PLR, ABL\}$ results (see Sections 7.5.2 and 7.5.3).

About impossible decoding areas in the (p, q) grid

An (n,k) AL-FEC code produces n-k parity packets from k source packets. With the gilbert model we can calculate the maximum number of packet losses supported by any AL-FEC code. We then transmit n packets over the network and the number of packets actually received equals to:

$$n_{received} = n * (1 - PLR) \tag{7.3}$$

Where $n_{received}$ must be at least equal to the necessary needed symbols n_{ne_succ} to success the decoding. When n_{ne_succ} = k (i.e, case of ideal codes) and using equations (7.2) and (7.3), we have:

$$q_{th} = \frac{p}{\frac{1}{CR} - 1}, \tag{7.4}$$

q_{th} represents the lowest value of q where the decoding is possible ($n_{received} \geq$ k).
This equation defines the limit of impossible decoding area for an AL-FEC code rate CR. We show in Figure 7.3 the impossible decoding areas for CR=($\frac{1}{2}, \frac{1}{3}, \frac{2}{3}$).

In our work, there are areas in the 21×21 grid where decoding is, in theory, not feasible as the number of packets received ($n_{received}$) is less than k, the code dimension. However, since we do stochastic simulations, using $\{p, q\}$ as transition probabilities, the actual number of packets received is not necessarily equal exactly to $n_{received}$.

About decoding overhead based evaluation methods

Our erasure recovery performance evaluation is not based on decoding overhead evaluation techniques, i.e. on the determination of the minimum number of packets that need to be received in excess to k for success decoding. This is deliberate. In our methodology we do not really care whether decoding was possible with a low overhead or not, especially as this overhead can differ according to the channel features[2].

[2] E.g. we identified a packet scheduling where the overhead was significantly better for low to medium *PLR* conditions, but significantly increases when approaching the theoretical limit. At the end, when

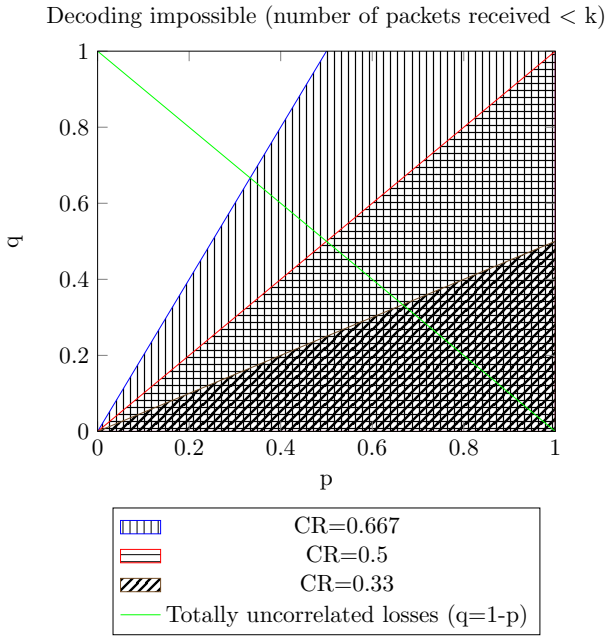


FIGURE 7.3: Impossible decoding area for $CR = (1/2, 1/3, 2/3)$

7.4 Transmission scheduling considered

To evaluate the behavior of GLDPC-Staircase codes as a function of the packet scheduling for different channels, we identify 22 transmission modes, classified into four categories:

- Category (1) where all the packets sent randomly. This is the reference scheduling;
- Category (2) where source packets are sent first, *sequentially*, followed by all repair packets using different orderings. This category is important to video streaming applications since it minimizes the transmission delays (there is no need to buffer all the source packets before beginning the transmissions as is the case with the other categories);
- Category (3) where source packets are sent first *randomly*, followed by all repair packets using different orderings;
- Category (4) where source and repair packets are sent randomly in blocks (i.e, source, LDPC/RS repair, and extra-repair packets);

Table 7.1 details the 22 schedulings that we considered. We believe that these schedulings

considering the total number of successful tests, this promising scheduling was significantly worse than several other schedulings.

Category (1)	**all packets are sent randomly. This is the reference scheduling**
T-Mode-0	Randomly send all source and repair packets (fully random order)
Category (2)	**source packets are sent first *sequentially*, followed by repair packets in a certain order**
Tx-Mode-1	LDPC/RS repair first sequentially and then extra-repair sequentially
Tx-Mode-2	Fully random order of all repair packets
Tx-Mode-3	Extra-repair first sequentially and then LDPC/RS repair sequentially
Tx-Mode-4	LDPC/RS repair first randomly and then extra-repair sequentially
Tx-Mode-5	LDPC/RS repair first sequentially and then extra-repair randomly
Tx-Mode-6	Extra-repair first randomly and then LDPC/RS repair sequentially
Tx-Mode-7	Extra-repair first sequentially and then LDPC/RS repair randomly
Tx-Mode-8	LDPC/RS repair first randomly and then extra-repair randomly
Tx-Mode-9	Extra-repair first randomly and then LDPC/RS repair randomly
Category (3)	**source packets are sent first *randomly*, followed by repair packets in a certain order**
Tx-Mode-10	Fully random order of all repair packets
Tx-Mode-11	LDPC/RS repair first randomly and then extra-repair randomly
Tx-Mode-12	Extra-repair first randomly and then LDPC/RS repair randomly
Tx-Mode-13	LDPC/RS repair first sequentially and then extra-repair sequentially
Tx-Mode-14	LDPC/RS repair first randomly and then extra-repair sequentially
Tx-Mode-15	LDPC/RS repair first sequentially and then extra-repair randomly
Tx-Mode-16	Extra-repair first sequentially and then LDPC/RS repair sequentially
Tx-Mode-17	Extra-repair first randomly and then LDPC/RS repair sequentially
Tx-Mode-18	Extra-repair first sequentially and then LDPC/RS repair randomly
Category (4)	**source and repair packets are sent randomly per block type (i.e, source, LDPC/RS repair, extra-repair)**
Tx-Mode-19	Randomly send extra-repair first, then randomly LDPC/RS repair, and then randomly source
Tx-Mode-20	Randomly send LDPC/RS repair first, then randomly extra-repair, and then randomly source
Tx-Mode-21	Randomly send all GLDPC repair first, then randomly source

TABLE 7.1: The 22 packet schedulings considered in this work.

cover a large set of possibilities and give good insights on the overall decoding performances, in particular those applicable to either file download and streaming applications, although the list is not exhaustive. We have no interest to the category of the transmitted packets scheduling, where the repair packets are sent sequentially before source packets, which involves more computing at the receiver to recover source packets.

7.5 Simulation results and discussion

7.5.1 Simulation Setup

Let us consider an object composed of K=1000 packets and GLDPC-Staircase codes, with a base code rate (i.e. associated to the LDPC-Staircase code) equals to $r_L = \frac{2}{3}$, and a global code rate which either equals to $r_G = \frac{1}{2}$ or $r_G = \frac{1}{3}$. Therefore, with $r_G = \frac{1}{2}$, we

have 500 LDPC/RS repair packets (i.e. produced by LDPC-Staircase or RS encoding) and 500 extra-repair packets (i.e. produced by RS encoding). With $r_G = \frac{1}{3}$, we still have 500 LDPC/RS repair packets but we now have 3 times more extra-repair packets, namely 1500 extra-repair packets.

In the following, we first provide general results obtained while varying the $\{p,q\}$ parameters, then we provide complementary results according to the $\{PLR, ABL\}$ parameters. It shows that if the two methods are equivalent in theory, they can highlight behaviors that would not be visible otherwise.

7.5.2 Results W.R.T. the $\{p,q\}$ parameters

Tables 7.2, 7.3, 7.4, 7.5, 7.6 and 7.7 provide the total number of successful tests (among a total of 441,000 tests), for which all the erased source packets can be recovered, as a function of the transmission scheduling, for code rates $r_G = \frac{1}{2}$ and $r_G = \frac{1}{3}$ and for (IT+RS) decoding and ML decoding techniques. Of course, the higher the number in the table is, the better in the sense that this packet scheduling enables FEC decoding to take place successfully for a higher number of different channels.

The case of (IT+RS) decoding From tables 7.2 and 7.3, we see that transmitting the LDPC/RS repair packets (randomly or sequentially) just after the source packets (randomly or sequentially), marked in gray, leads to very poor results in (IT+RS) decoding. By

Tx-Mode	1	2	3	4	5	6	7	8	9
r_G=1/3	87,673	224,636	231,169	86,933	87,671	238,336	235,796	86,952	241,383
r_G=1/2	80,140	135,920	142,678	86,837	87,642	142,997	143,654	86,854	147,726

TABLE 7.2: Total number of successful tests for scheduling category (2) under (IT+RS) decoding.

Tx-Mode	10	11	12	13	14	15	16	17	18
r_G=1/3	224,492	86,892	241,269	87,245	86,652	87,678	231,214	238,322	235,804
r_G=1/2	136,048	86,869	143,795	87,190	86,653	87,359	142,376	143,002	143,614

TABLE 7.3: Total number of successful tests for scheduling category (3) under (IT+RS) decoding.

