

UNIVERSITY OF BELGRADE
SCHOOL OF ELECTRICAL ENGINEERING

OMRAN AL RASHEED

**LOW COMPLEXITY DECODING ALGORITHMS
SUITABLE FOR APPLICATION IN ASYMMETRIC
CRYPTOSYSTEMS**

Doctoral Dissertation

Belgrade, 2015

UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET

OMRAN AL RASHEED

**ALGORITMI DEKODOVANJA MALE
KOMPLEKNOSTI POGODNI ZA PRIMENU U
ASIMETRIČNIM KRIPTOSISTEMIMA**

doktorska disertacija

Beograd, 2015

Mentor disertacije

dr Predrag N. Ivaniš, vanredni profesor,
Univerzitet u Beogradu, Elektrotehnički fakultet

Članovi komisije

dr Zoran Čiča, docent,
Univerzitet u Beogradu, Elektrotehnički fakultet

dr Goran T. Đorđević, vanredni profesor,
Univerzitet u Nišu, Elektronski fakultet

dr Aleksandra Smiljanić, redovni profesor,
Univerzitet u Beogradu, Elektrotehnički fakultet

dr Marija Rašajski, vanredni profesor,
Univerzitet u Beogradu, Elektrotehnički fakultet

Datum odbrane

List of Contents

List of Figures	I
List of Tables	IV
Abbreviations	V
Author's publications	VII
Abstract	VIII
1. Introduction	1
1.1. Public key cryptosystems	2
1.2. Post-quantum cryptography	5
1.3. Key size problem	8
1.4. Contributions and Outline	12
2. McEliece Cryptosystem	13
2.1. Goppa codes	14
2.2. Description of the McEliece cryptosystem	15
2.3. Modifications of McEliece cryptosystem based on LDPC codes	18
2.3.1. Permutation equivalent private and public codes	21
2.3.2. Non-permutation equivalent private and public codes ...	22
2.4. Moderate Density Parity Check (MDPC) codes	24
2.4.1. Procedure of the MDPC/QC-MDPC code-based McEliece	26
2.5. Implementations of McEliece cryptosystems	27
3. LDPC Codes and Iterative Decoding Algorithms	29
3.1. Linear Block Codes	30
3.2. Preliminaries	32
3.3. Encoding of LDPC codes	34
3.3.1. Progressive edge growth construction	34
3.3.2. Quasi cyclic construction	37
3.3.3. Encoding process	38
3.3.3.1. Example of Greedy Permutation Algorithm.....	41
3.4. Decoding of LDPC codes	42
3.4.1. Belief Propagation Algorithm	45
3.4.2. Reduced Complexity Decoders	47
3.4.3. Gallager A\B decoder	48
4. Improvement of the Bit-Flipping Algorithm and Faulty Decoding	51
4.1. Bit flipping algorithm	52
4.2. Weighted Bit Flipping algorithm and modified	53
4.3. Gradient Descent Bit flipping algorithm	56
4.4. Optimize GDBF algorithm to the BSC	58
4.4.1. Comparison between GDBF and other decoders.....	61
4.4.1.1. Motivating Examples.....	63
4.4.2. Probabilistic GDBF decoder for BSC	69

4.4.3. Find optimal p value for PGDBF	76
4.4.4. The Multiple Decoding attempts and Random re- Initializations (MUDRI) algorithm	78
4.4.5. Analysis of the MUDRI algorithm.....	82
4.5. Fault-Tolerant PGDBF and MUDRI Decoders for BSC	85
4.6. Hardware Realization and Complexity of PGDBF Algorithm.....	91
5. Performance and complexity of modified McEliece cryptosystem	95
5.1. The choice of the decoding algorithm	96
5.1.1. Numerical results for LDPC codes	98
5.1.2. Numerical results for MDPC codes	103
5.2. Computational complexity	106
5.3. Cryptanalysis of the McEliece cryptosystem	114
5.3.1. Attacks on the Dual code	114
5.3.2. Information Set Decoding Attacks	115
6. Conclusion and Perspective	119
References	122

List of Figures

1.1	Symmetric (a) and asymmetric (b) cryptosystems	1
1.2	Eavesdropper obtains the public key from public directory (a); Quantum computer can break security by reverse computing private key faster than a classical computer (b)	6
1.3	Post-quantum cryptography and broken cryptography by post quantum computers	8
2.1	Original procedure of the McEliece cryptosystem with Goppa codes	17
3.1	The basic channel coding model	31
3.2	The TG for the parity check matrix $\mathbf{H}(8,4)$	33
3.3	Bipartite graph before PEG procedure	35
3.4	The second stage from procedure PEG for v_6	36
3.5	Illustration of parity-check matrix of PEGReg (504, 252) LDPC code	37
3.6	Illustration of parity-check matrix of Tanner (155, 64) LDPC code	38
3.7	Approximate lower triangular (ALT) form	40
3.8	Systematic approximate lower triangular (SALT) form	40
3.9	Passed messages over TG for MP decoder at iteration l	45
3.10	FER performance on Tanner code (155,64) by passing decoding algorithms over BSC	50
3.11	FER performance on Tanner code (155,64) by passing decoding algorithms over AWGN	50
4.1	Simple telecommunication model for AWGNC using GDBF decoder	56
4.2	Convergence behavior and escape from local max	57
4.3	Illustration of the variable node processing unit for (a) GDBF and (b) BF algorithms over BSC	60
4.4 (a)	Performance of GDBF in soft and hard ways for Tanner (155,64), $L=100$	60
4.4 (b)	FER performance comparison for the (155,64) Tanner code, $L=100$	60
4.5	Weight-three error configurations un-correctable by BF during 3 iterations...	64
4.6	Weight-three error configurations correctable by general GDBF during two iterations	65
4.7	Weight-three error configurations correctable by Gallager A\B decoder and uncorrectable by GDBF algorithm.....	66
4.8	Weight-three error configurations correctable by Gallager A\B decoder and uncorrectable by GDBF algorithm.	66
4.9	Weight-four error configurations uncorrectable by GDBF algorithm where $\mathbf{b}^{(l)}$ gets stuck only on variable v_3 for each iteration	67
4.10	Weight-five error configurations correctable by GDBF algorithm.	67
4.11	Weight-six error configurations uncorrectable by GDBF algorithm where $\mathbf{b}^{(l)}$ gets stuck on variables v_6 and v_8 after 9 iterations	68
4.12	Illustration of the variable node processing unit for PGDBF (Algorithm 2) ...	70
4.13	Behavior of GDBF and PGDBF algorithms for five-error pattern on the Tanner code (155, 64)	73
4.14.	Mechanism of GDBF and PGDBF to correct the error patterns on the Tanner code (155, 64)	74-75
4.15	Impact of the parameter p on PGDBF optimized for the BSC. The plot is for the (155,64) Tanner code and QC(732.551), $\alpha=4\times 10^{-3}$, $\alpha=10^{-2}$ and $L=100$	77
4.16	Performance of PGDBF algorithm for the (155,64) Tanner code with different p values	77

4.17	MUDRI algorithm scheme	79
4.18	Performance of GDBF and PGDBF with optimal choice for five-error pattern	80
4.19	Adaption method to solve the error pattern by decrease the threshold value...	80
4.20	Probability distribution of the successful decoding in the l -th iteration of PGDBF, three-bit and five-bit error pattern, Tanner(155,64) code, $p=0.7$	83
4.21	Probability of unsuccessful decoding for the three-bit and five-bit error pattern, MUDRI with $\lfloor L/L_1 \rfloor$ attempts per L_1 iterations each, $p=0.7$	84
4.22.	FER as a function of number of iteration l , Tanner (155,64) code, $\alpha=0.01$, various decoding algorithms	84
4.23	Illustration of the variable node processing unit for PGDBF under faulty hardware	86
4.24	FER performance as a function of α under various decoding algorithms, Tanner code (155,64), $P_{\oplus}=10^{-2}$, $P_R=10^{-3}$, $L=100$	86
4.25	FER as function of parameter p , LDPC codes with $\gamma=3$ and $\gamma=4$, $\alpha=0.004$	88
4.26	FER as a function of probability of error XOR gates, $\alpha=0.004$, $P_R=0$, LDPC codes with $\gamma=3$ and girth-8, with various code rates and codeword lengths.....	89
4.27	FER as a function of probability of error in registers, $\alpha=0.008$, $P_{\oplus}=10^{-3}$, LDPC codes with $\gamma=3$, $\gamma=4$ and girth-8, various codeword lengths	90
4.28	FER as a function of crossover probability in BSC channel. The code is LS(2388,1793)(C_1), with $P_{\oplus}=10^{-3}$, $P_R=10^{-4}$, $p=0.7$ in the PGDBF and MUDRI and other decoding algorithms	91
4.29	Global architecture of PGDBF compared to the original GDBF.....	92
4.30	FER performance comparison of the different decoders on the Tanner code (155,64)	92
5.1	PGDBF decoders in the parallel form	97
5.2	Error correction performance as a function of the intentional errors for Tanner code (155,64) under BF, PGDBF, MUDRI, MUDRI-P, SPA and PGDBF-PR decoding algorithms	98
5.3.	Average number of iterations as a function of the intentional errors for Tanner code (155,64) under BF, PGDBF, MUDRI, MUDRI-P, SPA and PGDBF-PR decoding algorithms	99
5.4	Error correction performance as a function of the intentional errors for quasi-cyclic $\gamma_v=4$, $\rho_c=8$ and $N=1296$, under GDBF, PGDBF, MUDRI, PGDBF-PR and SPA decoding algorithms	101
5.5	Error correction performance as a function of the intentional errors for QC-LDPC PDFs code with $n=8168$, $p=1021$, $R=7/8$ and $\gamma_v=5$ under SPA, PGDBF, MUDRI-P and PGDBF-PR decoders	102
5.6	Error correction performance as a function of the intentional errors for QC-MDPC (9600,4800,90) under NBP, GDBF, 5-MUDR-P, and 10-MUDRI-P algorithms	104
5.7	Error correction performance as a function of the intentional errors for QC-MDPC (12288,3072,220) under NBP, GDBF, and 5-MUDR-P algorithms.....	104
5.8	Average number of iterations as a function of the intentional errors for QC-MDPC (12288,3072,220)	105
5.9	Comparison of the threshold during the decoding iterations for Tanner (155,64) code, $\gamma_v=3$	108
5.10	Comparison of threshold during the decoding iterations for QC-LDPC (8168,1021) code, $\gamma_v=5$	109

5.11	Comparison of the threshold during the decoding iterations for QC-MDPC (9600,4800,90) code, $\gamma_v=45$	109
5.12	Comparison of the threshold during the decoding iterations for QC-MDPC (12288, 3072, 220) code, $\gamma_v=55$	110
5.13	The computational complexity (per decoding process) comparison in the term of binary operations for Tanner code (155,64)	113
5.14	The computational complexity (per decoding process) comparison in the term of binary operations for QC-MDPC (12288,3072,220)	113

List of Tables

1.1	Key size comparisons ECC, RSA and McEliece bit size requirements for AES to achieve different bit-security level	9
1.2	Parameters of the original and modified McEliece cryptosystem	10
2.1	Performance comparison of microcontroller implementations for different public encryption schemes	28
2.2	Performance comparison of FPGA implementations for different public key encryption schemes	28
4.1	Hardware and throughput estimation for PGDBF with different RG implementation and for Min-Sum	93
5.1	LDPC decoders are used to represent Figure 5.2	100
5.2	Average number of iterations and number of errors that can be corrected with FER= 10^{-4}	100
5.3	Number of errors that can be corrected with FER= $2 \cdot 10^{-6}$ for quasi-cyclic $\gamma_v = 4$, $\rho_c = 8$ and $N=1296$	101
5.4	Polynomials used in the construction of rate-7/8 QC-LDPC code based on PDFs	102
5.5	FER performance of different decoders for QC-LDPC, $n=8168$ at $t=30$	103
5.6	The total number of operations for some LDPC decoders during a single iteration.....	107
5.7	Average overall complexity per frame (Real: real comparison or addition; Int.: integer comparison or addition; Bin.: binary operation)	111
5.8	QC-MDPC (12288,3072,220); complexity $\times 10^7$	112
5.9	Work factor of ISD for Goppa code (1024, 524, 50) McEliece cryptosystem	117
5.10	Peter's level security for QC-MDPC codes	118

Abbreviations

AES	<i>Advanced Encryption Standard</i>
ALT	<i>Approximate Lower Triangular</i>
ARQ	<i>Automatic Repeat Request</i>
AWGNC	<i>Additive White Gaussian Noise Channels</i>
BCH	<i>Bose-Chaudhuri-Hocquenghem</i>
BEC	<i>Binary Erasure Channel</i>
BER	<i>Bit Error Rate</i>
BF	<i>Bit-Flipping</i>
BP	<i>Belief-Propagation</i>
BSC	<i>Binary Symmetric Channel</i>
CCA	<i>Chosen Ciphertext Attacks</i>
DES	<i>Data Encryption Standard</i>
DLP	<i>Discrete Logarithm Problem</i>
DVB	<i>Digital Video Broadcast</i>
ECC	<i>Elliptic Curve Cryptography</i>
ECDLP	<i>Elliptic Curve Discrete Logarithm Problem</i>
FAID	<i>Finite Alphabet Iterative Decoders</i>
FER	<i>Frame Error Rate</i>
FPGA	<i>Field-Programmable Gate Arrays</i>
GBA	<i>Generalized Birthday Algorithm</i>
GDBF	<i>Gradient Descent Bit Flipping</i>
GRS	<i>Generalized Reed-Solomon</i>
IDES	<i>International Data Encryption Algorithm</i>
IEEE	<i>Institute of Electrical and Electronic Engineers</i>
IFP	<i>Integer Factorization Problem</i>
IMWBF	<i>Improved Modified Weighted Bit-Flipping</i>
IPEG	<i>Improved Progressive Edge Growth</i>
ISD	<i>Information Set Decoding</i>
IVRG	<i>Intrinsic-Value Random Generator</i>
LDPC	<i>Low-Density Parity-Check</i>
LFSR	<i>Linear Feedback Shift Register</i>
LLR	<i>Log Likelihood Ratio</i>
LS	<i>Latin Square</i>
MAJ	<i>Majority</i>
MAP	<i>Maximum-a-Posteriori</i>
MDPC	<i>Moderate Density Parity-Check</i>
M-GDBF	<i>Multi Gradient Descent Bit Flipping</i>
MIF	<i>Modified Inverse Function</i>
ML	<i>Maximum Likelihood</i>
MP	<i>Message Passing</i>
MS	<i>Min-Sum</i>

MUDRI	<i>Multiple Decoding attempts and Random re-Initializations</i>
MWBF	<i>Modified Weighted Bit-Flipping</i>
NBP	<i>Normalized Belief Propagation</i>
NGDBF	<i>Noisy Gradient Descent Bit Flipping</i>
OTM	<i>One-Time Pad</i>
PBF	<i>Probabilistic Bit Flipping</i>
PDFS	<i>Pseudo Difference Families</i>
PEG	<i>Progressive Edge Growth</i>
PGDBF	<i>Probabilistic Gradient Descent Bit Flipping</i>
PGP	<i>Pretty Good Privacy</i>
QC	<i>Quasi-Cyclic</i>
RAM	<i>Random Access Memory</i>
ROM	<i>Read Only Memory</i>
RS	<i>Reed-Solomon</i>
RSA	<i>Rivest, Shamir, Adleman public-key cryptography algorithm</i>
S/MIME	<i>Secure/Multipurpose Internet Mail Extensions</i>
SALT	<i>Systematic Approximate Lower Triangular</i>
SNR	<i>Signal-to-Noise Ratio</i>
SPA	<i>Sum-Product Algorithm</i>
SSH	<i>Secure Shell</i>
TBBF	<i>Two-Bit Bit Flipping</i>
TG	<i>Tanner Graph</i>
TMP	<i>Trusted Platform Module</i>
TSO	<i>Trapping Set Ontology</i>
WBF	<i>Weighted Bit-Flipping</i>
WF	<i>Work Factor</i>

Author's publications

Journals

- [A1] O. Al Rasheed, P. Ivanis, and B. Vasic, "Fault-tolerant probabilistic gradient-descent bit flipping decoder," *IEEE Communications Letters*, vol. 18, no. 9, pp. 1487-1490, September 2014. (IF=1.268)
- [A2] S. Brkic, O. Al Rasheed, P. Ivanis and B. Vasic, "On Fault Tolerance of the Gallager B Decoder Under Data-Dependent Gate Failures," *IEEE Communications Letters*, vol. 19, no. 8, pp. 1299-1302, August 2015. (IF=1.268)
- [A3] O. Al Rasheed, D. Radovic and P. Ivanis, "Performance analysis of iterative decoding algorithms for PEG LDPC codes in Nakagami fading channels," *Telfor Journal*, vol. 5, no. 2, 2013, pp. 97-102, November 2013.
- [A4] O. Al Rasheed, S. Brkic, P. Ivanis and B. Vasic, "Performance Analysis of Faulty Gallager-B Decoding of QC-LDPC Codes with Applications," *Telfor Journal*, vol. 6, no. 1, pp.7-11, November 2014.

Conferences

- [B1] O. Al Rasheed, P. Ivanis, "Complexity and Performance of QC-MDPC code-based McEliece Cryptosystems," in *Proceedings IEEE TELSIKS 2015*, Nis, Serbia, October 14th-17th, 2015, pp. 209-216.
- [B2] P. Ivanis, O. Al Rasheed and B. Vasic, "MUDRI: A fault-tolerant decoding algorithm," in *Proceedings IEEE International Conference on Communications (ICC)*, London, 8-12 June 2015, pp. 4291- 4296.
- [B3] O. Al Rasheed, D. Drajić, P. Ivanis and G. Djordjevic, "Complexity of the McEliece Cryptosystem based on GDBF Decoder for QC-LDPC Codes," in *Proceedings International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST 2014)*, vol. 2, June 2014, Nis, Serbia, pp. 321-324.
- [B4] O. Al Rasheed and P. Ivanis, "Analiza bit flipping dekodera LDPC kodova realizovanih pomocu nepouzdanih komponenti," *INFOTEH-JAHORINA*, vol. 13, March 2014, pp. 403-407.
- [B5] O. Al Rasheed, S. Brkic, P. Ivanis and B. Vasic, "Performance analysis of faulty Gallager B decoding of QC-LDPC Codes," in *Proceedings of 21st Telecommunication Forum (TELFOR 2013)*, November 2013, Belgrade, Serbia, pp. 323-326.
- [B6] O. Al Rasheed, D. Radovic, P. Ivanis, "Performances of Progressive Edge-Growth LDPC codes in Nakagami fading channel," in *Proceedings of 20st Telecommunication Forum (TELFOR 2012)*, November 2012, Belgrade, Serbia, pp. 560-563.

Abstract

Most of the public key cryptosystems depend on the hardness of either the factorization or the discrete logarithm problems. However, computation both of them can be done in polynomial time on a quantum computer. Therefore, there exists an urgent need for alternative cryptosystems that resist attackers designed on quantum computers. Code-based cryptography is one of the most promising alternatives for post-quantum cryptography where its security depends on the problem of decoding unknown error-correcting codes, which is known to be \mathcal{NP} -hard. McEliece cryptosystem belongs to this class of crypto-schemes and is based on the difficulty to decode an unknown linear code. Encoding and efficient decoding of a certain code in the presence of a fixed number of random errors are the key idea of McEliece cryptosystem. The main advantages of this system are fast encryption and decryption procedures, while the main drawback is the large public key size when the Goppa codes are originally used. For this reason, McEliece cryptosystem has not stood up to RSA for practical applications, i.e., it was assumed to be impractical for many applications. However, attempts to reduce the public key size by using alternative codes, while maintaining the same security levels, lead to promising results.

This thesis focuses on the QC-LDPC/MDPC code-based McEliece cryptosystems. QC-LDPC/MDPC codes are one of the most promising families of codes to replace the Goppa codes that still preserve the desired security properties of the cryptosystem. The drastic reductions in the public key size can be achieved by these codes. Security of the McEliece cryptosystem is directly determined by inability to detect the properties of codes from the intercepted sequences and the performance of an applied decoder. If the decoder has the advantage of being able to correct more errors, then more errors can be inserted during the encryption process, which have strong positive impact on the security. In addition, achieving low decoding failure rates is critical in McEliece cryptosystem such that the information bits can be extracted and sending the requesting retransmission can be avoided, which can help an attacker to detect the information bits.

As long as that the decoding process is the most complex part in code-based cryptography, the selection of an efficient decoding algorithm is crucial to the overall performance. The complexity of the iterative decoding algorithms for LDPC codes only grows linearly in the code length. There are several iterative decoding algorithms that can be applied in the same LDPC codes with different performance and complexity. It is well-known

that the Sum-Product Algorithm (SPA) has a superior performance but complexity of this algorithm is very high and decoding speed is limited.

The decoders proposed in the thesis achieve the good performance, comparing with the state-of-the art soft decoding algorithms, with lower implementation complexity. We discuss the possible use of Probabilistic Gradient Descent Bit Flipping (PGDBF) and Multiple Decoding attempts and Random re-Initializations (MUDRI) decoders which achieve a significant advantage in terms of performance with limited complexity.

The proposed decoding algorithm for LDPC/MDPC codes, as we will see, outperforms its competitors both in terms of computational complexity as well as decoding failure rate. A great flexibility in designing LDPC/MDPC decoding algorithms provides a possibility to improve the security level in McEliece cryptosystems.

Rezime

Većina algoritama sa javnim ključevima svoju sigurnost zasniva na težini faktorizacije brojeva na proste činioce ili problemu izračunavanja diskretnog logaritma. Ipak, oba pomenuta problema su rešiva u polinomijalnom vremenu korišćenjem kvantnih računara. Iz ovog razloga, javila se potreba za razvojem alternativnih kriptosistema koji uspešno odolevaju napadima izvedenim pomoću kvantnih računara. Dobro rešenje za post-kvantnu kriptografiju predstavljaju kriptografski algoritmi zasnovani na teoriji zaštitnih kodova, pri čemu se njihova sigurnost zasniva na problemu dekodovanja nepoznatih zaštitnih kodova, a poznato je da je to NP-kompletni problem. Tipičan predstavnik ove klase kriptografskih algoritama je McEliece kriptosistem. Kodovanje i efikasno dekodovanje određenog koda u prisustvu fiksnog broja slučajnih grešaka predstavlja ključnu ideju McEliece kriptosistema. Osnovne prednosti ovog sistema su velika brzina šifrovanja i dešifrovanja, dok osnovnu manu predstavlja veličina javnog ključa, posebno u originalnoj varijanti algoritma sa Goppa kodovima. Iz ovog razloga, McEliece kriptosistem nije smatran adekvatnom zamenom za RSA, tj. smatran je nepraktičnim za razne primene. Ipak, pokušaji da se redukuje veličina javnog ključa korišćenjem drugih klasa kodova, uz zadržavanje istog nivoa sigurnosti, doveli su do znatno unapređenih rešenja.

Ova teza pre svega se bavi McEliece kriptosistemima baziranim na QC-LDPC/MDPC kodovima. QC-LDPC/MDPC kodovi predstavljaju jednu od najpogodnijih familija kodova kojima se mogu zameniti Goppa kodovi a da se zadrži isti nivo sigurnosti kriptosistema. Primena ovih kodova dovodi do drastičnog smanjenja veličine javnog ključa. Sigurnost McEliece kriptosistema direktno je određena nemogućnošću da se odrede osobine koda iz presretnutih sekvenci, kao i performansama primenjenog dekodera. Ako dekođer ima sposobnost ispravljanja većeg broja grešaka, tada se više grešaka može utisnuti tokom procesa šifrovanja, što ima pozitivan uticaj na ostvareni nivo sigurnosti. Pored prethodno navedenog, postizanje male verovatnoće neuspešnog dekodovanja je kritično za uspešno funkcionisanje McEliece kriptosistema, da bi se izbegle retransmisije i tako otežalo napadaču da detektuje informacione bite.

Pošto je dekodovanje najsloženiji korak u procesu dešifrovanja, izbor efikasnog algoritma dekodovanja je ključan za performanse kompletnog kriptosistema. Kompleksnost iterativnog algoritma dekodovanja LDPC kodova raste linearno sa dužinom kodne reči. Postoji nekoliko iterativnih algoritama dekodovanja koji se mogu primeniti na isti LDPC kod,

od kojih svaki obezbeđuje različite performanse i svakog odlikuje drugačiji nivo kompleksnosti. Dobro je poznato da algoritam sumiranja i proizvoda (*Sum-Product Algorithm - SPA*) ima superiorne performanse, ali kompleksnost ovog algoritma je velika i brzina dekodovanja je ograničena.

Dekoderi predloženi u ovoj disertaciji postižu dobre performanse, uporedive sa standardnim algoritmima sa mekim odlučivanjem, uz značajno nižu kompleksnost implementacije. U disertaciji je razmotreno moguće korišćenje probabilističkog Gradient Descent Bit Flipping algoritma (PGDBF) i algoritma sa višestrukim pokušajima dekodovanja sa slučajnom reinicijalizacijom (*Multiple Decoding attempts and Random re-Initializations - MUDRI*), čija primena rezultuje značajnim poboljšanjem performansi uz ograničenu kompleksnost.

Predloženi algoritam dekodovanja za LDPC/MDPC kodove, kao što će biti pokazano, prevazilazi performanse drugih predloženih rešenja kako u pogledu kompleksnosti dekodera tako i u pogledu postignutog nivoa verovatnoće greške. Velika fleksibilnost u dizajniranju algoritama za dekodovanje LDPC/MDPC kodova rezultuje mogućnošću da se unapredi stepen sigurnosti McEliece kriptosistema.

Chapter 1

Introduction

Cryptography has been associated with design of algorithms that aspire to find solutions for security issues such as confidentiality, authenticity and integrity of sensitive data. Actually, cryptography stands as a defense line against any potential abuse in communication systems. Cryptography has become a significant research area, which is guided by the requirements of advanced information and communication technology. The primary objective of cryptography is to assure the confidentiality of sensitive data. Cryptosystems are designed to fulfill this purpose. A cryptosystem converts the information message, i.e., the plaintext, into encrypted message, i.e., the ciphertext, and allows singular recovery of the plaintext from the ciphertext by procedure called the decryption. A possible cryptosystem employs large keys, designed for the encryption and decryption algorithms. They are chosen in such a way that their evaluation by exhaustive search is impractical for an attacker.

The cryptosystems can be widely divided in two classes: (i) *asymmetric (public key-based)* and (ii) *symmetric (secret key-based)* systems, as it is illustrated in Figure 1.1.

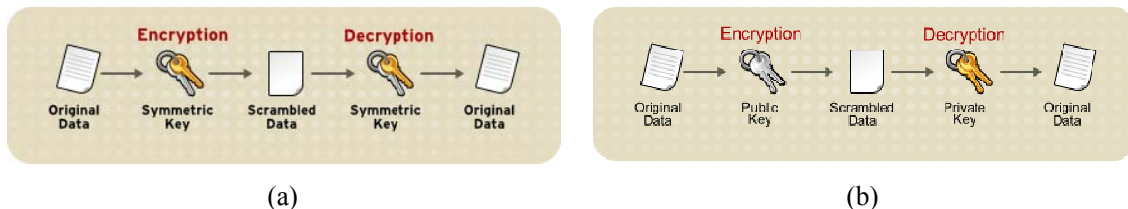


Figure 1.1. Symmetric (a) and asymmetric (b) cryptosystems

In the symmetric cryptosystem, the same key is used for the encryption and decryption procedures. The private key must remain secret and cannot be disclosed to a person or persons who are not intentional receivers of an encrypted message. In the case when the private key becomes compromised, any message encrypted using that key might be compromised. The private key cryptosystem is sometimes referred as the *symmetric key system*, since the same key is used in both ends of the system. In this kind, two main problems appear when the private key-based system is used: (i) *key generation* and (ii) *key distribution*. The system requires that $\binom{\mathcal{K}}{2}$ secret keys need to be distributed over a secure communication channels for \mathcal{K} communications partners, i.e., each information flow needs different secret key. Moreover, the secure communication channels can be difficult to achieve. Common examples of symmetric key encryption schemes are One-Time Pad (OTP) [1],[2], the *Data Encryption Standard (DES)* originally specified in [3], *International Data Encryption Algorithm (IDEA)* [4] and the *Advanced Encryption Standard (AES)* [5]. DES with 56-bit key was the most widely used symmetric-key encryption algorithm, and even after thirty years of crypto-analysis, the most practical attack remains exhaustive key search. Triple DES (or 3DES) uses the same algorithm, applied three times with different keys giving it an effective key length of 128 bits [6]. The AES is intended to replace the DES and Triple-DES. Its primitive is (usually) based on it representing a keyed pseudo random permutation. For near term AES-128 is advised for using and for long term it will be AES-256. Symmetric key encryption algorithms have the fastest implementations in hardware and software. Thus, they are very well suited to the encryption of large amount of data.

1.1. Public key cryptosystems

Whitfield Diffie and Martin Hellman [7] first introduced the notion of the *public-key cryptosystem* in mid-seventies. In their cryptosystem, different keys are assigned to the sender and receiver, which solve the problem of private keys distribution, which exists in the symmetric cryptosystems. Mathematical one-way functions are used to generate many of the pair keys. These functions need to be easy to implement in order to create the keys, but hard to invert.

The public-key encryption schemes are basically slower than symmetric-key encryption algorithms. Symmetric key operations are often based on low-level bit

manipulation primitives while public-key operations are often based on exponentiation of large integers. In addition, public keys must have many more bits than symmetric keys that achieve the same level of security. For this reason, public-key encryption is commonly used in practice for encryption of small data items and/or for transport of keys, subsequently used for data encryption by symmetric-key algorithms. Therefore, symmetric key and public key cryptography complement each other to provide cryptosystems used in practice.

Authentication and digital signatures are very important applications of public key cryptography. For instance, Pretty Good Privacy (PGP), first released in 1991, is a software package that provides encryption and authentication for e-mail and file storage applications. It uses Rivest, Shamir and Adleman (RSA) public key cryptosystems [8] for key transport and IDEA for bulk encryption of messages. Similar schemes are used for the Secure/Multipurpose Internet Mail Extensions (S/MIME) secure e-mail standard. Disk encryption systems like Microsoft's Bitlocker also uses a similar approach on systems equipped with Trusted Platform Module (TMP) chips. The bulk encryption of the data on the disk uses AES and then encrypted using RSA. The SSH (or Secure Shell) network protocol widely used for remote connections to Unix-like operation systems can also use public key cryptography for logging in operation.

In this context, *RSA*, *Rabin*, *ElGamal* and *Elliptic Curve* are the most famously used public-key cryptosystems [8-12]. The security of the known cryptosystem depends on the hardness of a number of theoretic problems such as *integer factorization problem* (IFP) and the computations of *discrete logarithm problem* (DLP) in finite fields or in groups of points on an elliptic curve.

RSA is a public key encryption system designed by Rivest, Shamir, and Adleman in 1978, as a signature and encryption scheme. It has received a lot of attentions and it is commonly known how to choose its parameters that guarantee the security against the best known attacks using current computer platforms. In this algorithm, both encryption and decryption are executed using exponentiation. The security of RSA system comes from the difficulty associated with factoring large prime numbers, which are incorporated into public and private keys. Thus, the ability to decipher plaintext from the use of a public key and ciphertext is the equivalent of factoring two large primes. The commonly used key size for RSA is 1024-bits, due to the progress on the factorization problem, it is expected that changing the key size is required. Therefore, the key sizes that are considered to be secure today are even longer.

In 1985, Koblitz and Miller suggested the use of the elliptic curves to design public key cryptography systems [11], [12]. An elliptic curve cryptography (ECC) based on elliptic curve theory that can be utilized to produce the faster, smaller and more efficient cryptographic keys instead of the traditional method of creation, which are based on the products of very large prime numbers. Equations based on elliptic curves have a characteristic that is useful for cryptography purposes: they are relatively easy to accomplish, and extremely hard to reverse. An ECC is used in the most advanced technologies as TLS, PGP and SSH on which advanced web and new IT world are based. In addition, ECC is based on the intractability of the Elliptic Curve Discrete Logarithm Problem (ECDLP) that is much more difficult to defy compared to IFP, at the equivalent key lengths. For comparison between RSA and ECC, it is extensively thought that the same level of security 128-bit, it can be achieved using ECC with the key size is 256 bits while RSA needs 3072-bits [13]. Efficient ECC hardware implementations for curves can be found in [14-16] which all give good performance at moderate resource requirements.

In 1998, Hoffstein, Pipher and Silverman introduced NTRU cryptosystem [17]. This system depends on the algebraic structures of certain polynomial rings. The *hard problem* which NTRU depends on it can be defined as a problem of finding a short vector in a given lattice. A mixing system used in the encryption procedure relies on polynomial algebra and reduction modulo two numbers. The decryption procedure uses un-mixing system whose validity depends on elementary probability theory. For comparison between NTRU, RSA and ECC, the procedure NTRU cryptosystem is significantly faster than the RSA and it is two orders of magnitude faster than ECC, but the keys of NTRU are longer than ECC keys [18].

Another important class of the public key cryptography is *code-based cryptography*. Typically, error-correcting codes are used for increasing the reliability of communication transmission, which is in all its forms subjected to the channel noise. Starting from observation that the Goppa codes allow fast and efficient error correction, Robert J. McEliece in 1978 proposed a public key cryptosystem based on the coding theory [19]. In fact, the cryptography system proposed by McEliece is not only the first encryption scheme based on the coding theory but over the years remain resistant to the attacks attempting to recover the secret key [20]. Soon after the McEliece's publication, a number of researchers adapted his idea and started to refine and improve the original concept. The Niederreiter cryptosystem is known as a coupled version of McEliece cryptosystem where Generalized Reed-Solomon (GRS) codes are used [21] and the new update version can be found in [22] where the Goppa codes are used. It uses the smaller public key sizes and equivalent security. They notice some

drawbacks of McEliece's scheme, among of them, the necessity for a *large key*, for 128-bit security the best known attacks force a key size is around 192192 bytes which prevent its widespread adaption. In the McEliece cryptosystem, the message is represented as a binary vector and mapped to a unique transmitted message by the redundancy bits. The proportion of useful data (non-redundant) transmitted in each codeword is called the transmission code rate. The transmission code rate used in the classical McEliece cryptosystem is 0.5 which is considered low compared with the full code rate in RSA. Worth mentioning that the McEliece cryptosystem is free to use counter to another public-key cryptosystems such as NTUR cryptosystem where is patented by the company *NTUR Cryptosystems* [23].

1.2. Post-quantum cryptography

Wineland [24] said: "*Perhaps the quantum computer will be built in this century. If so, it will change our lives in the same radical way as the classical computer transformed life in the last century*". Physicists are prophesying that large-scale quantum computers may be available in the next 15 to 20 years due to the technological progress. The quantum computers are a new class of computers which essentially differ from the classical computers. In quantum computing, the fundamental unit can hold both 0 and 1 value at the same time; this is known as a superposition of two states. These quantum bits are known as *qubit*. The quantum information theory is considered as a suitable framework to describe their computational features rather than classical information theory.

The quantum algorithms can be simply defined as the sequence of the basic of manipulations of qubits, and these algorithms specifically required for the quantum computers to operate intelligible upon information processing, which cannot be run on classical computers. Regardless that the large quantum computers exist today or not, the mathematicians try to improve the algorithms that convenient to carry out the functions on this modern computer architecture during the period of time preceding. It has been shown that some issues can be solved in remarkably less running time by quantum algorithms in contrast to the traditional algorithms running on the classical electronics circuit hardware, as shown in Figure 1.2.

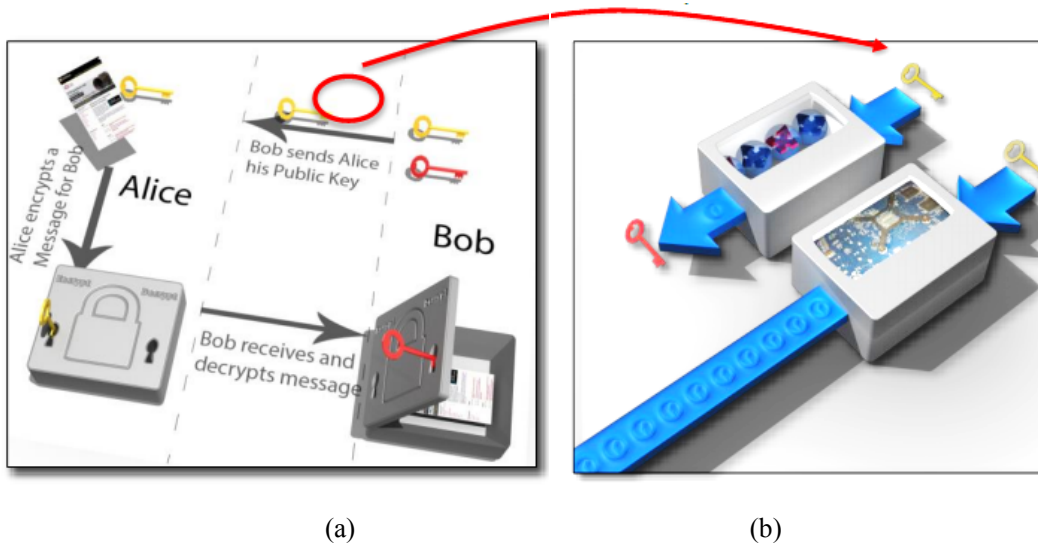


Figure 1.2. Eavesdropper obtains the public key from public directory (a); Quantum computer can break security by reverse computing private key faster than a classical computer (b)

The first algorithm for quantum computing introduced by Peter Shor in 1994 with great importance for effective implementations, and in 1996 he again considerably improved the original algorithm [25], [26]. Shor showed how to do so in a futuristic scenario. Actually, Shor's algorithm is considered as a probabilistic algorithm that divided into two parts, a classical part and a quantum algorithmic part. The original Shor's algorithm aims to decompose a non-negative composite integer N in its prime factors on a quantum computer. In 2001, Shor's algorithm was implemented on a 7-qubit quantum computer to factorize the number 15 by Chuang *et al* [27]. After ten years and using adiabatic quantum algorithms, the Chinese researches succeeded to factorize the number 143 [28]. The special feature of this algorithm is that its running time can be considered the smallest compared with any known algorithm and can be applied to solve the related DLP. However, in the past time there is a significant progress to solve the integer factorization and discrete logarithm problem using the classical computers [29-31].

"Is cryptography dead?"—Bernstein *et al.* asked in the beginning of the book [32]. In fact, the quantum computers will not end the secure of telecommunication system, but other public key cryptosystems require taking place of the famous cryptosystem as RSA and ECC.

It can be believed that the quantum computers have ability to solve problems in polynomial time and these problems are designed to be adamant for the classical computer. So, novel security model have to be improved to resist the quantum adversaries.

The *post-quantum cryptography* relates to cryptographic systems that are secure against attacks by conventional computers and withstand against attacks by all known

quantum algorithms which are applicable in future. Most of current symmetric cryptography algorithms like AES are considered to be relatively secure from attacks by quantum computers, by increase the key sizes of the symmetric cryptosystems. We say that AES-128 is a difficult for a classical computer to break while AES-256 would be difficult for a quantum computer. Therefore, the post-quantum symmetric cryptography will be similar to the current symmetric cryptography.

In the context of the quantum computers, fortunately, some classes of asymmetric cryptosystem can be considered as a post-quantum cryptography as shown in Figure 1.3. There are at least four classes of public key cryptosystem that survive according to [32]: hash-based as “*Merkle’s hash-tree signature system*”, code-based as “*McEliece encryption*”, lattice-based as “*NTRU encryption*” and multivariate-quadratic-equations as “*HFE signature scheme*” cryptosystems — to the best of our knowledge.

Today, two known quantum algorithms, Grover’s and Shor’s [25, 26, 33] are used to break the current state-of-the-art cryptosystems. In fact, there are no applicable with the same impact on the code-based schemes.

Bernstein used Grover’s quantum algorithms were applied on the McEliece cryptosystem in order to break it [36]. Fortunately, it was not able to break McEliece’s system or other code-based cryptosystems. Some adjustable parameters are suggested in the case of sturdy quantum computers like force the McEliece key size to quadruple. In [37] the authors argued that quantum computers have only a small impact on the McEliece public-key system, reducing the attacker’s decoding cost from 2^{140} to 2^{133} for a code with length 4096 and dimension 3556.

In short: McEliece cryptosystem, together with cryptographic schemes based on lattices, multivariate polynomials, or on hash functions, is one of the oldest public key cryptosystems and the oldest that is conjectured to be post-quantum secure.