Tx-Mode	0	19	20	21
r_G=1/3	237,238	241,875	241,851	241,966
r_G=1/2	142,037	143,883	143,816	143,858

TABLE 7.4: Total number of successful tests for scheduling categories (1) and (4) under (IT+RS) decoding.

contrast, sending the extra-repair packets (randomly or sequentially) just after the source packets (randomly or sequentially) is much more efficient with (IT+RS) decoding (because it allows to avoid several stopping sets). This is due to that the LDPC/RS repair packets are linked whereas the extra-repair packets are independent. We also note that sending extra-repair or LDPC/RS repair in a random order has a positive effect on decoding success with (IT+RS) decoding. Therefore, from these two tables, it appears that Tx-mode=9 and Tx-mode=12 are the best, with the additional benefit with Tx-mode=9 where source packets are sent in sequence, which avoids the extra delay required for the application to submit them all.

Let us now analyze the behavior of GLDPC-Staircase code where source packets are not sent at the beginning. In table 7.4, we determine the impact of the scheduling categories (1) and (4), where the packets are sent in random order. This table proves that sending the repair packets (totally randomly or randomly per block type) performs extremely well, in particular Tx-mode=21 where repair packets are sent first in random order, followed by source packets also in random order. However, these schedulings are not suitable to applications having strict latency constraints (e.g, streaming applications) because they require that the sender has all the source packets first before doing FEC encoding and before starting transmitting packets. Getting all these packets first can be an issue in term of delay if they are produced progressively.

We can conclude that in order to obtain good performance under (IT+RS) decoding while keeping delay to a minimum, we recommend using Tx-mode=9.

The case of ML decoding Let us now consider ML decoding. Tables 7.5, 7.6 and 7.7 show that under ML decoding the performance differences between the scheduling modes are less significant than for (IT+RS) decoding (the extremums are for Tx-mode=1 and Tx-mode=13 (i.e. a 3356 difference for rate $r_G = \frac{1}{3}$). However, it confirms that with ML

Tx-Mode	1	2	3	4	5	6	7	8	9
r_G=1/3	323,938	325,923	325,893	325,936	325,927	325,924	325,945	325,951	325,974
r_G=1/2	200,487	220,769	221,245	220,643	221,118	221,112	220,647	221,766	220,773

TABLE 7.5: Total number of successful tests for scheduling category (2) under ML decoding.

Tx-Mode	10	11	12	13	14	15	16	17	18
r_G=1/3	325,923	325,918	325,919	327,294	326,283	326,014	326,088	326,029	326,089
r_G=1/2	220,757	220,734	220,766	221,096	220,745	221,232	219,552	221,315	220,802

TABLE 7.6: Total number of successful tests for scheduling category (3) under ML decoding.

decoding too, the Tx-mode=9 approach performs extremely well (it is only 0.40% behind

Tx-Mode	0	19	20	21
r_G=1/3	325,905	325,918	325,919	325,917
r_G=1/2	220,788	220,773	220,737	220,725

TABLE 7.7: Total number of successful tests for scheduling categories (1) and (4) under ML decoding.

the optimum for $r_G = \frac{1}{3}$).

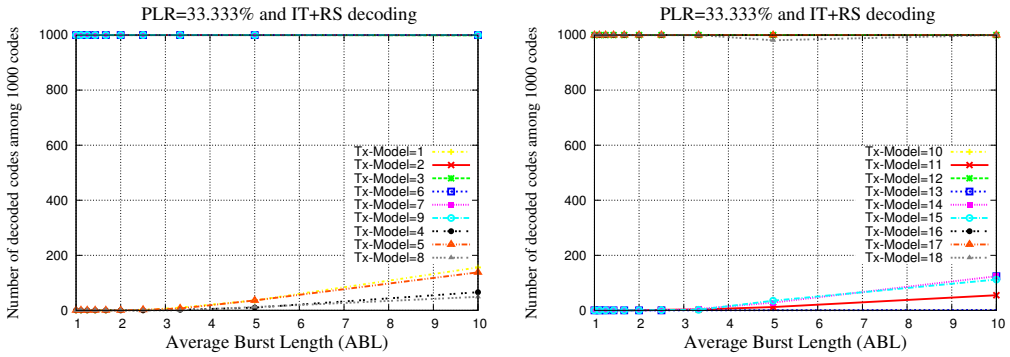
All things considered, it appears that Tx-mode=9 is the packet scheduling that offers the best compromise for both (IT+RS) and ML decoding.

7.5.3 Results W.R.T. the $\{PLR, ABL\}$ parameters

Let us now consider the PLR, ABL parameters. In order to assess the impacts of packet loss bursts on performance, we considered $ABL = \{1, 1.5, 2, \ldots, 20\}$ at $PLR = \{33.33\%, 50\%, 66.66\%\}$ for both (IT+RS) decoding and ML decoding[3].

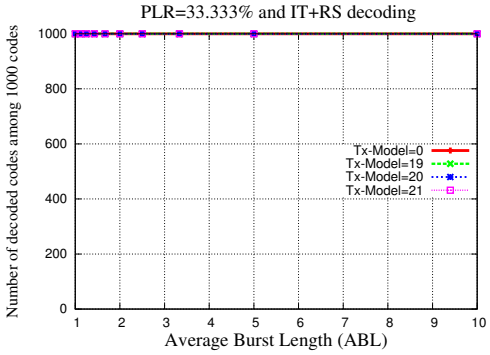
For (IT+RS) decoding only, Figure 7.4 and Figure 7.5 highlight major performance differences W.R.T. the decoding schedule at $PLR = 33.333\%$ and $PLR = 50\%$, whereas decoding becomes impossible at $PLR = 66.66\%$.
In the case of channels with low losses, Figure 7.4 shows that, for $PLR = 33.333\%$ and for different values of ABL, sending the extra-repair packets before LDPC/RS packets or sending all the packets randomly (the best modes) achieve the maximum number of success decoding. This is more than in modes where the LDPC/RS packets are sent in front the extra-repair packets (the worst modes). Despite the low value of number success decoding, these worst modes resist to the increase of ABL with superiority to the modes where packets are sent randomly.
Increasing PLR, Figure 7.5 provides the results for $PLR = 50\%$ case. We notice first that the worst modes mentioned previously are vanished. Therefore, with these modes, GLDPC-Staircase codes can decode only in low PLR. We can also see that categories (1) and (4) behave extremely well (it was also the case in Table 7.4). In addition, the increase of ABL causes a little decrease of the number of decoded codes. This figure also reveals that Tx-mode=9 performs reasonably well, but it is far from being the most efficient scheduling when the ABL increases: Tx-mode=3, 7 and 6 (16, 18 and 17 for source packets sent randomly) offer a better protection as the ABL increases, but they perform worse with very small values of ABL. One needs to be careful because the present curves are only a subset of the performance achieved when $PLR = 50\%$, whereas the results of Section 7.5.2 provide global results over the 441 channels considered in this study. Finally, at $PLR = 66.666\%$, the GLDPC-Staircase codes fail to decode in all modes.

[3] As explained before, tests are conducted only once for the 441 channels. Then we extract all the tests for which $PLR = p/(p+q) = 2/3$ exactly. Because of the way we iterate over (p,q), many such cases exist in the database. This is not necessarily the case for other values of PLR and running new tests could be needed then.



(a) Category (2)

(b) Category (3)



(c) Category (1) and (4)

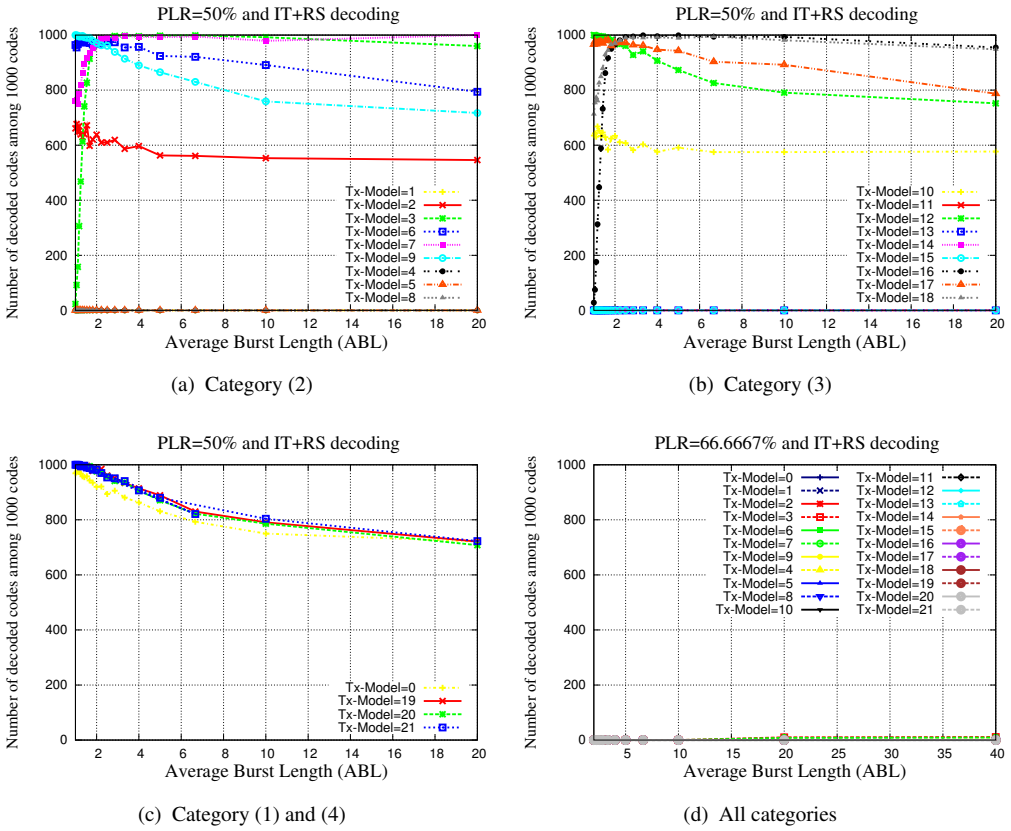
FIGURE 7.4: Total number of (IT+RS) successful decodings W.R.T. the ABL, with PLR =33.333%, when $r_G = \frac{1}{3}$.

Let us now consider ML decoding. Figure 7.6 shows that all the packet schedulings behave exactly the same for different values of ABL at $PLR = 33\%$ and 50%: no failure decoding. Figure 7.7 shows that they also behave similarly with $PLR = 66.66\%$: no impact of ABL until value of 20 and after that the increase of ABL causes the failure of decoding. This is perfectly in line with the results of Section 7.5.2.

Note that we are very close to the Shannon limit when $r_G = \frac{1}{3}$. The fact that decoding is still feasible with probability approx. 0.5 up to $ABL = 20$ packets is an excellent result that proves the high potential of GLDPC-Staircase codes.

7.6 Conclusions

In previous results we showed that these codes have erasure recovery performance quite close to ideal codes when packets are transmitted in a random order over memory-less erasure channels. Therefore, in this Chapter, we investigate the question of the performance of AL-FEC codes for the erasure channel as a function of packet scheduling at the sender. Since the type of channel (i.e. network conditions experienced by a given user) can have key impacts on performance, we considered a very broad range of channels



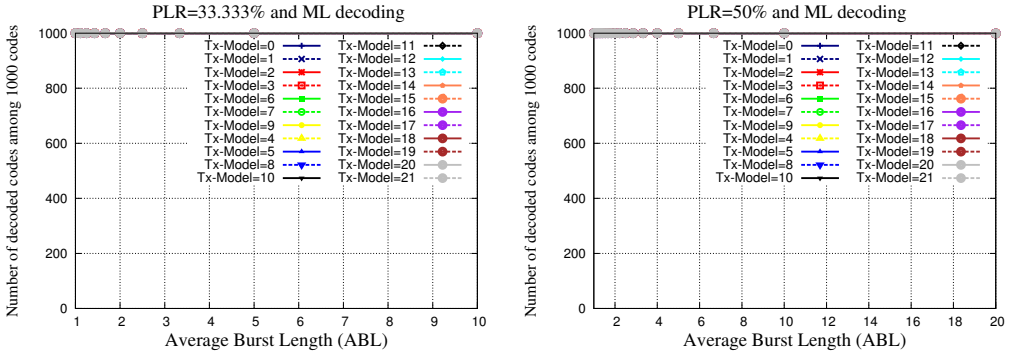
(a) Category (2)

(b) Category (3)

(c) Category (1) and (4)

(d) All categories

FIGURE 7.5: Total number of (IT+RS) successful decodings W.R.T. the ABL, with PLR =50% (top 3 figs.) and 66.66% (last fig.), when $r_G = \frac{1}{3}$.

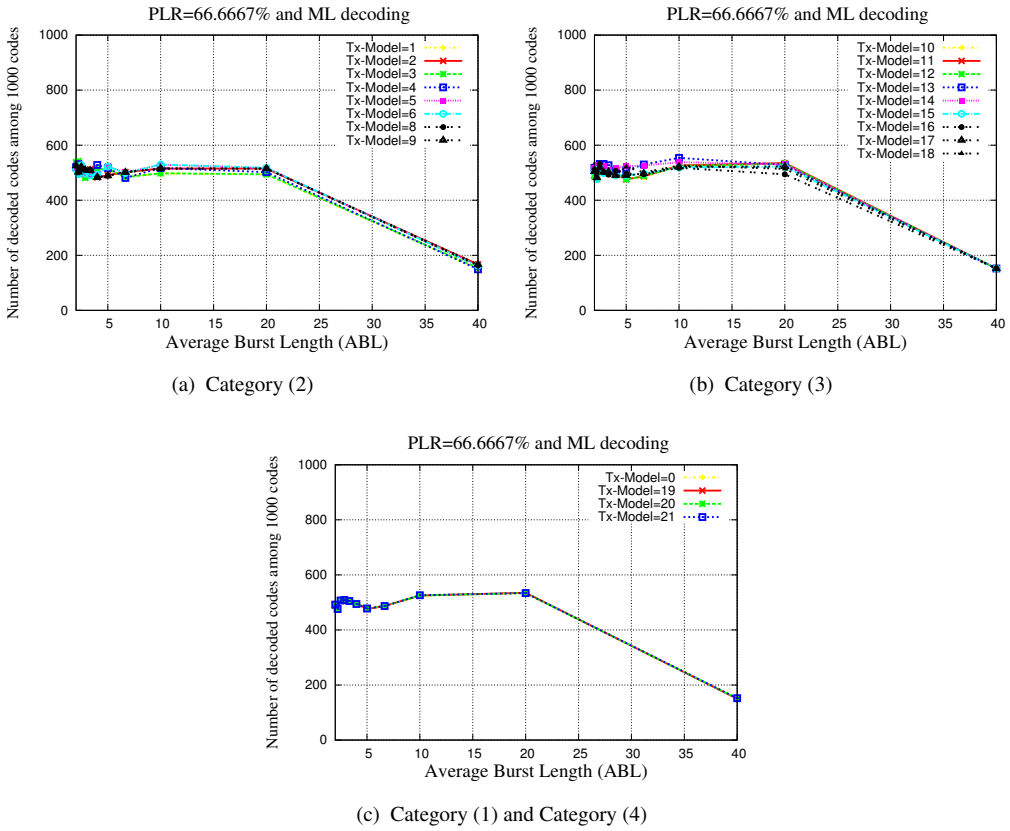


(a) All categories

(b) All categories

FIGURE 7.6: Total number of ML successful decodings W.R.T. the ABL, with PLR =33.333%, and PLR =50%, when $r_G = \frac{1}{3}$.

($21 \times 21 = 441$ different channels) using the well-known two state Gilbert model. We showed that conducting such an analysis is essential in practice in order to optimally use



(a) Category (2)

(b) Category (3)

(c) Category (1) and Category (4)

FIGURE 7.7: Total number of ML successful decodings W.R.T. the ABL, with PLR =66.66%, when $r_G = \frac{1}{3}$.

certain types of AL-FEC codes, for which the identity of the packets received does matter, in broadcast scenarios where there are as many channels as receivers.
Therefore, first of all, we detailed our methodology to address the problem; in a second step we applied it to the particular case of GLDPC-Staircase codes. By comparing the total number of successful decodings over all the channels considered, we show that a scheduling where source packets are sent first (in sequence) followed by extra-repair packets (randomly) and finally by LDPC/RS repair packets (randomly) leads to excellent performance both under (IT+RS) and ML decoding. It also has the key advantage of keeping the delay as low as possible since transmissions take place as soon as source packets are available, a key advantage with real-time streaming applications.
To the best of our knowledge, this is the first time that this question is addressed and a detailed methodology proposed. Note that this is a new question, specific to AL-FEC codes, since the flexibility they offer enables a sender to select the best packet scheduling according to the application and AL-FEC codes features: no such flexibility exists with physical layer FEC codes.

Part III

Conclusions and perspectives

Chapter 8

Conclusions and Perspectives

8.1 Conclusions

This thesis is a contribution to the design and optimization of FEC codes for the application or transport layer, so-called AL-FEC codes. These codes both ensure the *reliable* transmission of data packets and also optimize the large scale distribution of contents, in particular when retransmission is not feasible.

8.1.1 Design of GLDPC-Staircase codes

We focus on GLDPC-Staircase codes whose structure is based on LDPC-Staircase codes (as base code) and RS codes (as outer codes). These codes can be viewed as *an intelligent way of merging* two complementary AL-FEC codes. More precisely, GLDPC-Staircase codes benefit both from the *structured binary* LDPC codes, characterized by *low encoding complexity* and *good performance for medium to large objects* (where RS codes behave poorly), and also from the *non-binary, MDS, RS codes* that are *ideal codes for small objects* (i.e., where LDPC-Staircase codes behave poorly).
The coupling structure between RS and LDPC-Staircase codes is as follows: from a bipartite graph viewpoint, each check node of the base code now corresponds to an RS code (i.e., component code). This leads to a graph with powerful check nodes (compared to SPC check nodes) called *generalized check nodes*. These generalized check nodes produce *a potentially large number* of RS repair symbols (named *extra-repair symbols*) on demand, a feature that is well suited to situations where the channel conditions can be worse than expected, or to fountain like content distribution applications. The production of these extra-repair symbols allows to extend the initial LDPC Staircase code and *very small rates* are easily achievable. Moreover, GLDPC-Staircase codes are defined with *a property* (that is called scheme A in this document): the LDPC repair symbols produced in a staircase fashion are also defined as RS repair symbols, which significantly improves IT decoding. Thanks to this property, GLDPC-Staircase codes behave, in uniform and unified way, as a single AL-FEC scheme.