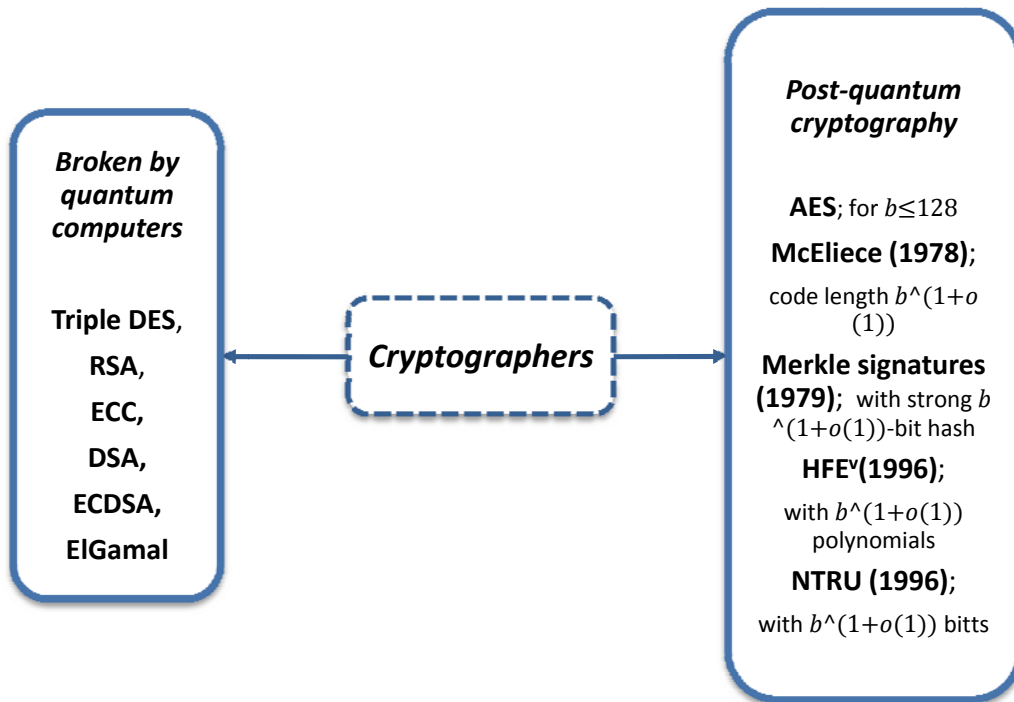


Figure 1.3. Post-quantum cryptography and broken cryptography by post quantum computers; b means 2^b ; $1+o(1)$ means something converges to 1 as $b \rightarrow \infty$

1.3. Key size problem

An appropriate key size of any cryptosystem is proportional to the security. The key size is chosen according to the all known attacks, which could be believed that it can constitute risk for the system. The size of key is measured in bits and increase the key size raises the difficulty of a brute-force attack exponentially. For instance, an average size of key in RSA cryptosystem must increase with the time when more efficient factoring algorithms are designed and when computers are getting faster. In 1978, the key of RSA was 200-digit long values — as the authors recommended, but in 2005, the largest (known) number forced by a general-purpose factoring algorithm was 200-digits (663 bits) [38]. At this time, the experts anticipate that the keys for these 1024 and 2048 bits long may be breakable in the near future. However, these choices considered as temporary solutions to confront the new attacks and development of technology. At all events, increase the key size of the cryptosystem leads to slow down of the encryption and decryption procedures. In addition, the memory increases which also must be taken into account.

On the other hand, the different sorts of cryptosystems demand widely different key lengths to preserve the security. Generally, the systematic cryptosystems require shorter keys

compared with the asymmetric cryptosystem as RSA which need longer key. Worth mentioning that ECC has the ability to keep its security with the key lengths similar to those of symmetric systems. For example, to provide security equivalent to AES, then RSA would have to range approximately between 3072 and 15000 bits long, which is so large that standard embedded hardware would be unable to provide reasonable levels of performance or throughput, while ECC key of 256 bits would provide at less than 1/2 the size as shown in Table 1.1 [13].

Table 1.1. Key size comparisons ECC, RSA and McEliece bit size requirements for AES to achieve different bit-security level

	AES-128 small	AES-192 medium	AES-256 large
Bit-security level	128	192	256
ECC	256	384	512
RSA	3072	7680	15360
McEliece	460 647	-	7 667 885

In this thesis we focus on code-based cryptosystem, especially on the McEliece cryptosystem. It is based on the problem of decoding a general linear code. Its security based on the hardness of correcting errors in linear codes and the hardness of distinguishing the public code description from random. It is considered immune to quantum computer attacks [20]. Therefore, there is a clear perspective to be a candidate for post-quantum encryption scheme.

In McEliece cryptosystem, the most of operations in the encryption and decryption procedures are based on the binary linear codes which can be implemented efficiently in dedicated hardware. Moreover, factoring problem (FP) and DLP based cryptosystem require computationally expensive multi-precision integer arithmetic while the operations on binary codes do not need it, so it is beneficial for small computing platforms. McEliece cryptosystem performs great in comparative criterion and has a time complexity of $O(n^2)$, while RSA has a time complexity of $O(n^3)$. Therefore, the longer key sizes in the McEliece cryptosystem lead to higher complexity but, in fact, it is very low compared with RSA, even when Goppa codes are used. Table 1.2 shows that the McEliece system and its Niederreiter [22] dual version (homologue) display lower encryption and decryption complexity than the

asymmetric schemes as RSA with respect to the same security level. Additionally, in some applications the profits of very fast encryption and decryption are surpassed by the costs of communicating and storing these keys. In the case of the post-quantum cryptography algorithms, larger key sizes are required than in commonly used "pre-quantum" public key algorithms. In [40] introduces the parameters that satisfy a minimum security level of the McEliece cryptosystem.

Table 1.2. Parameters of the original and modified McEliece cryptosystem

	Key Size ^a	Rate	k ^b	C_{enc}/k ^c	C_{dec}/k ^d	Ref
McEliece (1024,524)	67072	0.51	524	514	5140	[51]
Niederreiter (1024,524)	32750	0.57	284	50	7863	[51]
RSA,1024-bit mod. Public exp.17	256	1	1024	2402	738112	[51]
QC-LDPC	6144	0.75	12288	658	4678	[50]

^a Expressed in bytes

^b Information block length (bits)

^c Number of binary operations per information bit for encryption

^d Number of binary operation per information bit for decryption

Several attempts have been made to propose alternatives to the classical Goppa codes. The main motivation is to additionally reduce the size of the public and private keys. The recent solutions seek to use cyclic or quasi-cyclic codes (QC) in code-based public key cryptosystems. The main idea of this method consists in considering quasi-cyclic generator matrices of a code, so that rather than giving the whole generator matrix, it is only sufficient to give a few rows from which the whole matrix can be derived, i.e., only some rows of the generator matrix are stored in the memory (Block RAM), and the other rows can be obtained by simply rotating the store values. In [41] they used a very large set of subcodes of a given BCH code as a large set of Goppa codes with key of size 12 Kbits for length 2047 and 20 Kbits for length 4095, and for the modern cryptosystem to assure a good security the 500 Kbits needed for length 2047. The drawbacks of presented versions are that (i) the public code comes from a primitive BCH code i.e., an adversary is able to enumerate all of BCH codes for the suggested parameters and (ii) the permutation (used to hide the secret code) is too restrictive [42].

One such alternative is based on random Low Density Parity Check (LDPC) codes which is considered as a first implementation with McEliece cryptosystem [43].

Unfortunately, random-based LDPC codes do not allow to decrease the key size and the system will be unsecure. In [44] again exploited the QC-LDPC codes to overcome those limitations and the authors propose a public key size that is about 48 Kbits to achieve a security equivalent of 2048 bit RSA. This is a reduction of key size by a magnitude of ten from the original McEliece cryptosystem while maintaining the advantage of fast encoding and decoding as shown in Table 1.2. Some weaknesses of this method are summarized in [42], [45]. However, the authors responded in [46] when QC-LDPC codes are used in McEliece cryptosystem and recommended to change values of some parameters to avoid the known attacks. Recently, a variant of McEliece encryption scheme from Moderate Density Parity-Check (MDPC) codes and another from quasi-cyclic MDPC codes are introduced in [47]. Public key size in bits to achieve 128 bits of security is 9856 and private key is 19714 bits. The McEliece cryptosystem based on LDPC\MDPC codes use the suitable decoding algorithms for trying to correct all intentional errors added to the transmitted messages. The Sum-Product Algorithm (SPA) [48] and the Bit-Flipping (BF) [49] are the decoding algorithms usually used in the modified McEliece cryptosystem. The SPA based on the soft decision and requires high complexity to obtain the best performance while BF based on the hard decision and requires very low complexity with poor performance.

The main goal of this thesis is design low complexity decoder suitable for LDPC and MDPC codes that will result in additional reduction of complexity for the same security level. Dramatic increase in the dimension of the LDPC\MDPC codes employed in the McEliece cryptosystem is needed to guarantee the high level security in the quantum-computing. It is known that the size of the public keys increases linearly with the code dimension (length). Therefore, designing novel low complexity decoders employed in the McEliece cryptosystem is significant, since can reduce the complexity of a hall system. The decoders proposed in the thesis give high flexibility to approach the low complexity decoders with the relatively good performance. In this moment we discuss the possible use of Probabilistic Gradient Descent Bit Flipping (PGDBF) and Multiple Decoding attempts and Random re-Initializations (MUDRI) decoders which give a significant advantage in terms of complexity and performance.

The complexity of these decoding algorithms will be analyzed in details and the corresponding McEliece cryptosystem. From the point of view the security, impact of applications of these decoding algorithms on the security will be also given and practical implementations will be considered.

1.4. Contributions and Outline

The rest of the thesis is organized as follows:

Chapter 2 provides the background on McEliece cryptosystem based on error-correcting codes. It describes the original McEliece cryptosystem, depiction of its advantages and the main drawbacks. Some previous attempts of reducing the key size of this system are introduced. We focus on the QC-LDPC and QC-MDPC codes which are functional to reduce the key size of the McEliece cryptosystem with respect to the level security.

Chapter 3 provides the background on class of linear block codes, LDPC codes and iterative decoding algorithms. In this chapter, the definitions, notations and bipartite graph representation for LDPC codes are introduced. Also, we give short explanation for some constructions of LDPC codes and encoding process. It is then followed by short survey of the message-passing decoding algorithms. Furthermore, their advantages and disadvantages will be full presented through comparison.

In Chapter 4 some LDPC decoding algorithms designed for AWGNC are presented. After the introduction and comparisons of these decoding algorithms, the motivations of our work can be seen more obviously. We propose a new decoding approach based on Gradient Descent Bit Flipping (GDBF) algorithm. The GDBF algorithm will be adapted to work over BSC. This chapter discusses in detail analysis performance of the novel algorithm PGDBF over BSC. Also we provide the MUDRI decoder to approach to the better performance. Furthermore, performance of these decoders under faulty hardware components and their robustness are discussed. The rest of this chapter we introduce implementation of PGDBF decoder on hardware.

Chapter 5 introduces the most dangerous attacks against the McEliece cryptosystem reported in the literature. Performances of some LDPC\MDPC codes used in McEliece cryptosystem are evaluated. Security and complexity of the McEliece cryptosystem based on QC-MDPC and our proposed decoders are evaluated.

In chapter 6, we will present the conclusions on the proposed decoding algorithms and some notions for possible future works.

Chapter 2

McEliece Cryptosystem

Code based cryptography is one of the most promising applications for post quantum cryptography. To date, the McEliece cryptosystem is considered as a post-quantum cryptographic scheme, i.e., it is believed that to be renitent to quantum computers [32]. The McEliece cryptosystem is classified as asymmetric cryptographic system, and relies on the hardness of decoding a linear block code without any visible structure for the key. Two keys are used in this system, private key and public key, and they are related to the generator matrix of a binary linear block code. The important advantage of the McEliece system is its fast encryption and decryption procedures, which require significantly lower number of operations compared with other system.

The McEliece cryptosystem depends on employment a huge family of codes, i.e., a collection of codes with the same length and dimension, where choice of a code (from the set) can be unpredictable. Efficiency of the McEliece system depends on hardness to find the solution of the decoding problem for the third part (intruder).

In McEliece system cryptography, two keys are used: the first (private) key is a generator matrix of the code and the second (public) key is a matrix transmitted and assigned as a public code, i.e., the code which generated by the public generator matrix. The low cost of the encryption and decryption procedures in the McEliece system gives code based system a special significance. Furthermore, it can be observed that the speed of these procedures is high when compared to other state-of-the-art public key algorithms, like RSA or ECC cryptography system. Decoding of a long linear block code, with no information about structure of a code, leads to difficult decoding and is known to be an *NP complete problem* [52], where the classical decoding problem is assumed to be hard on average, i.e., it is

difficult to find a codeword of an arbitrary linear code with minimum distance to a given vector. Therefore, potential use of error correcting codes have cryptography comes natural. Although the McEliece cryptosystem is less complex then other public key systems, there are some communications systems where a less complexity is strongly desirable. Thus, simpler methods used in practice where were believed to be secure adequate in the sense that no possible attacks were investigated.

2.1. Goppa codes

McEliece cryptosystem is the first code-based cryptosystem, originally proposed using Goppa codes. For appropriate system parameters, McEliece system with Goppa codes still remains resistant to the all known cryptanalysis techniques but leads to very large public keys. Two main reasons are to use Goppa codes: (i) *diversity in choosing the desired code gives a large number of potential public keys*, and (ii) *efficient decoding algorithm capable to correct up to a certain number of errors*.

A number of algebraic codes are considered as subfield subcodes of generalized Reed-Solomon (GRS). These codes are defined as *alternant codes*, and include Goppa codes and Bose-Chaudhuri-Hocquenghem (BCH) codes. In [53], [54] Goppa introduced a family of linear block codes defined as Goppa codes. The Goppa code $\Gamma(L, g(x))$ is defined by Goppa polynomial $g(x)$, which is a polynomial of degree t over the extension field $GF(p^m)$ for p a prime, and an accessory subset L of $GF(p^m)$

$$g(x) = g_0 + g_1x + \dots + g_tx^t = \sum_{i=0}^t g_ix^i, \quad (2.1)$$

$$L = \{\alpha_0, \alpha_1, \dots, \alpha_{n-1}\} \subseteq GF(p^m), \quad (2.2)$$

which are not zeros of $g(x)$. A Goppa codes that are able to correct t errors is defined as a set of vectors $c = (c_0, c_1, \dots, c_{n-1})$ over $GF(p)$, such that

$$\sum_{i=0}^{n-1} \frac{c_i}{x - \alpha_i} \equiv 0 \pmod{g(x)}. \quad (2.3)$$

The set L is called the support of the code. The polynomial $g(x)$ with degree ≥ 1 is said to be *irreducible* over the field $GF(2^m)$ if it cannot be represented as a product of two polynomials (with coefficients of $GF(2^m)$) of nonzero degree. The code is called as *irreducible Goppa code* if the polynomial $g(x)$ is irreducible, and the code can have

maximum length $n = p^m$ when the support of the code contains all the elements of $GF(p^m)$. The parameters of a code are the size n , dimension k and minimum distance d where the dimension of the code satisfies $k \geq n - mt$ and the minimum distance of the code satisfies $d \geq t + 1$. Since any degree- t irreducible polynomial generates a various irreducible code, we can say the number of various irreducible Goppa codes with the same parameters and correction capability is very high. For each irreducible polynomial of t degree over $GF(2^m)$ there is a binary irreducible Goppa code with maximum length $n = 2^m$. The matrix of the Goppa codes has no intrinsic structure and the parity-check matrix of a Goppa codes has the following form

$$\mathbf{H} = \begin{bmatrix} \frac{1}{g(\alpha_0)} & \frac{1}{g(\alpha_1)} & \cdots & \frac{1}{g(\alpha_{n-1})} \\ \frac{\alpha_0}{g(\alpha_0)} & \frac{\alpha_1}{g(\alpha_1)} & \cdots & \frac{\alpha_{n-1}}{g(\alpha_{n-1})} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\alpha_0^{t-1}}{g(\alpha_0^{t-1})} & \frac{\alpha_1^{t-1}}{g(\alpha_1)} & \cdots & \frac{\alpha_{n-1}^{t-1}}{g(\alpha_{n-1})} \end{bmatrix}. \quad (2.4)$$

The parity-check matrix of Goppa codes has no essential structure, and thus there is no way to expedite its storage. Consequently, a whole parity-check matrix with $r \cdot n$ bits needs to be stored. Goppa codes employ a fast polynomial time decoding algorithm [55] and are easy to generate but hard to find. Any irreducible polynomial over a finite field \mathbb{F}_2^m of $n = 2^m$ elements can be used to create a Goppa code, but the generator matrices of Goppa codes are nearly random [56]. For any fixed length n , there are many different Goppa codes. Though the exact number of Goppa codes, given n and t , is not known, Ryan and Fitzpatrick [56] found a way to calculate upper bounds, which are exact for some small parameters. For example, the upper bound for the number of Goppa codes of length 128 which are able to correct at least 10 errors is approximately 1.04×10^{15} , while the upper bound for Goppa codes of the same length able to correct at least 15 errors is approximately 2.38×10^{25} . In fact, the number of Goppa codes increases exponentially with the length of the code and the degree of the generating polynomial [56].

2.2. Description of the McEliece cryptosystem

In the McEliece version of the proposed system, Bob chooses a polynomial of degree t at random and verifies that it is irreducible. The probability that a randomly chosen polynomial is irreducible is about $1/t$. After that, Bob computes his secret key as the $k \times n$ systematic generator matrix \mathbf{G} of the linear block code over \mathbb{F}_p . In addition, another two randomly

matrices are used as parts of the private key of Bob: a dense $k \times k$ non-singular “scrambling” matrix \mathbf{S} such that the codewords of the public code should appear random to an attacker, i.e., sends the chosen generator matrix to another one, and an $n \times n$ permutation matrix \mathbf{P} to reorder the coordinates.

The public key is saved in the public directory and any one can get it. Finally the public key is given by this form

$$\mathbf{G}' = \mathbf{S} \cdot \mathbf{G} \cdot \mathbf{P}. \quad (2.5)$$

So, the matrices \mathbf{P} and \mathbf{S} seek to hide the structure of \mathbf{G} , in the other words, try to disguise a decodable linear code in a way that the construction of the code can no longer be deduced. The produced public key \mathbf{G}' is used to carry out an encryption procedure on the message planned for the receiver Bob and is the generator matrix of a linear code with the same rate and minimum distance as that generated by \mathbf{G} . Alice uses the public key from the public directory to encrypt the message. The data transmission is assumed to be error-free and errors are purposefully injected into a codeword as a part of the encryption process. The original message is divided into k -bit blocks and the encrypted message \mathbf{x} is given by

$$\mathbf{x} = \mathbf{u} \cdot \mathbf{G}' + \mathbf{e} = \mathbf{c} + \mathbf{e}, \quad (2.6)$$

where \mathbf{u} is one of the information blocks and an \mathbf{e} is a local vector added at the transmitter and it can be controlled to the weight t and length n . If an encrypted message is transmitted over the noisy channel and additional error influences on the message, so it is important to take into account this influence that decoding algorithm can correct all the errors in order to allow the reconstruction of the original message.

When Bob receives the encrypted message from the channel, he uses an iterative inverse procedure to obtain the plaintext. He computes $\mathbf{x}' = \mathbf{x} \cdot \mathbf{P}^{-1}$ where \mathbf{P}^{-1} is the inverse of the permutation matrix \mathbf{P} (that coincides with its transpose)

$$\mathbf{x}' = \mathbf{x} \cdot \mathbf{P}^{-1} = (\mathbf{u} \cdot \mathbf{S} \cdot \mathbf{G} \cdot \mathbf{P} + \mathbf{e})\mathbf{P}^{-1} = \mathbf{u} \cdot \mathbf{S} \cdot \mathbf{G} + \mathbf{e} \cdot \mathbf{P}^{-1}. \quad (2.7)$$

Therefore, \mathbf{x}' is a codeword of the secret Goppa code chosen by Bob affected by the error vector $\mathbf{e} \cdot \mathbf{P}^{-1}$ of weight t . Remarkably, the permutation by \mathbf{P}^{-1} of the positions of the errors in the intentional error vector does not lead to change weight of the vector t , in the other words, it does not impact on the decoding process to correct the errors. Using Patterson’s algorithm, Bob can hence correct the errors and recover $\mathbf{u}' = \mathbf{u} \cdot \mathbf{S}$, and \mathbf{u} is obtained from an easily computed through multiplication by \mathbf{S}^{-1} .

The cryptography system is *secure against adaptive chosen ciphertext attacks CCA2* [57] (as a strongest security notion) if an attacker has no benefit in deciphering a given ciphertext. In addition, it is considered to be *indistinguishable* in the CCA2-model if an attacker has no profit in conclusion for a given ciphertext and two plaintexts which of them were encrypted. For example, encryption of the same message twice creates two various ciphertexts which can be distinguished to extract the original message as long as it is an improbable that the same added positions of error can be repeated for the two messages twice. According to an above decryption, the McEliece cryptosystem is not immune for the effect of chosen-ciphertext attacks [58]. That means it is unable to achieve the "*IND-CCA2 security*". However, [59] and [37] suggested making the system CCA2-secure by using the idea of scrambling the message inputs. In the other words, to ruin any relations of two dependent messages that can be exploited by an adversary. There are several versions of the McEliece cryptosystem which achieve CCA2 security and the γ -conversion of Kobara and Imai considered as the best candidate for quantum-immune encryption scheme with the best information rate [59], [57].

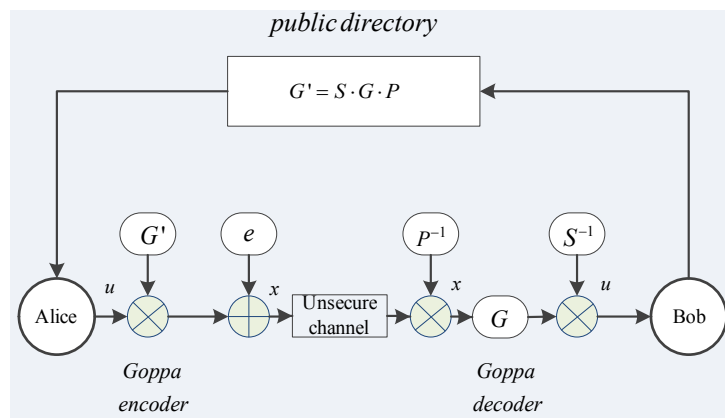


Figure 2.1 Original procedure of the McEliece cryptosystem with Goppa codes

The McEliece cryptosystem has some advantages as it is remarkably faster than RSA two or three orders of magnitude and it is resistant against attack, where there is no efficient algorithm to extract the parameters of the Goppa codes from a generator matrix of a permutation equivalent code and no chance for brute force attack on the McEliece cryptosystem because of the large possibilities for G , S and P :

- Number of irreducible polynomials of degree $t = 50$ is about 10^{149} for $n = 1024$.
- Possible random $k \times k$ scrambling matrices \mathbf{S} are 2^{524^2} for $k = 524$.
- Possible $n \times n$ permutation matrices \mathbf{P} are $1024!$

For all that, there are main limitations for this system as the large size of the public key. Without a doubt, any standard PC can contain millions of such keys and CPUs can smoothly treat the McEliece cryptosystem. It is not the case for small devices. The original version of the McEliece cryptosystem used Goppa codes with $n = 1024$, $k = 524$ and the small rate $R = 0.5$. For comparison, the McEliece uses the public key with 67072 bytes long while RSA with equivalent security uses public key with 256 bytes long. Some suggestions were made to change McEliece original method [41], [61-63], most of them alternate the Goppa codes with other family codes. However, most of them have been subsequently broken or inefficient compared to McEliece original system.

2.3. Modifications of McEliece cryptosystem based on LDPC codes

The classical McEliece cryptosystem based on the Goppa codes remains secure to this date but leads to very large public keys. There are a number of ways to overcome the drawbacks of the McEliece cryptosystem. They are based on reducing the representation of a linear code besides the matrices of the linear transformations. However, this usually leads to serious security flaws, since security of a system partially depends on error correction capability of Goppa codes. The chosen code families must achieve some conditions to ensure the security: (i) the codes family must to be large enough to keep away from any enumeration, (ii) a generator matrix of a permutation equivalent code must be obscure that no permit recover the secret code from the public code, i.e., does not give any information about the structure of the secret code, and (iii) the code has efficient algorithm to correct the codewords, i.e., the receiver is able to read the transmitted ciphertext over the unsecure channel.

We provide a brief overview of state-of-the-art modified McEliece-based cryptosystems. One of the modifications for the McEliece cryptography interested on raise an immunity of the system by choosing the parameters that would maximize resistance against some of known attacks [51], [64], [65]. On the other hand, Niederreiter introduced a randomized variant of the McEliece system [21] and proposed using Generalized Reed-Solomon (GRS) codes instead of Goppa codes, which it was proved in [66] that this choice was insecure. In 1994, Sidelnikov proposed employing Reed-Muller codes [63]. His scheme was afterwards broken in [34]. In 1996, Janwa and Moreno studied the use of algebraic

geometric codes [67], which was broken in [68]. Gaborit proposed using BCH codes in [41] and [69] where it was proved that this variant of McEliece cryptosystem can be broken. McEliece was modified in [70], [71] by considering quasi-cyclic or quasi-dyadic versions of Goppa codes where the key size can be reduced with a factor $t = \tilde{O}(n)$ smaller than generic Goppa codes. However, the added structure permits a drastic reduction of the number of unknowns in algebraic attacks. The kind of these attacks can be prevented by picking smaller cyclic or dyadic blocks for these schemes such that number of unknowns of the algebraic system will be increased.

Because of the sparse of the parity check matrix of the LDPC codes, many attempts have been studied to exploit this property to overcome those limitations of the first version of the McEliece cryptosystem, where their storage sizes increase linearly in the code length.

The first implementation for random-based LDPC codes with McEliece is proposed in [43], where the receiver, Bob, receives the encrypted message from the transmitter, Alice. Bob chooses a parity check matrix \mathbf{H} in the set Γ with high cardinality. The matrix \mathbf{H} is a sparse matrix with no structural properties, i.e. it is randomly constructed. To disguise the construction of the matrix \mathbf{H} , Bob chooses an $r \times r$ non-singular random *transformation matrix*, \mathbf{T} , and the public key is given by $\mathbf{H}' = \mathbf{T} \cdot \mathbf{H}$. The matrix \mathbf{T} keeps the sparse of the parity check matrix \mathbf{H} , so \mathbf{H} and \mathbf{H}' describe codes have the same distance properties. The impact of the matrix \mathbf{T} on the public key is appearance of many short length cycles in graph representations of \mathbf{H}' , which makes it high density matrix, and the attacker Eve is unable to exploit the public key to decrypt the secure code. In [43], the authors described that when \mathbf{T} is a sparse matrix, then there is a high chance to attack the secure code and recover the private key from the public key. Therefore, we need to have a dense \mathbf{T} and dense \mathbf{H}' to avoid this kind of attacks. Actually, these codes do not allow to reduce the key size and the system will be insecure (\mathbf{T} is a sparse), where the dual attack of the secret code is able to entirely break it with restricted complexity, unless when some its parameters are guardedly chosen. Because of these reasons, the random-based LDPC codes system is no longer vindicated. The main problem can be resolved with structured LDPC codes to obtain the sparse public keys.

The parity check matrix of the QC-LDPC codes as we will see in the Chapter 3, in general, can be entirely depicted by a single row of them (as the first row). On the other hand, the families of these codes can be modeled with a diversity of rates, even meaningfully larger than 0.5. For these purposes, exploit the QC-LDPC to overcome some drawbacks of the first attempt must be ensured that neither the public code nor its dual code allow any too sparse

representation. In the context of the code-based McEliece cryptosystem and with QC-LDPC codes, an interesting and a simple case of the “circulant block” form which is defined as a circulant matrices class. A circulant block is completely described by its first row (or column). A special class of quasi-cycles codes having the parity-check matrix in the form of a single row of circulant blocks. The private key is formed by the sparse parity check matrix \mathbf{H} , randomly chosen, having the following form

$$\mathbf{H} = [\mathbf{H}_0 | \mathbf{H}_1 | \dots | \mathbf{H}_{n_0-1}], \quad (2.8)$$

where each matrix \mathbf{H}_i , $i = 0 \dots n_0 - 1$, is a $p \times p$ circulant matrix given by

$$\mathbf{H}_i = \begin{bmatrix} m_0 & m_1 & m_2 & \dots & m_{(p-1)} \\ m_{(p-1)} & m_0 & m_1 & \dots & m_{(p-2)} \\ m_{(p-2)} & m_{(p-1)} & m_0 & \dots & m_{(p-3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_1 & m_2 & m_3 & \dots & m_0 \end{bmatrix}. \quad (2.9)$$

The first row of the matrix \mathbf{H}_i , $i = 0 \dots n_0 - 1$, can be denoted as h_i , $i = 0 \dots n_0 - 1$, which contains the positions of symbols 1, where $h_i = (m_0, m_1, \dots, m_{p-1})$ and it can be written as $h_i(x) = (m_0 + m_1x + \dots + m_{p-1}x^{p-1})$. The matrix \mathbf{H} is quasi-cyclic low density parity check code if each \mathbf{H}_i is sparse. It can be noted that, if all the h_i vectors have disjoint sets of differences modulo p , the cycles with length-4 in matrix \mathbf{H} disappear in its associated Tanner graph. The secret code can be easily constructed by randomly selecting n_0 vectors h_i with such property. This permits us to obtain large families of codes with identical parameters [72].

The matrix \mathbf{H} has a length $n = n_0 \cdot p$, dimension $k = k_0 \cdot p$ and redundancy $r = p$ are adopted, where n_0 is a small integer (e.g., $n_0 = 3, 4$), $k_0 = n_0 - 1$, and p is a large integer (on the order of some thousands), then the code rate is $R = (n_0 - 1)/n_0$. When the row and column weight of \mathbf{H}_i is equal to a constant, then the code is regular, otherwise is irregular code. Without loss of generality, we can consider that \mathbf{H}_{n_0-1} is non-singular (has full rank), so a systematic generator matrix (in reduced echelon form) for the code is $\mathbf{G} = [\mathbf{I} | \mathbf{P}]$, where \mathbf{I} represents the $k \times k$ identity matrix and

$$\mathbf{P} = \begin{bmatrix} (\mathbf{H}_{n_0-1}^{-1} \cdot \mathbf{H}_0)^T \\ (\mathbf{H}_{n_0-1}^{-1} \cdot \mathbf{H}_1)^T \\ \vdots \\ (\mathbf{H}_{n_0-1}^{-1} \cdot \mathbf{H}_{n_0-2})^T \end{bmatrix}, \quad (2.10)$$

where superscript T denoted transposition. It is important to select an efficient algorithm to calculate the inverse of a circulant matrix $\mathbf{H}_{n_0-1}^{-1}$ and it can be found in [72].

Two methods can achieve the desired purposes during QC-LDPC codes:

- *Like in the original McEliece cryptosystem, by using permutation equivalent private and public codes, and choosing a private code such that it does not admit any too sparse representation.*
- *By using non-permutation equivalent private and public codes, such that the sparse nature of the private code is lost in the public code.*

2.3.1. Permutation equivalent private and public codes

In this system, Bob chooses the private key as a parity-check matrix \mathbf{H} of a QC-LDPC code to accomplish the decoding process and correct the secure word with high efficiency. The QC-LDPC codes have good error correction capability and it is necessary to correct all the intentional errors in the secure codeword. Bob again selects two matrices: $k \times k$ non-singular random scrambling matrix \mathbf{S} and a random permutation matrix \mathbf{P} , that also considered as his secret key, and the public key is given as the first original version except for the fact that they used the inverse of \mathbf{S} and \mathbf{P} , and these procedures do not affect the properties of matrix, density and randomness. We have

$$\mathbf{G}' = \mathbf{S}^{-1} \cdot \mathbf{G} \cdot \mathbf{P}^{-1} = \mathbf{S}^{-1} \cdot \mathbf{G} \cdot \mathbf{P}^T, \quad (2.11)$$

where \mathbf{G} is a the corresponding generator matrix in systematic form, such that $\mathbf{H} \cdot \mathbf{G}^T = \mathbf{0}$. In the last relation, the private and the public codes are permutation equivalent. In the other word, their codewords coincide, except for a permutation of their bits. Of course, the permutation equivalent between the private key and the public key admits an Eve to search low weight codewords in the dual of the secret code, thus aiming at recovering \mathbf{H} , which can be achieved by Information Set Decoding (ISD) algorithms [39], as we will see in Chapter 5.

The encryption procedure is repeated, i.e., Alice divides the message into blocks with length k bits, gets the public key from the public directory, encrypts the message with public

key and adds the intentional errors with weight t . The decrypted procedure of this system is the same decrypted procedure of the original version. It is noticeable that to keep the QC nature in the public matrix, the matrices \mathbf{S} and \mathbf{P} must be designed by $k_0 \times k_0$ and $n_0 \times n_0$ blocks of $p \times p$ circulants, respectively. Generally, the eavesdropper Eve is unable to conclude the matrix \mathbf{H} from familiarity of the public key, and will not be able to correct the secure codeword by the efficient LDPC decoding algorithm. Therefore, Eve resorts to use some generic decoding algorithm, like ISD to correct the intentional errors.

2.3.2. Non-permutation equivalent private and public codes

The permutation equivalent permits the attacker to conclude the parity-check matrix \mathbf{H} , by searching for low weight codewords in its dual code that makes the matrix \mathbf{H} insecure. To solve this problem, in [72] introduced the solution through hiding the sparse structure of the secret code in a public code whose dual does not contain codewords with too low weight. As in the previous system, Bob chooses the private parity check matrix \mathbf{H} and computes the corresponding generator matrix \mathbf{G} in the systematic form. The remaining parts of the private key are the scrambling matrix \mathbf{S} with a $k \times k$ dimensions and another matrix \mathbf{Q} is a sparse random non-singular transformation matrix with $n \times n$ dimensions that the public code prevents sparse characteristic matrices. The matrix \mathbf{Q} has average row and column weight equal to m . The matrix \mathbf{S} instead is dense, with average row and column density around 0.5. If the matrix \mathbf{S} is sparse, then the average of row and column weight is $s \ll k$.

On the other hand, in the case of QC-LDPC codes, the matrices \mathbf{S} and \mathbf{Q} will have circulants structure where \mathbf{S} with $k_0 \times k_0$ and \mathbf{Q} with $n_0 \times n_0$ block of $p \times p$ circulants, so the public matrix preserves the QC nature. The public key is given by

$$\mathbf{G}' = \mathbf{S}^{-1} \cdot \mathbf{G} \cdot \mathbf{Q}^{-1}. \quad (2.12)$$

In the case when a suitable CCA2-secure conversion of the system used, then the public key \mathbf{G}' can be used in the systematic form and the scrambling matrix and permutation matrix are eliminate and $\mathbf{G}' = \mathbf{G}$.

Alice encrypts an intended message \mathbf{u} to $\mathbf{x} = \mathbf{u} \cdot \mathbf{G}' + \mathbf{e} = \mathbf{c} + \mathbf{e}$, where \mathbf{e} is an intentional errors vector with length n and weight t . In this system, the private key must be able to correct a number of errors $t' > t$. Bob runs the inverse procedure to get the corrected codeword as

$$\mathbf{x}' = \mathbf{x} \cdot \mathbf{Q} = \mathbf{u} \cdot \mathbf{S}^{-1} \cdot \mathbf{G} + \mathbf{e} \cdot \mathbf{Q}. \quad (2.13)$$

It is noticeable, that the codeword is affected by an error vector $\mathbf{e} \cdot \mathbf{Q}$ which has average weight $t' = t \cdot m$. By an efficient LDPC decoding algorithm, Bob can recover the original message after the decoding process and then through multiplication by \mathbf{S} .

In [73] introduced an efficient class of transformation matrices that have a limited propagation effect on the intentional error vectors such that permit to obtain the better disguise for the private key into the public one with a controlled error propagation effect.

The main purpose is avoiding the dual of the public code from containing very low weight codewords even when the private code is an LDPC code with very low weight codewords in its dual. Let assume that the secret LDPC matrix is \mathbf{H} , which defines a dual code with low weight codewords corresponding to its row, so this matrix results the codewords with weight row is ρ_c , where ρ_c is the row weight of the matrix \mathbf{H} , so

$$\begin{aligned} \mathbf{H} \cdot \mathbf{Q}^T \cdot \mathbf{G}'^T &= \mathbf{H} \cdot \mathbf{Q}^T \cdot (\mathbf{S}^{-1} \cdot \mathbf{G} \cdot \mathbf{Q}^{-1})^T \\ &= \mathbf{H} \cdot \mathbf{Q}^T \cdot (\mathbf{Q}^{-1})^T \cdot \mathbf{G}^T \cdot (\mathbf{S}^{-1})^T \\ &= \mathbf{H} \cdot \mathbf{G}^T \cdot (\mathbf{S}^{-1})^T = \mathbf{0}. \end{aligned} \quad (2.14)$$

Therefore, the matrix $\mathbf{H}' = \mathbf{H} \cdot \mathbf{Q}^T$ is a valid parity-check matrix for the public code and this matrix has a more density compared with a matrix \mathbf{H} . The weight of the rows of $\mathbf{H}' = \mathbf{H} \cdot \mathbf{Q}^T$ is about the product of their row and column weights, that is, $\rho_c \cdot m$. In fact, when two sparse binary vectors are summed together, it is very likely that the weight of the resulting vector is about the sum of their weights. Therefore, if the value of m is properly chosen, the minimum weight of the codewords in the dual of the public code can be made sufficiently high to make dual code attacks unfeasible. Therefore, Bob must be able to correct a number of errors that is about m times larger than the weight t of the intentional errors vectors added by Alice.

Therefore, it is proposed to adapt \mathbf{Q} matrices with block diagonal form, in which the circulant blocks along the main diagonal have row/column weight m

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{Q}_{n_0-1} \end{bmatrix}. \quad (2.15)$$

Due to its low density, matrix \mathbf{Q} is among the weakest components of the new system. In fact, an attack could be tempted on each of the first $n_0 - 1$ blocks along the main diagonal,

noted by \mathbf{Q}_i , $i \in [0, n_0 - 2]$, if \mathbf{Q}_i is known, the attacker could multiply the block in the i -th column of \mathbf{G}' by \mathbf{Q}_i , thus obtaining the i -th column of \mathbf{S}^{-1} (due to the reduced echelon form of \mathbf{G}) [69]. However, such approach would require, at least, $\binom{p}{m} \cdot p \cdot m$ binary operations, where $p \cdot m$ binary operations have been considered for multiplication by \mathbf{Q}_i .

Using LDPC codes instead Goppa codes in the McEliece cryptosystem creates some differences that should be taken into account. Error correction capability of Goppa codes associated with the Goppa polynomial of degree t (can correct the full design number of errors), while for LDPC codes is generally unknown. In addition, transformation matrix \mathbf{Q} has effect on the error correction performance, so design QC-LDPC codes with high error correction capability is required. On the other hand, the decoder at the receiver has an important role to extract the secure code with high probability. Despite all these procedures to guarantee correction the codeword, there is a small probability that Bob fails to recover the secret message. There are different methods can be executed to impede that occurrence. Because of the huge number of codes that can be achieved by the random-based approaches, as random difference families [44], Bob can carefully select QC-LDPC codes which have the relatively good properties, rather than the randomly selects the codes. The request retransmission can be used when Bob cannot correct the secure word, by sending the request to resend the same secure word but with different error positions or with lower weight of the intentional error vector \mathbf{e} . An Automatic Repeat Request (ARQ) protocol can allow Alice to know whether Bob is able to correct all the errors she has randomly introduced or not. In principle, this exposes the system to message-resend attacks, but simple modifications of the cryptosystem are known which prevent these attacks without significant drawbacks [59], [74].

2.4. Moderate Density Parity Check (MDPC) codes

Some kinds of attacks can risk the McEliece cryptosystem either by message attacks based on standard decoding algorithms or key recover attacks by finding low weight codewords in the dual of the public code, as we will see in Chapter 5. In addition, inappropriate choices of the parameters in the case of QC-LDPC maybe lead to non-secure system. Of course, using sparse matrices \mathbf{S} and \mathbf{Q} allows reducing the encryption and decryption complexity but according to the OTD attack which can recover the secret key [69] and Baldi advised to use density matrix \mathbf{S} instead sparse matrix to avoid this attack.

In [47] it is introduced new method that can protect the system from those attacks. Moderate Density Parity Check (MDPC) codes used instead LDPC codes where these codes. They have higher weight of rows and columns of the parity check matrix. Transformation matrix \mathbf{Q} that is used in the case of LDPC codes can be replaced with the permutation matrix \mathbf{P} , similarly as in the original McEliece with Goppa codes. Increase the density of the parity check matrix in the most situations leads to a greatly degraded error correction performance when compared to the standard LDPC codes performance. However, MDPC codes are still adequately valid to keep from the impressiveness of the standard decoding algorithms. The authors in [47] suggested employ Bit Flipping (BF) decoding algorithm to correct the intentional error vectors. However, the high weight of the rows ρ_c in the parity check matrix and the highly existence of short-cycles in the Tanner graph maybe lead to an increase number of iterations. To decrease the number of iterations, some modifications are inserted to the BF decoding algorithm [47].

There is only one variance between LDPC and MDPC codes where the LDPC codes have small constant weight of rows (typically less than 10) while for MDPC codes the weight of rows in the scale of $O(\sqrt{n \log n})$. In the form of single row of the circulant blocks and for regular codes, these codes can be formed as LDPC codes, i.e., QC-MDPC codes and for given parameters p , n_0 and γ_v , the number of various random matrices in the form (2.8) is

$$N_{MDPC} = \frac{1}{p} \binom{p}{\gamma_v}^{n_0}. \quad (2.16)$$

To keep away from some kinds of attacks in the case of LDPC-code base cryptography, it is necessary to avoid presence low weight codewords in the dual code of the public code. Eve is able to recover the secret key or a key equivalent to the secret one by using ISD techniques (as we will see in Chapter 5). An attacker can find a set of low weight codewords in the dual of the public code, and through any decoding algorithm like BP or perform syndrome decoding to evaluate the intentional error vector. The row weight value ρ_c should be chosen such that the message recovery and key recovery attacks are of the same order of complexity [47]. The ρ_c is selected according to

$$\rho_c = (1 + o(1)) \sqrt{\frac{n \ln n \ln(1 - k/n)}{\ln(k/n)}}. \quad (2.17)$$

The number of errors which can be corrected by the Bit Flipping algorithm is of order

$$t = (1 + o(1)) \frac{n}{4\rho_c} \ln \left(\rho_c \left(1 - k/n \right) \right). \quad (2.18)$$

Since the parity check matrix of these codes is already quite dense, then there is no need for a transformation matrix \mathbf{Q} , so a standard permutation matrix can be used, as in the original McEliece cryptosystem. In addition, using MDPC code in McEliece cryptosystem instead LDPC code considered as a good solution to avoid some kinds of attacks and solve the problem how to select parameters m and t to guarantee the security and successful decoding process. On the other hand, if the public code is MDPC, it contains moderate weight codeword in the dual space of the public code, and if Eve is able to find those codewords, than she is also able to decrypt the ciphertext, but the already cited ISD techniques (that are always NP), are not able to find moderate weight codewords in reasonable time.