To design GLDPC-Staircase AL-FEC codes in practice, we proposed to use systematic "quasi" Hankel-RS codes. This is motivated by:

1. very low RS construction times which allows to generate on the fly an RS code (outer code) with appropriate dimension and length values;

2. the possibility to immediately obtain the scheme A property (this is made possible by the column of "1", in the RS generator matrix, associated to the first repair symbol).

Independently of GLDPC-Staircase codes, we showed that systematic Hankel-RS codes also outperform Vandermonde-RS codes in terms of generator matrix construction time. This is a key advantage in all use-cases where the code dimension and length values are not known in advance.

Then, in an effort to analyze the impacts of the *scheme A* structure property on GLDPC-Staircase code performance, another structure (namely, *scheme B*) has been proposed. They differ on the nature of LDPC repair symbols being RS repair symbols as well or not. We proved that scheme A has a positive impact on the (IT+RS) decoding. More precisely, these "dual identity" repair symbols increase the decoding correction power of each generalized check node (the impacts of stopping sets on (IT+RS) decoding vanish) and accelerate the (IT+RS) decoding convergence that leads to small decoding overheads. However, it does not impact ML decoding performance. So globally, scheme A is the most appropriate code construction approach.

8.1.2 Hybrid IT/RS/binary or non binary ML decoding of GLDPC-Staircase codes

A good AL-FEC solution is not only a good code (with good correction capabilities) but also a good codec (with low encoding/decoding complexity). Therefore, in addition to the basic decoder of GLDPC-Staircase codes, called (IT+RS) decoder, we proposed a new decoder type called hybrid (IT/RS/ML) decoder. Thanks to a joint use of IT, RS and binary/non binary ML decoding, it achieves ML decoding performance with a lower complexity. The hybrid decoding starts with (IT+RS) decoding and then switches to the binary/non binary ML decoding if the former decoding type fails to recover the erased source symbols.
In order to optimize the number of operations over $GF(2^m)$ in ML decoding step, we proposed an optimization. This optimization consists in reducing the number of symbols decoded with the non binary ML decoding step (i.e., decoding a set of source symbols using binary ML decoding instead of non binary ML decoding). This optimization led to a little improvement of the decoding throughput.

8.1.3 Asymptotic analysis of GLDPC-Staircase codes through DE and EXIT methods

A method for the analysis of GLDPC-Staircase codes under (IT+RS) decoding and ML decoding on the erasure channel, based on the concept of EXIT and DE tools is described. This analysis allows us to investigate the decoding convergence and to determine the gap to the theoretical limit. First, we derived Density Evolution (DE) equations for the

two mentioned previously schemes under (IT+RS) decoding over erasure channels. This technique has then been combined with EXIT functions by generalizing the area theorem, which has been proposed for LDPC codes, for our codes in order to determine the upper bound on the ML decoding threshold. This analysis showed that GLDPC-Staircase codes under ML decoding are quite close to the theoretical limit.

8.1.4 Evaluation of GLDPC-Staircase codes

Tuning important internal parameters

The GLDPC-Staircase codes are characterized by important internal parameters:

- the extra-repair distribution, which defines how extra-repair symbols are distributed on the generalized check nodes. It impacts the decoding correction capabilities;
- the base code rate (the number of extra-repair symbols per generalized check nodes) for a given global code rate. It impacts the decoding correction capabilities;
- the parameter N1, which is the degree of the source variable nodes. It impacts both the correction capabilities and the decoding complexity.

We showed that N1=5 and a small number of extra-repair symbols distributed regularly over the generalized check nodes represent the best configuration for hybrid (IT/RS/ML) decoding.

Achieved performance over memory-less channels and random packets order

In finite length analysis, based on the best configuration of GLDPC-Staircase codes for hybrid decoding, we showed that these codes exhibit exceptional memory-less-erasure recovery capabilities. More precisely, these codes show very small decoding overhead close to ideal codes, low decoding failure probabilities, low error floor, and steep waterfall region. These results are obtained no matter the object size, which is the main problem of most AL-FEC codes (e.g., LDPC-Staircase codes, RS codes, and Raptor codes). With these excellent behaviors, GLDPC-Staircase codes outperform the rate-less Raptor codes, LDPC-Staircase codes and another construction of GLDPC codes [5, 41]. In addition, they have correction capabilities close to that of RaptorQ codes.

Achieved performance over a large set of channels and packets order

In practice AL-FEC performance is largely impacted by the packet transmission scheduling and the packet loss distribution observed by a receiver, in particular when they are used in a broadcast context. We investigated the packet scheduling impacts on these codes over a very broad range of erasure channels using the well-known two state Gilbert model. First, we proposed a methodology to assess this problem. Then, through this method, numerical results revealed that a scheduling where source packets are sent first, in sequence, followed by extra-repair packets, randomly, and finally by LDPC/RS repair packets, randomly, leads to excellent performance both under (IT+RS) and ML decoding. It also has the key advantage of keeping the delay as low as possible since transmissions take place as soon as source packets are available, a key advantage with

real-time streaming applications.

All the aforesaid results make GLDPC-Staircase codes ubiquitous codes. On one side, they are suited to streaming applications where the encoding is performed on small amounts of data in order to satisfy real-time constraints. On the other side, these codes are appropriate for bulk data transfer applications where the encoding is performed on large amounts of data (ideally a single block encompassing the whole file) and is only limited by the practical aspects (memory/CPU req. during decoding).

8.2 Future works

As perspectives of this work, there are a number of problems that can being the subject of future research. Here is a short list of some of the possible directions, focus mainly on the improvement of the encoding/decoding complexity of GLDPC-Staircase AL-FEC codes and on the study of the construction of these codes for Physical/MAC layers.

8.2.1 GLDPC-Staircase Encoding/Decoding optimization

For certain lightweight terminals (e.g. smart-phones), the encoding and decoding speeds must be high (equivalently, the CPU load must be low) and the maximum memory consumption kept to a minimum. Therefore, to use GLDPC-Staircase codes on these terminals, we propose three tracks that lead to reduce the complexity.

- **Hybrid (IT/RS/ML) decoding based on SGE method**: We proposed an optimization of the hybrid decoding to reduce the number symbols decoded in non binary way. This optimization has not a great impact on the decoding throughput (in section 4.2.3.2). In [129], the SGE method has been applied to the LDPC-Staircase codes and allows to achieve a gain of 579 Mbps (for k=8422, r=4211, code rate=2/3, N1=7, symbol size=1288, and 4202 erasures). Therefore an extension and adaptation of this method for GLDPC-Staircase codes seems possible.

- **Efficient GLDPC-Staircase encoding method**: We did not discuss how to encode efficiently RS codes based on Hankel matrix. In fact, these codes are defined by a generator matrix where the repair symbols part has a structured form: it features a cyclic representation, i.e, each row is obtained by shifting to the left the previous row by a single step. Thus, it is interesting to study and propose a fast encoding method taking advantage of the form of the matrix. This would potentially reduce the overall encoding complexity of GLDPC-Staircase codes.

- **Working only on GF(2)**: We know that operations over $GF(2^m)$ are expensive. Therefore, to reduce the complexity of GLDPC-Staircase codes, we propose to project the operations of GLDPC-Staircase encoding/ decoding over $GF(2)$. More precisely, we propose to use Hankel-RS codes working on their binary image representation. This approach allows circumventing the complexity of finite fields operations by operating over a sparse graph representation of the binary images of the RS codes. For the RS decoding, instead of using the well-known adaptive parity

check matrix based on BP decoding (ADP) which requires operations on the RS parity check matrix at each iteration of BP decoding [144, 145], we can derive a new approach. This approach consists in using a fixed improved binary RS parity check matrix (i.e, with low density and small number of short cycles using for example the greedy search algorithm) before using a BP decoding algorithm or an improved BP decoding algorithm (e.g., Contraction-based Message Passing (CMP) decoding method [146]). This work is currently under study.

8.2.2 Extension of GLDPC-Staircase coding scheme over an AWGN channel

We have shown that GLDPC-Staircase codes feature excellent performance when they are used as AL-FEC codes. But how do they behave when they are used as PHY-FEC codes, where the situation is completely different since data is subject to errors instead of erasures?

Over AWGN channel, the BP decoding of LDPC codes is a Soft Input Soft Output (SISO) decoding type whereas the basic decoding of RS codes is a Hard Input Hard Output (HIHO) decoding type. Therefore, to coupling these two codes to design GLDPC-Staircase codes, we propose to use a SISO decoding instead of HIHO decoding for RS codes. In fact, the HIHO does not fully exploit the error correction capability of the original code. Whereas, the SISO decoding uses the reliability of the received information in the decoding process in order to offer coding gains over HIHO decoding and have a small performance degradation compared to ML decoding. We propose to use the well known bit-flipping Chase-2 Soft Decision Decoding (SDD) [147] combined with Pyndiah approach [148] to obtain a SISO decoding type.