2.4.1. Procedure of the MDPC/QC-MDPC code-based McEliece

Here, a (n, r, ρ_c) -QC-MDPC codes are considered, such that the parity-check matrix has the form of (2.8), where $n = n_0 p$ is the code length, ρ_c is a row weight of the parity check matrix \mathbf{H} and $r = p$.

Procedure of the McEliece cryptosystem based on the MDPC/QC-MDPC can be described as follows:

Step 1: a parity check matrix \mathbf{H} is constructed, where $\mathbf{H} \in \mathbb{F}_2^{r \times n}$ of a t -error correction (n, r, ρ_c) -QC-MDPC code. The matrix \mathbf{H} considered as a private key in this system. The generator matrix \mathbf{G} can be constructed as $\mathbf{G} = [\mathbf{I} | \mathbf{P}]$ where $\mathbf{G} \in \mathbb{F}_2^{(n-r) \times n}$ in row reduced echelon form and \mathbf{P} has form of (2.10). The generator matrix \mathbf{G} considered as a public key of the McEliece cryptosystem and saved in the public directory. Using QC form of MDPC leads to significantly decrease of storage requirements, by storing the first rows of the circulant blocks.

Step 2: Alice uses the public key \mathbf{G} to encrypt the message $m \in \mathbb{F}_2^{(n-r)}$ and obtains the ciphertext $x \in \mathbb{F}_2^{(n)}$, an intentional error vector $e \in \mathbb{F}_2^n$ of $\text{wt}(e) \leq t$ which is generated randomly and added to the ciphertext.

Step 3: by LDPC/MDPC decoding algorithm ψ_H which specified to the sparse parity-check matrix H to refine the ciphertext form the error according to $mG \leftarrow \psi_H(mG + e)$ and the plaintext m can be obtained from the first $(n - r)$ positions of mG .

The description of this system does not use the permutation matrix P and scrambling matrix S which are considered in the original McEliece cryptosystem. The public key matrix G in the systematic form can be used without loss of security according to the CCA2-secure conversion [57]. The QC-MDPC variant has a public-key of size $(n - r)$ and the MDPC variant of size $(n - r)$.

2.5. Implementations of McEliece cryptosystems

The McEliece system rarely used in real world applications because of its large public key size. We here provide some examples of McEliece cryptosystem on embedded hardware and software platforms, which illustrate its applicability.

The McEliece cryptosystem was considered impracticable on such small and embedded systems because of the large size of the private and public keys. Almost, all devices could only supply a few hundred bytes of RAM and ROM which was a strict restriction for application designers. One of the solutions that can be used in the case when the public key is too large is retransmission of the public key for each encryption [75]. However, this solution has a passive effect on the transmission rate and the speed of the encryption procedure.

The first implementations of the McEliece cryptosystem introduced by [76] on a popular 8-bit AVR microcontroller and the Xilinx Spartan-3AN 1400 FPGA. They used the code-length 2048 for an estimated 80-bit security where an external memory is used for the public key structures, and the private key (parity check matrix) stored in 192 kB Flash memory. Another implementation introduced for the same code-length (2048) where spent 84% of slices and 50% of BRAMs (2700Kb) [77].

One of the important implementation of the McEliece cryptosystem was introduced in [78]. The low cost implementations that use QC-MDPC codes instead of Goppa codes are also presented in this chapter. The implementation is on 8-bit AVR along with the Xilinx Virtex-6 XC6VLX240T FPGA. For FPGA implementation, the public and secret key do not have to be stored in external memory as it was necessary in previous FPGA implementations of McEliece using Goppa codes. Only 4800 and 9600 bits are used for the public and secret

key at 80-bits security where in the beginning of the procedure the first row of the key is given and by rotating by one bit position of the row yields the next row and so forth. The QC-MDPC McEliece requires 0.59 k Byte which is only a fraction of the 100.5 k Byte public key of [77]. The results of this implementations show that QC-MDPC McEliece is better than other currently used public key systems with the same security levels, for more details show Table 2.1. and Table 2.2.

Table. 2.1 Performance comparison of microcontroller implementations for different public encryption schemes

Scheme	Platform	Time ms/op	Ref
Goppa code, $n = 1024, t = 40,$ 62-bit sec.	Infineon SLE76CF5120P, 16-bit CPU Enc@33MHz, Dec@33MHz	970 ms, 690 ms	[79]
Goppa code, $n = 2028, t = 50,$ 102-bit sec.	Infineon SLE76CF5120P, 16-bit CPU Enc@33 MHz, Dec@33 MHz	1390 ms, 1069 ms	[79]
Goppa code $n = 2048, t = 27,$ 80-bit sec.	AVR ATxMega 192, 8-bit CPU Enc@32MHz, Dec@32MHz	450 ms, 618 ms	[76]
QC-MDPC $n = 9600, t = 84,$ 80-bit sec.	AVR ATxmega256A3, 8-bit CPU Enc@ 32MHz, Dec@ 32MHz	800 ms, 2700 ms	[78]
ECC-P160 (SECG)	ATMega128 @8MHz	203 ms (at 32MHz)	[76]
RSA-1024	ATMega128 @8MHz	20748 ms (at 32MHz)	[76]

Table. 2.2 Performance comparison of FPGA implementations for different public key encryption schemes

Scheme	Platform/ f	Time/Op	Ref
Goppa code, $n = 2048, t =$ 27, 80-bit sec	Spartan-3AN 1400 FPGA, Enc@150MHz, Dec@85MHz	1.07 ms, 10.82 ms	[76]
Goppa code, $n = 2048, t =$ 50, 102-bit sec	Xilinx Virtex-5, Enc@163MHz, Dec@163MHz	0.5 ms, 1.4 ms	[77]
QC-MDPC, $n = 9600, t =$ 84, 80-bit sec	Xilinx Virtex-6, Enc@351.3MHz, Dec@190.6MHz	0.14 ms, 0.86 ms	[78]
ECC-P160 (SECG)	Spartan-3 1000-4	5.1ms (at 32MHz)	[76]
RSA-1024	Spartan-3E 1500-5	51ms	[76]

Chapter 3

LDPC Codes and Iterative Decoding Algorithms

Error control coding is one of the most used concepts for increasing the reliability of communication transmission and to utilize capacity as much as possible. Shannon in his paper mentioned that only the long codes have capability possible to operate near capacity, but they are impractical, i.e. it cannot be practically encoded and decoded. In the 1960s, R. Gallager invented low-density parity-check (LDPC) codes, which are nowadays known as Gallager's codes [49]. Iterative algorithms also proposed in the Gallager's original paper achieve the capacity approaching performance but these algorithms need complicated calculations which make them unusable at that time. In the 1990s MacKay and others rediscover these codes [80]. Near-capacity performance of LDPC codes can be achieved in practice by decoding algorithms whose complexity is linear in the code length. These codes are used in many techniques of communication as encryption and compression. LDPC codes have proved their superiority over many other classes of linear block codes. They are used in many applications since their ability to correct errors of the received message and low decoding complexity is highly desirable in modern communication systems. Some wireless and wire-line communications standards that specify LDPC codes are, digital video broadcast (DVB-S2), WiMAX wireless (IEEE 802.16e) and 10 gigabit Ethernet (IEEE 802.3an).

LDPC codes are linear block codes, constructed by sparse parity check matrix \mathbf{H} . There are many methods to construct matrix \mathbf{H} in order to obtain the best performance and simple way to implement encoding and decoding procedures. Structural LDPC codes with intermediate block lengths are popular in recent research, noticeably the algebraic constructions which are shown to perform within a fraction a dB away from the Shannon limit. Several of these LDPC code constructions include the Reed-Solomon based codes [81],

array-based codes [82], as well as Quasi-Cyclic (QC) LDPC codes [83]. These codes share the same property that their parity check matrices can be written as a two-dimensional array of component matrices of equal size, each of which is a permutation matrix. The main advantage of these constructions principle compared to randomly constructed codes is that their encoding procedure is easier to implement.

The decoding algorithms of the LDPC codes fall into highly intensive research arena. The decoding algorithms of the LDPC codes are iterative and depend on graph which consists of check and variable nodes [84]. Under an iterative algorithm and when the noise level is below some fixed point, the bit error probability for LDPC codes (when length tends to infinity) is arbitrary small [49].

Generally, decoding algorithms are divided into two types: (i) soft decision as a Sum Product algorithm (SPA) which requires high complexity to obtain the best performance [60], [48] and (ii) a hard decision as Bit Flipping (BF) algorithm with very low complexity but it has poor performance [49]. All researches are seeking to reduce its computations complexity for platform implementations.

3.1. Linear Block Codes

We assume that transmitted message is presented in a binary format and before transmission the data bits are organized in order to be able to correct errors. A block coding is one of mechanisms for adding redundancy. Basically with the linear block codes, there are two types of streams (sequences) when sending messages over noisy channel by error correcting codes. A sequence of K data bits is mapped into N code bits where $N > K$. The sequence bits K is called a data word while the corresponding sequence of N bits is called a codeword. An encoder at transmitter can add $(N-K)$ bits to the original message as redundant bits by different ways. The *systematic codes* are especially interesting, where the block comprising K information bits is not changed in the codeword, and $N-K$ control bits are added, forming a codeword of length N . This code is denoted as (N, K) code and *code rate* is $R=K/N$. The receiver recovers the message information by inverse operation. This procedure is called a decoding process and after the error correction, an information message is recovered.

Block code has capability for correcting a noisy codeword, this ability has been attributed to minimum distance d_{min} . This parameter denotes the minimum number of components that are changed by the noisy channel in such a way that converts the original codeword into another codeword of the same code.

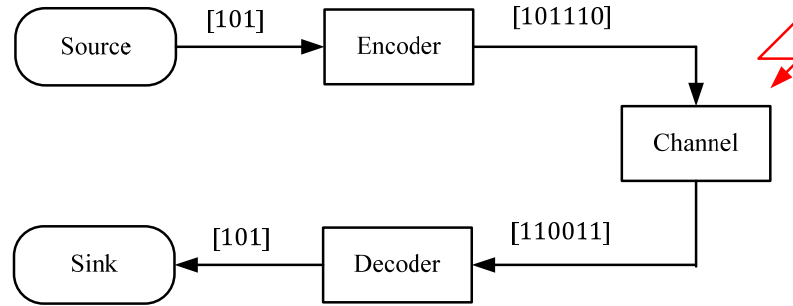


Figure 3.1. The basic channel coding model

In the other words, the error detection capability of the block code is determined by the minimum distance. If the number of errors is less than $(d_{min}-1)$ codeword cannot be converted into another codeword caused by the noisy channel. Linear code block can correct any noisy codeword with error pattern of weight $t = \lfloor (d_{min} - 1)/2 \rfloor$.

A linear code may be described in terms of a generator matrix \mathbf{G} or in term of a parity check matrix \mathbf{H} . LDPC codes are a class of linear block codes whose parity-check matrix has 1% or fewer 1 entries. If only binary codes are considered code can be described as a K -dimensional subspace \mathcal{C} of vector space \mathbb{F}_2^N of binary N -tuple over the finite field \mathbb{F}_2 . Given a $K \times N$ generator matrix \mathbf{G} , a codeword $\mathbf{x} \in \mathcal{C}$ is obtained by

$$\mathbf{x} = m \cdot \mathbf{G}, \quad (3.1)$$

where m is a binary row vector containing K bits. Accordingly, an $(N-K) \times N$ parity check matrix \mathbf{H} forms the null space \mathcal{C}^\perp so that $\mathbf{H} \cdot \mathbf{G}^T = \mathbf{0}$, where $(\cdot)^T$ denotes transposition and the dimension of matrix $\mathbf{0}$ (all element equal zero) are $(N-K) \times N$. It is obvious, because the subspaces are orthogonal each other. Therefore, instead of using generator matrix, at the receiving end, from the parity-check matrix \mathbf{H} should be found and verify to the received vector is in the subspace orthogonal to the subspace generated by \mathbf{H} . Solution of the above equation is very simple if matrix \mathbf{G} corresponds to a systematic code. In most designs, we need to transform \mathbf{H} into \mathbf{G} by using a transition matrix \mathbf{P} such $\mathbf{H} = [\mathbf{I}|\mathbf{P}]$ and $\mathbf{G} = [\mathbf{P}^T|\mathbf{I}]$, where \mathbf{I} is a unity matrix with dimensions $(N-K) \times (N-K)$.

Let a vector $\mathbf{x} = (x_1, x_2, \dots, x_N)$ denote a codeword of \mathcal{C} if and only if $\mathbf{x} \cdot \mathbf{H}^T = \mathbf{0}$ which means that every codeword \mathbf{x} is orthogonal to the rows of \mathbf{H} , where \mathbf{H}^T is the transpose of \mathbf{H} . In addition, a parity check matrix \mathbf{H} is used to verify if any N binary vector is a codeword that belongs to \mathcal{C} or not. For decoding the received word \mathbf{y} that can be any vector of the 2^N , the

following vector should be found $\mathbf{s} = \mathbf{y} \cdot \mathbf{H}^T$. It is called syndrome, dimension are $1 \times (N-K)$ and its element are parity checks. We can consider that variable nodes are the bits of the codewords. For example, if C is the binary linear code with the parity-check matrix

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}, \quad (3.2)$$

then C is the set of all binary 8-tuples that satisfy the four following equations expressed in the matrix \mathbf{H}

$$\begin{aligned} s_1 &= r_2 \oplus r_4 \oplus r_5 \oplus r_8 \\ s_2 &= r_1 \oplus r_2 \oplus r_3 \oplus r_6 \\ s_3 &= r_3 \oplus r_6 \oplus r_7 \oplus r_8 \\ s_4 &= r_1 \oplus r_4 \oplus r_5 \oplus r_7 \end{aligned} \quad (3.3)$$

To simplify the implementation on hardware and lower power consumption, a linear block LDPC codes are used. Considering that LDPC codes the best solution for many applications in the communication systems which used coding for error correction as transmission mode forward error correction (FEC), they have met a lot of interest to overcome their imperfection. In the next section, we introduce the basic concepts regarding LDPC codes, how they are constructed, encoded and in particular how they decoded with the various classifications.

3.2. Preliminaries

Following the known graphical representation of convolutional codes. Tanner in his pioneering work observed that it is possible to represent LDPC codes by a bipartite graph, later called Tanner graph or TG for brevity [84]. He introduced the principle of equivalence between the construction of codes by matrix \mathbf{H} and graphs and that the characteristics of a graph can be utilized to extract bounds on the minimum distance of the code. Each variable node corresponds to a code symbol of the codeword and each check node represents one check equation. The TG introduces an efficient way to describe iterative decoding algorithms for LDPC codes.

Let TG denote the Tanner graph of an (N, K) binary LDPC code C of rate $R=K/N$, as a linear block code, which consists of a set of N variable nodes $V(TG)=\{1, 2, \dots, N\}$ and a set of M check nodes $C(TG)=\{1, 2, \dots, M\}$. The parity check matrix \mathbf{H} is the bi-adjacency matrix of TG since it is composed by K linearly independent row vectors.

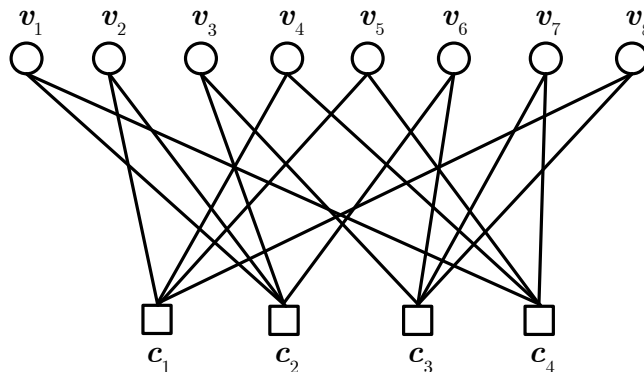


Figure 3.2. The TG for the parity check matrix $\mathbf{H}(8,4)$ given in (3.2)

Two nodes in TG are neighbors if there is an edge between them. An edge corresponds to the value 1 in the parity check matrix \mathbf{H} . The degree of a node v is the number of its neighbors and is denoted as γ_v . Graph TG is said to be γ -variable node regular if all variable nodes in V have the same degree, $\gamma_v = \gamma$. The degree of a check node c is ρ_c , and ρ -check-regular codes are defined analogously. We denote regular codes by (γ, ρ) . Otherwise, it's an irregular LDPC code. Degree of variable node γ_v determines ability of the LDPC codes to correct the errors. With high γ_v , node v can receive more information from its neighbors, which has a great opportunity for error correction during the decoding process. On the other hand, a check node can with high ρ_c helps its neighbors, variable nodes, which have low γ_v by sending useful information. For finite block length, the performances of irregular LDPC codes are better than those of regular LDPC codes in waterfall region. The sets of neighbors of nodes v and c are denoted as \mathcal{N}_v and \mathcal{N}_c , respectively. The length of the shortest cycles in the graph is called girth g of TG. The presence of cycles in the Tanner graph is not avoidable due to the restriction on the block length. All mechanisms for construction of LDPC codes seek to maximize the girth as much as possible. The negative impact of the short cycles in the bipartite graph is degraded when length of codes increases. However, avoidance of short cycles can be available for proportionately short-length LDPC codes when these codes are designed with an appropriate construction.

3.3. Encoding of LDPC codes

The LDPC codes can achieve near-capacity performance with iterative message-passing decoding and sufficiently long block sizes for random or pseudorandom constructions. An algebraic construction is one of the methods for constructing LDPC codes. This type is used for relatively short or medium long codes. When an application uses these code lengths, then it is preferred to use an algebraic construction since is no difference between them in the performance. However, there are many additional reasons that algebraic method is desirable in many applications. Construction principles of LDPC codes are evaluated in terms of error performance, as well as complexity of encoding and decoding operations, with is especially pronounced applications such as magnetic recording and optical communications. The performance of LDPC codes depends on the code constructions and decoding algorithms. There are many methods and each of them attempt to overcome problems typical for this class of codes such as short cycles and trapping set. Here we give short explanation for some constructions of LDPC codes which are used in our numerical results.

3.3.1. Progressive edge growth construction

The important LDPC code class is made by PEG method [85]. In this procedure the constructed graph created by adequately connecting variable and check nodes in an edge-by-edge manner. Based on a number of variable and check nodes and a variable - node - degree sequence, the optimum connection at the time is established. That way, the graph is updated an edge by an edge, taking care that girth is as large as possible. As a result, regular and irregular PEG LDPC codes can be produced. This is a very simple and efficient algorithm that provides flexibility in codeword length and code rate. In the improved algorithm IPEG [86], a check node is selected which maintains the highest degree of connectivity for the newly created cycles to the rest of the graph. That way these cycles will receive a great amount of information from the rest of the graph which will decrease their negative impact. The PEG LDPC codes are known to have the error capability performance as good as random codes in spite of they are composed by algebraic construction.

First, we determine N variable nodes $V = \{v_1, v_2, v_3, \dots, v_N\}$ and M parity check nodes $C = \{c_1, c_2, c_3, \dots, c_M\}$. Then, the degree distribution is defined as

$$\lambda(x) = \sum_{i \geq 2}^{\gamma_{\max}} \lambda_i x^i, \quad (3.4)$$

where λ_i presents fraction of edges emanating from variable nodes of degree i and γ_{\max} is a maximum degree of variable nodes. After that, by using density evolution, variable node degree sequence $\gamma_v = \{\gamma_1, \gamma_2, \gamma_3, \dots, \gamma_N\}$ has to be determined in non-decreasing order $\{\gamma_1 \leq \gamma_2 \leq \gamma_3 \leq \dots \leq \gamma_N\}$. The rest of the procedure will be explained in one illustrative example.

For example, $N=6$, $M=3$ and degree distribution $\lambda(x) = 0.8576x + 0.14237x^2$, by using density evolution [87] we get five variable nodes with degree two and one variable node with degree three. To illustrate the procedure of PEG we assume the PEG arrives at variable node v_6 to establish its edges, bipartite graph is given in Figure 3.3 and the parity check matrix is

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}. \quad (3.5)$$

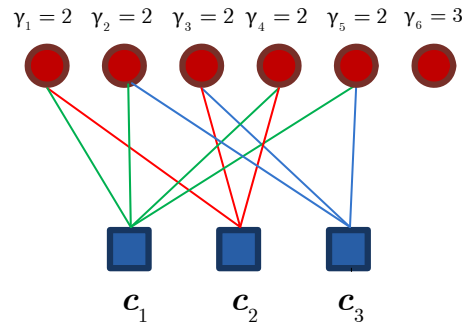


Figure 3.3. Bipartite graph before PEG procedure

Here, we define $\mathcal{N}_{v_i}^l$ as the set contains all of the check nodes reached by a sub-graph spreading from variable node v_i within depth l . The complementary of that set is donated as $\bar{\mathcal{N}}_{v_i}^l$, i.e., $\mathcal{CN}_{v_i}^l$, as shown in this example. Now for variable node v_6 , the PEG will put the connections from v_6 to three check nodes.

Stage 1: At the first connection step ($k=1$), PEG directly selects a check node which has the minimum degree. In this case we have two check nodes with degree of three (c_2 and

c_3), PEG randomly selects one of them, and we suppose the PEG adds the first edge to c_2 , as shown in Figure 3.4 (depth $l=0$).

Stage 2: At the second connection step ($k=2$), PEG algorithm starts from v_6 as follows: the variable v_6 has only one connection to c_2 so during depth (0):

$$\{\mathcal{N}_{v_6}^0 = \{c_2\}, \bar{\mathcal{N}}_{v_6}^0 = \{c_1, c_3\}\},$$

PEG moves to depth ($l=1$):

$$\{\mathcal{N}_{v_6}^1 = \{c_1, c_2, c_3\}, \bar{\mathcal{N}}_{v_6}^1 = \emptyset\},$$

the PEG algorithm calculates summation of check nodes in termination of every depth, so if summation is equal to M , then PEG algorithm stops. In this case, the summation is equal to three, so PEG returns to the previous depth and selects from $\bar{\mathcal{N}}_{v_6}^{l-1}$ check nodes one that has minimum degree (c_3), Figure 3.4.

Stage 3: As a third connection ($k=3$), like the previous stage, PEG algorithm adds an edge to c_1 and resulting parity check matrix is:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}. \quad (3.6)$$

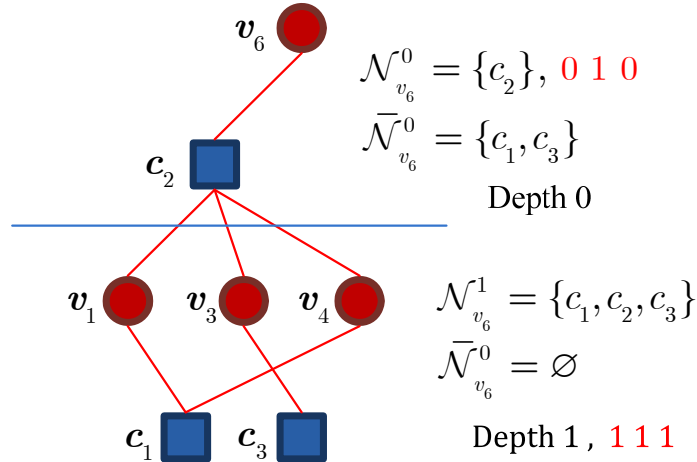


Figure 3.4. The second stage from procedure PEG for v_6

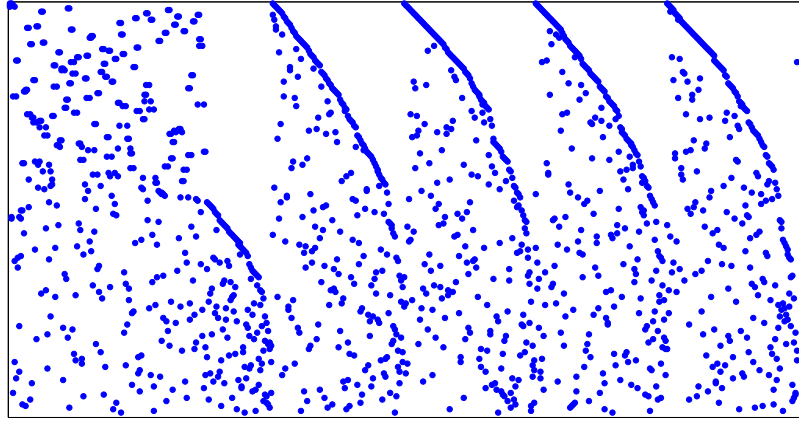


Figure 3.5. Illustration of parity-check matrix of PEGReg (504, 252) LDPC code; the blue dots correspond to positions of the value 1 in the parity check matrix

3.3.2. Quasi cyclic construction

The QC LDPC block codes are desirable to use in many applications. The main advantage of this construction principle compared to randomly constructed codes and PEG codes is that QC-LDPC encoding procedure is easier to implement using simple shift registers due to the regularity in their parity-check matrices. These codes consist of circulant submatrices, which could be either based on the identity matrix or a smaller random matrix. The parity check matrix \mathbf{H} of a QC-LDPC code is constructed by a concatenation of circulant submatrices as shown in the following

$$\mathbf{H} = \begin{bmatrix} I_1 & I_a & I_{a^2} & \cdots & I_{a^{k-1}} \\ I_b & I_{ab} & I_{a^2b} & \cdots & I_{a^{k-1}b} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ I_{b^{j-1}} & I_{ab^{j-1}} & I_{a^2b^{j-1}} & \cdots & I_{a^{k-1}b^{j-1}} \end{bmatrix}, \quad (3.7)$$

where I_x represent an identity matrix which rows are cyclically shifted to the left by positions x , and parameters a and b are two nonzero elements with multiplicative orders $o(a) = \rho$ and $o(b) = \gamma$, respectively. The parameters a and b should be chosen from Galois field $\text{GF}(m)$, where m is a prime number. For example, by choosing $m=31$, $a=2$, $b=5$, which produces regular code with parameters $\rho=5$ and $\gamma=3$ and $N=155$. It is so-called Tanner code (155,64). The regular and irregular codes can be designed by this form. Commonly, these codes suffer from restrictions on code length and rate with keeping the good performance. The main

problem lies in changing structure of parity check matrix to avoid that the number of ones increases which creates the short cycles. There are many methods to construct QC LDPC codes with different forms, and every method tries to overcome disadvantages mentioned above. Some of them are seeking to design LDPC codes with flexible code rate and length [35].

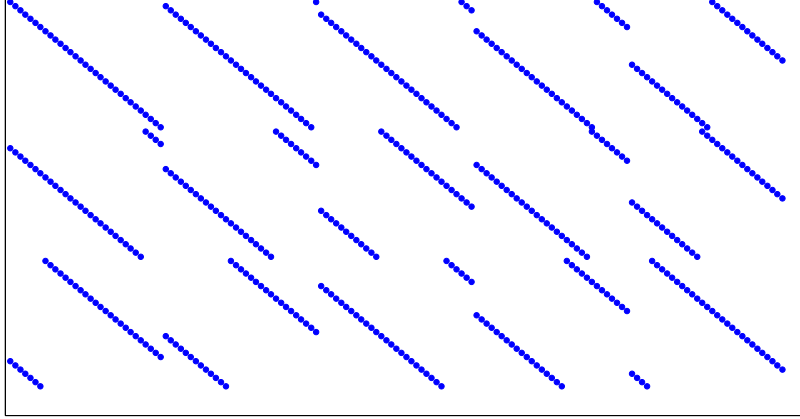


Figure 3.6. Illustration of parity-check matrix of Tanner (155, 64) LDPC code; the blue dots correspond to positions of the value 1 in the parity check matrix

The parity-check matrices obtained by PEG and QC code construction principles are illustrated in Figure 2.5 and Figure 2.6, respectively, where each 1 in the respective parity-check matrix is shown as a dot and each 0 is shown as a white space. The encoding procedure of QC LDPC codes is easier to implement and it can be implemented by using a series of shift registers, which allows its complexity to be proportional to code length [88].

3.3.3. Encoding process

Most of algorithms that design the parity check matrix \mathbf{H} do not give this matrix with the systematic form suitable for directly deriving the generator matrix $\mathbf{G} = [\mathbf{P} \mid \mathbf{I}_K]$. Instead, the Gaussian elimination (modulo-2) is used for converting the resulting matrix \mathbf{H} into a systematic form $\mathbf{H}' = [\mathbf{I}_{N-K} \mid \mathbf{P}^T]$, where \mathbf{I}_{N-K} is the identity submatrix of dimension $(N-K) \times (N-K)$. In general, the computation cost of reducing the matrix \mathbf{H} to systematic form is (N^3) and for actual encoding is (N^2) . The matrix \mathbf{H} can be written as $\mathbf{H} = [\mathbf{A} \mid \mathbf{B}]$, where \mathbf{A} is square non-singular matrix submatrix with dimension $N-K$. Result of $[\mathbf{A} \mid \mathbf{B}] * \mathbf{A}^{-1}$ equivalent to the systematic form of \mathbf{H} . This procedure is useful for PEG LDPC codes. The main problem for

some of constructions is that the matrix \mathbf{A} is singular, and then cannot be obtained the generator matrix.

It is useful to exploit the sparseness of matrix \mathbf{H} to reduce the cost complexity for encoding. In [83] the authors presented method (encoding by erasure decoding) that used the encoding process is exactly the same as decoding for transmission over the binary erasure channel (BEC) after supposing that bit nodes corresponding to parity bits are assumed to be unknown. However, success of this method is restricted to the case when matrix \mathbf{H} is free of stopping sets [89].

Here we investigate the approach presented by [90]. This approach is called a Greedy Permutation Algorithm which converts the \mathbf{H} matrix by permutation rows and columns into two forms: approximate lower triangular (ALT) form and systematic approximate lower triangular (SALT) form. The first work was presented by Richardson *et al* in [91]. The computations cost for encoding is $(N+h^2)$, where h is called a gap of the matrix \mathbf{H} and depends on its type and dimensions. This means that complexity of the encoding increases almost linearly with the code length if $h \ll N$. The gap is defined as the number of rows which decrease dimensions of the triangular submatrix \mathbf{T} . The value of h cannot be determined directly especially for large code lengths, so empirically we can test all possibilities such that the gap takes the smallest value.

The first step is transforming the parity check matrix \mathbf{H} into \mathbf{H}_1 with ALT form as illustrated in Figure 3.7. The main task is to obtain the diagonal structure for sub-matrix \mathbf{T} . The column with the minimum γ_{min} in the matrix \mathbf{H} is selected to permute with the column of index N of \mathbf{H}_1 . If there is more than one column with γ_{min} or the matrix \mathbf{H} is a regular code, then the selection can be random. Then γ_{min} positions of ones for this column are appeared at the right-down of \mathbf{H}_1 by permutation of rows. This column is excluded from any next procedure and the current \mathbf{T} submatrix arises to up from the row index $M-h$. Now all columns left of the column $(1, \dots, N-1)$ are used for formation the submatrix \mathbf{T} by searching the columns with the minimum weight on or above the main diagonal of \mathbf{T} to facilitate the procedure. By permutation the columns and rows, the procedure continues until we reach the first row with the main diagonal of the matrix \mathbf{T} . During the converting from \mathbf{H} to \mathbf{H}_1 , all permutations of the columns are saved to relocate bit-positions of the codeword. Then ALT form is obtained.

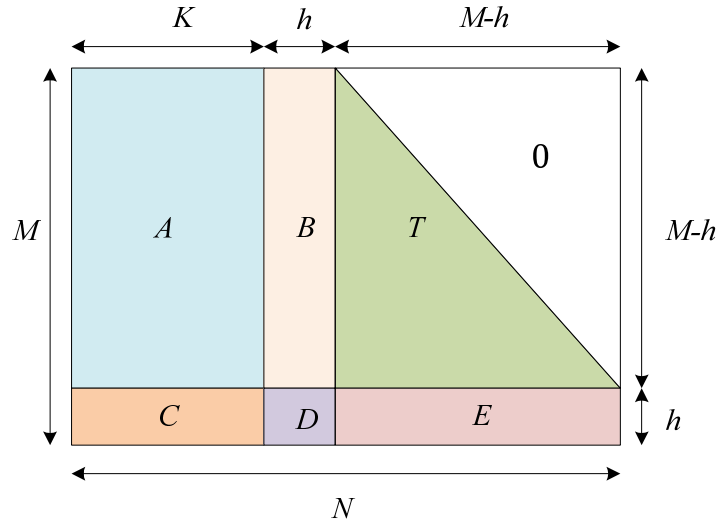


Figure 3.7. Approximate lower triangular (ALT) form

The second step is converting the H_1 ALT form to the matrix H_2 with the SALT form presented in Figure 3.8. Cancelling all ones in the submatrix E is performed by adding rows from the submatrix T and by Gaussian elimination over D and C . This creates identity submatrix D' . The gap h may be reduced during Gaussian elimination such that produces some linearly dependent rows at the bottom of the matrix which are eliminated from the matrix.

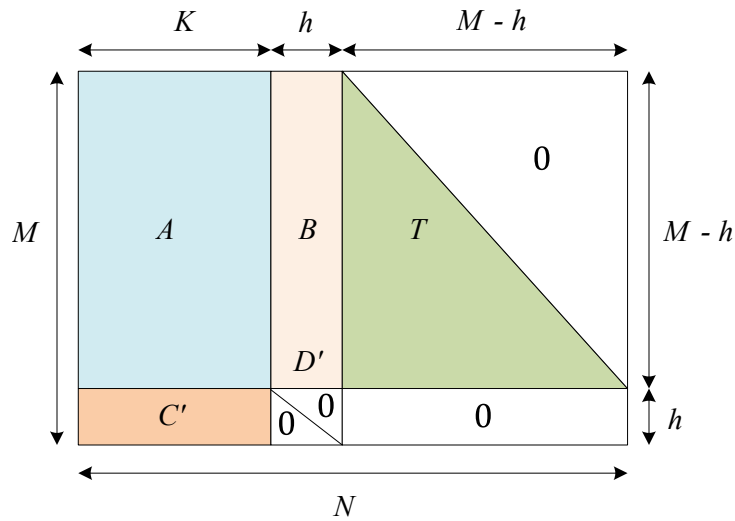


Figure 3.8. Systematic approximate lower triangular (SALT) form

The transmitted codeword \mathbf{x} is divided into three pieces u , p_1 and p_2 . According to the parity check condition $\mathbf{H} \cdot \mathbf{x}^T = \mathbf{0}$, equivalently the condition can be rewritten as:

$$\mathbf{A} \cdot u^T + \mathbf{B} \cdot p_1^T + \mathbf{T} \cdot p_2^T = \mathbf{0}, \quad (3.8)$$

where u represents a sequence of K information bits and p_1 denoted the first parity bits with size h and can be determined directly by multiplication with submatrix \mathbf{C}' as $p_1 = u \cdot \mathbf{C}'^T$. The last part p_2 of the codeword can be determined by back-substitution [91] as

$$p_2(l) = \sum_{j=1}^k \mathbf{A}_{l,j} \cdot u_j + \sum_{j=1}^h \mathbf{B}_{l,j} \cdot p_1(j) + \sum_{j=1}^{l-1} \mathbf{T}_{l,j} \cdot p_2(j), \quad (3.9)$$

where $p_2 = \{p_2(1), p_2(2), \dots, p_2(M - h)\}$.

It should be noted that for any regular LDPC code where $\gamma = 2$, the gap is always $h=0$ and complexity of the encoding is totally linear cost with the code length.

3.4. Decoding of LDPC codes

The error correction capability of LDPC codes depends on two elements: construction of the parity check matrix \mathbf{H} and an optimal decoding algorithm. Let assume that a codeword $\mathbf{x} = (x_1, x_2, \dots, x_N) \in \mathcal{C}$ is transmitted over a noisy channel and is received as a vectory $\mathbf{y} = (y_1, y_2, \dots, y_N)$. The optimal decision rule used to determine the estimate $\hat{\mathbf{x}}$ is the *maximum a posteriori* (MAP) decoding rule, which basically selects a codeword $\mathbf{x} \in \mathcal{C}$ that maximizes the *posteriori probability* $\Pr(\mathbf{y}|\mathbf{x}) = \prod_{i \in V} \Pr(y_i|x_i)$, i.e., $\hat{\mathbf{x}} = \arg \max_{\mathbf{x} \in \mathcal{C}} \Pr(\mathbf{x}|\mathbf{y})$. The MAP decoding problem can be reduced to *maximum likelihood* (ML) decoding problem under assumption that all codewords are equally likely to be transmitted.

Message Passing (MP) decoders have a significant attribute such that their procedure is comfortably described over a bipartite graph. For an infinite code length and assuming that the Tanner graph is a cycle-free or has a tree structure, the Belief-Propagation (BP) algorithm proposed by Gallager is considered as the optimal iterative decoding algorithm with linear complexity.

During MP decoding messages are exchanged between a set of variable nodes V and a set of check nodes C along edges of the Tanner graph, in order to calculate a posteriori probability of a codeword bit associated to each variable node. Two update functions are used in MP decoders, check node update function which updates the output messages of the check

nodes and variable node update function which updates the output messages of the variable nodes. These functions are symmetric functions on the incoming messages. This procedure is carried out until either $\hat{\mathbf{x}}$ is a codeword or maximum number of iterations is reached.

In an MP algorithm, nodes along edges of the TG iteratively exchange the messages and every node uses incoming messages from its neighbors to calculate outgoing messages. These nodes carry out local decoding operations along the edges of the TG, e.g., calculations at a particular node are performed independent of the comprehensive structure of the code. In addition, the computations are distributed, i.e., computations are performed by all nodes in the graph and so an effective decoder implementation can be realized. The number of messages exchanged in an iteration of a message passing decoder is the same as the number of edges in the TG; a sparser TG means fewer computations.

Only in the first iteration, the incoming messages at the variable nodes which are received from the channel, are immediately passed along the edges to the related neighboring check nodes and there no extrinsic messages from the check nodes in the first iteration. Every check node carries out the local decoding operations to determine the outgoing messages according to the incoming messages received from the neighboring variable nodes. After that outgoing messages from the check nodes are sent back to the related variable nodes and variable nodes perform the local decoding operations to determine the outgoing messages depending on the incoming messages received from the neighboring check nodes. The complete iteration is defined when the extrinsic message has passed in both directions along every edge. After every complete iteration the estimated codeword $\hat{\mathbf{x}}^l$ is tested if it is valid codeword or not by calculating the syndrome. The decoding process continues if the result is a negative until maximum number of iteration is reached or the valid codeword is found.

We suppose input sequence to the MP decoder is defined by $\mu_{ch} = (\mu_{ch1}, \mu_{ch2}, \dots, \mu_{chN})$. Let $\mu^l(v_i, c_j)$ be an extrinsic message passed by a variable node v_i to the neighboring check node c_j in the l -th iteration and $\mu^l(c_j, v_i)$ the extrinsic message passed by a check node c_j to the neighboring variable node v_i in the l -th iteration. Let $(\mathcal{N}(v_i), v_i)$ denote a set of all incoming messages to the variable node v_i and $\mu^l(\mathcal{N}(v_i) \setminus c_j, v_i)$ denote a set of all incoming messages to variable node v_i except from check node c_j . Sets $(\mathcal{N}(c_j), c_j)$ and $\mu^l(\mathcal{N}(c_j) \setminus v_i, c_j)$ can be clarified in a similar manner.

Two functions are required to complete the decoding process: variable update function fct_v to update the extrinsic message to the check node and check update function fct_c to update the extrinsic message to the variable node, where:

$$\mu^{l+1}(v_i, c_j) = \text{fct}_v \left(\mu_{chi}, \mu^l(\mathcal{N}(v_i) \setminus c_j, v_i) \right), \quad (3.10)$$

$$\mu^{l+1}(c_j, v_i) = \text{fct}_c \left(\mu^l(\mathcal{N}(c_j) \setminus v_i, c_j) \right). \quad (3.11)$$

After each complete iteration l at the variable node v_i , a *posteriori information* for each variable node is computed as

$$\tilde{\mu}_{v_i} = \mu_{chi} + \sum \mu^l(\mathcal{N}(v_i), v_i), \quad (3.12)$$

and a bit value is estimated by

$$\hat{x}_i^l = \begin{cases} 0 & \text{if } \tilde{\mu}_{v_i} < 0 \\ 1 & \text{if } \tilde{\mu}_{v_i} > 0, \\ \text{sign}(\mu_{chi}) & \text{otherwise} \end{cases} \quad (3.13)$$

where the sign function outputs a 0 if the sign of the argument is positive, and a 1 if it is negative.

The MP decoder converges if an estimated codeword $\hat{\mathbf{x}}^l$ for some number of iteration l is a valid codeword, i.e. $s^l = \mathbf{0}$ and the decoding failed if not, i.e. $s^l = \mathbf{1}$, where $s^l = (s_1^l, s_2^l, \dots, s_M^l)$ is a syndrome vector calculated as, multiplying the temporarily decoded bit sequence $\hat{\mathbf{x}}^l$ with the transpose of the \mathbf{H} , i.e., $s^l = \hat{\mathbf{x}}^l \mathbf{H}^T$. The MP algorithm runs until a valid codeword is found or the maximum number of iteration is reached.

The MP decoder is an optimal (i.e. maximum likelihood ML decoding) for those codes whose graph is cycles free otherwise is a sub-optimal due to closed paths (cycles) in the graph. So, if the codes have cycles then the MP decoder will perform close to ML decoder. Furthermore, the overall decoding complexity is linear with the code length.

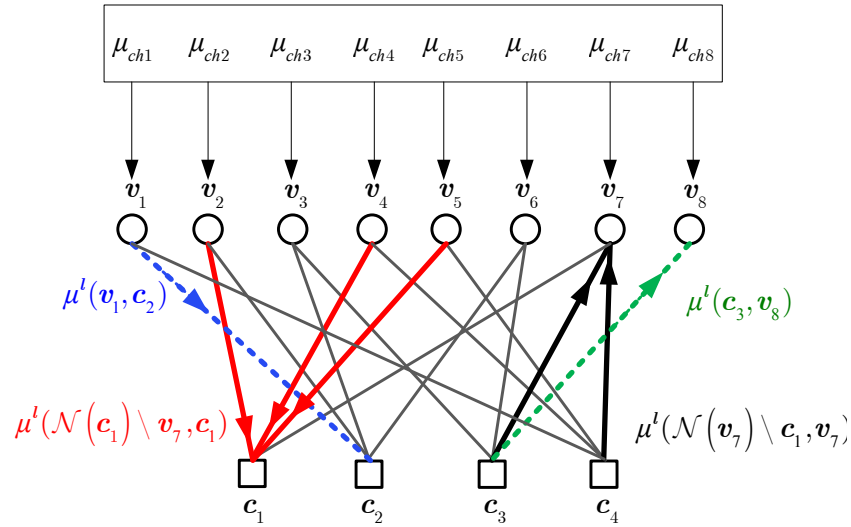


Figure 3.9. Passed messages over TG for MP decoder at iteration l

3.4.1. Belief Propagation Algorithm

The Belief Propagation (BP) algorithm, also known as Sum-Product Algorithm (SPA), is considered to be one of the most important algorithm in field of an error correction and represents the state-of-the art algorithm for decoding of LDPC codes. This algorithm belongs to a class of MP algorithms, where the messages passed between nodes along the edges are probabilities, i.e., this algorithm runs in the probability domain and intends to find the most probable transmitted codeword at every decoding iteration until reaches to the valid codeword. BP algorithm estimates a *posteriori* probability of each message symbol as a function of the received signal, the code information, expressed in the case of LDPC codes as parity equations, and the channel attributes. The BP also works in the *log likelihood ratio* (LLR) domain for numerical stability and to reduce the complexity calculations of the probabilistic approach where the value of the check node is determine by multiplication of probabilities for 0 and 1 in the probability domain. The operations are converted to summation in LLR domain where the messages are real-valued. Here we consider BP algorithm in LLR domain.

The *priori information* in LLR domain is given by

$$\mu_{chi} = \log \left(\frac{\Pr(y_i | x_i = 0)}{\Pr(y_i | x_i = 1)} \right), \quad (3.14)$$

for the Binary Symmetric Channel (BSC) we can write

$$\mu_{chi} = \begin{cases} \log\left(\frac{1-\alpha}{\alpha}\right) & \text{if } y_i = 0 \\ \log\left(\frac{\alpha}{1-\alpha}\right) & \text{if } y_i = 1 \end{cases}, \quad (3.15)$$

where α is a value of crossover probability in BSC and for AWGN we have $\mu_{chi} = \frac{2y_i}{\sigma^2}$ where σ^2 is a variance of a zero-mean Gaussian process. Let v be a variable node with a γ degree. Hence, $\gamma-1$ incoming messages to variable node v are needed to determine an extrinsic message from node v and these messages can be defined as $(\mu_1, \mu_2, \dots, \mu_{\gamma-1})$. Similarly, let c be a check node with a degree ρ . Similarly, $\rho-1$ incoming messages to care required to calculate an extrinsic message from node c and defined as $(\mu_1, \mu_2, \dots, \mu_{\rho-1})$. The update functions of the BP decoder are given as follows

$$fct_v(\mu_{chi}, \mu_1, \mu_2, \dots, \mu_{\gamma-1}) = \mu_{chi} + \sum_{j=1}^{\gamma-1} \mu_j, \quad (3.16)$$

$$\begin{aligned} fct_c(\mu_1, \mu_2, \dots, \mu_{\rho-1}) &= \prod_{j=1}^{\rho-1} \text{sign}(\mu_j) \cdot \phi \left[\sum_{j=1}^{\rho-1} \phi(|\mu_j|) \right] \\ &= 2 \tanh^{-1} \left(\prod_{j=1}^{\rho-1} \tanh \left(\frac{\mu_j}{2} \right) \right), \end{aligned} \quad (3.17)$$

where $\phi(x) = -\log\left(\tanh\left(\frac{x}{2}\right)\right)$ and sign denotes the standard signum function.

The BP algorithm is a powerful algorithm with high error correction capability, but its applicability in practice is limited, due to its complexity and low throughput. In any event, the presence of cycles of relatively short lengths in the Tanner graph cannot be avoided, which leads to degradation of BP performance. Performance degradation of BP decoder is visible at some fixed point of channel error probability which is called the error floor phenomenon. This event can be noticed when Frame Error Rate (FER) curve slowly changes its tendency while the Signal-to-Noise (SNR) takes adequately large value. BP decoder at the error floor region cannot reach to arbitrary low error probability. Regardless of these drawbacks, the BP algorithm is significant and over the years there were some attempts to reduce computations complexity of the decoder in the LLR domain. The central idea concentrates about

simplification the check node updating function of messages, and leads to a much simpler Min-Sum (MS) algorithm. Although, the Min-Sum (MS) algorithm has lower complexity than BP, there is a significantly gap between their performances. However, λ -min decoding [92] is algorithm such tries to decrease the gap between BP and min-sum algorithms and sometimes offers the same performance. The complexity of λ -min decoding is lower than the complexity of the BP decoder. Another algorithm for that aims to reduce the performance gap is the *offset* min-sum decoder proposed by Chen *et al* [93]. The *offset parameter* seeks to reduce the overestimate of the outgoing message which is calculated by check node updating function. There are methods to pick this parameter [93] and could be determined or could be changed as a function of the SNR.

A number ways are presented which try to reduce the complexity of procedures in the standard BP algorithm, and also computation cost, while preserving the performance up to certain level. Here we present a brief survey of simplified BP decoding algorithms.

3.4.2. Reduced Complexity Decoders

The largest values of the function $\phi(x)$ depend on the minimum values of x , so the function fct_c that calculates an extrinsic message received by a variable node depends on the minimum absolute value of the incoming messages to the check node. We assume the vector $[t_1, t_2, \dots, t_\lambda]$ is the λ smallest absolute value, then the fct_c can be approximated as:

$$fct_c(\mu_1, \mu_2, \dots, \mu_{\rho_c-1}) \approx \left(\prod_{j=1}^{\rho_c-1} \text{sign}(\mu_j) \right) \phi[\phi(|t_1|) + \phi(|t_2|) + \dots + \phi(|t_\lambda|)], \quad (3.18)$$

this algorithm is called λ -min algorithm where $\lambda < \rho_c$.

Min-sum algorithm also approximates the function fct_c in such a way that takes only the minimum absolute value which corresponds to the largest magnitude of the incoming messages. The check-node update function of the MS decoder is given by

$$fct_c(\mu_1, \mu_2, \dots, \mu_{\rho_c-1}) \approx \left(\prod_{j=1}^{\rho_c-1} \text{sign}(\mu_j) \right) \min_{j \in \{1, \dots, \rho_c-1\}} (|\mu_j|). \quad (3.19)$$

The gap between the standard BP and the min-sum decoders is large specifically in the low to mid SNR regions, but its complexity is lower than BP algorithm. One important algorithm which narrows the gap is *offset* min-sum algorithm [93]. The update function of the

check node is modified by inserting one factor to reduce an excessively high estimate of the extrinsic message from a check node using the min-sum algorithm. This factor is called an *offset* or a *correction* factor. The modified function is given by

$$fct_c(\mu_1, \mu_2, \dots, \mu_{\rho_c-1}) = \left(\prod_{j=1}^{\rho_c-1} \text{sign}(\mu_j) \right) \max \left(\min_{j \in \{1, \dots, \rho_c-1\}} (|\mu_j|) - \eta, 0 \right). \quad (3.20)$$

It can be noted from fct_c that an extrinsic message is zero if the magnitude of an incoming message is less than offset factor η . There are many methods to determine value of this factor that assume that it could be fixed or code dependent. In the second case the offset factor value can be obtained by a brute force simulation.

On the other hand, the authors in [94] proposed a very simple and powerful self-correction algorithm for the min-sum decoding of LDPC codes. In that algorithm, an update variable function is modified by erasing unreliable messages, while the check node update function is the same as in the min-sum algorithm. The basic idea is track sign changes of a variable node message in the two consecutive iterations and deletes forces this message to 0 if a change is detected. After updating the extrinsic check messages (c_j, v_i) at iteration l and determine a *posteriori information* $\tilde{\mu}_{v_i}$, the extrinsic variable messages are calculated as:

$$\mu_l(v_i, c_j)^{\text{tmp}} = \tilde{\mu}_{v_i} - \mu^l(c_j, v_i), \quad (3.21)$$

$$\mu^l(v_i, c_j) = \begin{cases} \mu^l(v_i, c_j)^{\text{tmp}} & \text{if } \text{sgn}(\mu^l(v_i, c_j)^{\text{tmp}}) = \text{sgn}(\mu^{l-1}(v_i, c_j)) \\ 0 & \text{otherwise} \end{cases}. \quad (3.22)$$

When the message $\mu^{l+1}(v_i, c_j)$ takes a *zero value* and arrives to the check node to have both negative and positive signs in the next iteration we update the new variable node message by $\mu^{l+1}(v_i, c_j) = \mu^{l+1}(v_i, c_j)^{\text{tmp}}$.

3.4.3. Gallager A\B decoder

Gallager A\B algorithm is a hard decision iterative decoder. In spite of its simplicity, it belongs to the class of MP decoders. Gallager A\B decoder is a sub-optimal decoder for BSC channel with binary format.

Gallager-B decoder is a hard decision decoder, which means that messages passed between nodes in Tanner graph are binary. The small range of values makes this decoder as a

hard decision, as opposite to a soft decision decoder which consumes a wide range of values. The procedure of Gallager-B decoder is as follows:

The input of the Gallager-B decoder is defined by $\mathbf{y} = \{y_1, y_2, \dots, y_N\}$ as a binary received signal. The update check function fct_c is given by:

$$fct_c(\mu_1, \mu_2, \dots, \mu_{\rho_c-1}) = \left(\sum_{j=1}^{\rho_c-1} \mu_j \right) \bmod 2, \quad (3.23)$$

and the update variable function is defined as:

$$fct_v(\mu_1, \mu_2, \dots, \mu_{\gamma_v-1}) = \begin{cases} 1 & \text{if } \sum_{j=1}^{\gamma_v-1} \mu_j \geq b_{v,l} \\ 0 & \text{if } \gamma_v - 1 - \sum_{j=1}^{\gamma_v-1} \mu_j \leq b_{v,l} \\ y_v & \text{otherwise} \end{cases}, \quad (3.24)$$

where b_v represents a threshold of the majority voter function in the variable v , which in general can depend on iteration l -th and should be greater than $(\gamma_v-1)/2$ and smaller than γ_v . After predefine number of iterations the final decision of transmitted bit is made on the basis of majority of its estimate (v_i, c_j) .

In a special case, when the threshold b_v is fixed at $b_v = \gamma_v - 1$ for all iterations and all variable nodes. Worth mentioning that Gallager-B decoder is more powerful than algorithm-A while for $\gamma_v = 3$, both algorithms have the same performance.

Performance of this algorithm is lower in both the waterfall and the error regions compared with BP. The error correction capability of regular $\gamma = 3$ LDPC codes on the BSC decoded under the Gallager A\B algorithm has been analyzed in [95], [96]. We can summarize their results. For $g \geq 10$, where g is the girth of the Tanner graph representation of a code, Gallager A\B can correct all error patterns with up to $(g/2-1)$ errors in the at most $g/2$ iterations. This means that there is a relation between capability error correction and girth when $g \geq 10$ under Gallager A\B algorithm. However, when $g \leq 8$ it was shown that the girth is not sufficient condition for assurance correcting the errors with weight $(g/2-1)$.

The Figure 3.10 and Figure 3.11 represent FER performance on Tanner code (155,64) by passing decoding algorithms over BSC and AWGNC, respectively.

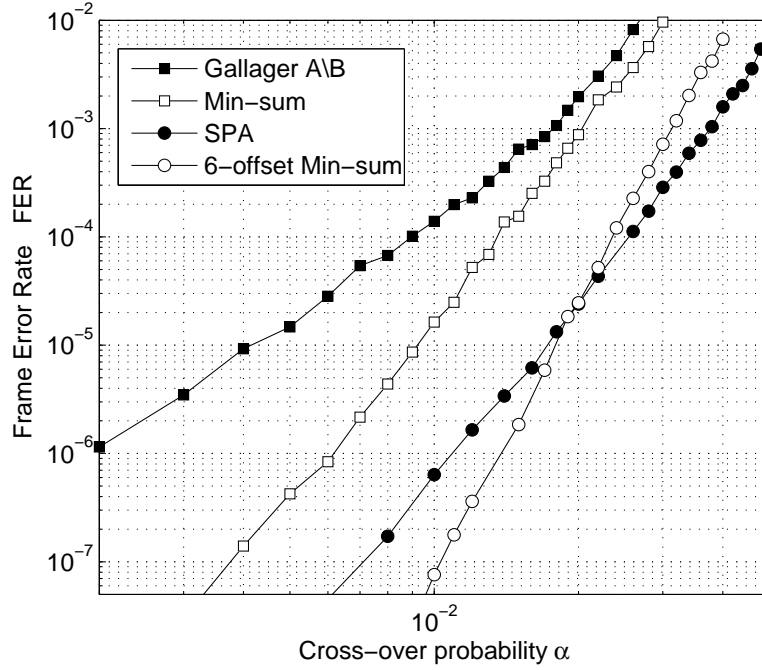


Figure 3.10. FER performance on Tanner code (155,64) by passing decoding algorithms over BSC

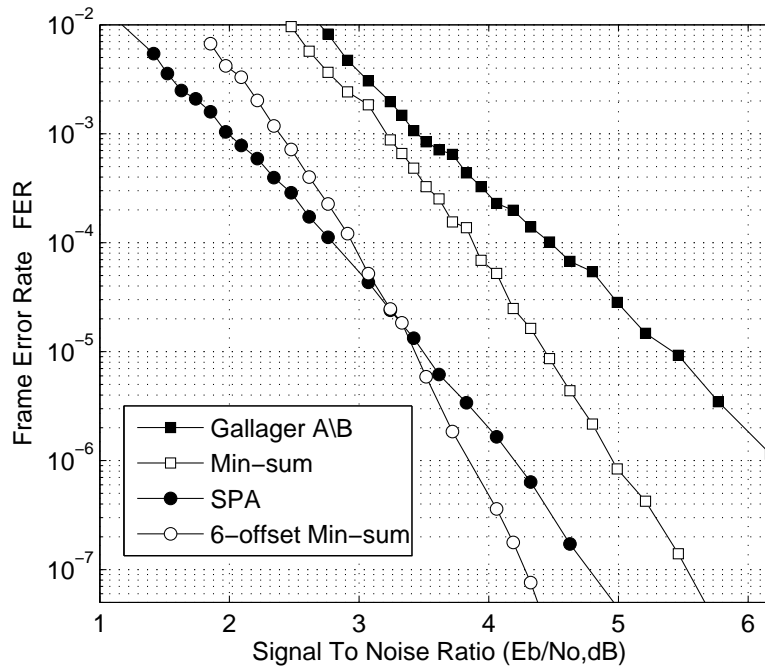


Figure 3.11. FER performance on Tanner code (155,64) by passing decoding algorithms over AWGN

Chapter 4

Improvement of the Bit-Flipping Algorithm and Faulty Decoder

All attempts for achieving the better balance between complexity of LDPC decoder and error performance, launched from two opposite directions: the first direction aims to diminish the high complexity of BP decoding with maintaining the high performance as much as possible as the mentioned algorithms in Chapter 3. The second direction aims to improve performance of hard-decision decoding to be closed to BP performance. In our research we follow the second direction.

There are many differences between high and low complexity decoding algorithms. High-complexity decoding algorithms such as Believe Propagation (BP) algorithm can correct more errors created by the channel noise, and thus, they should be used when the complexity and computation time is not important. Low-complexity decoding algorithms, however, are more attractive when we demand the fast decoders for high-throughput systems or delay-sensitive applications. Low-complexity decoding algorithms have two main disadvantages. First, they have an inferior threshold of decoding compared to high-complexity algorithms so, to ensure the better performance, a lower code rate should be used. Second, because of their relatively poor performance, some decoding algorithms need large number of iterations to obtain a wanted bit or frame error rate.

Bit flipping algorithms are the fastest, least complex and easy to implement in hardware. Original BF decoding was introduced by Gallager in his seminal work. The BF decoders are hard-decision decoders and their performance gap can be significantly wider compared to for example the Gallager A\B. BF algorithm is a rather simple because its procedure only requires calculation the number of unsatisfied check nodes for every bit in

code, which can be done by the logical operations. Based on the specified threshold for each bit, the bit is flipped if number of unsatisfied check nodes is larger than the threshold.

We begin by introducing some algorithms designed for the Additive White Gaussian Noise Channels (AWGNCs). These algorithms depend on the real values of the received signal to approach the better performance with lower computations cost. Our proposed algorithm is derived from Gradient Descent Bit Flipping (GDBF) algorithm which is designed only for an AWGNC [100]. By inserting some modifications on this algorithm, we can optimize this algorithm for BSC with sufficient performance. The optimization is performed by analyzing some of harmful trapping sets in the TG.

4.1. Bit flipping algorithm

Bit flipping BF is a simple iterative algorithm and it is classified with the hard decoding algorithms. In the BF algorithm the number of satisfied and unsatisfied check nodes that are connected to the certain variable node is determined and compared with a predefined threshold. The threshold is designed to optimize the error performance. Let $\psi_s^l(v)$ denote a number of satisfied check nodes and $\psi_{us}^l(v)$ the number of unsatisfied check nodes, that are connected to variable node v . The check node c is satisfied if $s_c^{l-1} = 0$ and unsatisfied if $s_c^{l-1} = 1$ before a new iteration l . When $\psi_{us}^l(v) > \Delta_v$ then the value of the variable node is flipped, where Δ_v is a threshold for each variable node v and it can be empirically selected to obtain the better performance and in general it could be selected as $\gamma_v/2$. The procedure is repeated until all parity check equations are satisfied or a determined maximum number of iterations is reached. The BF decoder is given in Algorithm 0.

In recent years, there are a number of improvements of the BF algorithm. Many of these algorithms depend on joining the bit-flip decision with the received symbols where the bit is represented in bipolar form [+1,-1]. Adding the received symbols in the bit-flip decisions improves performance of the decoding. The most of previous modifications of the BF algorithm, including some kind of reliability information of the received symbols in their decoding decisions, are designed for the AWGNC or any soft information channel.

Algorithm 0 Parallel Bit Flipping Algorithm over BSC

```

Input:  $\mathbf{y}$  received word
 $\forall v \in V : \hat{x}_v^{(0)} \leftarrow y_v$ 
 $\mathbf{s}^{(0)} \leftarrow \mathbf{x}^{(0)} H^T (\forall c \in C : s_c^{(0)} \leftarrow \bigoplus_{u \in \mathcal{N}_c} \hat{x}_u^{(0)})$  calculate the syndrome
 $l=0$ 
while  $\mathbf{s}^{(0)} \neq \mathbf{0}$  and  $l \leq L$  do
 $\forall v \in V$ : Compute  $\psi_{us}^l(v)$ 
 $v=1$ 
while  $v \leq N$  do
    if  $\psi_{us}^l(v) > \frac{\gamma v}{2}$  then
 $\hat{x}_v^{(l+1)} \leftarrow 1 \oplus \hat{x}_v^{(l)}$  flip bit
    else
 $\hat{x}_v^{(l+1)} \leftarrow \hat{x}_v^{(l)}$ 
    end if
 $v \leftarrow v + 1$ 
end while
 $\mathbf{s}^{(l+1)} \leftarrow \mathbf{x}^{(l+1)} H^T$  syndrome check
 $l \leftarrow l + 1$ 
end while
Output:  $\hat{\mathbf{x}}^{(l)}$ 

```

4.2. Weighted Bit Flipping algorithm and modified

Kou *et al* introduced the Weighted Bit-Flipping (WBF) algorithm [97]. In this algorithm a magnitude of the received symbol is used as a simple measure of the reliability of a received symbol. The checksums for each check node is weighted by the minimum magnitude of the received symbols which participate in the same check node. The bit-flip decision is determined by summation all weighted syndrome of check nodes, which are connected to the same bit (variable node). This summation can be defined as an estimation criterion of reliability the symbol. The bit with minimum estimation criterion value is selected to be flipped. The estimation criterion in the WBF algorithm, can be chosen in such a way that only one bit is flipped during a iteration, which slowdowns the convergence. The WBF algorithm significantly increases the complexity of computations as the minimum value estimation is a global function, which means that it is carried out over all variable nodes.

We assume the message $\boldsymbol{\chi} = (\chi_1, \chi_2, \dots, \chi_N)$ is transmitted in the bipolar format $\chi_v = \mp 1$ across a noisy channel that adds a vector of independent, identically distributed Gaussian noise, $\boldsymbol{\eta}$, to the message. At the receiver, a vector of samples, $\boldsymbol{\eta} = (\eta_1, \eta_2, \dots, \eta_N)$, is

obtained and given by $\boldsymbol{\eta} = \boldsymbol{\chi} + \mathbf{n}$, where χ_v and η_v denote the value of the sample associated with variable node v .

As we said that improvement of the BF algorithm can be achieved by taking into account the reliability of the received samples. In weighted bit flipping algorithm WBF for AWGNC, an information reliability of the received sample can be considered by its magnitude $|\eta_v|$. The reliability information is determined for each check node by selecting the minimum reliability of the variable nodes which are connected to a particular check node. Information reliability for each check node β_c can be defined as:

$$\beta_c \triangleq \{\min\{|\eta_v|\} : v \in \mathcal{N}_c\}.$$

An estimated sample at the beginning of the decoding process is defined by $\hat{\chi}_v^{(0)} = \text{sign}(\eta_v)$ while an *inversion function* $\Delta_v^{(l)}$ is given by the sum of the weighted bipolar syndromes and can be defined as:

$$\Delta_v^{(l)}(\boldsymbol{\chi}, \boldsymbol{\eta}) \triangleq \sum_{c \in \mathcal{N}_v} \beta_c \prod_{u \in \mathcal{N}_c} \hat{\chi}_u^{(l)}. \quad (4.1)$$

This function gives the measure of invalidness of symbol assignment on v and the sign of an estimated bit $\hat{\chi}_v^{(l)}$ is flipped if $\Delta_v^{(l)}(\boldsymbol{\chi}, \boldsymbol{\eta}) = b^{(l)}$. The threshold is defined as

$$b^{(l)} = \min(\Delta^{(l)}(\boldsymbol{\chi}, \boldsymbol{\eta})),$$

while $\Delta^{(l)}(\boldsymbol{\chi}, \boldsymbol{\eta}) = (\Delta_1^{(l)}(\boldsymbol{\chi}, \boldsymbol{\eta}), \Delta_2^{(l)}(\boldsymbol{\chi}, \boldsymbol{\eta}), \dots, \Delta_N^{(l)}(\boldsymbol{\chi}, \boldsymbol{\eta}))$.

Zhang *et al* modified the WBF algorithm and added another term to the last summation [98]. In MWBF, a term is added to the inversion function which depicts the effect of the received symbol. For AWGNC if reliability of the symbol is denoted as $|\eta_v|$, represent the absolute value of the received symbol, we have the following criterion function:

$$\Delta_v^{(l)}(\boldsymbol{\chi}, \boldsymbol{\eta}) \triangleq \sum_{c \in \mathcal{N}_v} \beta_c \prod_{u \in \mathcal{N}_c} \hat{\chi}_u^{(l)} + \alpha \cdot |\eta_v|, \quad (4.2)$$

where the weighting factor α is a positive real number optimized for different signal-to-noise ratios [98].

Performance of MWBF is better than WBF but has the same problem of the high computation complexity, which is the negative side paid for the performance improvement. Another flipping algorithm is presented by Jiang *et al* [99]. In that paper, improvement of MWBF is made by taking into account that reliability of checksums involving this bit should

exclude the bit itself if its reliability is the minimum. In the other word, remedies the update of the check message in the min-sum algorithm. IMWBF algorithm demands additional computational complexity before decoding iterations.

The WBF, MWBF and IMWBF are a single-bit flipping algorithms and need a large number of iterations during the decoding process to reach the valid codeword. Wadayama *et al* designed a new formulation for bit-flipping decoding based on the gradient descent model [100]. The GDBF decoding algorithm is designed to correct the errors over AWGNs as shown in Figure 4.1. This model is optimized for maximum likelihood ML decoding problem by combining the correlation of received samples with the syndromes. This model is called the objective function. The gradient descent inversion function for each bit is based on the partial derivative of the objective function. The error performance of GDBF algorithm is superior compared to the WBF, MWBF and IMWBF algorithms especially if the number of allowed iteration is limited to a small number. To reduce the number of iterations during GDBF decoding, the authors of GDBF modified the condition for bit-flipping. After specifying an inversion threshold, all bits are flipped if their inversion functions are smaller than this threshold.

Most of the proposed algorithms improve original BF algorithm, but they depend on the soft information from a channel. Miladinovic and Fossorier introduced a new algorithm Probabilistic BF (PBF) on the BSC [101]. In their algorithm, a probabilistic parameter p is used to convert the BF algorithm from the determinism to the probabilism. It uses an addition condition for bit-flip decision such that the bit is flipped if number of unsatisfied check sums is greater than the predetermined threshold, but the flipping will occur only with some probability $p \leq 1$. Probabilistic values of p may be increased during decoding process with a certain step. For $\gamma = 3$ a significantly improvement is obtained and possible improvement quickly decreases as γ increases. PBF algorithm is designed for practical LDPC codes are represented by Tanner graphs which contain cycles. Because of the probabilistic nature of PBF algorithm the convergence of the decoding process is delayed.

Recently, a new class of algorithms is designed to improve BF decoding on the BSC in [102]. These algorithms use two bits to represent variable node and another two bits to represent check node and are called two-bit bit flipping (TBBF) algorithms. For variable node, an additional bit can refer to strength of a variable node and the algorithms may decrease its strength based on a combination of satisfied and unsatisfied check nodes. For check node, an additional bit can refer to its reliability. In an efficient manner for failure

analysis of these algorithms, a concatenation of TBBF algorithms is used to give excellent improvement and a good performance complexity tradeoff has been proposed.

4.3. Gradient Descent Bit flipping algorithm

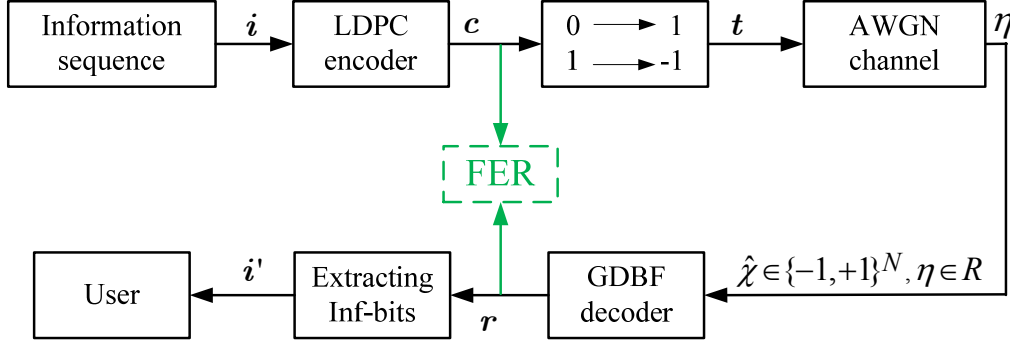


Figure 4.1. Simple telecommunication model for AWGNC using GDBF decoder

The BF algorithm can be seen as a process to minimize a hidden objective function with certain number of iterations. An objective function is designed to converge to the correct codeword with low complexity. The main idea behind the GDBF algorithm is combination two decoding processes. The first process is maximum likelihood (ML) decoding and the second process is sum of the bipolar syndromes of $\hat{\chi}$. The ML decoding problem for an AWGNC is process for finding an estimated codeword at the particular iteration which gives the largest correlation to a given received signal η . In the other words, ML decoding aims to find $\hat{\chi} \in C$ that maximizes the correlation

$$\hat{\chi} = \arg \max_{\chi \in C} \sum_{u=1}^N \hat{\chi}_u \eta_u . \quad (4.3)$$

Actually, finding a set of all possible messages must subject to the parity constraint and this is a complexity method. One of the simple methods to solve ML decoding problem is joint it with the second process as a *penalty term*, so an objective function for GDBF algorithm can be defined as

$$f^{(l)}(\hat{\chi}, \eta) \triangleq \sum_{v=1}^N \hat{\chi}_v^{(l)} \eta_v + \sum_{r=1}^M \prod_{u \in \mathcal{N}_c} \hat{\chi}_u^{(l)} , \quad (4.4)$$

where $\hat{\chi} \in \{-1 + 1\}^N$ and it is noticeable that the second term is maximized when $\hat{\chi} \in C$, $\sum_{r=1}^M \prod_{u \in \mathcal{N}_c} \hat{\chi}_u = M$, i.e. all syndromes are equal to +1, and when $\hat{\chi} \notin C$ at least one of the syndromes is equal to -1 due to parity breach. An objective function is a nonlinear function and has many local maxima as shown in Figure 4.2. Wadayama in [100] defined an inversion function as

$$\Delta_v^{(l)}(\hat{\chi}, \boldsymbol{\eta}) = \hat{\chi}_u^{(l)} \frac{\delta f^{(l)}(\hat{\chi}, \boldsymbol{\eta})}{\delta \hat{\chi}_u^{(l)}} = \hat{\chi}_v^{(l)} \eta_v + \sum_{c \in \mathcal{N}_v} \prod_{u \in \mathcal{N}_c} \hat{\chi}_u^{(l)}, \quad (4.5)$$

i.e. move direction and position of the guess toward the objective function. An iteratively procedure for GDBF algorithm with respect to each symbol for convergence of the codeword, flips the sign of bits for which $\Delta_v^{(l)}(\hat{\chi}, \boldsymbol{\eta}) = b^{(l)}$. It can be noted that GDBF algorithm tries to maximize an objective function during iterations in a gradient ascent manner. An objective function can be used as a tool to describe the decoding process. In each iteration, maybe only a single bit is flipped corresponding to the condition. Wadayama [100] introduced a *MultiGDBF algorithm* in order to increase the speed of algorithm such that the bits are flipped in parallel, i.e., every bit is flipped whose inverse function value is higher than the threshold operation. To increase the decoding speed and to improve the stability, a sign of any bit $\hat{\chi}_v^{(l)}$ is changed if $\Delta_v^{(l)}(\hat{\chi}, \boldsymbol{\eta}) < \theta$, where $\theta < 0$ is a threshold parameter. Worth mentioning that procedure of GDBF gets stuck at the certain local maximum, i.e., decoder cannot converge to a correct codeword even if the number of iterations tends to infinity. There are many mechanisms to escape from the local maximum and to approach to the global maximum by inserting a random perturbation in the inversion function [103].

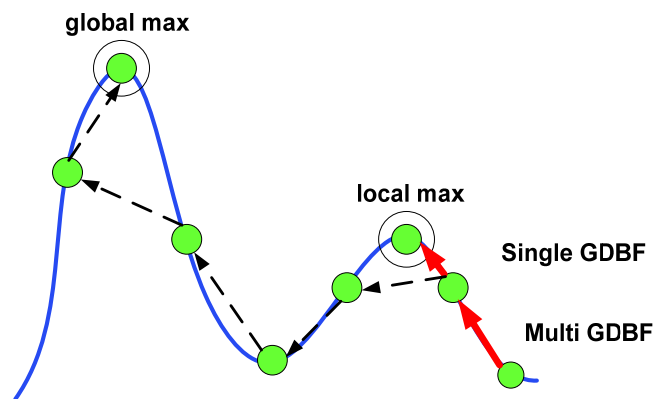


Figure 4.2. Convergence behavior and escape from local max

The GDBF decoding is in way soft-decision decoding and to make the decisions, required real addition operations to compute an inversion function. It is well known that a real addition operation is much more complex than a logical operation, the computational complexities of GDBF decoding are dominated by the total number of real additions needed to decode a received sequence. In our proposed algorithm we optimize GDBF algorithm to work over BSC and most computations can be done by a logical operations.

4.4. Optimize GDBF algorithm to the BSC

To adapt GDBF for BSC, we first rewrite the polar based inverse function in a binary form. Let $\mathbf{x} = (x_1, x_2, \dots, x_N)$ denote a codeword of C that is transmitted over a BSC with crossover probability α and let $\mathbf{y} = \{y_1, y_2, \dots, y_N\}$ is be a vector received by a decoder from BSC where $\mathbf{y} = \mathbf{x} \oplus \mathbf{e}$ and $\mathbf{e} = (e_1, e_2, \dots, e_N)$ denotes the *error pattern* introduced by the BSC, and \oplus is the component-wise modulo-two sum. Let $\hat{x}_v^{(l)} = 1 - 2x_v^{(l)}$ and $\eta_v = 1 - 2y_v$, by using modul-2 arithmetic, the inverse function can be written into:

$$\Delta_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) = 2 - 2(\hat{x}_v^{(l)} \oplus y_v) + \gamma_v - 2 \sum_{c \in \mathcal{N}_v} \bigoplus_{u \in \mathcal{N}_c} \hat{x}_u^{(l)}. \quad (4.6)$$

This equation has constant terms and variable terms, as we have above mentioned that GDBF algorithm tries to maximize an inverse function and because of the negative values of the some variable terms, the modified inverse function (MIF) can be rewritten for BSC, where the above expression is minimized by maximization of the following modified inverse function

$$\begin{aligned} \Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) &= 1 - 0.5 \cdot \Delta_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) \\ &= \hat{x}_v^{(l)} \oplus y_v + \sum_{c \in \mathcal{N}_v} \bigoplus_{u \in \mathcal{N}_c} \hat{x}_u^{(l)} - 0.5 \cdot \gamma_v. \end{aligned} \quad (4.7)$$

For γ - variable-regular codes, the modified inverse function can be simplified as

$$\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) = \hat{x}_v^{(l)} \oplus y_v + \sum_{c \in \mathcal{N}_v} \bigoplus_{u \in \mathcal{N}_c} \hat{x}_u^{(l)}. \quad (4.8)$$

The values of the modified inverse function MIF are always positive and the range of these values is restricted to the set of positive integer values $[0, \gamma+1]$. To minimize the

expression (4.6), the maximum value of the modified inverse function is found at the l -th iteration, and let $b^{(l)} = \max_v(\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}))$ be the largest value and as a counterpart, in AWGNC the minimum value, $\min_v(\Delta^{(l)}(\hat{\mathbf{x}}, \boldsymbol{\eta}))$, is found. The bit with maximum value is flipped to get the convergence, the new decoder over BSC is given in Algorithm 1. Many variable nodes maybe satisfy the relation $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) = b^{(l)}$, which has a negative impact on the algorithm convergence. If the flipping decision was wrong, the “flip messages” would propagate through the short cycles in TG. According to (4.8), for a (γ, ρ) -regular code it is necessary to calculate the parities in the neighboring check nodes, by using ρ -input exclusive or (XOR) gates. An additional two-input XOR gate is required to check if the v -th bit of the current estimate is the same as the bit initially estimated from the channel. The value of $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y})$ is equal to the number of non-zero outputs of the XOR gates. Combinational logic at the variable nodes is based on a set of majority logic (MAJ) gates, each having $\gamma+1$ inputs and adaptable threshold. The output of the v -th MAJ gate in the l -th iteration is non-zero only if the modified inverse function value is equal to the threshold value $b^{(l)}$. This threshold is the same for every variable node as presented in Figure 4.3 (a). The Figure 4.3 shows the variable node processing unit for GDBF and BF algorithms over BSC, it can be noted that both algorithms have the same structure and for GDBF there is two-input XOR logic gate to XOR an initial estimated value of the bit with current estimated value for the same bit.

How threshold adaptation can be realized?

The threshold can be adapted in hardware by three steps for every iteration:

- The threshold may be initialized to the maximum value $b^{(l)} = \gamma + 1$.
- The threshold is decremented when all MAJ logic gate outputs are zeros, for instance by using N-input OR gate.
- When the output of at least one MAJ gate is not equal to zero, the threshold is set to $b^{(l)} = \max_v(\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}))$.

This is important for hardware implementation, as it significantly simplifies global operation of maximization of MIF.

It seems interesting to investigate the performance of GDBF in binary model and compare it with GDBF in way soft-decision. We can compute the probability of error by this relation $\alpha = 0.5 \operatorname{erfc}(\sqrt{(E_b/N_0)})$ as a performance measure from BSC to AWGNC where

$erfc$ is a complementary error function [104]. Figure 4.4 shows important performance degradation for the GDBF in hard-decision compared with GDBF in soft-decision.

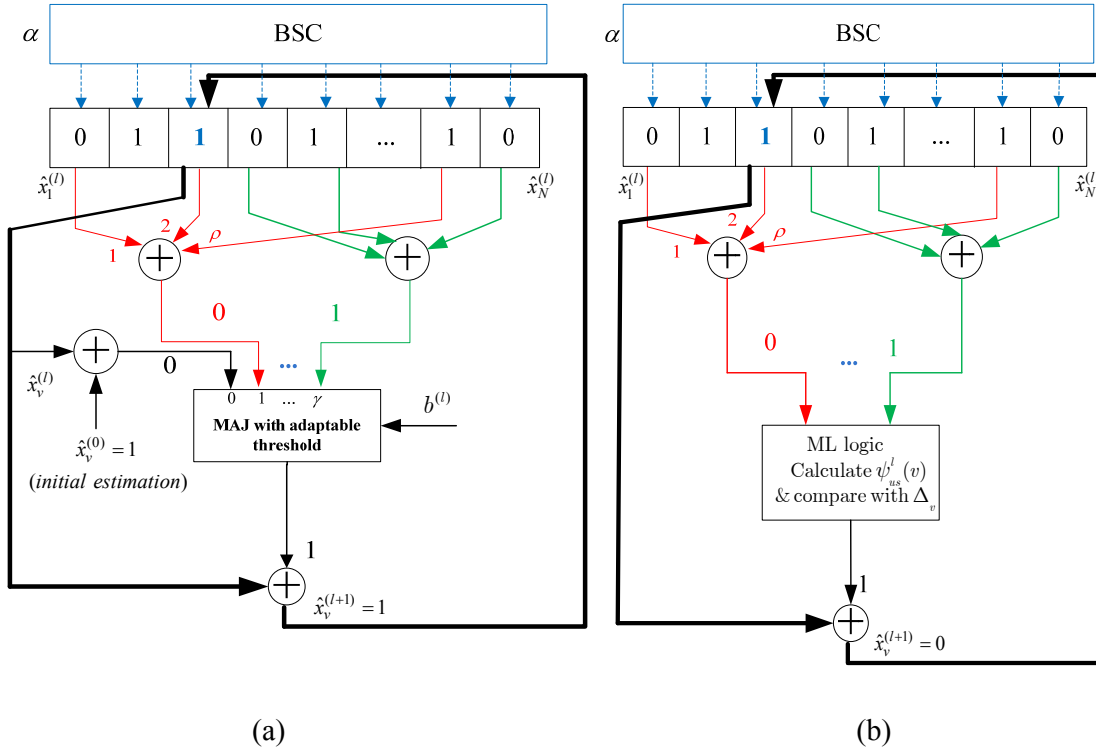


Figure 4.3. Illustration of the variable node processing unit for (a) GDBF and (b) BF algorithms over BSC

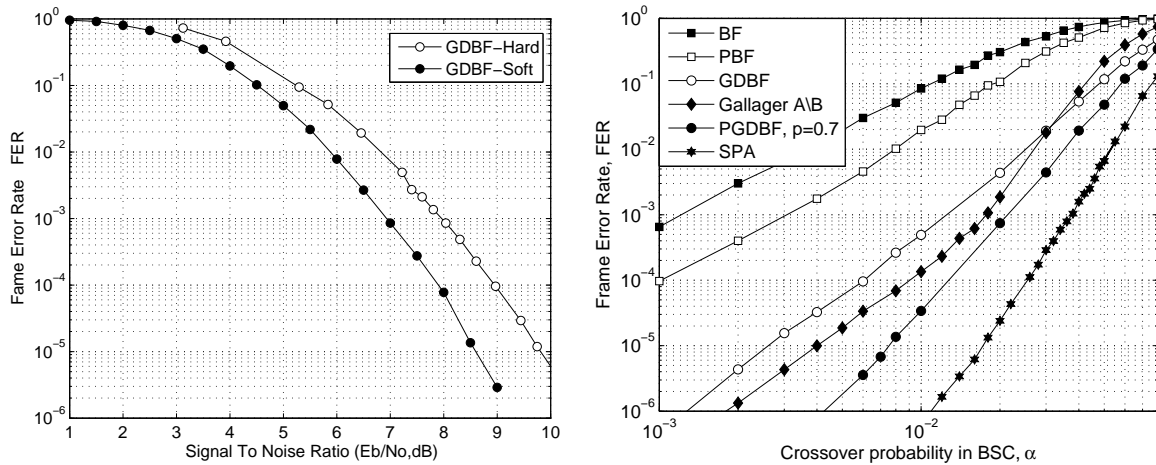


Figure 4.4 (a). Performance of GDBF in soft and hard ways for Tanner (155,64), $L=100$

Figure 4.4 (b). FER performance comparison for the (155,64) Tanner code, $L=100$

Algorithm 1 GDBF Algorithm over BSC

```

Input:  $\mathbf{y}$  received word
 $\forall v \in V : \hat{x}_v^{(0)} \leftarrow y_v$ 
 $\mathbf{s}^{(0)} \leftarrow \mathbf{x}^{(0)} H^T (\forall c \in \mathcal{C} : s_c^{(0)} \leftarrow \bigoplus_{u \in \mathcal{N}_c} \hat{x}_u^{(0)})$  calculate the syndrome
 $l=0$ 
while  $\mathbf{s}^{(0)} \neq \mathbf{0}$  and  $l \leq L$  do
 $\forall v \in V$ : Compute  $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y})$ 
 $b^{(l)} \leftarrow \max_v (\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}))$  find the maximum value
 $v=1$ 
while  $v \leq V$  do
    if  $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) = b^{(l)}$  then
         $\hat{x}_v^{(l+1)} \leftarrow 1 \oplus \hat{x}_v^{(l)}$  flip bit
    else
         $\hat{x}_v^{(l+1)} \leftarrow \hat{x}_v^{(l)}$ 
    end if
     $v \leftarrow v + 1$ 
end while
 $\mathbf{s}^{(l+1)} \leftarrow \mathbf{x}^{(l+1)} H^T$  syndrome check
     $l \leftarrow l + 1$ 
end while
Output:  $\hat{\mathbf{x}}^{(l)}$ 

```

4.4.1. Comparison between GDBF and other decoders

Before analyzing the failure of the GDBF decoder over BSC to approach the same performance of GDBF over AWGNC, let us explain why GDBF has a superior performance compared to BF and PBF decoders over BSC for waterfall and error-floor regions. We show that a number of error pattern corrected by the BF algorithm is correctable with GDBF. This is an absolutely evident because the gap between two algorithms is very large as shown in Figure 4.4 (b). Also GDBF decoder has good convergence behavior only for high crossover probability values of the BSC and is not for lower values when compared to Gallager A/B on the well-known Tanner code (155,64) with $\gamma_v=3$ and $\rho_c=5$ [105], [83]. We consider in our simulations this code for two reasons: the difference between its minimum distance $d_{min} = 20$ and its minimum pseudo-distance $w_p^{min} \simeq 10$ is large, that means the difference in the guaranteed error correction capability between customary iterative decoders is anticipated to be large. Second, the (155,64) Tanner code is adequately small and structured, so that a brute

force checking of whether all errors get up to the determined weight- t are corrected by decoding variety set can be done by Monte-Carlo simulations with rational computation time.