Since the RS codes work on symbols instead of bits, we also propose to use symbol-flipping chase-2 decoding strategy on RS codes instead of bit-flipping Chase-2 decoding to improve the finite length performance [149]. This improvement can cause a high decoding complexity, therefore we can propose a strategy and metric which allow to reduce and adjust the size of the list efficiently. This work is currently under study.

Part IV

Appendices

Appendix A

Performance evaluation metrics and tools for correcting codes

A.1 Performance evaluation metrics

A.1.1 Correction capabilities tools

The role of a correction code being to correct errors and/or erasures, the correction capabilities of the code are thus the main metric.

A.1.1.1 Distance to the capacity Δ

From a theoretical point of view, it is desired that a code be as close as possible to the channel capacity. A code that reaches the capacity of the channel is called perfect code. On the erasure channel, a perfect code is a [n, k]-code capable of correcting $n-k$ erasures, i.e, it reaches the *Singleton bound*. Therefore, the first metric is the distance to the perfect code. This quantity is called the distance to the capacity Δ and is computed as the difference between the capacity of the code, C_c, and the capacity of the channel c_{ch}. In the case of the binary erasure channel the capacity is equal to $c_{ch} = 1 - p$, where p is the erasure channel probability. The capacity of the code is equal to $C_c = 1 - \varepsilon$, where ε is the maximum erasure probability correctable by the code. Thus Δ equals to:

$$\Delta = C_c - c_{ch} = C_c - 1 - \varepsilon \tag{A.1}$$

The distance to the capacity allows to assess the correction capabilities of code from the viewpoint of information theory. This quantity is often deduced from the probabilities of erasures that can be corrected by the code. It may therefore be obtained *analytically* or from *simulations*.

A.1.1.2 Decoding failure probability curve

In practice we are often interested in the reliability of a system and in particular the probability that it does not fulfill its function. And for erasures codes we are interested in the probability that the decoding fails (*Decoding failure probability*). On erasure channels, the decoding failed when the erased data symbols have not been rebuilt. These error decoding probabilities are often expressed as a function of the erasure channel probability

for a fixed code rate. However, it is also expressed as a function of the overhead. As mentioned above, metrics based on error probabilities are used to measure the quality of service provided to the upper layer.

Block error probability : The probability that the decoding fails is called block error probability. It corresponds to the probability that the decoding is not complete (i.e, erased source symbols cannot be recovered) given a set of received symbols.

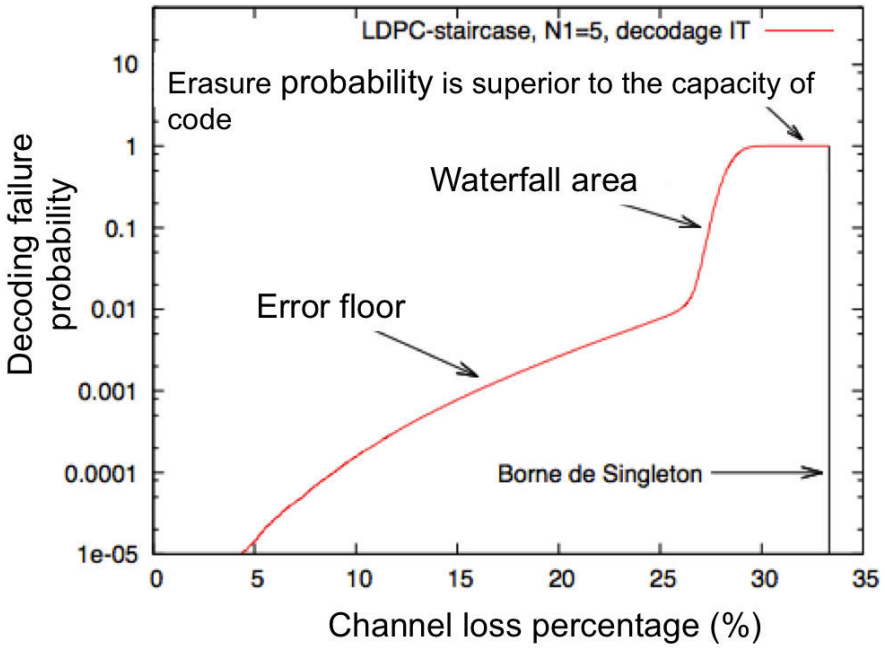


FIGURE A.1: Interest areas on a curve of block error probability according to the erasure rate (k = 1000, R = 2/3).

Symbol error probability : As we mentioned earlier, a decoding failure does not mean that the data will be not used. Indeed, if the code is systematic, part of the source symbols could be received. In addition, some decoders, such as iterative decoders, allow to reconstruct symbols progressively as the symbols are received. Even if the decoding is terminated with failure at the block level, part of the reconstructed source symbols can be exploited [150].

Curves terminologies : For perfect codes the decoding error probability is deduced directly from the code size and the number of received symbols. There is therefore a clear boundary between the success decoding area and decoding failure area. In the general case, this boundary is not very spruce, and the decoding failure probability curve as a function of the quality of the channel possess a particular form. The curve is usually divided into three areas (see figure A.1) :

- When the erasure channel probability is superior to the capacity of code, erasures are too numerous to be corrected, therefore the decoding failure probability is equal to **1**.
- The *error floor*, which corresponds to the area where the **erasure probability is low**. This area shows the very low failure rate of decoding "intrinsic" to the code, same as persistent if the channel conditions are improving. We seek to build a code with the **lowest error floor**. Note that the error floor area is not always visible, since the error probability in this area may be too small to be captured by the experiments.
- The *waterfall area* which lies between the previous two areas. In this area, the block error probability evolves quickly from low values of error floor to a maximum value 1. The closer this area to the theoretical limit, the better.

A.1.2 Decoding inefficiency ratio

In the previous section, we considered the decoding failure probability as a function of the amount of received information. In this section we consider the amount of information **necessary** to enable a successful decoding. This requires the use of *decoding inefficiency ratio*, which is defined as the ratio between the average amount required to decode and the number of source symbols (i.e, the dimension of the code symbols K). The inefficiency ratio is expressed as:

$$\text{Decoding inefficiency ratio} = \frac{\text{Number of symbols required for successful decoding.}}{K} \tag{A.2}$$

When the decoding inefficiency ratio is low, better is the correction capabilities (for perfect code, the decoding inefficiency ratio equals to 1). From the decoding inefficiency ratio we can define the additional *overhead* denoted by ε which corresponds to the distance from the decoding inefficiency ratio of the code and that of a perfect code. The decoding inefficiency ratio of a perfect code equals to *1*, then the *overhead* is expressed as follows:

$$\varepsilon = \text{decoding inefficiency ratio} - 1 \tag{A.3}$$

The decoding will succeed from $K(1+\varepsilon)$ received symbols. Therefore, the overhead is the number of symbols over K which are required to success the decoding, which is equal to $K.\varepsilon$.

A.1.3 Algorithmic complexity metrics

The complexity of system will determine the feasibility of its use in practice. Correcting codes are often a component that is plugged in series in the communication stack: if the flow is too low, they can become a bottleneck that will slow down the entire system. In addition, the codes may be used in embedded terminal, with limited processing and memory capabilities (for example in the "Digital Video Broadcasting - Handheld Satellite", Satellite mobile (DVB-SH) where the terminals are mobile receivers). The implementation of error-correcting codes can both be done in software and in hardware. Erasure codes for application level being typically implemented in software, we will focus on the critical aspects of the complexity of such an implementation.

A.1.3.1 Theoretical complexity

To estimate algorithms cost, it is common to use the theoretical complexity metric. This complexity reflects the number of operations carried out by the algorithm based on the size of the entries. It is obtained by decomposing the algorithm basic operations such as addition and multiplication. Notation is commonly used in $O(.)$ for this complexity. The function f is a big O of g if:

$$\lim_{n\to+\infty} sup \mid \frac{f(n)}{g(n)} \mid < \infty \tag{A.4}$$

We say that f is dominated by g. The cost function of an algorithm is then compare with usual function whose behavior is known as n grows. Thus we say that an algorithm has a constant cost if $f(n) \in O(1)$, linear if $f(n) \in O(n)$ and quadratic if $f(n) \in O(n^2)$.
The theoretical complexity gives us a glimpse of how that the complexity of the algorithm will evolve depending on the size of the input. But this complexity does not take into account the cost of basic operations. It is therefore a first assessment of the cost of the algorithm, but will mostly not sufficient to determine whether an encoding or decoding algorithm is faster than another.

A.1.3.2 Encoding/decoding throughput

A more practical metric in the target platform to evaluate the complexity of an algorithm is to measure the execution time. In the case of components of communication systems we consider more generally the *throughput*.
In a general, the throughput (average, minimum or maximum) achievable by an encoder or decoder, is a metric that will determine whether an implementation of an algorithm is used or not. When two systems are compared on the basis of their throughput given a set of parameters (length and dimension of the code, size of the symbols, the channel parameters), we must take the precaution to carry out experiments on the same platform, especially since the algorithms can be optimized for specific architectures microprocessors (the size of the cache memories in particular has a very large influence).
Metrics based on throughput decoding have the advantage of presenting the complexity in practice. However, they require the implementation of an optimized codec, which may require a significant amount of work, where the theoretical complexity requires just an analysis of the algorithm.