We briefly discuss some harmful constructions in the LDPC code which impede the GDBF algorithm to achieve the convergence. As we know that curves of linear codes in general show purposely tendency towards partition into two distinct regions. The gain is high at the beginning, we talk certainly about good codes, but this phenomenon appears within the first few decibels. With higher SNR the quality of the signal increases and a negative impact of the channel decreases and the number of errors reduces. It is expected that the performance of the code are meaningfully better and it is ability of LDPC codes to shove this gain to be adjacent to the theoretical limit [106]. At some point, this gain may stop abruptly and this event because of some harmful constructions inside of the LDPC codes which are called *trapping sets*.

For GDBF algorithm the situation maybe be different, the tendency of the GDBF curve is relatively constant for all crossover probability values of BSC α , this case is blamed on low-weight codewords, which imply poor d_{min} and on the other hand on the nature of this algorithm which concerns only on the maximum value of an inverse function to flip the bits such that prevent to correct another error pattern in the graph as we will see in the next. We begin with a short debate of trapping sets and related matters. For our work we use the same definitions of the trapping set for another decoders and expand the concept of the trapping set to the GDBF decoder, which can to be analyzed efficiently than other more complex MP decoders. We show trapping sets that GDBF can correct while Gallager A\B and BF algorithms cannot and contrarily.

Definition 1. *A variable node is regarded to be eventually correct if there exists a positive integer l_s such that for all $l \geq l_s, \hat{x}_i^{(l)} = 0$.*

We assume that the all-zero codeword was transmitted and this legitimate assumption for Gallager A\B, the Bit Flipping, BP algorithms operating over BSC [87]. Under this assumption, a variable node is correct if it is 0 and corrupt if it is 1. Let $\mathbf{F}(\mathbf{y})$ is the set of variable nodes that are not eventually correct.

Definition 2. *In a Tanner graph TG and for an iterative decoding algorithm, a trapping set is a non-empty set of variable nodes that are not eventually correct. A set of variable nodes \mathbf{T} is called an (a, b) trapping set if it contains \underline{a} variable nodes and the subgraph induced by these variable nodes has \underline{b} odd-degree check nodes [107].*

Definition 3. For transmission over the BSC, \mathbf{y}' is a fixed point of the decoding algorithm if and only if there exists a positive integer l_s such that $\text{supp}(\mathbf{y}') = \text{supp}(\hat{\mathbf{x}}^{(l)})$ for all $l \geq l_s$. If $\mathbf{F}(\mathbf{y}) \neq \emptyset$ and \mathbf{y}' is a fixed point, then $\mathbf{F}(\mathbf{y}) = \text{supp}(\mathbf{y}')$ is called a fixed set.

For the BF and Gallager A\B algorithms the adequate and essential requirements are defined for a set of variable nodes to form a fixed set given by theorem [96].

Theorem 1. Let C be an LDPC code with d_v -left regular graph TG. Let \mathcal{T} be a subset of a variable nodes with induced subgraph \mathcal{T} . Let the checks in \mathcal{T} be partitioned into two disjoint subsets; \mathcal{O} consisting of checks with odd degree and \mathcal{E} consisting of checks with even degree. Then \mathcal{T} is a fixed set for the Gallager A/B algorithm iff: (I) Every variable node in \mathcal{T} has at least $\lfloor \frac{d_v}{2} \rfloor$ neighbors in \mathcal{E} and (II) No $\lfloor \frac{d_v}{2} \rfloor$ of \mathcal{O} share a neighbor outside \mathcal{T} .

The previous Theorem is used to generate the trapping set ontology (TSO) which is a database of trapping sets that is organized as a hierarchy based on their topological relations such that is used to define relevant trapping sets independent of a given code [108].

The main purpose is to obstructions that forbid Algorithm 1 to coincide with soft way. Analyzing this problem will redound to develop the algorithm with different versions as we will show in the next.

4.4.1.1. Motivating Examples

We will take some examples to show how the general GDBF can correct error pattern with small number of iterations and its failure for another error patterns. Let \circ denotes a correct variable node while \bullet a corrupt variable node at the end of the $(l-1)$ th iteration, and \square denotes a satisfied check node and \blacksquare an unsatisfied check node at beginning of l th iteration. Let C be a regular LDPC code with column-weight $\gamma=3$ and $g=8$.

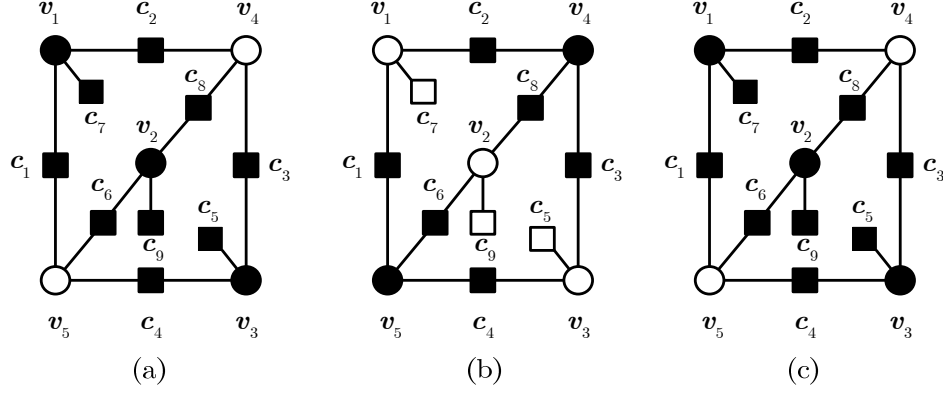


Figure 4.5. Weight-three error configurations uncorrectable by BF during 3 iterations

Figure 4.5 presents an error pattern with weight-three error and v_1 , v_2 , and v_3 are initially corrupt variables and v_4 and v_5 are initially correct variables and the BF decoding is employed. At the beginning of the decoding c_1 , c_2 , c_3 , c_4 , c_5 , c_6 , c_7 , c_8 and c_9 are unsatisfied check nodes as shown in Figure 4.5 (a). $\psi_{\text{us}}^1(v_1) = \psi_{\text{us}}^1(v_2) = \psi_{\text{us}}^1(v_3) = 3 \geq \Delta_v$, where $\Delta_v = 2$, so these variable are flipped to become correct variable nodes. On the other hand, the variable v_4 and v_5 are flipped to become corrupt variable nodes because of $\psi_{\text{us}}^1(v_4) = \psi_{\text{us}}^1(v_5) = 3 \geq \Delta_v$ at the end of the first iteration, as illustrated in Figure 4.5 (a). In the second iteration as shown in Figure 4.5 (b), $\psi_{\text{us}}^2(v_1) = \psi_{\text{us}}^2(v_2) = \psi_{\text{us}}^2(v_3) = 2 \geq \Delta_v$ and $\psi_{\text{us}}^2(v_4) = \psi_{\text{us}}^2(v_5) = 3 \geq \Delta_v$, so the procedure is repeated and all variables are flipped to return to the initial state with the same error pattern as in shown in Figure 4.5 (c). The set of corrupt and correct variable nodes after decoding process alternate between $\{v_1, v_2, v_3\}$ and $\{v_4, v_5\}$ and finally the decoder cannot converge. The BF algorithm fails to correct this error pattern because it uses only syndrome as a criterion to flip the bit or not.

On the other hand, GDBF algorithm can correct above situation during two iterations, the decoder calculates also number of unsatisfied check nodes for each variable node and adds the sum of the received bit with the estimated bit, $\hat{x}_v^{(l)} \oplus y_v$, to get the $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y})$. At the beginning of the first iteration, $\Lambda_{v_1}^{(1)}(\hat{\mathbf{x}}, \mathbf{y}) = \Lambda_{v_2}^{(1)}(\hat{\mathbf{x}}, \mathbf{y}) = \Lambda_{v_3}^{(1)}(\hat{\mathbf{x}}, \mathbf{y}) = 3$ and also $\Lambda_{v_4}^{(1)}(\hat{\mathbf{x}}, \mathbf{y}) = \Lambda_{v_5}^{(1)}(\hat{\mathbf{x}}, \mathbf{y}) = 3$, so all variable nodes are flipped at end of the first iteration. For the second iteration, there are two corrupt variable nodes v_4 and v_5 and three correct variable nodes v_1 , v_2 , and v_3 , so $\Lambda_{v_1}^{(1)}(\hat{\mathbf{x}}, \mathbf{y}) = \Lambda_{v_2}^{(1)}(\hat{\mathbf{x}}, \mathbf{y}) = \Lambda_{v_3}^{(1)}(\hat{\mathbf{x}}, \mathbf{y}) = 3$ and $\Lambda_{v_4}^{(2)}(\hat{\mathbf{x}}, \mathbf{y}) = \Lambda_{v_5}^{(2)}(\hat{\mathbf{x}}, \mathbf{y}) = 4$ and decoder flips only v_4 and v_5 with maximum lambda. The GDBF succeeds to correct the above situation only in two iterations Figure 4.6.

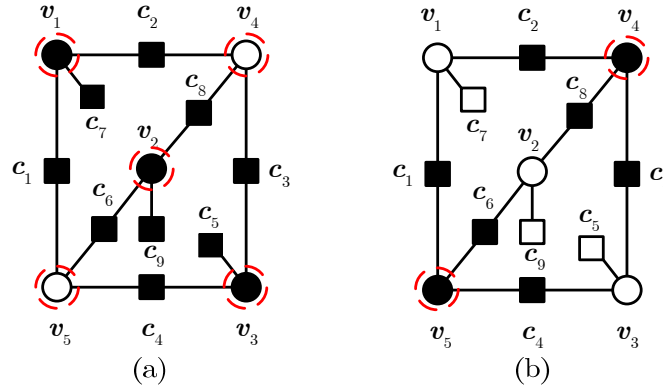


Figure 4.6. Weight-three error configurations correctable by general GDBF during two iterations

Unfortunately, we cannot say that GDBF is capable of correcting any weight-three error pattern in a girth $g=8$. The proof of this note can be illustrated by the next example. Let use the same error pattern but we consider that v_1 , v_4 , and v_5 are initial corrupt variable nodes and v_2 and v_3 are initial correct variable nodes as shown in Figure 4.7 (a). At the beginning of the first iteration we have $\Lambda_{v_2}^{(1)}(\hat{\mathbf{x}}, \mathbf{y}) = \Lambda_{v_3}^{(1)}(\hat{\mathbf{x}}, \mathbf{y}) = \Lambda_{v_4}^{(1)}(\hat{\mathbf{x}}, \mathbf{y}) = \Lambda_{v_5}^{(1)}(\hat{\mathbf{x}}, \mathbf{y}) = 2$ and $\Lambda_{v_1}^{(1)}(\hat{\mathbf{x}}, \mathbf{y}) = 1$, so the decoder will flip all variable nodes with maximum lambda=2. At the second iteration an inverse function for each variable nodes as: $\Lambda_{v_2}^{(2)}(\hat{\mathbf{x}}, \mathbf{y}) = \Lambda_{v_3}^{(2)}(\hat{\mathbf{x}}, \mathbf{y}) = \Lambda_{v_4}^{(2)}(\hat{\mathbf{x}}, \mathbf{y}) = \Lambda_{v_5}^{(2)}(\hat{\mathbf{x}}, \mathbf{y}) = 4$ and $\Lambda_{v_1}^{(2)}(\hat{\mathbf{x}}, \mathbf{y}) = 3$, so decoder flips all variable nodes with maximum inverse function, $\Lambda_v^{(2)}(\hat{\mathbf{x}}, \mathbf{y}) = 4$, Figure 4.7 (b). After flipping the variable nodes the situation returns to the initial case with the same corrupt and correct variable nodes, the same situation for error pattern in Figure 4.8. Finally, the GDBF decoder fails to correct these error patterns which can be corrected by Gallager A\B decoder which explains why Gallager A\B has superior performance than GDBF algorithm in the error-floor region. Although that GDBF algorithm cannot correct these patterns, we will see in the next how we can solve this problem and maybe introduce feature of this decoder for tolerant the fault caused by the noise. Some error patterns are presented in Figure 4.9 and Figure 4.10 to show how GDBF based on binary values can correct the errors or not.

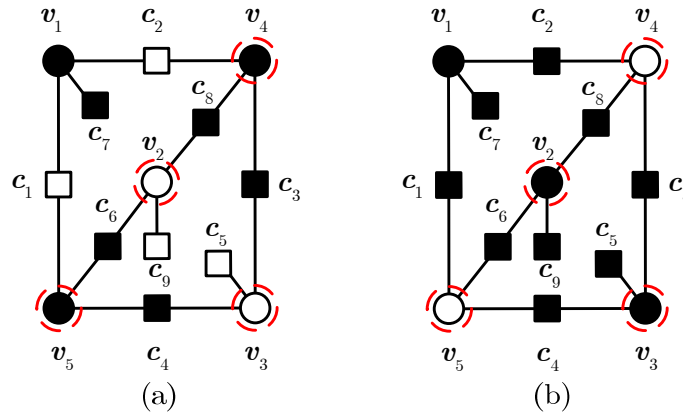


Figure 4.7. Weight-three error configurations correctable by Gallager A\B decoder and uncorrectable by GDBF algorithm.

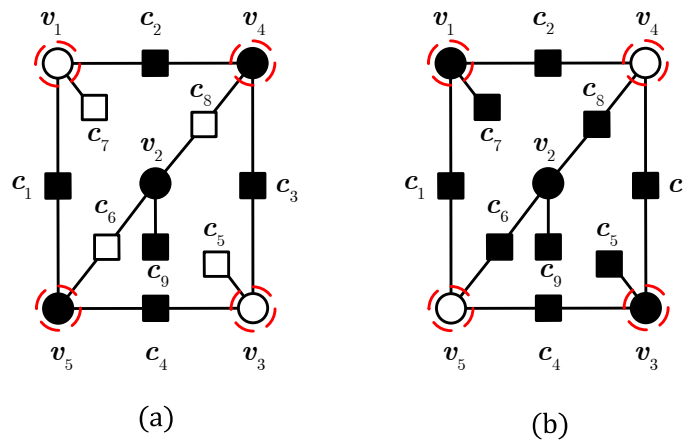


Figure 4.8. Weight-three error configurations correctable by Gallager A\B decoder and uncorrectable by GDBF algorithm.

In Figure 4.9, four bit error pattern is presented and cannot be corrected by GDBF algorithm. In the first iteration only v_3 has two unsatisfied check nodes and has to be flipped. In the next iteration there are three variable nodes with one unsatisfied check node but only v_3 has the value different from the value initially received from the channel. As a general conclude the maximum inverse function gets stuck only on the variable node v_3 in two successive iterations and the same variable node v_3 is flipped for every iteration and GDBF algorithm fails to correct the trapping set and it can be considered as a fixed set according to the Definition 3.

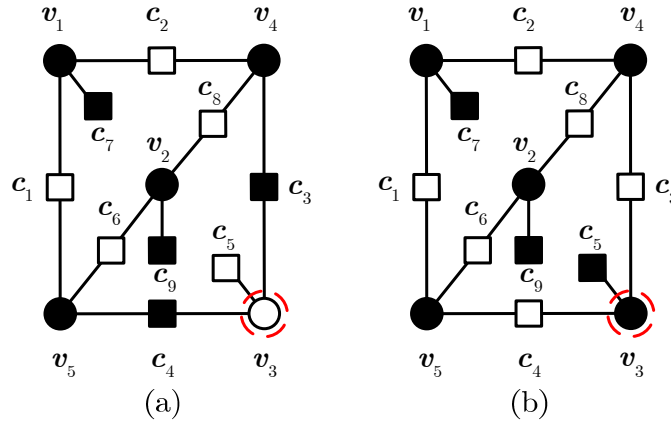


Figure 4.9. Weight-four error configurations uncorrectable by GDBF algorithm where $b^{(l)}$ gets stuck only on variable v_3 for each iteration.

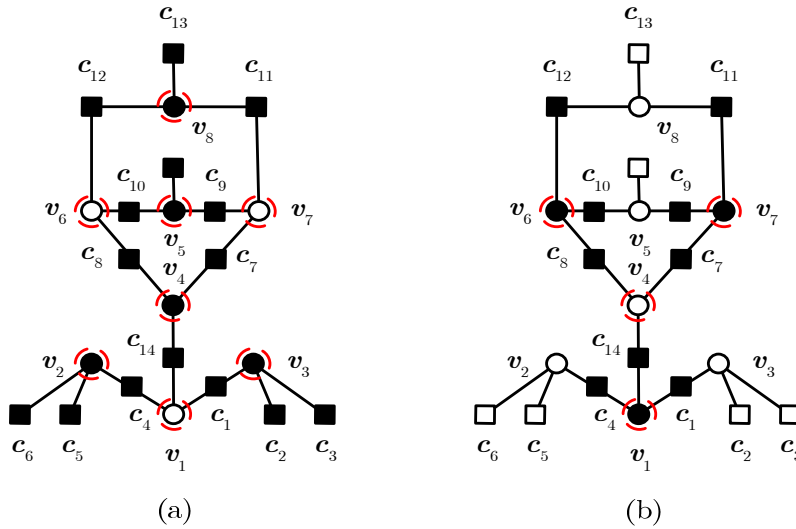


Figure 4.10. Weight-five error configurations correctable by GDBF algorithm.

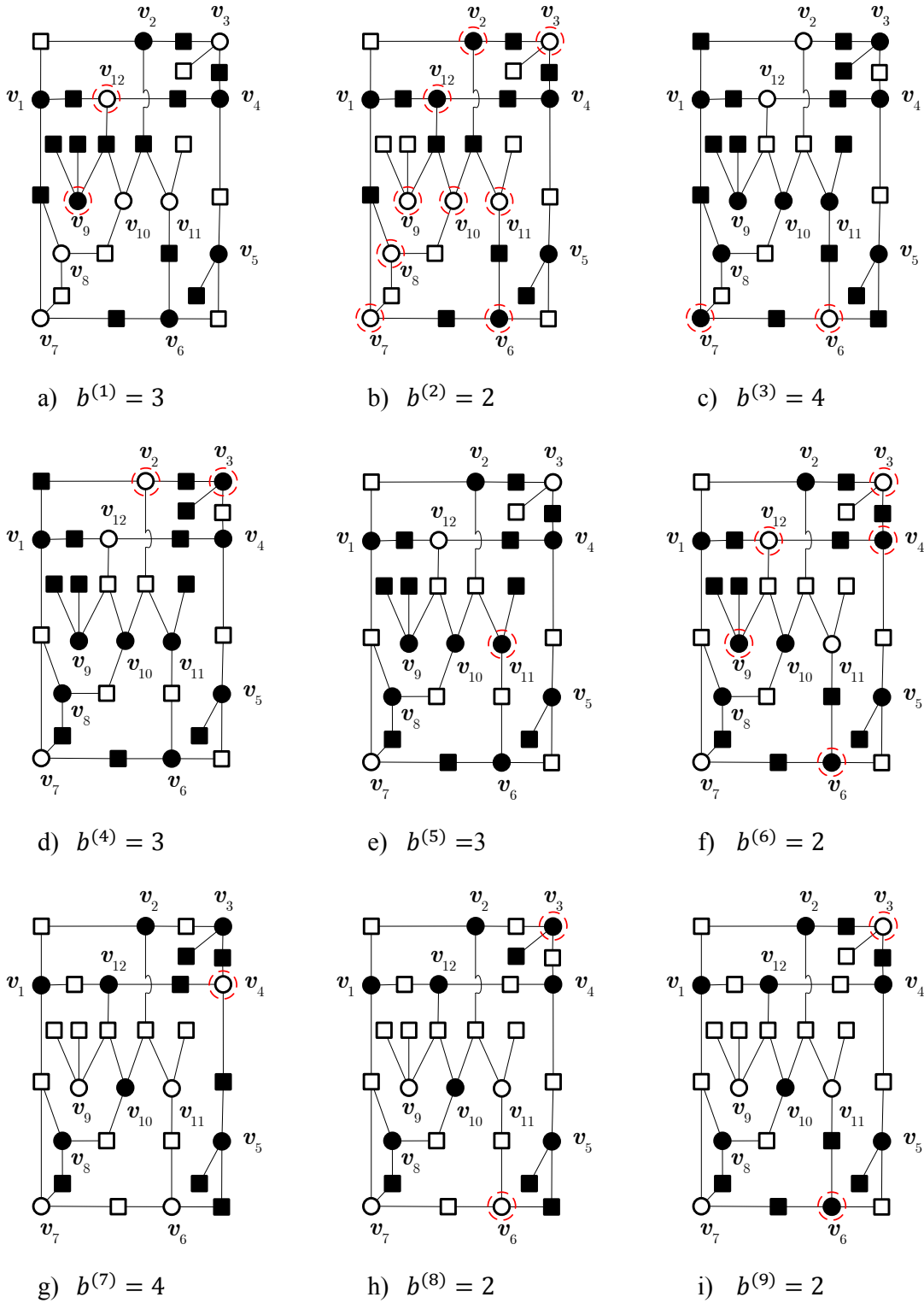


Figure 4.11 Weight-six error configurations uncorrectable by GDBF algorithm where $b^{(l)}$ gets stuck on variables v_6 and v_8 after 9 iterations

4.4.2. Probabilistic GDBF decoder for BSC

We have seen that GDBF algorithm strategy over BSC creates periodic error patterns. This is not only the consequence of Algorithm 1 but also of presence of short cycles in the code graph. Because of the deterministic nature of Algorithm 1 and at some points of the decoding, the set of some variable nodes is repeated after a number of iterations and Algorithm 1 is trapped into an infinite periodic pattern, which cannot be resolved whatever number of iterations. To improve performance of this algorithm it is logically to overcome the drawbacks which make the GDBF algorithm fails to break some error patterns. The stagnancy of GDBF algorithm at some points of the decoding can be fractured by flipping some variable nodes which realize the condition $\Delta_v^{(l)}(\hat{\mathbf{x}}, \boldsymbol{\eta}) = b^{(l)}$. The proposed algorithm is called *Probabilistic GDBF* algorithm (PGDBF) where a particular bit x_v , for which $\Delta_v^{(l)}(\hat{\mathbf{x}}, \boldsymbol{\eta}) = b^{(l)}$, will not be flipped automatically-instead it will be flipped with a predefined probability $p < 1$. In Algorithm 2 this is done by multiplying the flipping decision with Bernoulli $(1, p)$ random variable a_v . In hardware, it can be realized by adding to each variable node processor one AND gate and a generator of Bernoulli random variables a_v with $\Pr(a_v = 1) = p$, as shown in Figure 4.12.

Algorithm 2 Probabilistic GDBF Algorithm over BSC

```

.....▷same as GDBF
while  $v \leq V$  do
    if  $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) = b^{(l)}$  then
 $\hat{x}_v^{(l+1)} \leftarrow a_v \oplus \hat{x}_v^{(l)}$     flip bit
    else
 $\hat{x}_v^{(l+1)} \leftarrow \hat{x}_v^{(l)}$ 
    end if
     $v \leftarrow v + 1$ 
end while
.....    ▷same as GDBF

```

Our work is motivated by the *Probabilistic BF* algorithm proposed by Miladinovic and Fossorier [101]. In PBF, code bits with a number of unsatisfied check sums higher than a fixed threshold are flipped with some probability, which is adapted throughout the iterations.

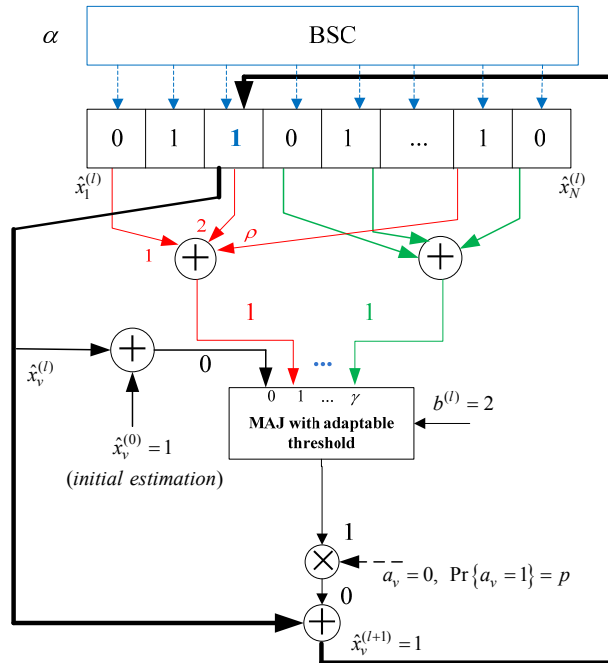


Figure 4.12. Illustration of the variable node processing unit for PGDBF (Algorithm 2)

In PGDBF algorithm, the probability p which defines the flipping criterion, is an independent parameter and does not depend on the crossover probability of the BSC and it is a constant parameter for all iterations. Optimization of parameter p depends on the γ , column weight, of the LDPC code. PGDBF algorithm selects a set of some variable nodes which need to be flipped. It is possible that these nodes are correct or incorrect, so the transaction with partition of the variable nodes will slow down the decoding process especially for error pattern which is corrected directly by GDBF algorithm. However, PGDBF uses stochastic method for solving the trapping sets, and this is an important affair to approach the convergence point for codes that have short cycles. On the other hand, PGDBF uses the probabilistic mechanism to change the error pattern to another pattern that can be corrected. Due to its randomness, PGDBF has a big chance to correct an error pattern as the number of iteration increases. In general, PGDBF algorithm can correct all periodic error patterns with sufficient number of iteration under some conditions, except the situation in Figure 4.9 where only one variable node has the maximum value.

The proposed probabilistic decoding algorithm achieves gain in both performance and decoding time in the waterfall and error-floor regions of the error performance curve compared to the Gallager A\B decoder, especially for $\gamma=3$. The following two examples are

intended to give perception on the performance of the PGDBF algorithm and the performance gain that can be achieved over GDBF algorithm for BSC.

Let us consider a three-bit error pattern shown in Figure 4.7, if GDBF is applied (for which $p=1$), the largest inverse function value $b^{(1)} = 2$ is associated with variable nodes v_2, v_3, v_4 and v_5 in the first iteration, for the second iteration the $b^{(2)} = 4$ is associated with the same variable nodes, as presented in Figure 4.5 (b). Note that these nodes have different value compared to the initial values. As it results in the fixed set Definition 3, this error pattern cannot be corrected by the GDBF algorithm. Now if the PGDBF is applied, the four bits with the largest MIF are not flipped automatically but are only the *candidates for flipping*. As a best solution to solve this pattern is by selection only the corrupt variable nodes v_4 and v_5 which have $b^{(1)} = 2$, and successful decoding can be resulted only exactly after two iterations. Let denote v_4 and v_5 as a *flipping sequence* that results in a successful decoding after two iterations and let s_l the probability that a given error pattern is successfully decoded in the l -th iteration. In our example, probability of flipping sequence $f = (f_2, f_1) = ((1), (0,0,1,1))$ is $s_2 = p^3(1-p)^2$.

It is clear that $s_1 = 0$, as this pattern cannot be corrected in the first iteration. Note that other flipping choices resulting in different flipping sequences might lead to the successful decoding but, possibly, in a larger number of iterations. We refer to such flipping sequences as suboptimal. In our case this number of iterations is $l > 2$. As there may be many suboptimal flipping sequences, the closed form expression for s_l is complicate to obtain. However, its numerical value can be easily estimated by using Monte Carlo simulation. The probability of unsuccessful decoding at the L iteration is obtained as:

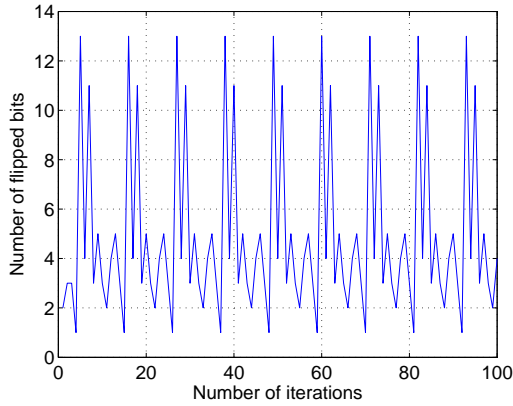
$$p_{PGDBF}(L) = 1 - \sum_{l=1}^L s_l \quad (4.9)$$

The second example is illustrated in the Figure 4.13 and Figure 4.14, we take error pattern on the Tanner code (155,64) where the corrupt error nodes are $v_1, v_{37}, v_{98}, v_{147}$ and v_{152} . Algorithm 1 cannot correct this pattern and the same procedure of bit flipping is repeated for some number of iterations over and over but without any usefulness as shown in Figure 4.14 (a). If we use Algorithm 2 to correct this pattern there is a variety of choices how the Algorithm 2 selects the path (flipping sequence) in order to converge. If the selection of flipping candidates is optimal in the beginning of decoding, the decoder can correct the error pattern in small number of iterations as shown in Figure 4.14 (b), where the number of

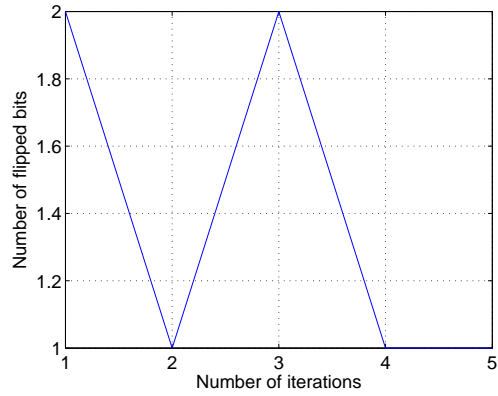
iteration at the successful decoding is 5. Figure 4.13 (c) shown that PGDBF have succeeded to correct this pattern after 28 iterations. In both scenarios, PGDBF have succeeded to converge on a correct message, however, in the last situation, there is some local degradation due to the unsuitable choice of flipped bits. This local degradation does not cause degradation in performance, but prolongs the decoding, as the it number of iterations necessary to find a solution increases. To obtain the better performance with small number of iterations, PGDBF parallel decoders are used in the same time to correct the same error codeword, i.e. all decoders fed with the same input and working in parallel. Maybe some decoders get the convergence in small number of iterations, but we cannot decide the output of that decoder is the correct codeword, i.e. miscorrection. Hence, some decision rules are introduces to find a final decoded word [101]

- *If there is only one codeword among the candidate words, which is final decoded word.*
- *If there is more than one codeword among candidate word, choose the one closet in Hamming distance to the received word.*
- *If none of the candidate words obtained from decoders working in parallel is a codeword, the final decoded word is obtained using majority logic rule for each based on all the candidate words.*

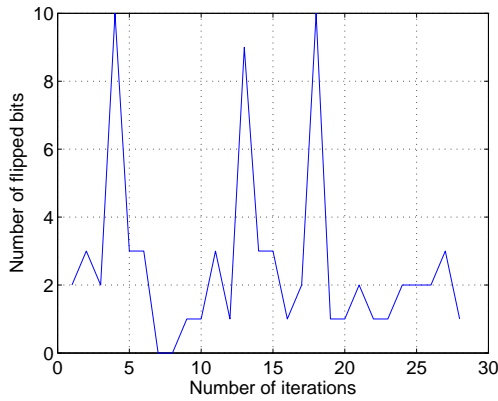
These rules are presented for wholeness of the decoding instruction, but if it is shown that the probability that two decoders having the same sequence as input and working in parallel converge to different codewords at the same time, is relatively small. Moreover, when one decoder converges the decoding stops.



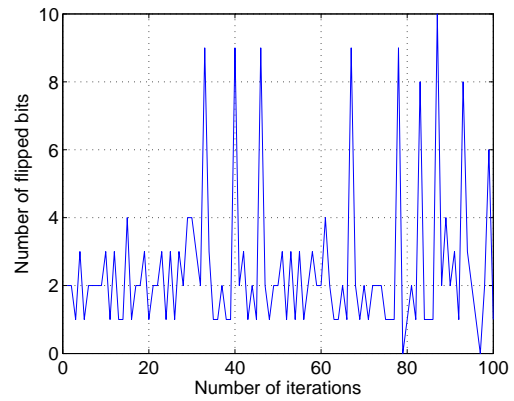
(a) Number of flipped bits vs Number of iteration for GDBF behavior, unsuccessful decoding.



(b) Number of flipped bits vs Number of iteration for PGDBF behavior, successful decoding.

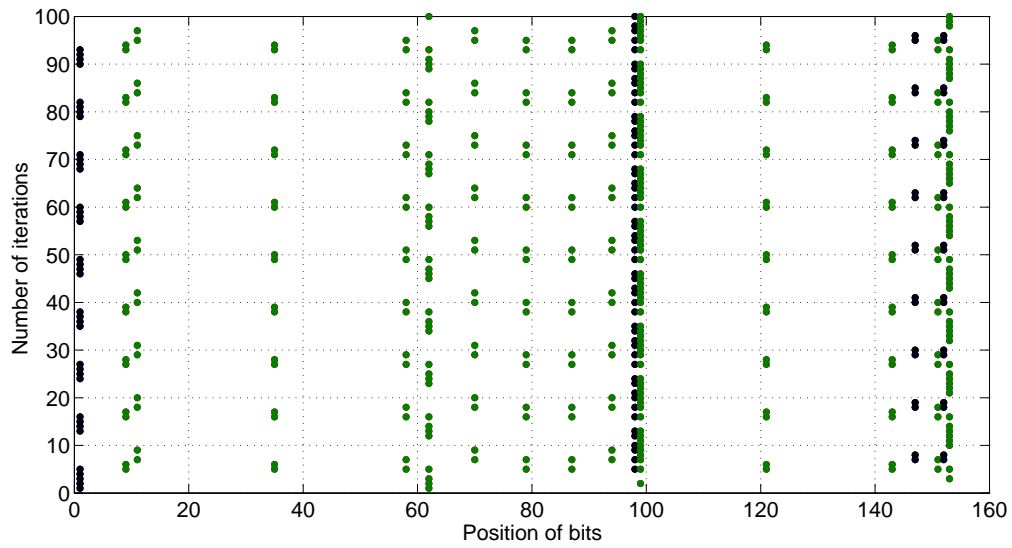


(c) Number of flipped bits vs Number of iteration for PGDBF behavior, successful decoding.

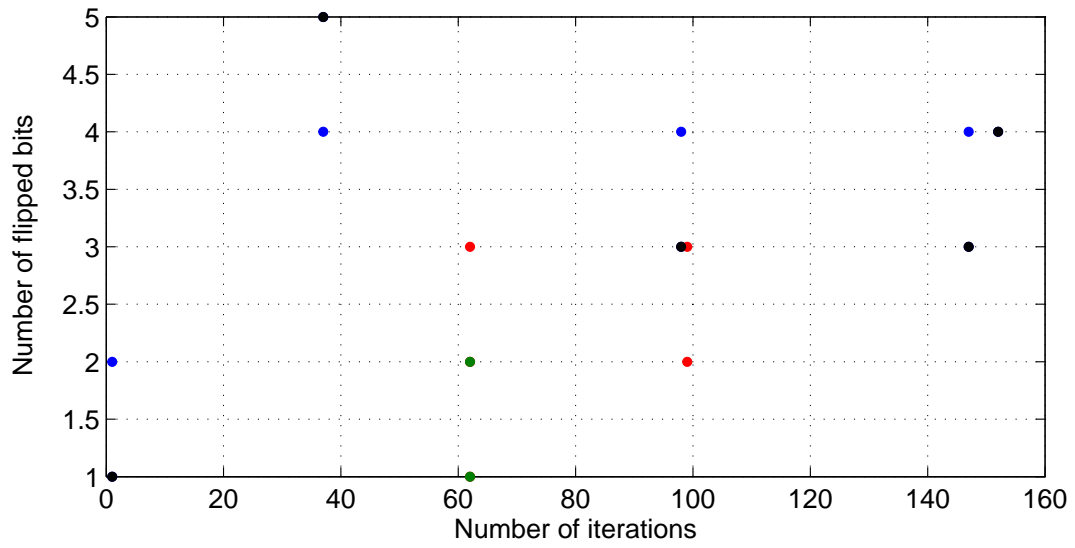


(d) Number of flipped bits vs Number of iteration for PGDBF behavior, unsuccessful decoding.

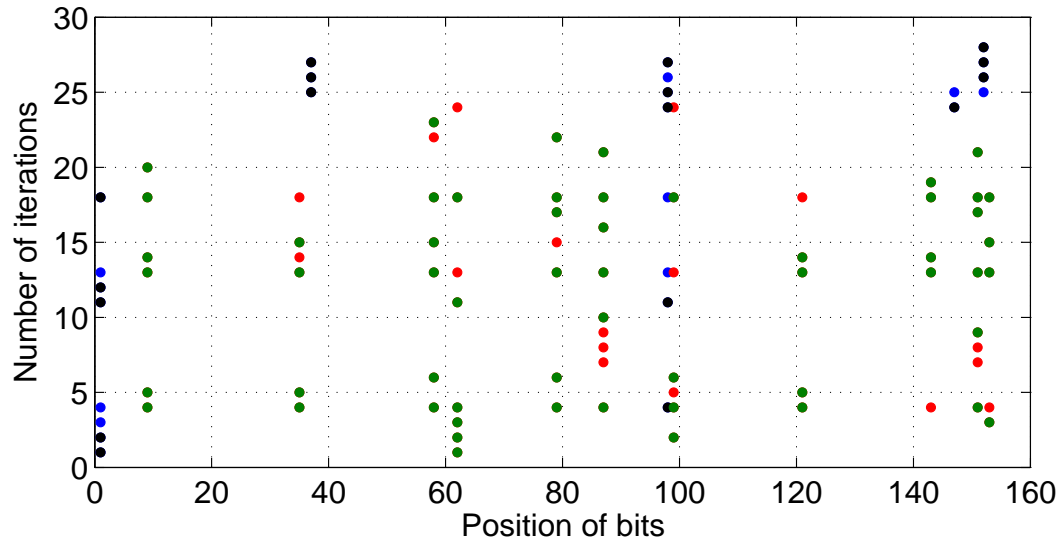
Figure 4.13. Behavior of GDBF and PGDBF algorithms for five-error pattern on the Tanner code (155, 64)



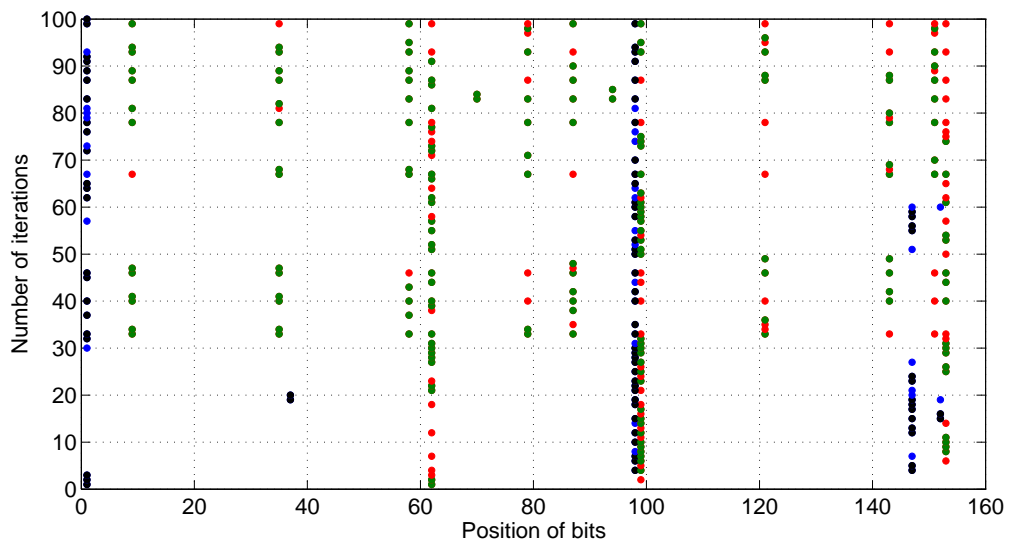
(a) Procedure of bit flipping by Algorithm 1, unsuccessful decoding for 100 iterations



(b) Scenario of bit flipping by Algorithm 2, successful decoding is obtained after 5 iterations



(c) Scenario of bit flipping by Algorithm 2, successful decoding is obtained after 28 iterations



(d) Scenario of bit flipping by Algorithm 2, unsuccessful decoding for 100 iterations

Figure 4.14. Mechanism of GDBF and PGDBF to correct the error patterns on the Tanner code (155, 64)

In Figure 4.14, the black points denote to initial corrupt variable nodes and have the largest MIF value and are in the flipping sequence, and blue points for initial corrupt nodes and have largest MIF value but are not in the flipping sequence. The green and red points are the same situations but for initial correct variable nodes, respectively.

4.4.3. Find optimal p value for PGDBF

Because of the probabilistic nature of the PGDBF algorithm, there are several possible scenarios to solve the trapping sets. In principle, the main idea is to escape from the constant situation of the trapping set by aid Bernoulli $(1, p)$ random variable. In general, to obtain the best performance it is important to select the optimal value p . There exists no general way to optimize the parameter p but can be optimized through Monte-Carlo simulation. In the proposed algorithm, the optimal value of parameter p is constant for all iterations during the decoding process, in contrast to *ProbabilisticBF* algorithm where the parameter p has initial value 0.1 and is increased by step is 0.1 if an estimated codeword equal to the previous estimated codeword. Figure 4.15 shows the FER for the two codes (155,64) Tanner code and QC(732,551) and fixed α , for various values of the parameter p . Numerical results are presented for non-faulty case, it can be observed that the PGDBF decoder has the best performance for $p \approx 0.7$ for two codes with different lengths, where $\gamma=3$ and it can be noticed that PGDBF significantly reduces the FER compared to the GDBF (where $p=1$). On the other hand, parameter p for PGDBF algorithm does not depend significantly on the BSC crossover probability, in contrast to NGDBF where the variance of the noise inserted to the variable nodes has to be approximately equal to the variance of the noise in the channel [103].

In order to estimate the convergence speed of the PGDBF decoder, the average number of iterations is used as an appropriate measure. Figure 4.16 shows the relation between an estimated iterations number for the successful decoding with BSC crossover probability. A PGDBF algorithm is presented with different values of parameter p . It can be observed that for $p = 0.7$, the PGDBF has faster convergence speed compared to other values with better performance. For $p = 1$, it can be noticed that GDBF decoder has faster convergence for a few iterations number and it can be explicated because of the deterministic nature of GBDF decoder in a way hard-decision.

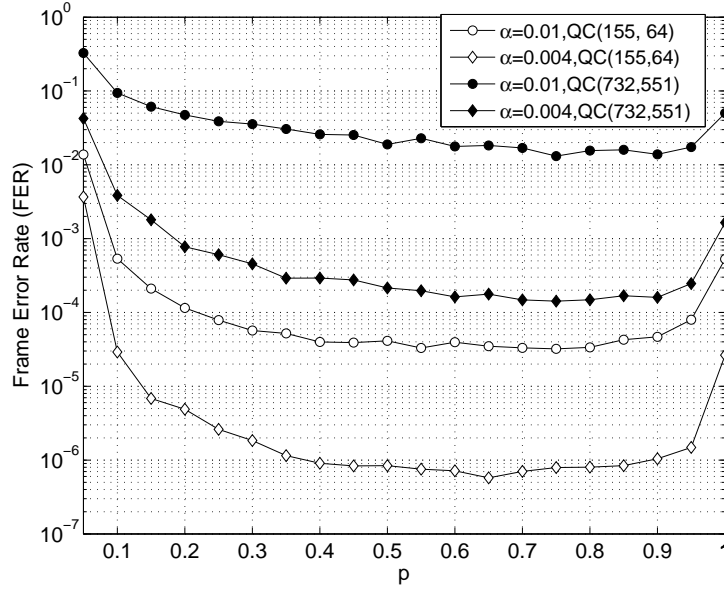


Figure 4.15. Impact of the parameter p on PGDBF optimized for the BSC. The plot is for the (155,64) Tanner code and QC (732,551), $\alpha=4 \times 10^{-3}$, $\alpha=10^{-2}$ and $L=100$

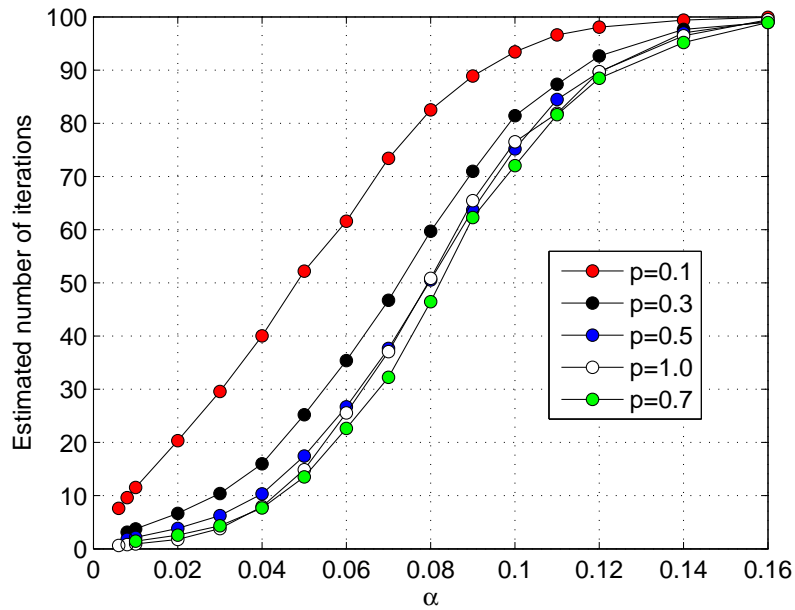


Figure 4.16. Performance of PGDBF algorithm for the (155,64) Tanner code with different p values

4.4.4. The Multiple Decoding attempts and Random re-Initializations (MUDRI) algorithm

We have seen from above that some bad choices of the bits which have the largest value of MID lead to unsuccessful decoding. Wrong choices in the beginning of the decoding process will prolong the decoding and/or reduce the probability that will be finish with success. In general, the decoding process will have a good chance to correct codeword errors as the number of iterations increases. By using equation (4.9) we are able to estimate $p_{PGDBF}(L)$ for any error pattern. However, there are some error patterns which have high values of $p_{PGDBF}(L)$ even for high values of L , and one such error pattern is shown in Figure 4.18 with the same situation in Figure 4.7 and 4.8. In the first iteration, $b^{(1)} = 3$ is associated with the variable nodes v_2, v_4 and v_5 . The PGDBF update rule allows an independent flipping of all these variables (2^3 possible choices), but only some of them are actually flipped. If only v_5 is flipped (with the probability $p(1-p)^2$), the error pattern at the beginning of the second iteration looks like the one shown in Figure 4.18 (b). In this case, $b^{(2)} = 2$ and six bits are considered for flipping, with 2^6 possibilities for the flipping choices in this step. If only the bits that are incorrectly received are chosen for flipping (v_1, v_3, v_6 and v_7), with the probability $p^4(1-p)^2$, the decoding process is successfully completed. As only one flipping sequence results in decoding after two iterations, the corresponding probability is obtained by multiplying the probabilities in two successive steps as $s_2 = p^5(1-p)^4$.

However, if a wrong choices are made in a few iterations at the beginning of decoding, it does not have to be completed successfully even for large value of L . Therefore, we propose the modification of the PGDBF algorithm. If the syndrome has non-zero value after L_1 iterations, the decoding is stopped and repeated $\lfloor L/L_1 \rfloor$ times starting from the received word for the other flipping random choices Figure 4. 17.

If the random sequences are independent, the probability that decoding fails is

$$p_{MUDRI}(L, L_1) = \left(1 - \sum_{l=1}^{L_1} s_l \right)^{\lfloor \frac{L}{L_1} \rfloor}. \quad (4.10)$$

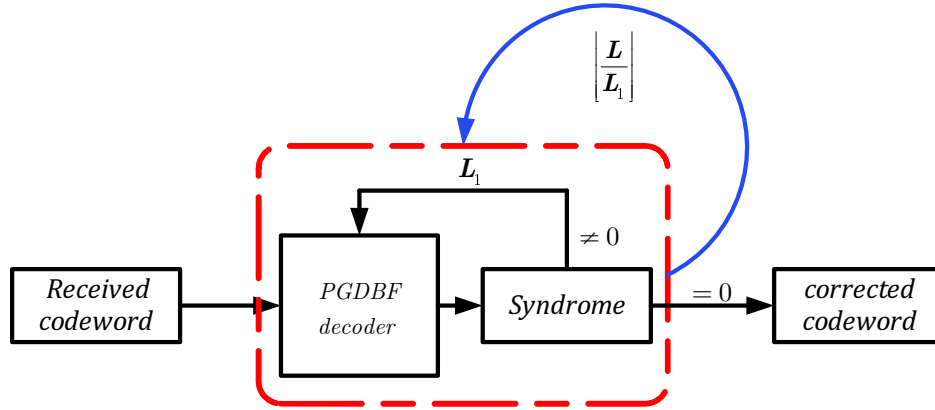


Figure 4.17. MUDRI algorithm scheme

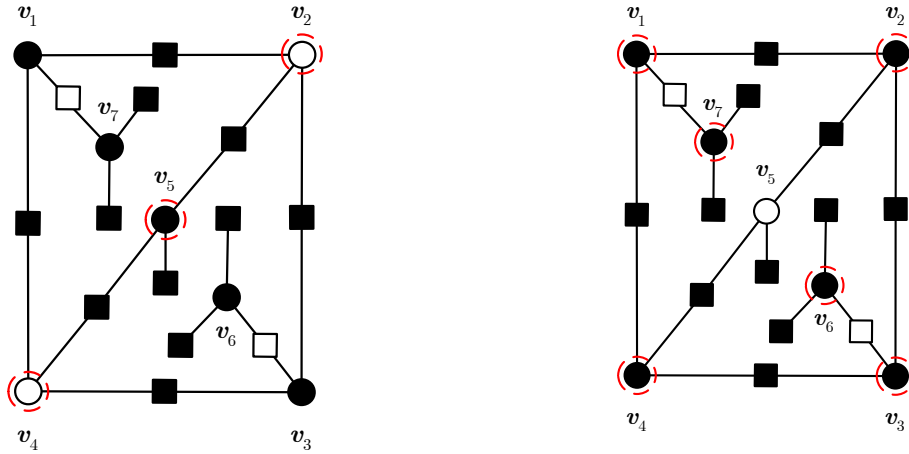
In the special case when $L_1 = L$, we have a single attempt with L iterations, and the above expression reduces to Equation (4.9). The probability of unsuccessful decoding can be minimized with the proper choice of this parameter L_1 .

To return to the Figure 4.9 with the situation that MIF gets stuck on one variable node in every iteration, and used many attempts does not lead to any improvement. In such a situation we propose decrementing the threshold in variable nodes until it reaches the second largest value, i.e.

$$b_{mod}^{(l)} = \max_v \Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}). \quad (4.11)$$

In our example $b_{mod}^{(2)} = 1$, the nodes with $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) \geq 1$ are flipped and the decoding is successful after the second iteration (Figure 4.19 (c)).

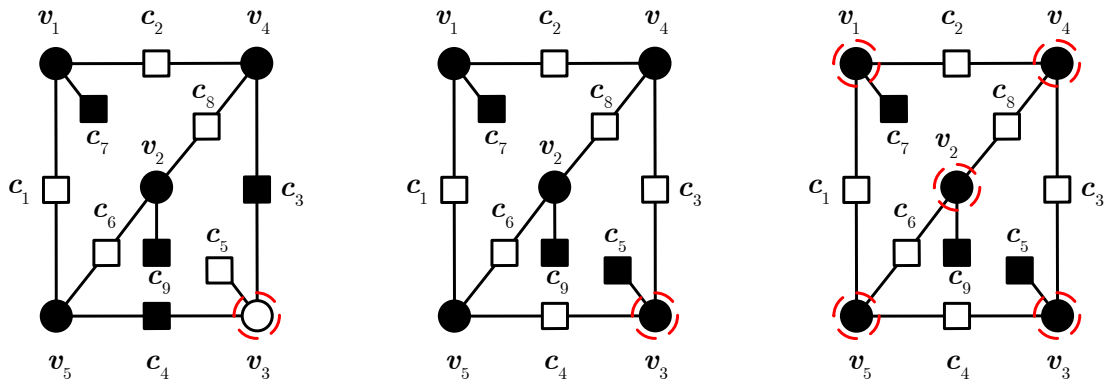
The above modifications are combined with the PGDBF algorithm to obtain the Multiple Decoding attempts and Random re-Initializations (MUDRI) decoding algorithm, formally given in Algorithm 3. The modification is applied under the condition that $in^{(l)} = in^{(l-1)} = 1$ and that in two successive iterations the maximal MIF value corresponding to the same bit in the codeword (denoted by $v_f^{(l)}$).



(a) A five-bit error pattern uncorrectable by using GDBF.

(b) The second iteration of PGDBF, after the first iteration with the optimal choice.

Figure 4.18. Performance of GDBF and PGDBF with optimal choice for five-error pattern



a) A four-bit pattern critical in PGDBF.

b) The second iteration if $b^{(l)} = \max_v(\Lambda_v^{(l)}(x, y))$.

c) The second iteration if $b^{(l)} = \max_v(\Lambda_v^{(l)}(x, y)) - 1$.

Figure 4.19. Adaption method to solve the error pattern by decrease the threshold value

Algorithm 3 MUDRI decoder

Input: \mathbf{y} received word

$$\forall v \in V : \hat{x}_v^{(0)} \leftarrow y_v$$

$$\mathbf{s}^{(0)} \leftarrow \mathbf{x}^{(0)} H^T \left(\forall c \in C : s_c^{(0)} \leftarrow \bigoplus_{u \in \mathcal{N}_c} \hat{x}_u^{(0)} \right) \text{ calculate the syndrome}$$

$$n=0, l=0,$$

while $\mathbf{s}^{(0)} \neq \mathbf{0}$ and $n \leq \lfloor L/L_1 \rfloor$ **do**

$$l=0, in^{(0)} = 0, v_f^{(0)} = 0, \forall v \in V : \hat{x}_v^{(0)} \leftarrow y_v$$

$$\mathbf{s}^{(0)} \leftarrow \mathbf{x}^{(0)} H^T \left(\forall c \in C : s_c^{(0)} \leftarrow \bigoplus_{u \in \mathcal{N}_c} \hat{x}_u^{(0)} \right)$$

while $\mathbf{s}^{(l)} \neq \mathbf{0}$ and $l \leq L_1$ **do** $\forall v \in V$: Compute $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y})$

$$b^{(l)}, in^{(l)}, v_f^{(l)} \leftarrow FUN\left(\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}), in^{(l-1)}, v_f^{(l-1)}\right)$$

for $\forall v \leq N$ **do****if** $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) \geq b^{(l)}$ **then**

$$\hat{x}_v^{(l+1)} \leftarrow a_v \oplus \hat{x}_v^{(l)} \text{ flip bit}$$

else

$$\hat{x}_v^{(l+1)} \leftarrow \hat{x}_v^{(l)}$$

end if**end for**

$$\mathbf{s}^{(l+1)} \leftarrow \mathbf{x}^{(l+1)} H^T \text{ syndrome check}$$

$$l \leftarrow l + 1$$

end while

$$n \leftarrow n + 1$$

end while**Output:** $\hat{\mathbf{x}}^{(l)}$

Algorithm 4 FUN: Adaption of threshold in MAJ gates**Input:** $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}), in^{(l-1)}, v_f^{(l-1)}$

$$b^{(l)} \leftarrow \max_v \left(\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) \right)$$

for $\forall v \in V$ **do****if** $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) = b^{(l)}$ **then**

$$in^{(l)} = in^{(l-1)} + 1$$

$$v_f^{(l)} \leftarrow v$$

end if**end for****if** $l > 1$ and $in^{(l)} = in^{(l-1)} = 1$ and $v_f^{(l)} = v_f^{(l-1)}$ **then**

$$b^{(l)} \leftarrow \max_2 \left(\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y}) \right)$$

end if**4.4.5. Analysis of the MUDRI algorithm**

To evaluate the algorithm performance, we first consider the decoding of the error patterns presented in the mentioned examples, illustrated in Figures 4.7 (a) and Figure 4.18 (a), for the case these patterns appear in the Tanner (155, 64) code. The probability of successful decoding at exactly l iterations is estimated by using Monte Carlo simulation, and the corresponding probability distributions are presented in Figure 4.20.

As expected, the probability that a three-bit error pattern is not successfully decoded steadily decreases with the increase of the parameter L , and we obtain $p_{PGDBF}(100) = 3 \times 10^{-5}$ for the standard PGDBF algorithm. On the contrary, the simulation results show that the five-bit error pattern Figure 4.18 (a) is either corrected in 14 or less iterations, or it cannot be corrected at all ($s_l \approx 0$ for $l > 14$). In this case, the probability of decoding failure is estimated as $p_{PGDBF}(14) = 0.8768$. The increase of L cannot help by itself, but combined with the proposed modification with multiple attempts, it results in lowering probability of unsuccessful decoding. Further optimization of the parameter L_1 also results in lowering p_{MUDRI} , as presented in Figure 4.21. It can be noticed that the best results are obtained for approximately $L_1 = 6$ decoding iterations per attempt. In Figure 4.22, the frame error rate (FER) as a function of number of iterations is presented for $\alpha = 0.01$. Although it is not convenient to adapt parameter L_1 for every error pattern, the simulations indicated that the minimal value of FER (i.e. $p_{MUDRI}(L, L_1)$ average over all received error pattern) is achieved for $L_1 \approx 25$ for Tanner (155,64) code and this parameter is somewhat larger for longer codes.

It is interesting to notice that while the PBF, GDBF and Gallager-B decoders need not more than 30 iterations to converge, after which their FER performance has reached the lowest possible value, the PGDBF continues to improve its FER performance up to 100 iterations and results in significant gain compared to the GDBF. The MUDRI, with ten attempts per each of $L_1 = 25$ iterations, results in an order of magnitude lower FER when compared to the PGDBF. The algorithm performance further improves with the increase of parameter L , to approximately $FER = 6 \times 10^{-7}$ when $L = 2000$.

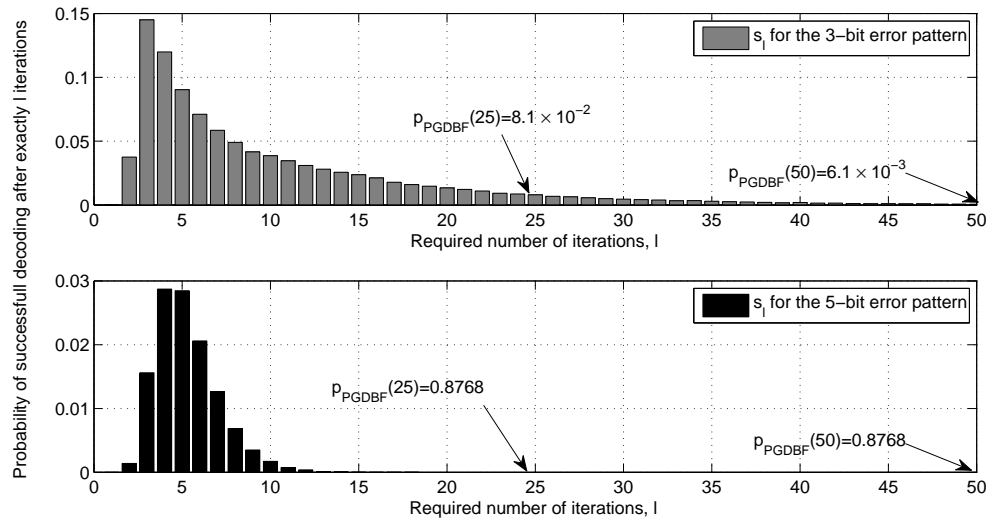


Figure 4.20. Probability distribution of the successful decoding in the l -th iteration of PGDBF, three-bit and five-bit error pattern, Tanner(155,64) code, $p=0.7$

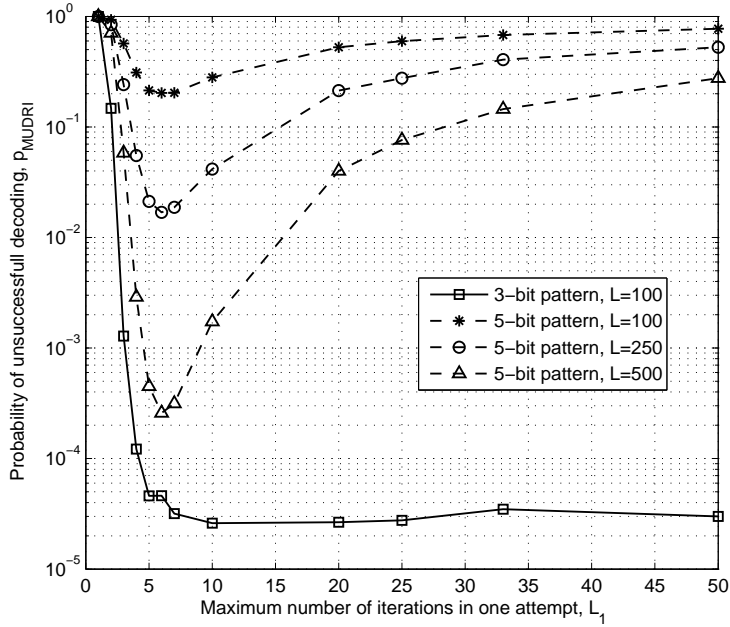


Figure 4.21. Probability of unsuccessful decoding for the three-bit and five-bit error pattern, MUDRI with $\lfloor L/L_1 \rfloor$ attempts per L_1 iterations each, $p=0.7$

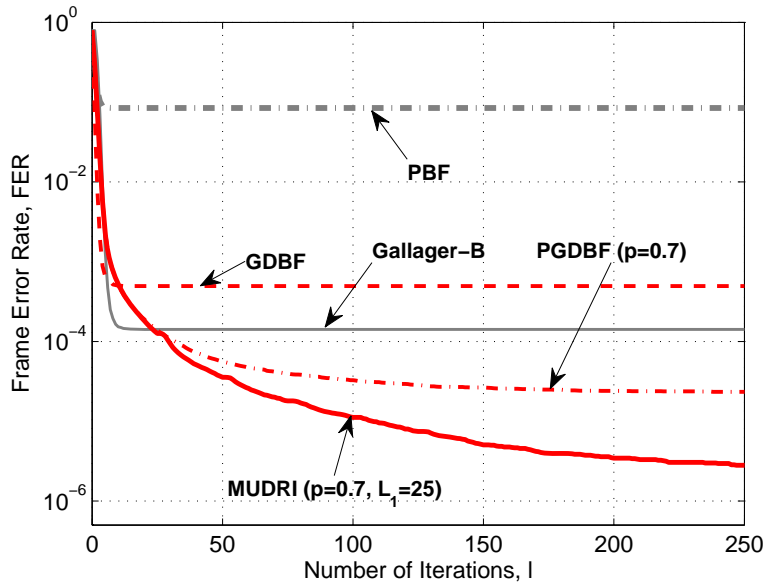


Figure 4.22. FER as a function of number of iteration l , Tanner (155,64) code, $\alpha=0.01$, various decoding algorithms

4.5. Fault-Tolerant PGDBF and MUDRI Decoders for BSC

In general, the error correction coding techniques used with the customary systems of communication are exploited with supposition that the operation of an error correction encoder and decoder are deterministic and noise exists only in the transmission or storage channel. While appropriate in systems where the reliability of registers and logic gates used in the decoder is many orders of magnitude higher than the reliability of the channel, this assumption is invalid if digital logic in the decoder is built of faulty components. Because of the high integration factor of integrated circuits together, low power consumption requirements and variations in the technological process makes MOS and emerging semiconductor devices inherently unreliable [109]. Two main reasons lead to unreliable computing, the first one is declining of the consumption energy such that the level of the signal will be close to the noise level and noise immunity is decreased. The second reason is created when the decoder is built from faulty components, then the errors happening at the gate level affect the operations carried out by the decoder. However, traditional von Neumann-type triple modular redundancy architectures that ensure fault tolerance are inefficient in handling such increased unreliability thus requiring solutions based on error control coding. Recently there was a surge in research in fault-tolerant decoders. Vasic and Chilappagari [110] established an information theoretical framework for analysis and design of faulty decoders for low-density parity-check (LDPC) codes. They have also analyzed bit-flipping decoding [110] or one-step majority logic (MAJ) decoding [111], [112]. Methods for performance analysis of more complex decoders built from unreliable hardware based on the sum-product algorithm (SPA) [113] and its suboptimal (min-sum algorithm) version [114] have been also developed for transient failure model. In the similar context, finite-alphabet decoders (FAID) were analyzed by Huang and Dolecek in [115]. Density evolution analysis of the simplest message-passing algorithm (Gallager-B) implemented in noisy hardware is given in [116] and [117].

With an aim of demonstrating the robustness of the algorithm to the hardware failures, we consider the canonical transient von-Neumann logic gate failure mechanism in which the failures in different gates and in different time instants are independent and identically distributed. The failures manifest themselves as random bit flips at the gate outputs. All XOR gates have probability of failure P_{\oplus} , and failures in the register where $\hat{\mathbf{x}}^{(l)}$ is stored occur with probability P_R . We also assume that MAJ gates are reliable, i.e. $P_{MAJ} \approx 0$, Figure 4.23.

Although optimistic, this can be readily realized by using, for example, larger transistors in MAJ gates.

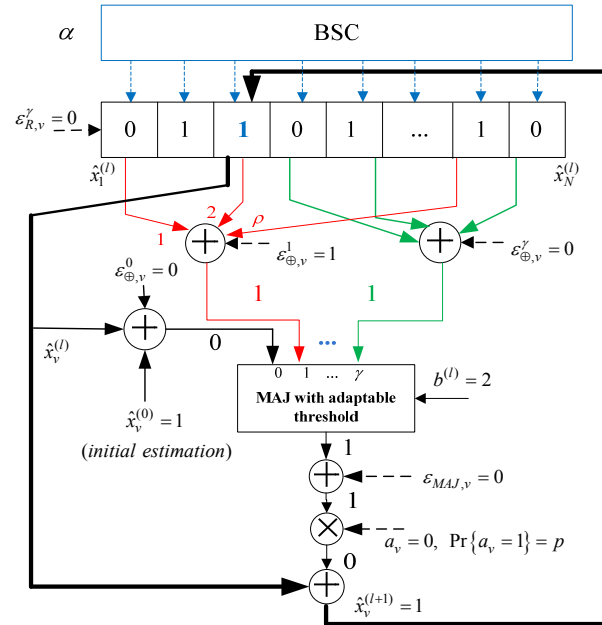


Figure 4.23. Illustration of the variable node processing unit for PGDBF under faulty hardware

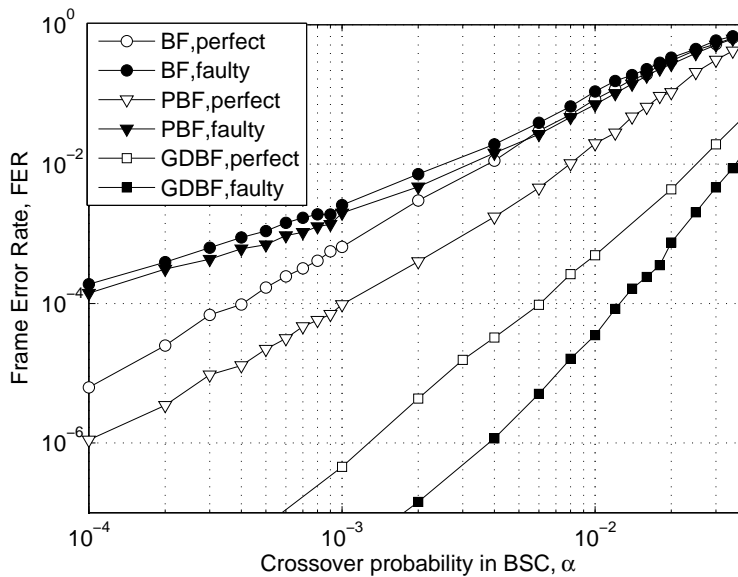


Figure 4.24. FER performance as a function of α under various decoding algorithms, Tanner code (155,64), $P_\oplus=10^{-2}$, $P_R=10^{-3}$, $L=100$

Now we present the numerical results of Monte Carlo simulations, in Figure 4.24, the FER performance of the (155,64) Tanner code is presented for the maximum number of decoding iterations $L = 100$. The results are presented for the BF, PBF and GDBF algorithms. In the case of non-faulty hardware in the decoder, GDBF algorithm results in lower FER values compared to the BF and PBF algorithms. As expected, the performances of faulty BF and PBF decoders are significantly degraded. For the case when $P_{\oplus}=10^{-2}$, $P_R=10^{-3}$, the performance of the two are approximately the same. On the other hand, for the same failure rates, performance of the GDBF is improved compared to the non-faulty decoder case! This surprising effect is related to the finite set of possible values of $\Lambda_v^{(l)}(\hat{\mathbf{x}}, \mathbf{y})$.

For longer codes, we present the FER performance of two codes with similar codeword lengths but different column weight. The performance of (2388, 1793) code (code C_1) with girth-8 and $\gamma = 3$ based on Latin Squares [134] and (2212, 1880) code (code C_2) with girth-6 and $\gamma = 4$ are determined as a function of parameter p for $L = 100$ and $L_1 = 50$ and presented in Figure 4.25. When $p = 1$, the algorithm realized in faulty hardware has lower FER than the algorithm implemented in perfect hardware, and the performance can be further improved by reducing the parameter p for both codes. For C_1 the best performance is obtained for $p \approx 0.7$ in the non-faulty case, while the lowest FER is obtained for $p \approx 0.8$ when $P_{\oplus} = 10^{-3}$, and $P_R = 10^{-4}$. This corresponds to the previously results for short quasi-cyclic codes with girth-8 and $\gamma = 3$. For C_2 (with $\gamma = 4$), the best performance are obtained if $p \approx 0.9$ for the non-faulty case, while for the faulty implementation the optimum value of p is slightly larger.

The FER performance of QC and LS codes with various code rates are presented in Figure 4.26, as a function of the parameter P_{\oplus} . If $p = 1$, the best performance is achieved for the non-zero value of P_{\oplus} . On the other hand, if $p = 0.7$ the FER is significantly reduced for small values of P_{\oplus} , when compared to the $p = 1$ case. More importantly, when $p = 0.7$ the FER is almost insensitive to P_{\oplus} in a wide range of P_{\oplus} values, up to a certain threshold, and is dominantly determined by the codeword length. The threshold can be estimated as $P_{\oplus,th} = 5/N$ for the codes with $\gamma=3$ and girth-8.

In Figure 4.27, we present the FER performance for five LDPC codes with various code constructions (QC, PEG, LS), column weights and codeword lengths (available in [118]), and for the case when $\alpha = 0.008$, $P_{\oplus} = 10^{-3}$ and $p = 0.7$. It is clear that the MUDRI decoder has approximately same performance, up to a certain threshold of P_R . The value of P_R where FER doubles with respect to the non-faulty case is dominantly determined by the codeword length. For the codes with $\gamma = 3$ and girth-8, this threshold is estimated to be $P_{R,th} =$

$1/(2N)$. Although the codes with $\gamma = 4$ and girth-6 have lower error correction capability, they are somewhat less sensitive to the logic gate failures.

The FER performance of C_1 for various decoders is presented in Figure 4.28. It can be noticed that the Gallager-B outperforms the GDBF for lower values of crossover probability in BSC channel and the GDBF is more effective in the water fall region. In the presence of gate failures, the performance is degraded for the Gallager-B decoder, but is improved for the GDBF ($p = 1$). The performance of MUDRI with $p = 0.7$ outperforms all hard decision algorithms for the analyzed crossover probability, and the increase of the parameter L results in additional performance improvement. In addition, the MUDRI is less sensitive to hardware failures when compared to the Gallager-B and PGDBF.

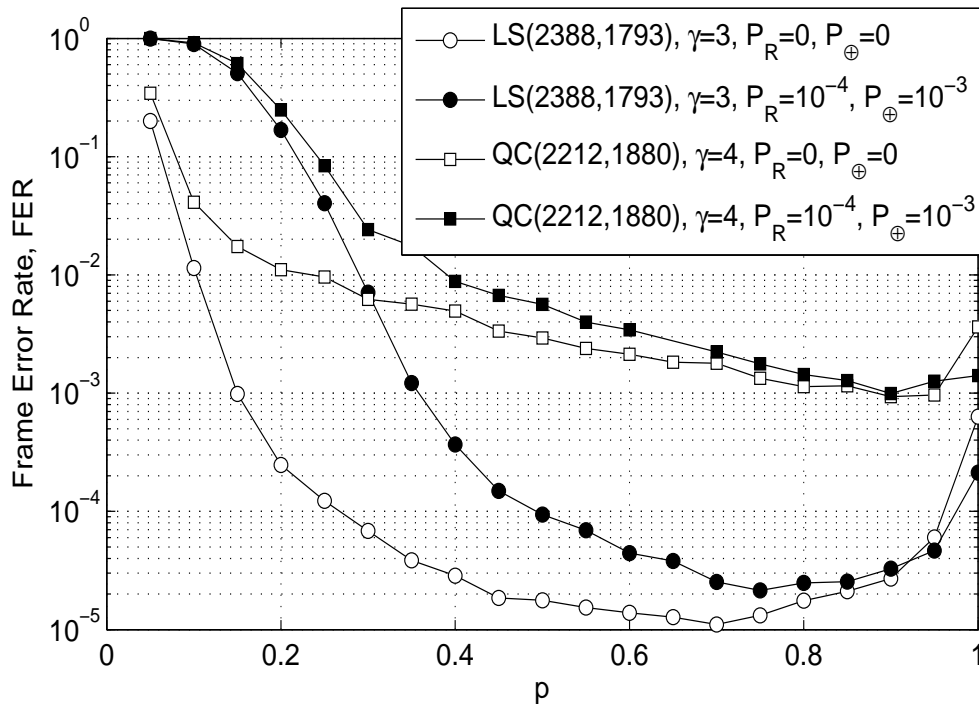


Figure 4.25. FER as function of parameter p , LDPC codes with $\gamma=3$ and $\gamma=4$, $\alpha=0.004$

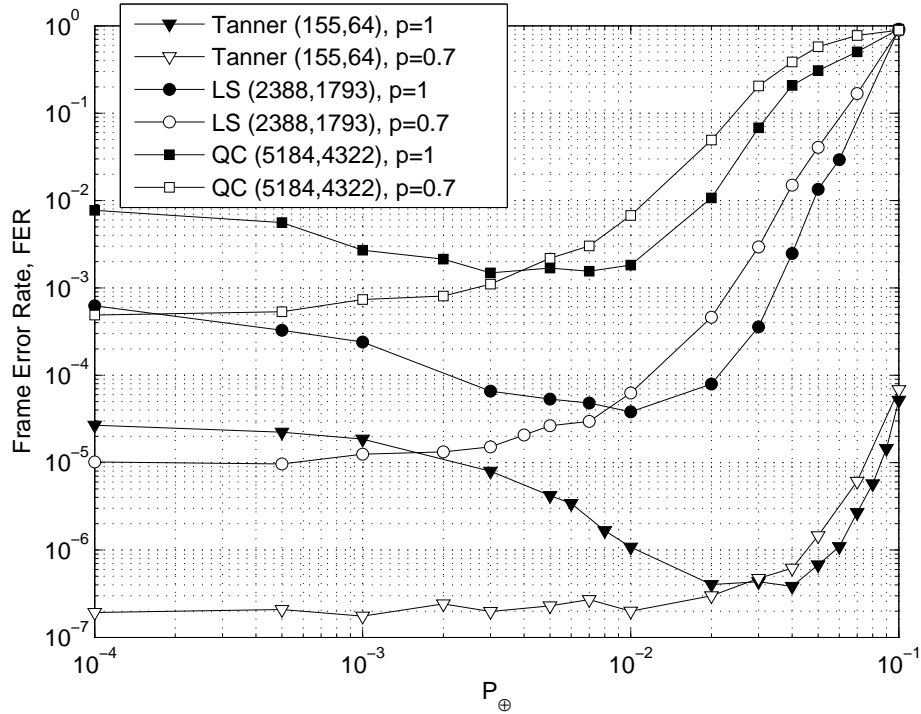


Figure 4.26. FER as a function of probability of error XOR gates, $\alpha=0.004$, $P_R=0$, LDPC codes with $\gamma=3$ and girth-8, with various code rates and codeword lengths

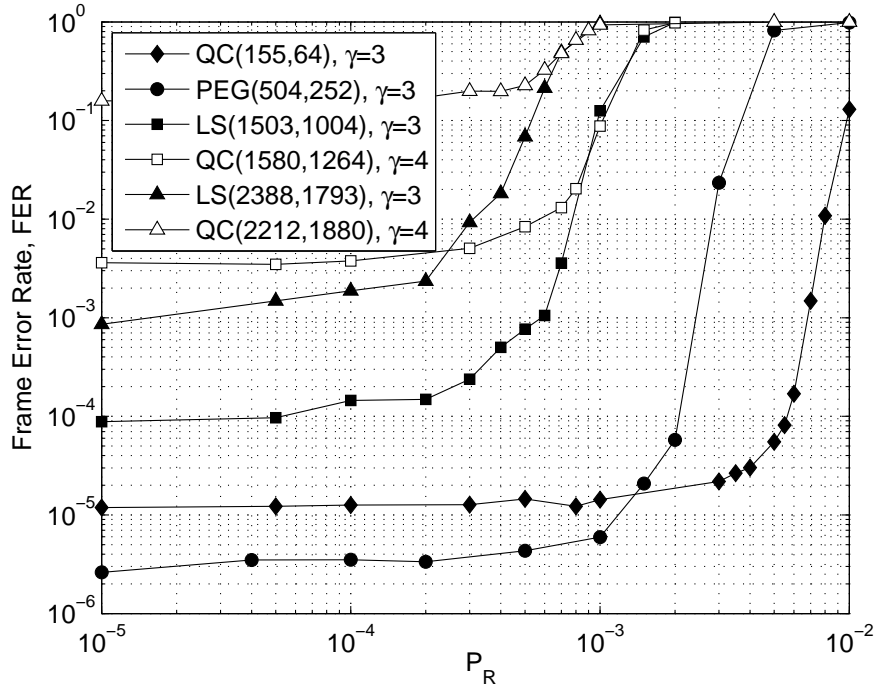


Figure 4.27. FER as a function of probability of error in registers, $\alpha=0.008$, $P_{\oplus}=10^{-3}$, LDPC codes with $\gamma=3$, $\gamma=4$ and girth-8, various codeword lengths

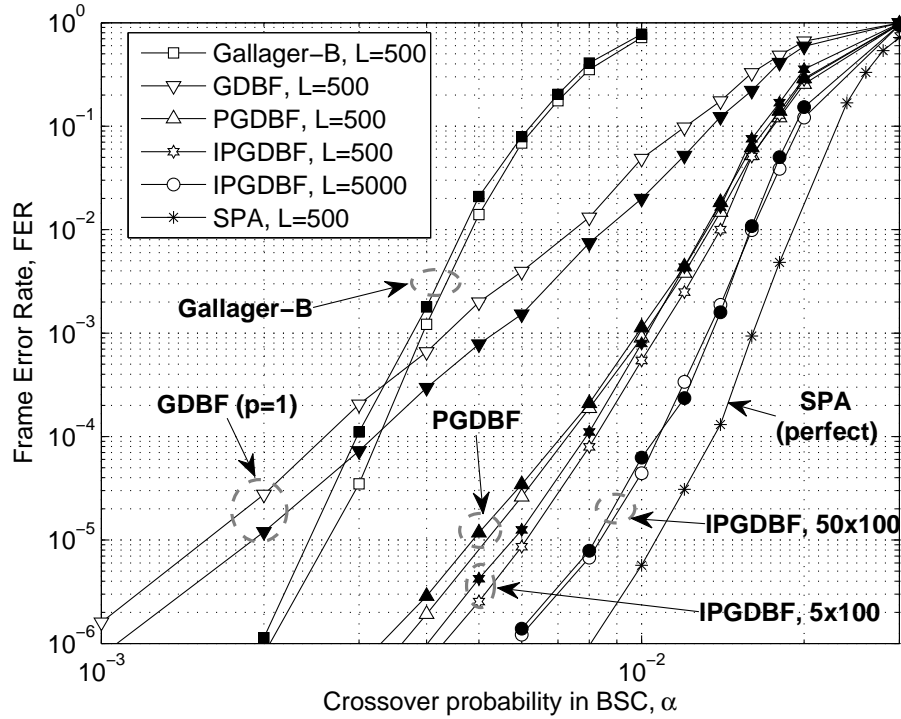


Figure 4.28. FER as a function of crossover probability in BSC channel. The code is LS(2388,1793)(C_1), the empty markers corresponds to perfect hardware and full markers to faulty hardware with $P_{\oplus}=10^{-3}$, $P_p=10^{-4}$, $p=0.7$ in the PGDBF and MUDRI and other decoding algorithms

4.6. Hardware Realization and Complexity of PGDBF Algorithm

As we have seen from the above discussions and the obtained results that improvement of performance for PGDBF can be realized by using the random generator to solve the trapping sets in the Tanner graph. Therefore improvement of the PGDBF decoder needs extra hardware resources where the hardware-exhausted random generators blocks have to be implemented for every variable node processing unit.