A.1.3.3 Maximum memory consumption

This metric consists of the maximum amount of information which will be required by the algorithm. Although the available memory of PC increases significantly, the AL-FEC codes can be used with huge files (several GB for video files). In addition, in constrained environment (e.g., mobile devices) have limited memory capacities, often shared with other applications.

Finally, the memory is organized according to a hierarchy whose capabilities and access time vary by several orders of magnitude. The intelligent use of memory space allows significant gains in terms of speed.

Bibliography

Bibliography

[1] F. Mattoussi, V. Roca, and B. Sayadi. Complexity comparison of the use of vandermonde versus hankel matrices to build systematic mds reed solomon codes. *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2012.

[2] F. Mattoussi, V. Savin, V. Roca, and B. Sayadi. Optimization with exit functions of gldpc-staircase codes for the bec. *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2012.

[3] F. Mattoussi, V. Roca, and B. Sayadi. Design of small rate, close to ideal, gldpc-staircase al-fec codes for the erasure channel. *IEEE Global Communications Conference (Globecom)*, 2012.

[4] F. Mattoussi, V. Roca, and B. Sayadi. Good coupling between ldpc-staircase and reed-solomon for the design of gldpc codes for the erasure channel. *IEEE Wireless Communications and Networking Conference (WCNC)*, 2013.

[5] E. Paolini. *Iterative Decoding Methods Based on Low-Density Graphs*. PhD thesis, Universitá degli Studi di Bologna, 2007.

[6] Emm-efec, selection of the fec. *document S4- 121367, 3GPP TSG-SA4 meeting 71, Bratislava, Slovakia. [Online]. Available: http://www.3gpp.org/ftp/tsg sa/WG4 CODEC/TSGS4 71/ Docs/S4- 121367.zip*, November 2012.

[7] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948. URL `http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf`.

[8] H. Bai and H. Aerospace. Error modeling schemes for fading channels in wireless communications: A survey. *IEEE Communications Surveys and Tutorials*, 5:2–9, 2003.

[9] M. Yajnik, S. B. Moon, J. Kurose, and D. Towsley. Measurement and modeling of the temporal dependence in packet loss. In *IEEE INFOCOM*, volume 1, pages 345–352, March 1999.

[10] 3GPP. Multimedia broadcast/multicast service (mbms); protocols and codecs. TS 26.346, 3rd Generation Partnership Project (3GPP).

[11] H. Ernst, L. Sartorello, and S. Scalise. Transport layer coding for the land mobile satellite channel. *59th IEEE Vehicular Technology Conference (VTC)*, May 2004.

[12] K. Bouchireb and P. Duhamel. Transmission schemes for scalable video streaming in a multicast environment. *In International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 419–424, August 2008.

[13] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss. Delay-tolerant networking : an approach to interplanetary internet. *IEEE Communications Magazine*, 41(6):128–136, June 2003.

[14] L. Vicisano L. Rizzo M. Luby, J. Gemmell and J. Crowcroft. Asynchronous layered coding (alc) protocol instantiation. *IETF Request for Comments, RFC3450*, December 2002.

[15] T. Paila, M. Luby, R. Lehtonen, V. Roca, and R. Walsh. Flute - file delivery over unidirectional transport. *IETF RMT Working Group, Request For Comments, RFC 3926*, October 2004.

[16] Digital video broadcasting (dvb) second generation farming structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broad- band satellite applications. *DRAFT EN 302 307 DVBS2-74r15, European Telecommunications Standards Institute*, 2003.

[17] R. G. Gallager. Low density parity check codes. *PhD thesis, MIT Press, Cambridge, MA*, September 1960.

[18] M. Sipser and D. A. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42(6):1710–1722, November 1996.

[19] D.J.C.Mackay. Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 45(2):399–431, March 1999. URL `availableathttp://www.inference.phy.cam.ac.uk/mackay/CodesGallager.html`.

[20] Y. Kou, S. Lin, and M. Fossorier. Low-density parity-check codes based on finite geometries: a rediscovery and new results. *IEEE Transactions on Information Theory*, 47(7):2711–2736, November 2001.

[21] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Improved low-density parity-check codes using irregular graphs. *IEEE Transactions on Information Theory*, 47(2):585–598, February 2001.

[22] T. Richardson, A. Shokrollahi, and R. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):619–637, February 2001.

[23] T.J. Richardson and R.L. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47(2): 599–618, February 2001.

[24] C. Berrou and A. Glavieux. Near optimum error correcting coding and decoding : turbo-codes. *IEEE, Transactions on Information Theory*, 44(10):1261–1271, October 1996.

[25] S. Y. Chung, G. D. Forneya, T. J. Richardson, and R. Urbanke. On the design of low-density parity-check codes within 0.0045 db of the shannon limit. *IEEE Communication Lettres*, 5(2):58–60, February 2001.

[26] M. P. C. Fossorier. Quasi-cyclic low-density parity-check codes from circulant permutation matrices. *IEEE Transactions on Information Theory*, 50(8):1788–1793, August 2004.

[27] H. Xiao and A. H. Banihashemi. Graph-based message-passing schedules for decoding ldpc codes. *IEEE Transactions on Communications*, 52(12):2098–2105, December 2004.

[28] H. Song, R. M. Todd, and J. R. Cruz. Low density parity check codes for magnetic recording channels. *IEEE Transactions on Magnetics*, 36(5):2183–2186, September 2000.

[29] D. MacKay and R. Neal. Near Shannon limit performance of low density parity check codes. *IET Electronics Letters*, 33(6):457 – 458, 1997.

[30] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 47(2):569–584, Ferbruary 2001.

[31] P. Oswald and M. A. Shokrollahi. Capacity-achieving sequences for the erasure channel. *IEEE Transactions on Information Theory*, 48(12):3017–3028, December 2002.

[32] M. Chiani and A. Ventura. Design and performance evaluation of some high-rate irregular low-density parity-check codes. *Proceding of IEEE Global Telecommunications Conference, San Antonio, USA*, November 2001.

[33] T. Richardson. Error floors of ldpc codes. *Proceding of Allerton Conference on Communication, Control and Computing*, October 2003.

[34] A. Amraoui, A. Montanari, and R. Urbanke. How to find good finite-length codes: From art towards science. *Proceding 4th International Symposium on Turbo Codes and Related Topics*, 2006.

[35] C. Di, R. Urbanke, and T. Richardson. Weight distribution of low-density parity-check codes. *IEEE Transactions on Information Theory*, 52(11):4839–4855, November 2006.

[36] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, September 1981. ISSN 0018-9448. doi: 10.1109/TIT.1981.1056404. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1056404`.

[37] J. Boutros, O. Pothier, and G. Zémor. Generalized Low Density (Tanner) Codes. *IEEE International Conference on Communications (ICC)*, 1:441–445, 1999.

[38] M. Lentmaier and K. Zigangirov. On Generalized Low-Density Parity-Check Codes Based on Hamming Component Codes. *IEEE Transactions on Communications*, 3 (8):248–250, 1999.

[39] G. Yue, L. Ping, and X. Wang. Generalized low-density parity-check codes based on hadamard constraints. *IEEE Transactions on Information Theory*, 53(3):1058–1079, March 2007.

[40] J. Chen and R. Tanner. A hybrid coding scheme for the gilbert-elliott channel. *IEEE Transactions on Communications*, 54(10):1787–1796, October 2006.

[41] N. Miladinovic and M. Fossorier. Generalized LDPC Codes with Reed-Solomon and BCH Codes as Component Codes for Binary Channels. *IEEE Global Telecommunications Conference (GLOBECOM)*, 2005.

[42] I. Djordjevic, O. Milenkovic, and B. Vasic. Generalized Low-Density Parity-Check Codes for Optical Communication Systems. *Lightwave Technology*, 23(5):1939–1946, 2005.

[43] Y. Wang and M. Fossorier. Doubly Generalized LDPC Codes. *IEEE International Symposium on Information Theory*, pages 669–673, July 2006.

[44] E. Paolini, M. Fossorier, and M. Chiani. Analysis of Doubly-Generalized LDPC Codes with Random Component Codes for the Binary Erasure Channel. *Proceedings of Allerton Conference on Communications, Control and Computing*, 2006.

[45] Mark Watson, Ali Begen, and Vincent Roca. *Forward Error Correction (FEC) Framework*, 2011. IETF Request for Comments, RFC 6363 (Standards Track/Proposed Standard).

[46] V. Roca, C. Neumann, and D. Furodet. Low density parity check (ldpc) staircase and triangle forward error correction (fec) schemes. 2008. IETF Request for Comments, RFC 5170 (Standards Track/Proposed Standard).

[47] J. Lacan, V. Roca, J. Peltotalo, and S. Peltotalo. Reed-solomon error correction scheme. *IETF RMT Working Group, Request for Comments, RFC 5510*, April 2009.

[48] M. Cunche V. Roca and J. Lacan. Simple low-density parity check (ldpc) staircase forward error correction (fec) scheme for fecframe. *IETF Request for Comments, RFC 6816 (Standards Track)*, December 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6816. txt.