Actually there is no problem to implement GDBF decoder designed for BSC in the hardware, but the main problem is how to design random generator that produces the binary sequences with the optimal value of p for the different characteristics of LDPC codes.

In [119], the authors presented several implementations of our proposed algorithm and in this section we just explain their designs of the random generator and obtained performance on GDBF and PGDBF algorithm implementations.

Two models of random generator were presented. The first design uses the linear feedback shift register (LFSR) to obtain the conventional implementation of the random

generator and the second design is called Intrinsic-Value Random Generator (IVRG) such that uses binary sequences produced by the LDPC decoder. They showed that both implementation of the PGDBF improve greatly the error correction performance, while maintaining the large throughput. In the case of LFSR-PGDBF, the performance gain requires large hardware components while in the case of IVRG-PGDBF only the extended expense is only 10%.

The idea of IVRG design is to use the values of the check nodes and interprets these values as a random source of bits. The used values are already created by the existing hardware block of the GDBF decoder. In this way, IVRG design economizes the hardware component compared with the LFSR design.

Figure 4.29 shows the global architecture of PGDBF compared and GDBF, and it can be noted in the context of hardware the GDPF and PGDBF have the same structure for the check nodes and for the finder of the maximum value of the inverse function for all variable nodes.

In [119], the authors implemented the GDBF and PGDBF decoders for the case of Tanner code (155,64) with $\gamma_v= 3$ and $\rho_c= 5$.The results shown that the probability of the IVRG output is unstable during the iterations can be estimated around range $0.88 < p_{opt} < 0.92$.

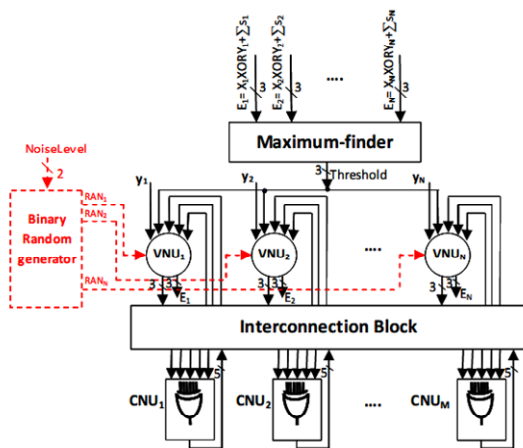


Figure 4.29. Global architecture of PGDBF compared to the original GDBF

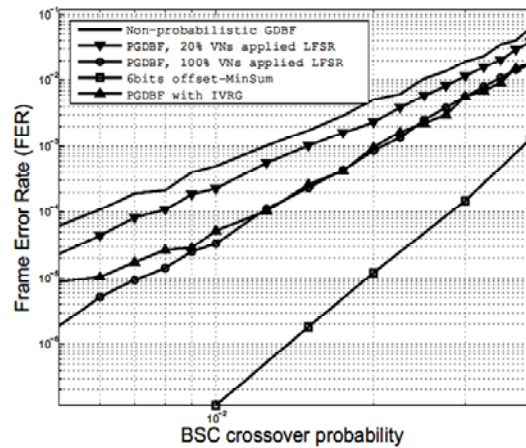


Figure 4.30. FER performance comparison of the different decoders on the Tanner code (155,64)

Figure 4.30 shows performance of the Tanner code (155,64) with respect to the BSC crossover probability for the two designs of the random generator. Performance of GDBF is improved for two solutions and significant gain is obtained. Impact of the imprecise of the random generator implemented with IVRG appears in the error-floor region where the performance of PGDBF backs down compared in the case of LFSR-PGDBF. However, decline in the performance for this code begins when $FER < 10^{-4}$ in the case of IVRG-PGDBF while the matched performance in the water region for the both solutions.

Table 4.1. Hardware and throughput estimation for PGDBF with different RG implementation and for Min-Sum

	1-bit Register	Slice LUTs	F_{\max} (MHz)	Throughput (Mbps)
GDBF	946	2151	132.721	4114.3
PGDBF (IVRG)	1038	2412	132.721	4114.3
PGDBF (LFSR)	9161	3545	134.56	4202.36
Offset min-sum	13694	15359	237.185	197.5

Table 4.1 represents the hardware component required to implement GDBF, PGDBF for two designs and 6-Min-Sum algorithms. The hardware results obtained by using FPGA Xilinx virtex 6 of 40nm technology and the results are the maximum frequency and throughput. The throughput calculated by $f_{\max} * N / (I_{\text{aver}} * S)$, where f_{\max} , I_{aver} , S are the maximum frequency, the average iteration number and the number of clock cycles needed for one iteration, respectively. The results are $S = 1$ for the PGDBF algorithms and $S = 10$ for the offset Min-Sum.

From Table 4.1 it can be noted that IVRG-PGDBF design needs an additional 92 1-bit-registers compared with the GDBF decoder. On the other hand, LFSR-PGDBF design needs additional 82515 1-bit registers compared with the GDBF decoder. Therefore, IVRG-PGDBF decoder improves the performance of GDBF decoder with low complexity rather than LFSR-PGDBF decoder.

For the Slice LUTs required, it can be noticeable that IVRG-PGDBF needs 261 more slices than the GDBF (12.1%) and this number for LFSR-PGDBF is 1394 (64.8%). Although the PGDBF decoder for two designs needs more hardware components to achieve the better performance than GDBF, the throughput of PGDBF decoder approximately remains the same

as for GDBF (less than 2% mismatch). In terms of decoding speed, it can be note that Min-Sum decoder is far more complex than BF type decoders.

Chapter 5

Performance and Complexity of Modified McEliece Cryptosystem

Under a specific LDPC (or MDPC) decoding algorithm over BSC, an LDPC code is considered to have a t -threshold error correction capability if it can correct all error patterns of weight t or less. The determined threshold of an LDPC code has critical part in some applications, such as flash memories, data-storage devices, optical communication and code-based cryptography. In the other words, many systems require extremely low error rates, solving the error-floor problem has been a critical issue.

For some classes of codes, like Reed-Solomon (RS) and Bose-Chaudhuri-Hocquenghem (BCH) codes, this threshold can be exactly determined. In general, most of the decoding algorithms for LDPC codes do not exactly guarantee that all error sequences with t (or less) errors can be corrected. An essential reason for this problem comes from the fact that in spite of the error floor analysis which includes the identification of code's trapping sets, there are still doubts in declaring whether a certain iterative decoder succeeds in correcting all t -error patterns. In theory, the threshold is computed by assuming the code-length to be infinite and when there are no cycles of the length less or equal to double number of iteration in the Tanner graph associated to the parity-check matrix. This means that the probability of error does not depend on particular error positions [120].

Replacement of Goppa codes by QC-LDPC (or MDPC) codes in the McEliece system, on one hand, decreases the key sizes with relatively good code rate, and on the other hand may lead to inability to decide what is the intended message because of the failure possibility of the decoding process. The correction of a large number of errors is not interested for cryptography system but only a number which ensures an adequate security

level, so it is important to find a solution which employs the secure code with high correction probability.

Generally, the attacks on McEliece code-based cryptosystem can be divided on two types of attacks. Without the knowledge of the private key, the first type of attacks tries to recover the plaintext from the ciphertext. This attack attempts to obtain an error vector e used for encrypting a ciphertext, and it can be found as the lowest weight codeword in the extended code [121]. The second type of attacks aims to retrieve the private key from the available public one which is called structural attack. When an intentional error vector has low weight, the decoding attacks against the McEliece cryptosystem are considered more dangerous than structural attacks and provide the smallest work factor (WF). Therefore, increasing the number of errors during the encryption step will make this type of attacks more difficult. In fact, McEliece cryptosystem suffers from chosen ciphertext attacks, where a given plaintext can be encrypted to give different ciphertexts. Therefore, an attacker can compare these different ciphertexts to obtain the original plaintext. Many methods were proposed to protect the system from chosen ciphertext attacks [59] [123], i.e., CCA2-secure variants.

The LDPC decoders can correct more errors with the larger code block, for the same code rate. But for the some class of LDPC codes the situation looks different. Fossorier in [124] proved that girth of cyclic and QC codes whose parity-check matrix has no zero blocks is at most 12. That means, the known relationship $g \propto \log_{(\gamma-1)(\rho-1)} N$, [49] will not be valid for the girth g of such codes. Therefore, the cyclic and QC codes have poor performance at very long code block lengths, where the small girths discourage their usage in a number of applications. However, in practical performance of LDPC (or MDPC) codes can be evaluated by simulation under a certain decoding algorithm. Therefore, we concentrate on the decoding algorithms that can achieve the better performance, i.e., lower FER, such that can increase security of the system.

5.1 The choice of the decoding algorithm

One of the solutions for this problem is using more sophisticated decoding algorithms with better error correction capability [48]. Surely, this comes at price of a significantly increased decoding complexity as in the BP decoding algorithm.

In Chapter 4, we proposed novel algorithms to obtain a good performance of LDPC codes with faster decoder and low complexity. We have seen that PGDBF decoder over BSC is twenty orders of magnitude faster than offset min-sum decoder for Tanner code (155,64).

Also we have noted that performance of PGDBF decoder is improved for higher number of iterations, which means that this algorithm has superior response to the number of iterations. The main purpose of our research is to reduce the complexity of the LDPC decoder for any application and especially for McEliece cryptosystem based on LDPC/MDPC codes. In this chapter, we introduce an optimization solution for reducing the computational complexity of LDPC (or MDPC) decoders in the McEliece cryptosystem with some tradeoffs between complexity, security and performance.

A significant improvement would be obtained by using multiple PGDBF decoders. Hence, we suggest using three schemes:

- (i) MUDRI decoder, i.e., one decoder with multiple attempts in serial form, as mentioned in Chapter 4.
- (ii) MUDRI-P decoders, i.e., multiple decoders that work in the parallel form.
- (iii) PGDBF-PR decoders, i.e., multiple PGDBF decoders in the parallel form with periodical Resets, where the threshold of the bit-flip decision holds two values: the first and second largest values of the MIF after some number of iterations. The main idea of this decrement (decrease the threshold to the second largest value of MIF) is to help the decoder to correct some error patterns that cannot be corrected according to the used threshold.

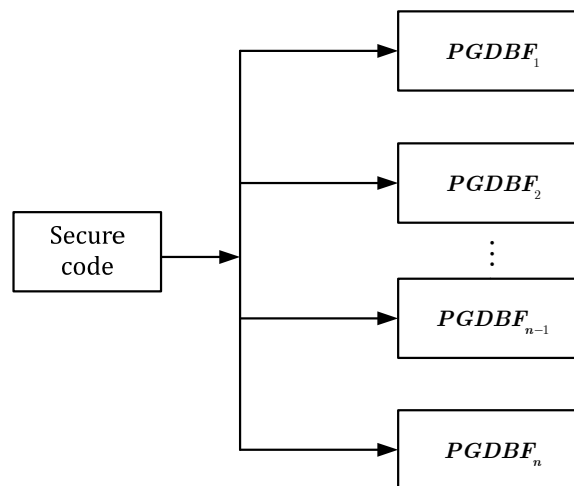


Figure 5.1. PGDBF decoders in the parallel form

For the parallel form, as shown in Figure 5.1, multiple decoders are used and this increases the complexity but helps to decrease the demanded number of iterations required to converge to the correct codeword. In this case, probability that decoding converges to a codeword with a small number of iteration increases, since the decoding stops when the codeword is reached for the first time. On the other hand, this realization also increases the probability that the decoder chooses the codeword that is different than actually transmitted one. Probability of such event depends on the distance profile of the selected code and can be minimize using rules described in [101].

5.5.1 Numerical results for LDPC codes

We begin with Tanner LDPC (155,64) code to illustrate the optimization between the performance and complexity cost. Figure 5.2 shows error correction performance as a function of the intentional errors under difference decoding algorithms represented in Table 5.1.

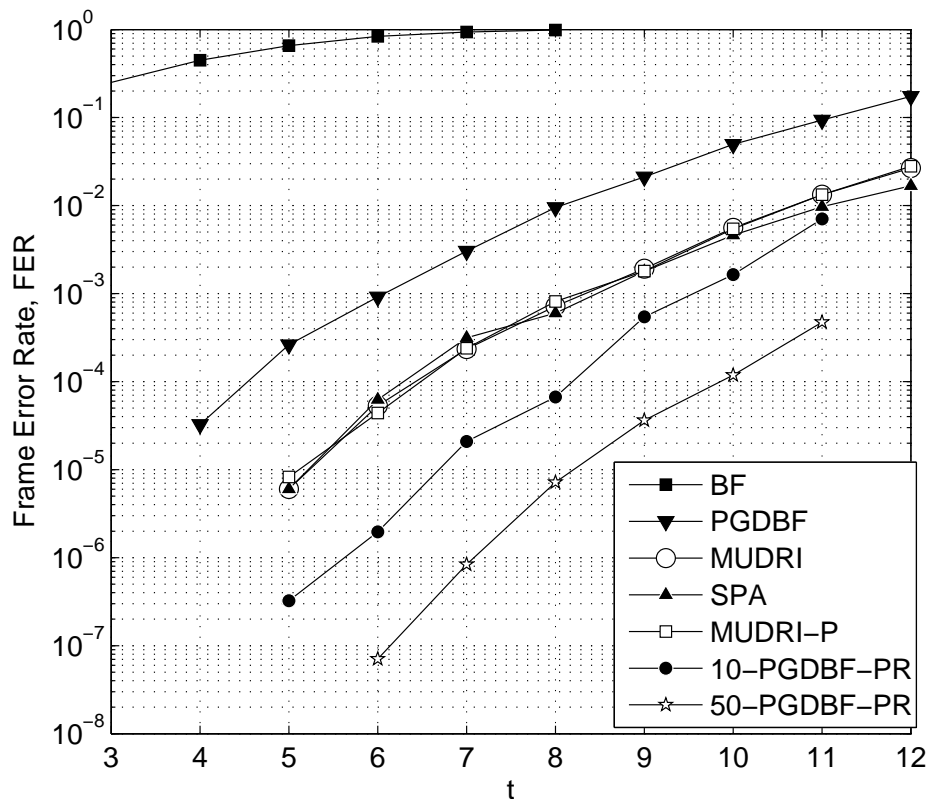


Figure 5.2. Error correction performance as a function of the intentional errors for Tanner code (155,64) under BF, PGDBF, MUDRI, MUDRI-P, SPA and PGDBF-PR decoding algorithms

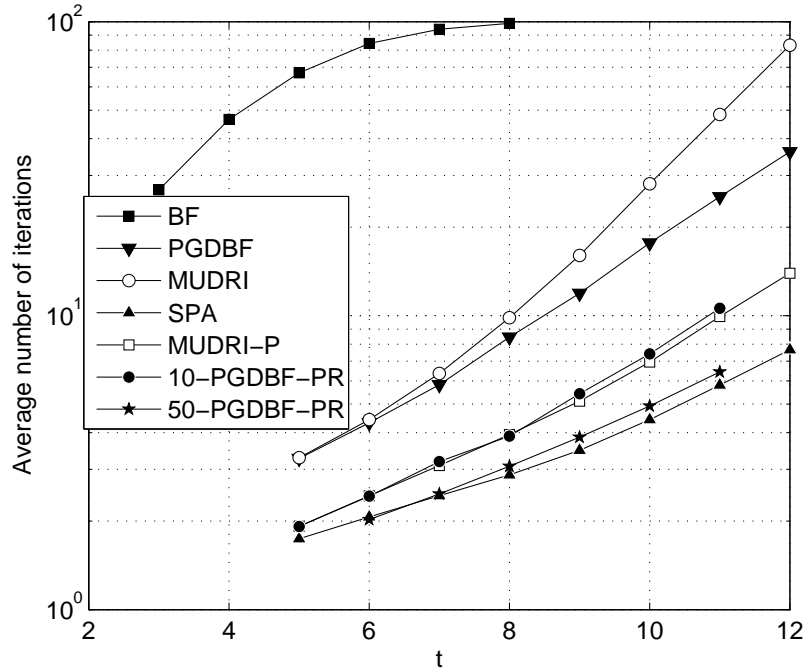


Figure 5.3. Average number of iterations as a function of the intentional errors for Tanner code (155,64) under BF, PGDBF, MUDRI, MUDRI-P, SPA and PGDBF-PR decoding algorithms

The maximum number of iterations for each decoder or attempt is $L=100$. Figure 5.2 shows the MUDRI and MUDRI-P decoders have the same performance as the SPA decoder. The better performance can be achieved by using LDPC decoding algorithms based on hard decision. The same procedure of the PGDBF decoder but after number of iterations *Reset*, the threshold holds two values: the first and second largest values of the inverse functions MIF, i.e., $\max_{1_v}(\Lambda_v^{(Reset)}(\hat{\mathbf{x}}, \mathbf{y}))$ and $\max_{2_v}(\Lambda_v^{(Reset)}(\hat{\mathbf{x}}, \mathbf{y}))$. In our simulations, we use this procedure with the parallel decoders. The effectiveness of this procedure can be is higher for the larger number of decoders.

On the other hand, the average numbers of iterations as a function of t errors for various decoders for Tanner code (155,64) are presented in Figure 5.3. It can be shown that MUDRI-P and MUDRI decoders require more iteration for the larger t . Also we can note that for $t=5$, the required average number of iterations is approximately the same for SPA and MUDRI-P while is large for MUDRI decoder.

In short, increasing the maximum number of iterations L for MUDRI or increasing the number of decoders for MUDRI-P schemes will guarantee superior performance. This

improvement depends on the number of errors which can be corrected and the time to carry out the decoding process.

Table 5.1. LDPC decoders are used to represent Figure 5.2

Method	Number of decoders\attempts*	Total number of iterations
BF	1	100
PGDBF	1	100
MUDRI	10*	1000
SPA	1	100
MUDRI-P	10	1000
PGDBF-PR	10	1000, Reset =10
PGDBF-PR,	50	50000, Reset =10

Table 5.2 shows the number of errors t that can be corrected at performance $FER = 10^{-4}$ for Tanner code (155,64). It can be noted that periodical decrement of the threshold during the decoding process significantly improves the performance, especially with the increase in the number of decoders.

Table 5.2. Average number of iterations and number of errors that can be corrected with $FER = 10^{-4}$

Decoder	t	It_{avr}
PGDBF	4	1.8
MUDRI	6	4.42
MUDRI-P	6	2.43
SPA	6	2
10-PGDBF-PR	8	3.89
50-PGDBF-PR	10	4.93

The performance of the same decoders for a longer quasi-cyclic LDPC code, with parameters $\gamma_v = 4$, $\rho_c = 8$ and $N = 1296$ are shown in Figure 5.4 and Table 5.3. One can observe that the PGDBF remarkably outperforms the GDBF. The MUDRI decoder significantly outperforms PGDBF decoder in the waterfall while the decrease in the error probabilities slows down in the error floor region. It can be seen that the PGDBF-PR surpasses the MUDRI at the error floor region and the difference between them at FER performance $2 \cdot 10^{-6}$ is about 14 errors.

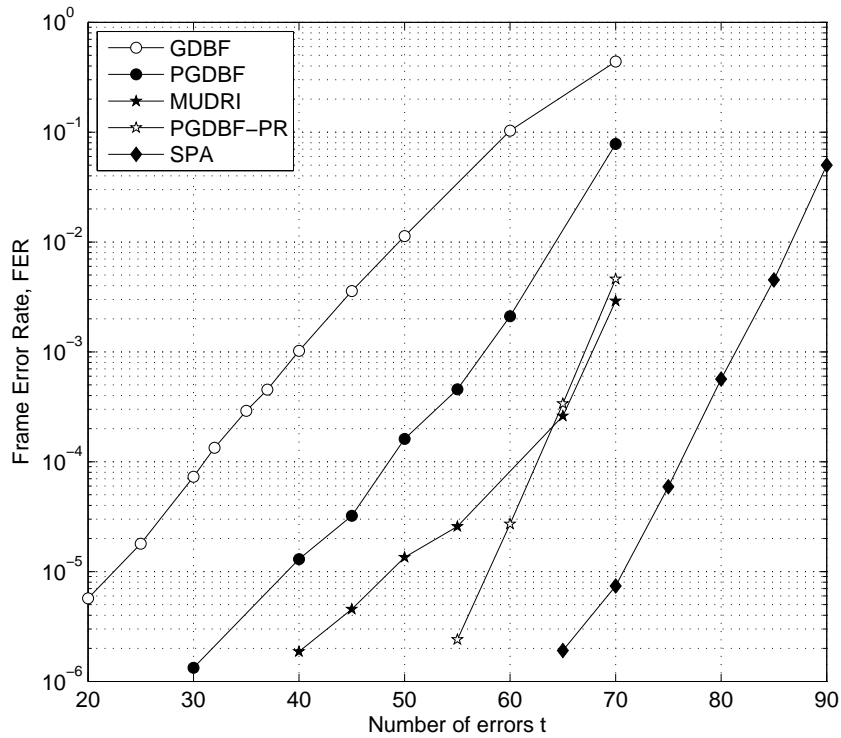


Figure 5.4. Error correction performance as a function of the intentional errors for quasi-cyclic $\gamma_v=4$, $\rho_c=8$ and $N=1296$, under GDBF, PGDBF, MUDRI, PGDBF-PR and SPA decoding algorithms

Table 5.3. Number of errors that can be corrected with $\text{FER}=2 \cdot 10^{-6}$ for quasi-cyclic $\gamma_v=4$, $\rho_c=8$ and $N=1296$

Decoder	Number of decoders/attempts*	t
GDBF	1	14
PGDBF	1	31
MUDRI	10*	40
PGDBF-PR	10	54
SPA	1	65

For the larger code block, we consider the QC-LDPC codes based Pseudo Difference Families (PDFs) in our simulations [126]. The code's parameters are $p = 1021$, $\gamma_v = 5$, $R = 7/8$, while eight base blocks of PDF can be selected such that an element appears only once among their differences. The Table 5.4 shows the polynomials obtained by choosing eight base blocks with distinct differences [125]. Using all polynomials in the Table 5.4 results in a regular code with minimum distance ≤ 10 and the associated Tanner graph has 40840 edges.

Table 5.4 Polynomials used in the construction of rate-7/8 QC-LDPC code based on PDFs

Polynomial	Value
$a_1(x)$	$x^{277} + x^{409} + x^{597} + x^{814} + x^{966}$
$a_2(x)$	$x^{83} + x^{201} + x^{900} + x^{905} + x^{974}$
$a_3(x)$	$x^{167} + x^{183} + x^{347} + x^{582} + x^{763}$
$a_4(x)$	$x^{27} + x^{213} + x^{319} + x^{588} + x^{895}$
$a_5(x)$	$x^{199} + x^{322} + x^{773} + x^{817} + x^{952}$
$a_6(x)$	$x^{180} + x^{181} + x^{399} + x^{425} + x^{857}$
$a_7(x)$	$x^{445} + x^{561} + x^{646} + x^{682} + x^{729}$
$a_8(x)$	$x^{179} + x^{268} + x^{451} + x^{526} + x^{618}$

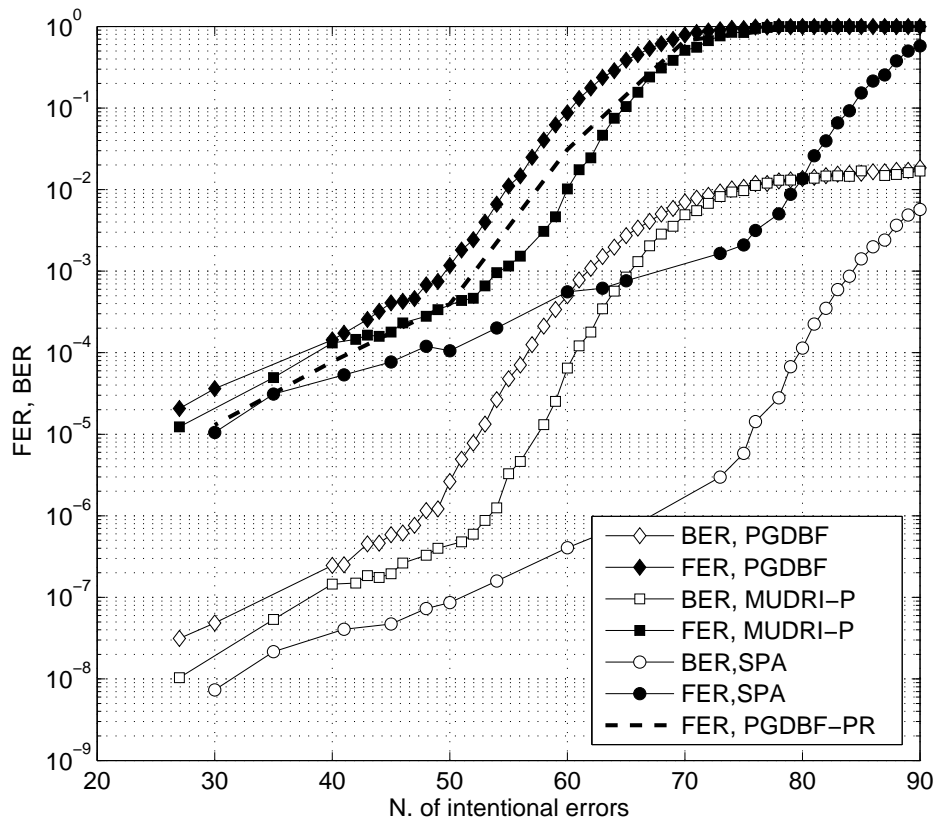

 Figure 5.5. Error correction performance as a function of the intentional errors for QC-LDPC PDFs code with $n=8168$, $p=1021$, $R=7/8$ and $\gamma_v=5$ under SPA, PGDBF, MUDRI-P and PGDBF-PR decoders

Table 5.5. FER performance of different decoders for QC-LDPC, $n=8168$ at $t=30$

Decoder	Number of decoders	FER
PGDBF	1	3×10^{-5}
MUDRI-P	5	2×10^{-5}
SPA	1	10^{-5}
PGDBF-PR, reset=10	5	1.3×10^{-5}

Figure 5.5 shows error correction performance as a function of the intentional errors for QC-LDPC PDFs code with $n = 8168$, $p = 1021$, $R = 7/8$ and $\gamma_v = 5$ under SPA, PGDBF, MUDRI-P and PGDBF-PR decoding algorithms, where the $L = 100$ for every decoder. It can be noticed that SPA has superior performance than the other decoders for all values of the errors. For the error $t \leq 30$, and reset=10, the PGDBF-PR decoder can achieve the same performance as the SPA decoder, as shown in Table 5.5.

5.1.2. Numerical results for MDPC codes

The normalized belief propagation (NBP) algorithm [133] represents a popular modification of belief-propagation (BP) algorithm in which the reliability of messages computed in the each decoding iterations is reduced, by some criterions. This modification is particularly effective for short low-density parity-check codes, where the existence of cycles makes the original BP algorithm perform suboptimal. QC-MDPC codes have short cycles because of the large row or column weights which degrade the performance of the BP algorithm. Therefore, it is preferred to use NBP as a decoding algorithm rather than original BP to correct the channel errors for QC-MDPC codes.

In the following, two examples are given to illustrate the decoding performance for QC-MDPC codes, we consider GDBF, 5-MUDRI-P, 10-MUDRI-P, MUDRI and NBP decoders. In Figure 5.6, the performances of the QC-MDPC code with (9600, 4800, 90) parameters and code rate = 0.5 are presented for the maximum number of the decoding iterations $L=100$. The situation of the MDPC codes is different than LDPC codes, where GDBF and PGDBF have the same performance for MDPC codes in terms of FER and use of a random generator does not help itself to correct more codewords, in addition, the decoder becomes slower.

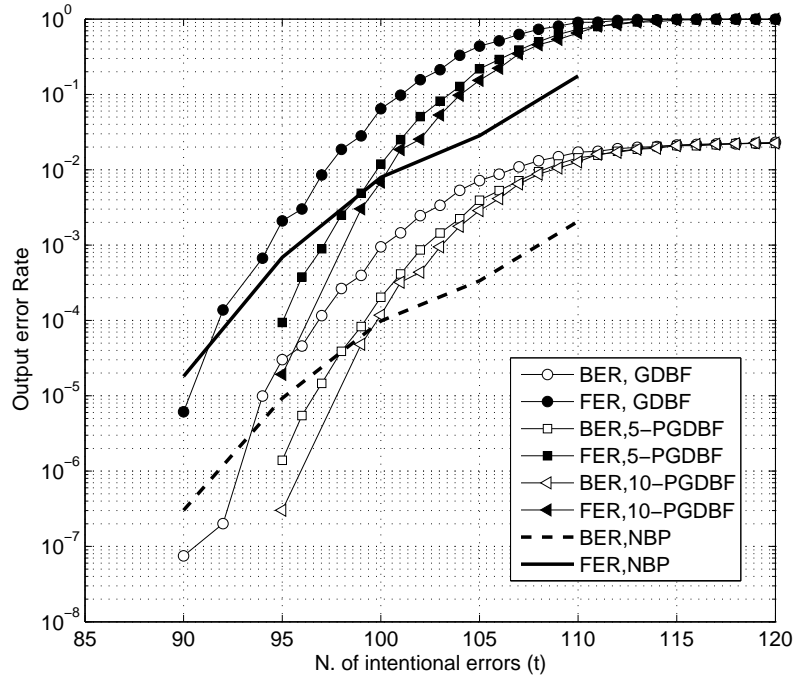


Figure 5.6. Error correction performance as a function of the intentional errors for QC-MDPC (9600,4800,90) under NBP, GDBF, 5-MUDR-P, and 10-MUDRI-P algorithms

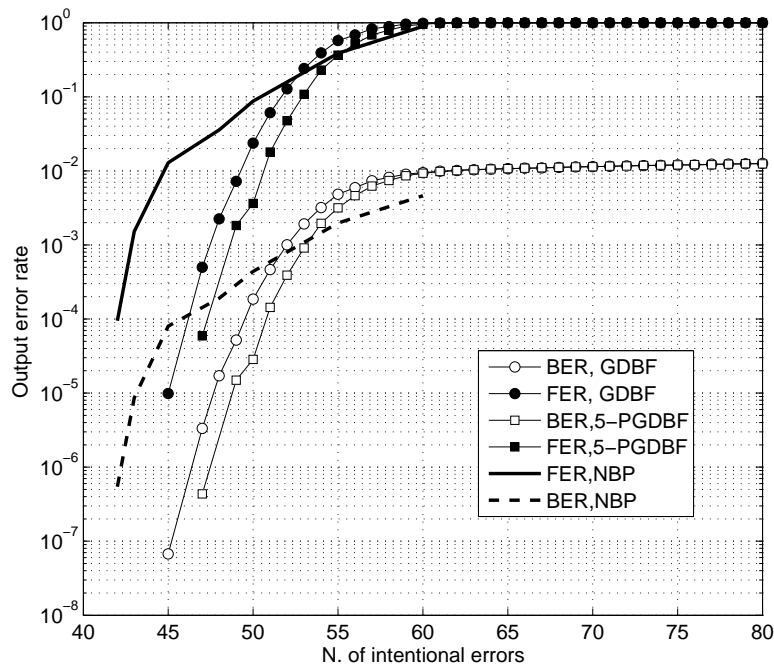


Figure 5.7. Error correction performance as a function of the intentional errors for QC-MDPC (12288,3072,220) under NBP, GDBF, and 5-MUDR-P algorithms

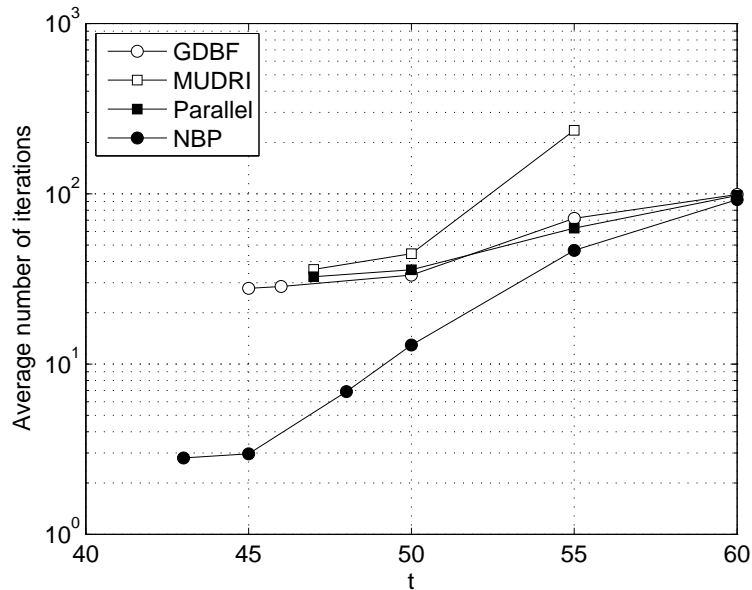


Figure 5.8. Average number of iterations as a function of the intentional errors for QC-MDPC (12288,3072,220)

It can be noted that NBP decoder has better performance only in the waterfall region (when the number of intentional errors $t > 100$) than the GDBF, but this performance with low error correction probability ($FER=0.01$). Because of the high column degree of these codes, the short-cycles in their Tanner graphs prevent the NBP decoder to keep its performance in correcting the codewords with small errors. On the other hand, the GDBF decoder has better performance than the NBP for the number of intentional errors < 92 and this performance can be improved by using multiple decoders in the parallel form. For this code we can determine the GDBF threshold (confirmed through simulation) $t = 91$ with $FER=10^{-5}$, $t = 93$ with $FER=10^{-5}$ for 5-MUDRI-P and $t= 94$ with $FER =10^{-5}$ for 10-MUDRI-P. In fact, interpretation of this phenomenon returns to the probabilistic nature of the PGDBF decoder in correction the error patterns. Improvement of the performance for these codes guarantees that most of transmitted messages can be corrected with the high probability that avoids using another technique like Automatic Repeat Request (ARQ) for retransmission the same message with the other intentional error vector, which can be attacked. Berson proved that the McEliece public key cryptosystem is unable to keep safe for any message which is sent more than once to a receiver using different random intentional error vectors [127]. In this case, a suitable CCA2-secure conversion is used.

The second QC-MDPC code is presented in Figure 5.7, with (12316,3079,220) parameters and code rate = 3/4. We can see that performance of GDBF decoder with higher code rate is better than performance of the NBP when $t > 52$. The GDBF error correction capability is $t = 45$ where $\text{FER} = 10^{-5}$ and $t = 47$ for 5-MUDRI-P decoders at the same FER performance.

In Figure 5.8 we illustrate average number of iterations as a function of the inserted errors for QC-MDPC-based scheme (12288,3072,220) under NBP, GDBF, 5-MUDRI-P and MUDRI decoders. It can be noticed that the average number of iterations for parallel decoders is smaller than average iterations for MUDRI decoder. An average number of iterations of NBP for $t < 45$ is very small compared to the average number of iterations of PGDBF decoders, that means increase the complexity of procedure to correct the codewords in spite of the PGDBF decoders have better performance than NPB. Regardless of this situation, the complexity of PGDBF decoder is smaller than NBP and sometimes it is preferred to use a faster decoder although it needs larger number of iterations.

5.2. Computational Complexity

The syndrome computing is the same for all decoding algorithms; therefore, the complexity comparisons are realized without the syndrome complexity. The computational complexity of PGDBF algorithm consists mainly of three parts: (i) calculating inverse functions, (ii) finding the threshold, and (iii) flipping the bits.

- Calculating inverse functions

Check node calculates XORs from the received messages and sends the result to all neighbour variable nodes, so at the check node, there is $(\rho_c - 1)$ binary sums and the total number of operations at the check nodes is $M(\rho_c - 1)$. For every variable node, there is one XOR logic gate between an estimated bit at l -th iteration and original received bit. Therefore, hence, $M(\rho_c - 1) + N$ binary operations are needed. Also $N\gamma_v$ integer additions are needed to calculate the inverse functions for all variable nodes per iteration.

- Finding the threshold (maximum value of the inverse functions)

Usually, maximum-finder is used to find the maximum value, but in PGDBF decoder we can exploit that the values of an inverse function are integer values and restricted in the range $[0, \gamma+1]$. Therefore, no need to use maximum-finder such that the threshold may be initialized to the maximum value $(\gamma+1)$, and the threshold is decreased by step

1 when all MAJ logic outputs are zeros, for instance by using N-input OR gate. The number of decrements of the inverse function value to obtain the maximum value differs in the case of LDPC and MDPC codes. Figures 5.9 and 5.10 represent values of the threshold during decoding iterations for Tanner code (155,64) with $\gamma_v = 3$ and QC-LDPC (8168,1021) code with $\gamma_v = 5$, respectively. However, for some values of t the threshold may take one value with the high average value. For instance, for the Tanner code (155,64) and $t=3$, the threshold =3 with average is 89%. Therefore, the threshold can be initialized with value 3 for $t=3$ rather than 4 value, i.e., $(\gamma+1)$, and in this way the number of decrements is decreased. In general, $\leq (\gamma_v+1)$ integer comparisons are needed for every variable node per iteration.

- Flipping the bits

After finding the threshold, the bit is flipped with probability p . This can be realized by adding to each variable node processor one AND gate. Therefore, it needs N AND logic gate and N XOR logic gate to flip the bit.

Table 5.4 presents the summarizing the computational complexity which includes binary/integer/real additions and comparison for BF, PGDBF and SPA decoders. Note that the random generator for the PGDBF decoder did not take into account in computational complexity. In [119] emphasized that IVRG scheme has advantage over LFSR in term of implementation and there is a simple difference between GDBF and PGDBF as shown in Table 4.1. Any way, it is still an open problem how to realize the precise random generator with low complexity. In addition, it can be realized by the specified matrix where its rows or columns are used instead of random generator.

Table 5.6. The total number of operations for some LDPC decoders during a single iteration

Operation	BF	PGDBF	SPA
Binary Operations	$N + M(\rho-1)$	$3N + M(\rho-1)$	-
Integer additions	$N(\gamma-1)$	$N\gamma$	-
Integer Comparisons	N	$\leq N(\gamma+1)^*$	-
Real Multiplications	-	-	$2N(\gamma^2+4\gamma)$
Real additions	-	-	$2N(2\gamma+1)$

* It is a maximum number of comparisons and an average number can be relatively determined according to the number of error t as in Figures 5.8-5.11 for LDPC and MDPC codes.

Figures 5.9 - 5.12 present the rate of the threshold value of the bit-flipping decision in the form of a histogram for Tanner (155,64), QC-LDPC (8168,1021), QC-MDPC (9600,4800,90) and QC-MDPC (12288, 3072, 220) codes, respectively.

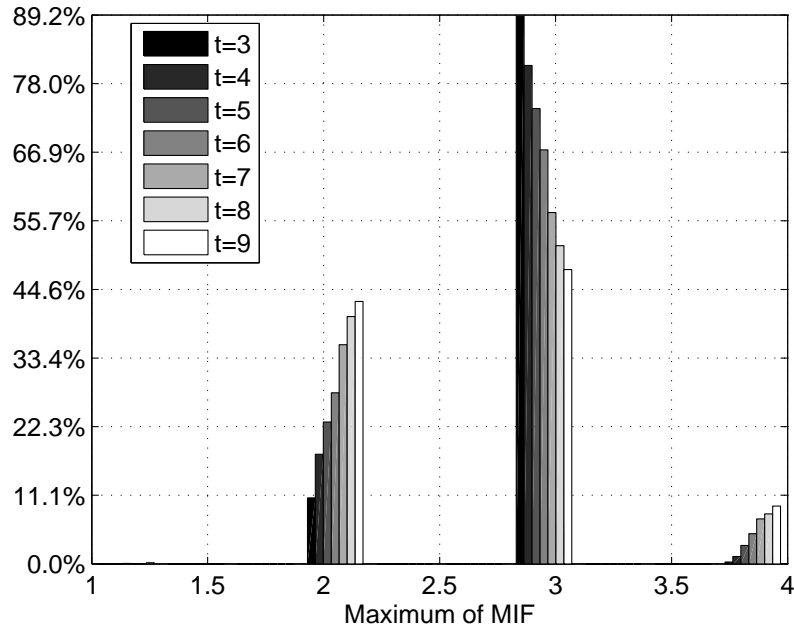


Figure 5.9. Comparison of the threshold during the decoding iterations for Tanner (155,64) code, $\gamma_v=3$

From Figure 5.9, it can be seen that for $t=3$, there are about 89% of all threshold values at which threshold equals to 3. In this case, the initial threshold can be started by value 3 instead of value 4 and by this way the number of decrements is reduced. Also this idea is applied when $t \in [4,5,6]$, where the rate of 3 value is very high. By this way, the computational complexity during the iterations is significantly reduced.

Figure 5.10 interprets how the threshold during the most of iterations holds in the high rates of values 4 and 5, which ensures that the comparison for every variable node is done mostly two times.