[49] J. Lacan A. Bouabdallah V. Roca, M. Cunche and K. Matsuzono. Simple reed-solomon forward error correction (fec) scheme for fecframe. *IETF Request for Comments, RFC 6865 (Standards Track)*, February 2013. [Online]. Available: http://www.ietf.org/rfc/rfc6865. txt.

[50] A. Yamada, H. Matsuoka, T. Ohya, R. Kitahara, J. Hagiwara, and T. Morizumi. *in IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB'11)*, June 2011.

[51] M. Cunche. Codes al-fec hautes performances pour les canaux déffacements : variations autour des codes ldpc. May 2010. Thesis.

[52] M. Cunche, V. Savin, V. Roca, G. Kraidy, A. Soro, and J. Lacan. Low-rate coding using incremental redundancy for GLDPC codes. In *IEEE International Workshop on Satellite and Space Comm*, pages 299–303, 2008.

[53] C. Méasson. *Conservation laws for coding*. PhD thesis, École Polytechnique Fédérale de Lausanne (EPFL), 2006.

[54] . URL `http://en.wikipedia.org/wiki/Entropy_information_theory`.

[55] M. Jiang, Z. Shi C. Zhao, and Y. Chen. An improvement on the modified weighted bit flipping decoding algorithm for ldpc codes. *Communications Letters, IEEE*, 9 (9):814–816, September 2005.

[56] A. Nouh and A. Banihashemi. Bootstrap decoding of low-density parity-check codes. *Communications Letters, IEEE*, 6(9):391–393, sep 2002.

[57] P. Elias. Coding for two noisy channels. *in Proceedings 3rd London Symposium Information Theory*, pages 61–76, September 1955.

[58] . URL `http://en.wikipedia.org/wiki/Mutual_information`.

[59] R. Singleton. Maximum distance q-nary codes. *IEEE Transactions on Information Theory*, 10(2):116–118, April 1964.

[60] M. Luby, M. Mitzenmacher, A. Shokrollah, and D. Spielman. Analysis of low density codes and improved designs using irregular graphs. *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 249–258, 1998. URL `http://dblp.uni-trier.de/db/conf/stoc/stoc1998.html#LubyMSS98`.

[61] C. Di, D. Proietti, I. E. Telatar, T. J. 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, June 2002.

[62] X. Hu, E. Eleftheriou, and D. Arnold. Regular and irregular progressive edge-growth tanner graphs. *IEEE Transactions on Information Theory*, 51(1):386–398, January 2005.

[63] H. Jin, A. Khandekar, and R. McEliece. Irregular repeat-accumulate codes. *in Proc. 2nd International Symposium on Turbo Codes and Related Topics, Brest, France*, pages 1–8, September 2000.

[64] D. Divsalar, C. Jones, S. Dolinar, and J. Thorpe. Protograph based ldpc codes with minimum distance linearly growing with block size. *in Proceedings of IEEE Global Telecommunications Conference*, November/December 2005.

[65] N. Alon and M. Luby. A linear time erasure-resilient code with nearly optimal recovery. *IEEE Transactions on Information Theory*, 42(6):1732–1736, November 1996.

[66] Tang. H, J Xu, Y Kou, S Lin, and K. Abdel-Ghaffar. On algebraic construction of gallager and circulant low-density parity-check codes. *IEEE Transactions on Information Theory*, 50(6):1269–1279, June 2004.

[67] D. J. C. MacKay and R. M. Neal. Near shannon limit performance of low density parity check codes. *Electronics Letters*, 33(6):457–458, March 1997.

[68] B. Ammar, Honary B., Y. Kou, Xu J., 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, June 2004.

[69] B. J. Frey and D. J. C. MacKay. Irregular turbo codes. *in Proceedings of the 37th Allerton Conference on Communication, Control, and Computing*, 1999.

[70] H. Tang, J. Xu, S. Lin, and K. A. S. Abdel-Ghaffar. Codes on finite geometries. *IEEE Transactions on Information Theory*, 51(2):572–597, February 2005.

[71] Y. Y. Tai L. Chen S. Lin L. Lan, L. Zeng and K. Abdel-Ghaffar. Construction of quasi-cyclic ldpc codes for awgn and binary erasure channels: a finite field approach. *IEEE Transactions on Information Theory*, 53(7):2429–2458, July 2007.

[72] T.R. Oening and Jaekyun Moon. A low-density generator matrix interpretation of parallel concatenated single bit parity codes. *IEEE Transactions on Magnetics*, 37 (2):737–741, March 2001.

[73] J.Garcia-Frias and Wei Zhong. Approaching shannon performance by iterative decoding of linear codes with low density generator matrix. *IEEE Communication lettre*, 7(6):266–268, June 2003.

[74] T.J.Richardson and Jaekyun Moon. Efficient encoding of low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):638–656, February 2001.

[75] D. Divsalar, H. Jin, and R. J. Mceliece. Coding theorems for turbo-like codes. *In Proceedings 36 Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, USA*, pages 201–210, Sept 1998.

[76] S. lin and D. J.costello. Error control coding (2nd edition). *Prentice Hall: Englewood Cliffs, NJ. ISBN 0-13-017973-b*, 2004.

[77] V. V. Zyablov and M. S. Pinsker. Decoding complexity of low-density codes for transmission in a channel with erasures. *Problemy Peredachi Informatsii*, 10(1): 15–28, January-March 1974.

[78] A. Shokrollahi. Ldpc codes: An introduction. *Digital Fountain, Inc., Tech. Rep*, page 2, 2003.

[79] H. Pishro-Nik and F. Fekri. On decoding of low-density parity-check codes over the binary erasure channel. *IEEE Transactions on Information Theory*, 50(3):439–454, March 2004.

[80] G. Miller and D. Burshtein. Bounds on the maximum likelihood decoding error probability of low density parity check codes. *Proceedings. IEEE International Symposium on In Information Theory*, 2000.

[81] Solving large sparse linear systems over finite fields. *in Advances in Cryptology, LNCS 537, Springer-Verlag*, 1991.

[82] Reduction of huge, sparse matrices over finite fields via created catastrophes. *Experimental Mathematics*, 1(2), 1992.

[83] M. Cunche and V. Roca. Optimizing the error recovery capabilities of ldpc-staircase codes featuring a gaussian elimination decoding scheme. *In 10th IEEE International Workshop on Signal Processing for Space Communications (SPSC, 2008), Rhodes Island, Greece*, October 2008.

[84] E. Paolini, G. Liva, B. Matuz, and M. Chiani. Generalized ira erasure correcting codes for hybrid iterative/maximum likelihood decoding. *Communications Letters, IEEE*, 12(6):450–452, June 2008.

[85] E. Paolini, M. Varrella, M. Chiani, B. Matuz, and G. Liva. Low-complexity ldpc codes with near-optimum performance over the bec. *CoRR*, abs/0804.2991:274–282, April 2008. URL `http://dblp.uni-trier.de/db/journals/corr/corr0804.html#abs-0804-2991`.

[86] S. ten Brink. Convergence of iterative decoding. *IEEE Electronics Letters*, 35(10): 806–808, May 1999.

[87] S. ten Brink. Iterative decoding trajectories of parallel concatenated codes. *in Proc. 3rd IEE/ITG Conference on Source and Channel Coding, Munich, Germany*, pages 75–80, January 2000.

[88] S. ten Brink. Convergence behavior of iteratively decoded parallel concatenated codes. *IEEE Transactions on Communications*, 49(10):1727–1737, October 2001.

[89] M. Ardakani and F. R. Kschischang. A more accurate one-dimensional analysis and design of irregular ldpc codes. *IEEE Transactions on Communication*, 52(12): 2106–2114, December 2004.

[90] M. Ardakani and F. R. Kschischang. Designing irregular ldpc codes using exit charts based on message error probability. *in Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, July 2002.

[91] H. El Gamal and A. R. Hammons. Analyzing the turbo decoder using the gaussian approximation. *IEEE Transactions on Information Theory*, 47(2):671–686, February 2001.

[92] D. Divsalar, S. Dolinar, and F. Pollara. Low complexity turbo-like codes. *in Proceedings of the 2nd International Symposium on Turbo Codes and Related Topics, Brest, France*, pages 73–80, 2000.

[93] S. ten Brink M. Tüchler and J. Hagenauer. Measures for tracing convergence of iterative decoding algorithms. *in Proceedings 4th International IEEE/ITG Conference on Source and Channel Coding*, pages 53–60, January 2002.

[94] M. G. Luby, M.Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann. Practical loss-resilient codes. *In Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 150–159, 1997.

[95] A. Ashikhmin, G. Kramer, and S. ten Brink. Extrinsic information transfer functions: model and erasure channel properties. *IEEE Transactions on Information Theory*, 50(11):2657–2673, November 2004.

[96] C. Méasson, A. Montanari, and R. Urbanke. Maxwell's construction: The hidden bridge between maximum-likelihood and iterative decoding. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, page 225, 2004.

[97] D. MacKay and M. Postol. Weaknesses of margulis and ramanujan-margulis low-density parity-check codes. *Electronic Notes in Theoretical Computer Science*, 74: 97–104, 2003. URL: http://www.elsevier.nl/locate/entcs/volume74.html.

[98] 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.

[99] S. Kim, J.-S. No, H. Chung, and D.-J. Shin. Girth analysis of tanner's (3,5) qc ldpc codes. *in Proceedings of IEEE International Symposium on Information Theory (ISIT)*, pages 1632–1636, September 2005.

[100] J. Lu, J. M. F. Moura, and U. Niesen. Grouping-and-shifting designs for structured ldpc codes with large girth. *in Proceedings of IEEE International Symposium on Information Theory (ISIT)*, page 236, June-July .

[101] 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, August 2004.

[102] H. Song, J. Liu, and B. V. K. Vijaya Kumar. Large girth cycle codes for partial response channels. *IEEE Transactions on Magnetics*, 40(4):3084–3086, 2004.

[103] J. Lu, J. M. F. Moura, and U. Niesen. A class of structured ldpc codes with large girth. *in Proceedings of IEEE International Conference on Communications*, 1: 425–429, June 2004.

[104] H. Zhang and J. M. F. Moura. The design of structured regular ldpc codes with large girth. *in Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)*, 7:4022–4027, December 2003.

[105] Y. Xiao and M. H. Lee. Construction of good quasi-cyclic ldpc codes. *in Proceedings of IET International Conference on Wireless Mobile and Multimedia Networks (ICWMMN)*, page 172, November 2006.

[106] J. Fan and Y. Xiao. A method of counting the number of cycles in ldpc codes. *in Proceedings of the 8th International Conference on Signal Processing (ICSP '06)*, 3:2183–2186, November 2006.

[107] A. Orlitsky, K. Viswanathan, and J. Zhang. Stopping set distribution of ldpc code ensembles. *IEEE Transactions on Information Theory*, 51(3):929–953, March 2005.

[108] S. Laendner and O. Milenkovic. Algorithmic and combinatorial analysis of trapping sets in structured ldpc codes. *Proceedings of Wireless Communications*, June 2005.

[109] O. Milenkovic, E. Soljanin, and P. Whiting. Asymptotic spectra of trapping sets in regular and irregular ldpc code ensembles. *IEEE Transactions on Information Theory*, 53(1):39–55, December 2006.

[110] Y. Mao and A. H. Banihashemi. A heuristic search for good low-density parity-check codes at short block lengths. *in Proceding IEEE International Conference Communication*, June 2001.

[111] T. Tian, C. Jones, J. Villasenor, and R. D. Wesel. Construction of irregular ldpc codes with low error floors. *in Proceedings IEEE International Conference on Communications*, 2003.

[112] I. Reed and G. Solomon. Polynomial codes over certain finite fields. *Society for Industrial and Applied Mathematics Journal.*, 8(2):300–304, June 1960.

[113] F. J. McWilliams and N.J.A. Sloane. The theory of error correcting codes. 1977. North-Holland Publishing Company, ISBN: 0 444 85009 0.

[114] J. C. Moreira and P. G. Farrell. Essentials of error-control coding. *John Wiley Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England. ISBN-13 978-0-470-02920-6 (HB)*, 2006.

[115] M. Blaum. A course on error correcting codes. 2001.

[116] R. E. Blahut. Algebraic codes for data transmission. *Cambridge,UK: Cambridge University Press*, 2002.

[117] V. Guruswami and A. Vardy. Maximum-likelihood decoding of reed- solomon codes is np-hard. *in Proceedings of the sixteenth annual ACM- SIAM symposium on Discrete algorithms*, pages 470–478, January 2005.

[118] D. J. C MacKay. Digital fountain codes. available at http://www.inference.phy.cam.ac.uk/mackay/DFountain.html.

[119] D. J. C. MacKay. Fountain codes. *IEE Proceding Communication*, 152(6):1062–1068, December 2005.

[120] W. Ryan and S. Lin. Channel codes: Classical and modern. *Cambridge University Press, New York, ISBN-13 978-0-511-64182-4*, 2009.

[121] M. Luby. Lt codes. *Proceedings of the 43 rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 271–280, 2002.

[122] A. Shokrollahi. Raptor codes. *IEEE/ACM Transactions on Networking*, 14:2551–2567, 2006.

[123] M. Luby, A. Shokrollahi, M.Watson, and T. Stockhammer. Raptor forward error correction scheme for object delivery. *IRFC 5053*, October 2007.

[124] ETSI. Etsi, ip datacast over dvb-h : Content delivery protocols (cdp). 2006. Technical Report TS 102 472 v1.2.1, ETSI.

[125] M. Luby, A. Shokrollahi, M.Watson, T. Stockhammer, and L. Minder. Raptorq forward error correction scheme for object delivery. *IRFC 6330*, August 2011.

[126] A. U. Pandya, S. D. Trapasiya, and S. S. Chinnam. Comparative analysis of al-fec raptor and raptorq over 3gpp embms network. *International Journal of Electronics and Communication Engineering (IJECE)*, 2(2):169–178, May 2013.

[127] S. Lassen, R. Karp, and A. M. Shokrollahi. Systems and processes for decoding a chain reaction code through inactivation. *US Patent 7030785*, April 2006.

[128] V. Roca and C. Neumann. Design, evaluation and comparison of four large block fec codecs, ldpc, ldgm, ldgm staircase and ldgm triangle, plus a reed- solomon small block fec codec. June 2004. Research Report 5225, INRIA.

[129] V. Roca, M. Cunche, C. Thienot, J. Detchart, and J. Lacan. Rs + ldpc-staircase codes for the erasure channel: Standards, usage and performance. In *9th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. URL `http://hal.inria.fr/hal-00850118`.

[130] J. Lacan and J. Fimes. A construction of matrices with no singular square submatrices. *in Proceeding International Conference on Finite Fields and Applications*, pages 145–147, 2003.

[131] J. Lacan and J. Fimes. Systematic mds erasure codes based on vandermonde matrices. *IEEE Communications Lettres*, 8(9):570–572, September 2004.

[132] R. M. Roth and G. Seroussi. On generator matrices of mds codes. *IEEE Transactions on Information Theory*, 31(6):826–830, November 1985.

[133] J. Fimes and J. Lacan. Estimation of the number of singular square submatrices of vandermonde matrices defined over a finite field. *Technical Report ENSICA*, 2003.

[134] I. Shparlinski. On singularity of generalized vandermonde matrices over finite fields. *preprint*, 2000.

[135] I. Gohberg and V. Olshevsky. Fast algorithms with preprocessing for matrix-vector multiplication problems. *Journal of Complexity*, 10(4):411–427, December 1994. URL `http://www.sciencedirect.com/science/article/pii/S0885064X84710211`.

[136] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM SIGCOMM computer communication review*, 27(2):24–36, April 1997.

[137] J. Bolot and A. V. Garcia. The case for fec-based error control for packet audio in the internet. *ACM International Conference of Multimedia*, 1996.

[138] C. C. Tan and N. C. Beaulieu. On first-order markov modeling for the raleigh fading channel. *IEEE Transactions on Communications*, 48(12):2032–2040, 2000.

[139] J.Bolot, S.Fosse-Parisis, and D.Towsley. Adaptative fec-base error control for interactive audio in the internet. *Annual Joint Conference of the IEEE Computer and Communications Societies Proceedings (INFOCOM)*, 1999.

[140] H. Sanneck, G. Carle, and R. Koodli. A framework model for packet loss metrics based on loss runlength. *SPIE/ACM SIGMM Multimedia Computing Network Conference*, pages 177–187, 2000.

[141] E. N. Gilbert. Capacity of burst-noise channel. *Bell System. Technical Journal*, 39: 1253–1266, 1960.

[142] E.O.Elliot. A model of the switched telephone network for data communications. *Bell system Technical Journal*, 44(1):89–109, 1963.

[143] A. Konrad, B. Zhao, A. Joseph, and R. Ludwig. A markov-based channel model algorithm for wireless networks. *Fourth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2001.

[144] J. Jiang and K Narayanan. Iterative soft input soft output decoding of reed-solomon codes by adapting the parity check matrix. *IEEE Transactions on Information Theory*, 52(8):3746–3756, 2008.

[145] B. Sayadi A. Alloum and V. Roca. Reed solomon codes on graph for dvb-sh streaming services. *Wireless World Research Forum (WWRF)*, 2009.

[146] Badri N. Vellambi and Faramarz Fekri. An improved decoding algorithm for low-density parity-check codes over the binary erasure channel. In *GLOBECOM*, page 5, 2005. URL `http://dblp.uni-trier.de/db/conf/globecom/globecom2005.html#VellambiF05`.

[147] D. Chase. A class of algorithms for decoding block codes with measurement information. *IEEE Transactions on Information Theory*, IT-18(1):170–182, January 1972.

[148] R. Pyndiah. Near optimum decoding of product codes: Block turbo codes. *IEEE Transactions on communications*, 46(8):1003–1010, August 1998.

[149] D. C. Cunha, J. Portugheis, and V. V. do Nascimento. Symbol-wise chase decoding of q-ary block codes over bi-awgn channels. *IEEE communications letters*, 15(2): 229–231, 2011 2011.

[150] C. Neumann and V. Roca. Analysis of fec codes for partially reliable media broadcasting schemes. *In 2nd ACM International Workshop on Multimedia Interactive Protocols and Systems (MIPS,2004)*, 2004.

BIBLIOGRAPHY

BIBLIOGRAPHY