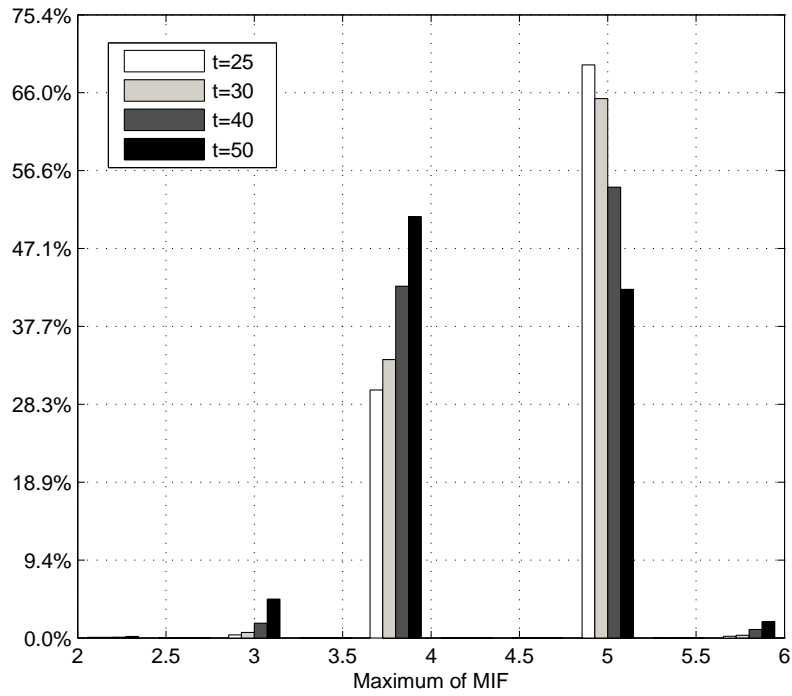


Figure 5.10. Comparison of threshold during the decoding iterations for QC-LDPC (8168,1021) code, $\gamma_v=5$

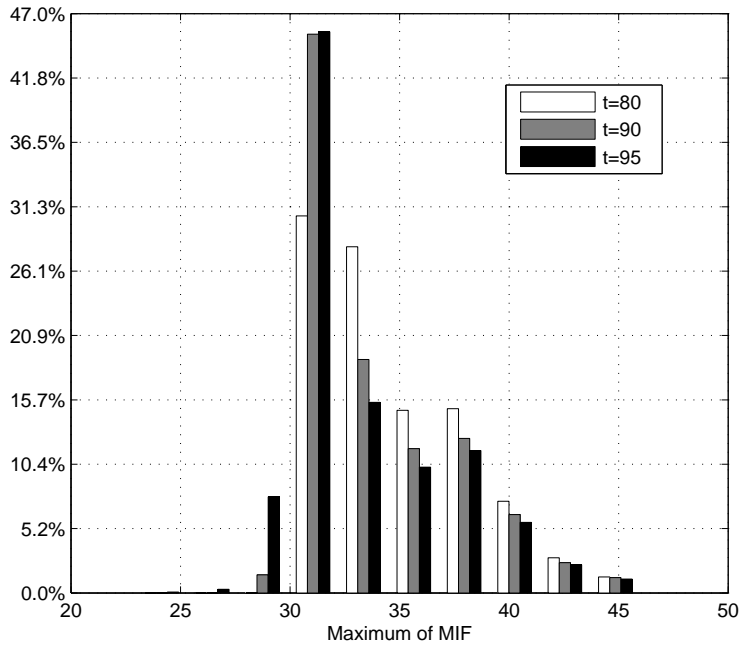


Figure 5.11. Comparison of the threshold during the decoding iterations for QC-MDPC (9600,4800,90) code, $\gamma_v=45$

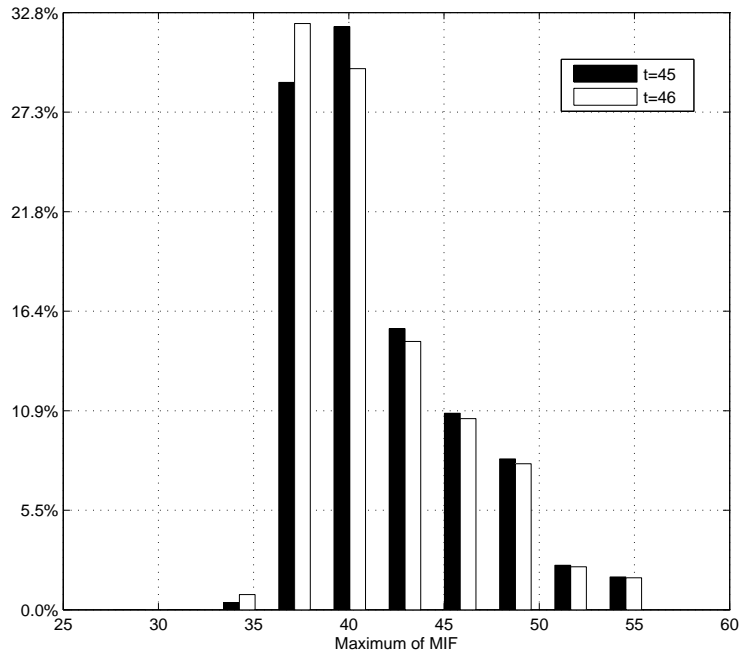


Figure 5.12. Comparison of the threshold during the decoding iterations for QC-MDPC (12288, 3072, 220) code, $\gamma_v=55$

From Figures 5.11 and 5.12, it can be observed that for QC-MDPC codes, the higher rate of the threshold value during the decoding process is small than 50% for some t errors, and in this case the initial threshold value starts with value which has the higher rate, for instance, initial threshold value in Figure 5.11 can be started with value 30 for $t = 80$, and the initial threshold is increased with step1 until obtain the maximum value of MIF.

Table 5.7. Average overall complexity per frame (Real: real comparison or addition; Int.: integer comparison or addition; Bin.: binary operation)

Tanner code (155,64); complexity $\times 10^3$				QC-LDPC (8168,1021) code; complexity $\times 10^5$			
t		5	6	7	30	35	40
BF	$I_{t_{avr}}$	67	84	94			
	Bin.	35.3	44.2	49.5			
	Int.	31.1	39	43.7			
	FER	0.6	0.8	0.9			
PGDBF	$I_{t_{avr}}$	3.2	4.3	5.8	8.3	9.7	11.55
	Bin.	2.6	3.5	4.8	5.3	6.2	7.4
	Int.	2.728	3.6	4.9	28.4	33.2	39.6
	FER	$2 \cdot 10^{-4}$	$9 \cdot 10^{-4}$	$3 \cdot 10^{-3}$	$3.6 \cdot 10^{-5}$	$7 \cdot 10^{-5}$	10^{-4}
MUDRI	$I_{t_{avr}}$	3.2	4.4	6.3			
	Bin.	2.6	3.6	5.2			
	Int.	2.728	3.75	5.37			
	FER	$6 \cdot 10^{-6}$	$5.3 \cdot 10^{-5}$	$2 \cdot 10^{-4}$			
10-MUDRI-P	$I_{t_{avr}}$	1.9	2.4	3	6.6	7.9	9.3
	Bin.	15.9	20	25.9	21.2	25.4	29.9
	Int.	16.1	20.46	25.57	113.20	135.5	159.52
	FER	$6 \cdot 10^{-6}$	$5.3 \cdot 10^{-5}$	$2 \cdot 10^{-4}$	$2 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$8 \cdot 10^{-5}$
10-MUDRI-PR, reset=10	$I_{t_{avr}}$	1.9	2.4	3.1	6.7	7.9	9.2
	Bin.	15.9	20	25.9	21.5	25.4	29.58
	Int.	17.670	22.320	28.830	114.9	135.5	157.8
	FER	$3.2 \cdot 10^{-7}$	$2 \cdot 10^{-6}$	$2 \cdot 10^{-5}$	$1.3 \cdot 10^{-5}$	$3 \cdot 10^{-5}$	$7 \cdot 10^{-5}$
50-MUDRI-PR, reset=10	$I_{t_{avr}}$	-	2	2.4			
	Bin.	-	83.7	100.4			
	Int.	-	93	111.6			
	FER	-	$7 \cdot 10^{-8}$	$8.3 \cdot 10^{-7}$			
SPA	$I_{t_{avr}}$	1.7	2	2.4	2.34	2.68	3
	Real	14.7	17.3	20.8	21.4	24.5	27.44
	FER	$6 \cdot 10^{-6}$	$6.2 \cdot 10^{-5}$	$3 \cdot 10^{-4}$	10^{-5}	$3 \cdot 10^{-5}$	$5 \cdot 10^{-5}$

Table 5.7 shows the simulated average complexity of various BF, PGDBF and the SPA algorithms for decoding a frame at different t errors. As an integer (or real) comparison requires the same computational complexity as that of an integer (or real) addition (hardware implementation of comparison can even be simpler than addition). Both are, thus, counted equally.

Tables 5.7 and 5.8 provide useful information for studying tradeoffs between performance and complexity. For the PGDBF decoder, the majority logic gate is used for every variable node to calculate the MIF with threshold comparison. Therefore, in the term of binary operation, the maximum number of binary operations that is needed for one frame for PGDFB is

$$C_{PGDBF} = [M(\rho_c - 1) + 3N + N(\gamma + 1)].$$

According to the implementation in [122], the decoding complexity in binary operations of the LLR-SPA is given by [50]

$$C_{SPA} = I_{ave} N [q(8\gamma + 12R - 11) + \gamma],$$

where I_{ave} is an average number of iterations, R is the code rate and q is the quantization number of bits used for the decoder, where $q = 6$ has been considered in our computations.

Table 5.8. QC-MDPC (12288,3072,220); complexity $\times 10^7$

t		45	46	47
PGDBF	I_{avr}	27.8	28.5	30
	Bin.	1.97	2.02	2.12
	Int.	2	2.06	2.17
	FER	10^{-5}	$2 \cdot 10^{-5}$	$5 \cdot 10^{-5}$
MUDRI	I_{avr}	33.3	34.1	35.9
	Bin.	2.3	2.4	2.5
	Int.	2.41	2.47	2.6
	FER	$3 \cdot 10^{-6}$	$6 \cdot 10^{-6}$	10^{-5}
5-MUDRI-P	I_{avr}	30	32	32.6
	Bin.	10.64	11.35	11.56
	Int.	10.87	11.59	11.81
	FER	$3 \cdot 10^{-6}$	$6 \cdot 10^{-6}$	10^{-5}
NBP	I_{avr}	3	4	5
	Real	24.74	32.99	41.23
	FER	$1.3 \cdot 10^{-2}$	$1.7 \cdot 10^{-2}$	$2 \cdot 10^{-2}$

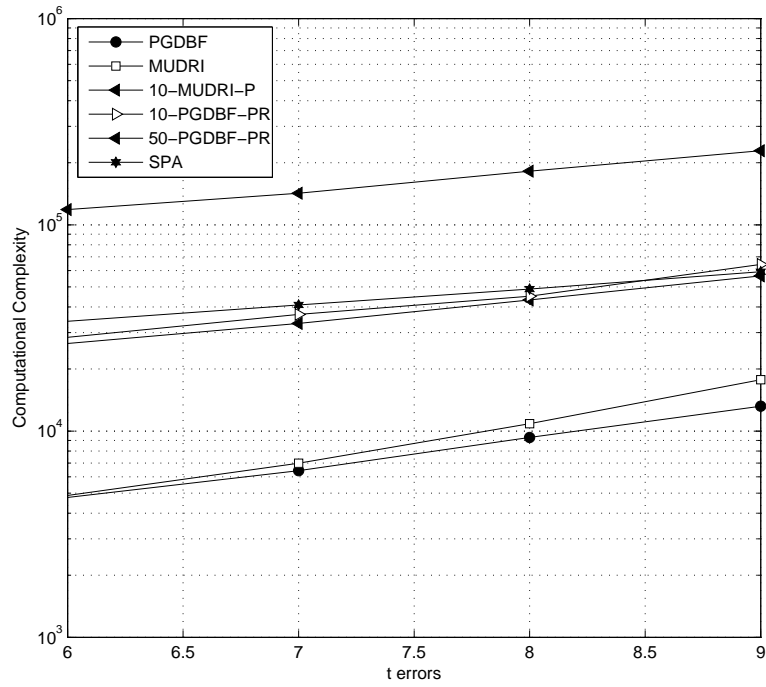


Table 5.13. The computational complexity (per decoding process) comparison in the term of binary operations for Tanner code (155,64)

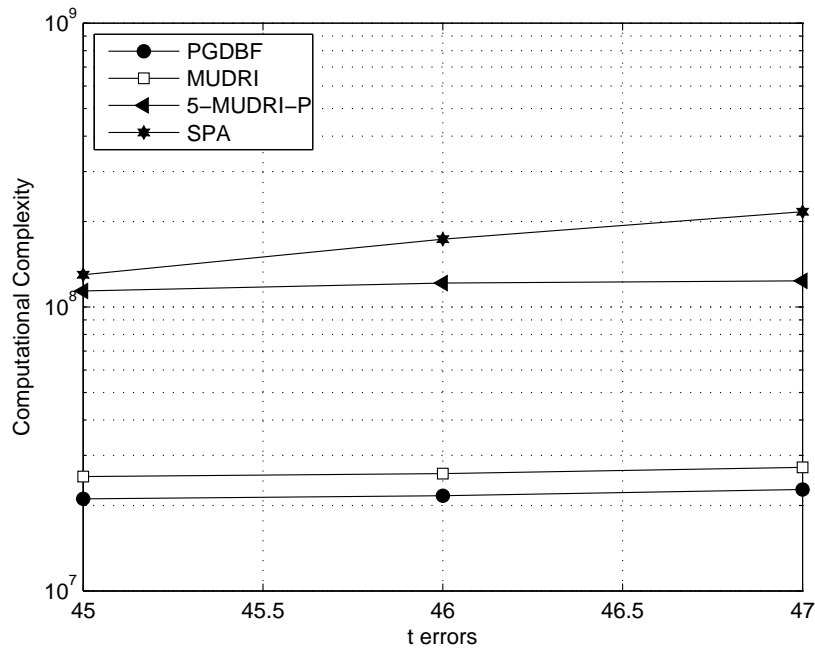


Table 5.14. The computational complexity (per decoding process) comparison in the term of binary operations for QC-MDPC (12288,3072,220)

Figures 5.13 and 5.14 represent the computational complexity in term of binary operations for the Tanner code (155, 64) and QC-MDPC (12288,3072,220), respectively. The better choice is MUDRI decoder which has two advantages, low complexity and good performance. The required average number of iterations approximately is the same for MUDRI and parallel schemes in the case of small t . It is preferred to use the parallel form when the difference between minimum distance of the code is small ≤ 10 , i.e., the probability of miss-correction is relatively high where the decoder only decides according to the syndrome sum if equals to zero. In this case, MUDRI cannot improve the performance whatever number of attempts or iterations.

5.3. Cryptanalysis of the McEliece cryptosystem

The best ways to select the secure parameters that avoid all attacks are comprehension and evaluation of the complexity of the decoding technique. An attack can be usually measured by the work factor, which defined as the average number of binary operations needed to have a successful work. We can say that work factors as high as 2^{80} or larger are important enough to guarantee that the system is protect at the certain technology.

The most dangerous attacks against public key cryptography based on LDPC (or MDPC) codes are attack on the Dual Code and Information Set Decoding (ISD) and the security level of the system could be decided as the smallest work factor of these attacks.

5.3.1. Attacks on the Dual Code

LDPC codes in McEliece cryptosystem risk its security. The main problem of using LDPC codes in the McEliece system is easy to observe the low weight parity check rows in the dual of the public code as codewords with low weights. There exists an easy method to find dual low weight codewords of an LDPC code and employ them to reconstruct a sparse parity check matrix, that is a straightforward attack against LDPC based McEliece cryptosystem.

When the dual of the secret code contains very low weight codewords for the McEliece cryptosystem based on LDPC codes, then the vulnerability for this system may increase and an attacker can directly recover the parity check matrix H and becoming able to perform LDPC decoding without waste. In [128] it is said that when a sufficiently large set of redundant check sums of small enough weight can be found, then an attacker can perform bit flipping decoding based on such parity check equations. Therefore, the dual attack do not

exist only when LDPC code is used for McEliece cryptosystem, but also can be subject to attacks based on low weight codewords in the dual of the public code.

Return to the form (2.8), let code rate R , column weight γ_v , row weight ρ_c , code length n and dimension r , be parameters of the dual of the secret codes and can be generated by matrix \mathbf{H}' . For weight $w \leq \rho_c m$ the dual code has at least $A_w \geq r$ codewords of weight w .

The number of codewords A_w with weight w should be in some way known. Then the work factor of the attack can be precisely evaluated, but on the other hand estimation of the number of codewords A_w is a difficult problem. Since for this case $\rho_c \ll n$ and that sparse vectors are most likely sums of vectors of higher weight then it is assumed that $A_w = r$.

Stern's algorithm can be used to search for minimum weight codewords in the dual of the secret code [121]. In this case, the probability of finding, in one iteration, a codeword with weight w is $\leq A_w P_w$, where A_w represents the number of w -weight codewords and P_w is expressed by [129]

$$P_w = \frac{\binom{w}{g} \binom{n-w}{k/2-g}}{\binom{n}{k/2}} \cdot \frac{\binom{w-g}{g} \binom{n-k/2-w+g}{k/2-g}}{\binom{n-k/2}{k/2}} \cdot \frac{\binom{n-k-w+2g}{l}}{\binom{n-k}{l}}, \quad (5.1)$$

the values of parameters g and l must be optimized as a function of the total number of binary operations.

The average number of iterations needed in order to find a w -weight codeword is hence $\geq c = (A_w P_w)^{-1}$ and it can be considered that each iteration of the algorithm requires a number of binary operations expressed by

$$B = \frac{(n-k)^3}{2} + k(n-k)^2 + 2gl \binom{k/2}{g} + \frac{2g(n-k) \binom{k/2}{g}^2}{2^l}, \quad (5.2)$$

with k and r interchanged, so the total work factor is $\geq W = cB$.

5.3.2. Information Set Decoding Attacks

Information set decoding (ISD) is considered as the best known attack algorithm especially when the code structure is not known [39]. In spite that the general decoding problem is \mathcal{NP} hard, a precise choice of system parameters (n, k) is demanded to guarantee that the security level of McEliece cryptosystem is high enough. The ISD is the best known technique when the weight of errors t is smaller than the Gilbert-Varshamov distance, which is defined as the smaller integer d_0 such that

$$\binom{n}{d_0} \geq 2^r.$$

In some cases it is preferred to use the Generalized Birthday Algorithm (GBA) [130] when t is larger than d_0 where GBA is more efficient than ISD.

An ISD algorithm tries to find a set of free error positions of the deformed ciphertext when it is transmitted. In most situations the random generator of the intentional error vector generates this vector in the random way, so eavesdropper could take, arbitrarily, with some possibilities a piece of the ciphertext which is not changed by any intentional error. Also as an addition condition to guarantee the continuous of this attack is code's generator matrix to these positions is invertible as will be interpreted in the following. In the other words, ISD tries to find an intentional error vector \mathbf{e} which has effect on the ciphertext (\mathbf{e} is a correctable error vector, i.e., has weight $\leq t$) and this mission can be completed by employing algorithms which search for the minimum weight codewords in a linear block code [121].

The original message then can be computed by multiplying the encrypted vector by the inverse of the sub-matrix constructed by the selected columns of the generator matrix. Let k information bits and information vector is \mathbf{u} then only k elements selected from \mathbf{x} and \mathbf{e} vectors for ISD algorithm. At the fixed positions k of the \mathbf{x} and \mathbf{e} and corresponding to the columns of the generator matrix \mathbf{G}' . Using subscript k to denote the vectors and matrix reduced in size and according to the condition, \mathbf{G}'_k is invertible matrix (non-singular matrix, even if this is always true only for maximum distance separable (MDS) codes), and we have

$$\mathbf{x}_k = \mathbf{u} \cdot \mathbf{G}'_k + \mathbf{e}_k. \quad (5.3)$$

In the case when all values of k selected positions of the error vector \mathbf{e} are zeros, i.e., $\mathbf{e}_k = \mathbf{0}$ then it can be written $\mathbf{x}_k = \mathbf{u} \cdot \mathbf{G}'_k$. As long as the sub-matrix \mathbf{G}'_k is known, an eavesdropper can obtain the plaintext by $\mathbf{u} = \mathbf{x}_k \cdot \mathbf{G}'_k^{-1}$. The probability that the vector \mathbf{e} has k zeros symbols in fixed positions is given by

$$P\{e_k = 0\} = \frac{\binom{n-k}{k}}{\binom{n}{k}} = \prod_{i=0}^{k-1} \left(1 - \frac{t}{n-i}\right), \quad (5.4)$$

and work factor for this attack can be evaluated with considering the cost of each matrix inversion as

$$WF_{ISD} \sim k^3 \frac{1}{P\{e_k = 0\}}. \quad (5.5)$$

The ISD algorithm is very simple and for binary linear codes, while the improvement of ISD algorithm are introduced later by Lee–Brickell [65], Leon [131], Stern [121], Peters [20], and Becker et al. [132]. It is worth mentioning that the Lee–Brickell algorithm is a decoding algorithm whereas Stern’s algorithm originally was designated for finding the low-weight codewords in a binary linear code.

In the case when $\mathbf{e}_k \neq 0$, then

$$\mathbf{u}' = \mathbf{x}_k \cdot \mathbf{G}'_k^{-1} + \mathbf{e}_k \cdot \mathbf{G}'_k^{-1}, \quad (5.6)$$

where \mathbf{e}_k is an error vector chosen randomly with weight $\leq t$. In principle, all possible \mathbf{u}' should be considered and examined through a deterministic procedure [65]. One can noticed that considering a subset of all possible \mathbf{e}_k vectors, namely those with weight less than or equal to a given integer j , can be convenient for the eavesdropper.

The work factor of Lee and Brickell’s algorithm can be evaluated as follows

$$W_j = T_j(\alpha k^3 + N_j \beta k),$$

where $T_j = 1 / \sum_{i=0}^j \binom{t}{i} \binom{n-t}{k-i} / \binom{n}{k}$, $N_j = \sum_{i=0}^j \binom{k}{i}$ and α and β are integers ≥ 1 .

The complexity of decoding general linear codes depends on all three code parameters (n, k, t) . Table 5.9 summarizes the work factor of different ISD algorithms for the Goppa code (1024, 524, 50).

Table 5.9. Work factor of ISD for (1024, 524, 50) McEliece cryptosystem

Algorithm	Log. Of binary work factor	Year
Adams-Mejier	80.7	1986
Lee-Brickell	70.89	1988
Stern	66.21	1989
Canteaut-Chabanne	65.5	1994
Canteaut-Chabaud	64.1	1998
Bernstein-Lange-Peters	60.4	2008
Finiasz-Sendrier	59.9	2009

Indeed, increase the Hamming weight of intentional error vector increases the security of the message where the ISD attack (in \log_2) increases linearly in the number of intentional errors. Security of QC-MDPC is determined by the work factor $WF_{dec}^{QC}(n, r, t)$ and here we

investigate the *decoding attack* where the security of the message is related to the hardness of decoding errors [47],

$$WF_{dec}^{QC}(n, r, t) = \frac{WF_{isd}(n, r, t)}{\sqrt{r}}, \quad (5.7)$$

where WF_{isd} is cost for decoding t errors by ISD algorithm [39].

Table 5.10 shows work factor of the Peter's attack [20] where the level security of the system linearly increases with weight of error vectors.

Table 5.10 Peter's level security for QC-MDPC codes

n		GDPF	5-PGDBF	10-PGDBF	NBP
9600	t	91	93	94	88
	WF	99.8456	101.7256	102.6656	97.035
12288	t	45	47	-	42
	WF	40.9475	41.4066	-	40.2975

Chapter 6

Conclusions and Perspectives

McEliece cryptosystem was rediscovered after long time of research as an interesting alternative cryptosystem to existing systems as RSA and ECC. The quantum algorithms are considered to be the major challenge for the public cryptosystem. Despite that the original McEliece cryptosystem was not reported broken by quantum computers, it has been rarely considered in practical applications due to the large keys and low transmission rate when Goppa codes are used. The low complexities of the encryption and decryptions procedures used in the McEliece cryptosystem compared to other systems make this system more attractive in a number of applications. It can be implemented on small devices like PDAs, USB Tokens and mobile phones if the public key size is reduced. Most researches used highly structured codes which can be stored more efficiently. In this thesis, the McEliece cryptosystem based on QC-LDPC and MDPC codes is analyzed, that are widely accepted as a possible modification of the original version while maintaining robustness to known security threats.

The larger part of this thesis was dedicated to develop a new class of decoders for LDPC/MDPC codes. First, we have optimized GDBF decoding algorithm to work over the binary symmetric channel. In fact, the original GDBF decoder is designed to work with soft-decisions and change it to be pure hard decision decoding technique leads to performance degradation. The deterministic nature and simplicity of the GDBF decoder allow simple and comprehensive analysis. The hard decision bit-flipping in the GDBF decoder is related to the maximum value of the modified inverse function (MIF). As we illustrated that failure of that algorithm is related to its deterministic nature, we have developed a novel algorithm that combine the idea of GDBF and probabilistic approach (PGDBF) [A1]. The value of MIF suggests that a variable node should be flipped, but in the PGDBF decoder it is flipped with a

predefined probability $p < 1$ rather than being flipped automatically. It has been shown that the optimal value of parameter p depends on the column weight of the code. The corresponding flipping probability is fixed during iterations, the tuning of parameters is simpler compared to previously proposed algorithms, and results in a minor increase of the decoding latency. It has been shown that FER performance of the PGDBF is significantly better than GDBF for various classes of LDPC codes and various code rates.

By using the analysis of trapping sets, we have developed an improved version of the algorithm (the MUDRI algorithm) [B1]. The proposed algorithm significantly increases the probability of correcting error patterns un-correctable by the existing variants of bit-flipping algorithm, especially in the case when large maximum number of iterations is permitted. In MUDRI decoder, multiple decoding attempts with random re-initializations are used. Moreover, the presented decoding algorithms are robust to the logic gate failures. We have shown that the critical probability of failure in XOR logic gates and registers is rather insensitive to the code construction method and rate, and it is mostly determined by the codeword length. Furthermore, we have shown that the proposed decoder not only has large immunity to gate failures but, surprisingly, can utilize the hardware failures to improve the decoding performance.

A relevant issue concerns on that the work factor of the information set decoding attacks (inlog_2) increases linearly with number of inserted errors to the transmitted messages during encryption. Therefore, starting from observation that the security of the message in McEliece cryptosystem is related to the hardness of decoding t -error patterns and improving of decoding abilities has impact on the security of the system, we introduced optimal solution for improvement performance of the decoder with relatively low complexity. As long as the decoding process in the code-based cryptosystem is mostly expensive procedure, we proposed some decoding algorithms that can reduce the complexity. It has been observed that the multiple PGDBF decoders can be exploited with independent parallel decoding or with successive decoding. They are presented in the QC-LDPC/MDPC code-based McEliece cryptosystems. The improvement performance provided by these schemes is large, which allows correcting more errors and helps to protect the messages.

As the decoder is the most complex part of the code based cryptosystem, the main challenge is to identify the solution with increased flexibility in the term of trade-off between security and complexity. It has been shown that the PGDBF decoder requires only binary logical operations and integer additions which can be implemented with simple

combinational logic gates, compared to the SPA algorithm which exploits the real valued messages. It has shown in [119] that the PGDBF decoder can be sufficiently implemented with high throughput and hardware resources required for the random generators can be reduced using IVRG method. The computational complexity is expressed in terms of the binary operations needed for each frame. The complexity of decoder is evaluated without taking into account the storage and transport complexity, although it is also important for efficient implementation. Taking these aspects in evaluating the overall complexity is not an easy task, and it is usually ignored in existing literature. However, it is worth to notice that efficient FPGA implementation of PGDBF algorithm (that includes computing, storage and transport of the messages through the edges of the graph) results in significantly increased throughput when compared with state-of-the-art algorithms [119]. Also, it has been shown that the multiple PGDBF decoders offer superior trade-off between security, latency and decoding complexity. In this way the system designers can choose a fast procedure with moderate the complexity and high security.

Our future research focuses on identifying the flipping sequences resulting in a minimum number of iterations requires for successful decoding of critical error patterns. On the other hand, if we allow restriction of the positions where the errors are inserted in the codeword during the encryption, this would result to the intentional creation of the trapping sets and further increase of the cryptosystem security. Also, it would be interesting to develop the deterministic algorithm based on GDBF that outperforms PGDBF without using a random generator with fast convergence, that would further reduce the cryptosystem complexity.

References

- [1] Frank Miller, “Telegraphic code to Insure Privacy and Secrecy in the Transmission of Telegrams,” Charles M. Cornwell, New York, 1882.
- [2] S. M. Bellovin, “Frank Miller: Inventor of the One-Time Pad,” *Cryptologia*, vol. 35, no. 3, pp. 203–222, Jul. 2011.
- [3] National Bureau of Standards, “Data encryption standard,” Federal Information Processing Standards Publication 46, U.S. Department of Commerce, January 1977.
- [4] X. Lai and J. Massey, “A Proposal for a New Block Encryption Standard,” *Proc. of Advances in Cryptology EUROCRYPT '90, Berlin*, Springer-Verlag, 1991, pp. 389–404.
- [5] National Institute of Standards and Technology, “The Advanced Encryption Standard,” Federal Information Processing Standards Publication (FIPS) 197, 2001.
- [6] National Institute of Standards and Technology, “Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher,” Special Publication SP800-67, 2004.
- [7] Whitfield Diffie and Martin E. Hellman, “New directions in cryptography,” *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.
- [8] R. L. Rivest, A. Shamir, and L. Adleman, “A Method for obtaining digital signatures and public-key cryptosystems,” *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.
- [9] M. Rabin, “Digitalized signatures and public-key functions as intractable as factorization,” *Technical Report MIT/LCS/TR-212*, MIT Laboratory for Computer Science, 1979.
- [10] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” *IEEE Trans. Inform. Theory*, vol. 31, no. 4, pp. 469–472, 1985.
- [11] V. Miller, “Use of elliptic curves in cryptography,” *Advances in Cryptography Crypto '85, Springer-Verlag New York, LNCS*, vol.218, pp. 417–426, 1986.
- [12] N. Koblitz, “Elliptic curve cryptosystem,” *Mathematics of Computation*, vol. 48, 1987, pp. 203–209.
- [13] Sources: FIPS 186-2. NIST SP 800-57. ANSI X9.30.1 - 2002
- [14] V. S. Dimitrov, K. U. Jarvinen, M. J. Jacobson, W. F. Chan, and Z. Huang, “FPGA Implementation of Point Multiplication on Koblitz Curves Using Kleinian Integers,”

- in *Cryptographic hardware and Embedded Systems-CHES'06*, pp. 445-459, 2006.
- [15] T. Güneysu and C. Paar, “Ultra high performance ECC over NIST primes on commercial FPGAs,” in *Cryptographic Hardware and Embedded Systems – CHES 2008*, ser. LNCS, vol. 5154, pp. 62–78, 2008.
- [16] S. S. Roy, C. Rebeiro, and D. Mukhopadhyay, “A Parallel Architecture for Koblitz Curve Scalar Multiplications on FPGA Platforms,” in *DSD*, pages 553–559, 2012.
- [17] J. Hoffstein, J. Pipher and J. H. Silverman, “NTRU: A ring-based public key cryptosystem,” *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, vol. 1423, pp. 267–288, 1998.
- [18] J. Hermans et. al., “Speed Records for NTRU,” *CT-RSA 2010*, LNCS 5985, pp. 73–88, 2010.
- [19] R. J. McEliece, “A public-key cryptosystem based on algebraic coding theory,” *DSN Progress Report*, pp. 114–116, 1978.
- [20] D. J. Bernstein, T. Lange, and C. Peters, “Attacking and defending the McEliece cryptosystem,” in *Post-Quantum Cryptography*, ser. Lecture Notes in Computer Science. Springer Verlag, vol. 5299, pp. 31–46, 2008.
- [21] H. Niederreiter, “Knapsack-type cryptosystems and algebraic coding theory,” in *Problems Control Inform. Theory*, vol. 15, no. 2, pp. 159-166, 1986.
- [22] Y. X. Li, R. Deng, and X. M. Wang, “On the equivalence of McEliece’s and Niederreiter’s public-key cryptosystems,” *IEEE Trans. Inform. Theory*, vol. 40, no. 1, pp. 271–273, Jan. 1994.
- [23] <http://www.ntru.com>
- [24] Nobel Committee for Physics. Particle control in a quantum world. http://kva.se/Documents/Priser/Nobel/2012/fysik/pop_fy_en_12.pdf, Website accessed 2013-03-14.
- [25] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, 1997.
- [26] P. W. Shor, “Algorithms for Quantum Computation: Discrete Logarithms and Factoring,” in *Proc. 35th Annual Symp. on Foundations of Computer Science*, Nov. 1994, pp. 124-134.
- [27] L. M. K. Vandersypen, M. Steen, G. Breyta, C. S. Yannoni, M. H. Sherwood and I. L. Chuang, “Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance,” *Nature*, vol. 414, pp. 883-887, 2001.

- [28] N. Xu *et al.*, “Quantum factorization of 143 on a dipolar-coupling nuclear magnetic resonance system,” *Phys. Rev. Lett.*, vol. 108, p. 130501, Mar. 2012.
- [29] A. K. Lenstra and H. W. Lenstra, Jr., editors, “The development of the number field sieve,” *Lecture Notes in Computer Science*, Springer-Verlag, vol. 1554, 1993.
- [30] A. K. Lenstra and E. R. Verheul, “Selecting cryptographic key sizes,” *Journal of Cryptology* 14, pp. 255-293, 2001.
- [31] S. Cavallar, B. Dodson, A. K. Lenstra, W. M. Lioen, P. L. Montgomery, B. Murphy, H. J. J. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. C. Leyland, J. Marchand, F. Morain, A. Muffett, C. Putnam, C. Putnam, and P. Zimmermann, “Factorization of a 512-bit RSA modulus,” In *Advances in Cryptology – EUROCRYPT 2000*, Springer, Berlin / Heidelberg, 2000.
- [32] Daniel J. Bernstein, Johannes Buchmann and Erik Dahmen, “Post-Quantum Cryptography,” Springer, 2009.
- [33] Lov K. Grover, “A fast quantum mechanical algorithm for database,” in *Proc. 28th ACM Symp. On Theory of Computing (STOC)*, pp. 212–219, Philadelphia, May, 1996.
- [34] L. Minder and A. Shokrollahi, “Cryptanalysis of the Sidelnikov cryptosystem,” In *EUROCRYPT 2007*, vol. 4515 of Lecture Notes in Comput. Sci., pages 347–360, Barcelona, Spain, 2007.
- [35] M. Beermann, F. Wickert, P. Vary, “Highly flexible design of multi-rate multi-length quasi-cyclic LDPC codes,” *8th International Symposium on Turbo Codes and Iterative Information Processing (ISTC 2014)*, pp. 37 – 41, 2014.
- [36] D. Bernstein, “Grover vs. McEliece,” In *the third international workshop on Post-Quantum Cryptography PQCRYPTO 2010*, Lecture Notes in Computer Science.
- [37] R. Overbeck and N. Sendrier, “Code-based cryptography,” in *Post-quantum cryptography*, Springer, 2009, pp. 95–145.
- [38] F. Bahr, M. Böhm, J. Franke, T. Kleinjung, “Factorization of RSA-200,” public announcement on May 9th, 2005.
- [39] E. Prange, “The use of information sets in decoding cyclic codes,” *IEEE Transactions on Information Theory*, vol. 8, no. 5, pp. 5–9.
- [40] R. Niebuhr, et al. “Selecting Parameters for Secure McEliece-based Cryptosystems,” *Informational Journal of Information Security*, vol. 11, pp. 137-147, issue 3, June 2012.
- [41] P. Gaborit, “Shorter keys for code based cryptography,” in *Proceedings of the 2005*

- International Workshop on Coding and Cryptography (WCC 2005)*, pp. 81–91, Bergen, Norway, March 2005.
- [42] A. Otmani, J. P. Tillich, and L. Dallot, “Cryptanalysis of two McEliece cryptosystems based on quasi cyclic codes,” in *Proc. First International Conference on Symbolic Computation and Cryptography (SCC 2008)*, Beijing, China, April 2008.
- [43] C. Monico, J. Rosenthal, and A. Shokrollahi, “Using low density parity check codes in the McEliece cryptosystem,” in *IEEE International Symposium on Information Theory (ISIT'2000)*, page 215, Sorrento, Italy, 2000.
- [44] M. Baldi and G. F. Chiaraluce, “Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes,” in *IEEE International Symposium on Information Theory*, pp. 2591–2595, Nice, France, March 2007.
- [45] V. G. Umana and G. Leander, “Practical key recovery attacks on two McEliece variants,” in *Proceedings of the Second International Conference on Symbolic Computation and Cryptography, (SCC 2010)*, pp. 27-44, Royal Holloway, University of London, Egham, UK, 23–25 June 2010.
- [46] M. Baldi, M. Bodrato, and F. Chiaraluce, “A new analysis of the McEliece cryptosystem based on QC LDPC codes,” in *Security and Cryptography for Networks*, volume 5229 of Lecture Notes in Computer Science, pages 246–262. Springer Berlin / Heidelberg, 2008.
- [47] R. Misoczki, JP. Tillich, N. Sendrier, and P. Barreto, “MDPC-McEliece: New McEliece variants from moderate density parity-check codes,” in *Proc. IEEE International Symposium on Information Theory (ISIT2013)*, Istanbul, Turkey, pp. 2069–2073. 2013.
- [48] J. Hagenauer, E. Offer, and L. Papke, “Iterative decoding of binary block and convolutional codes,” *IEEE Trans. Inform. Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.
- [49] R. G. Gallager, *Low-density parity-check codes*, M.I.T. Press, 1963.
- [50] M. Baldi, “LDPC codes in the McEliece cryptosystem: attacks and countermeasures,” *NATO Science for Peace and Security Series - D: Information and Communication Security, IOS Press*, vol. 23, pp. 160–174, 2009.
- [51] A. Canteaut and F. Chabaud, “A new algorithm for finding minimum-weight words in a linear code: application to McEliece’s cryptosystem and to narrow-sense BCH codes of length 511,” *IEEE Trans. Inform. Theory*, vol. 44, 367–378, January 1998.
- [52] E. Berlekamp, R.J. McEliece and H.C.A. Van Tilborg, “On the inherent intractability of certain coding problems,” *IEEE Transactions on Information Theory*, vol. 24, no. 3,

- pp. 384-386, May 1978.
- [53] V. D. Goppa, "Rational Representation of Codes and (L, g) Codes," *Probl. Pered. Inform.*, vol. 7, pp. 41–49, September 1971.
 - [54] V. D. Goppa, "A new class of linear error-correcting codes," *Probl. Peredach Inform.*, vol. 6, no. 3, pp. 24-30, Sept. 1970.
 - [55] P. Loidreau, "Strengthening McEliece Cryptosystem," In *Proc. Of ASIACRYPT 2000*, Springer–Verlag, 2000.
 - [56] P. Fitzpatrick, J.A. Ryan, "On the number of irreducible Goppa codes," *Workshop on Coding and Cryptography 2003*, 2003.
 - [57] K. Kobara and H. Imai, "Semantically secure McEliece public-key cryptosystems-conversions for mceliece pkc," in *K. Kim, editor, Public Key Cryptography*, LNCS, vol. 1992, pp. 19–35. Springer, 2001.
 - [58] H. M. Sun, "Further cryptanalysis of the McEliece public-key cryptosystem," *IEEE Communications Letters*, vol. 4, no. 1, pp. 18–19, 2000.
 - [59] D. Engelbert, R. Overbeck, and A. Schmidt, "A summary of McEliece-type cryptosystems and their security," *Journal of Mathematical Cryptology*, vol. 1, no. 2, pp. 151–199, May 2007.
 - [60] F. R. Kschischang, B. J. Frey and H. A. Loeliger, "Factor graphs and the sum product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, February 2001.
 - [61] E. M Gabidulin, A.V. Paramonov, and O.V. Tretjakov, "Ideals over a non-commutative ring and their applications to cryptography," in *Proc. Eurocrypt '91*, volume 547 of LNCS. Springer Verlag, 1991.
 - [62] E. M Gabidulin, A .V. Ourivski, B. Honary, and B. Ammar, "Reducible rank codes and their applications to cryptography," in *IEEE Transactions on Information Theory*, vol. 49, no. 12, pp. 3289-3293, Dec. 2003.
 - [63] V. M. Sidelnikov, "A public-key cryptosystem based on binary Reed-Muller codes," *Discrete Mathematics and Applications*, vol. 4, no. 3, 1994.
 - [64] C. M. Adams, H. Meijer, "Security-Related Comments Regarding McEliece's Public-Key Cryptosystem," in *Proc. of CRYPTO '87*, LNCS 293, pp. 244-233, Springer, 1988.
 - [65] P. J. Lee, E. F. Brickell, "An Observation on the Security of McEliece's Public-Key Cryptosystem," *EUROCRYPT*, 1988.

- [66] V.M. Sidelnikov and S.O. Shestakov, “On the insecurity of cryptosystems based on generalized Reed-Solomon codes,” *Discrete Math. Appl.*, vol. 1, no. 4, pp. 439–444, 1992.
- [67] H. Janwa and O. Moreno, “McEliece public key cryptosystems using algebraic-geometric codes,” *Designs, Codes and Cryptography*, vol. 8, no. 3, pp. 293-307, August 1996.
- [68] C. Faure and L. Minder, “Cryptanalysis of the McEliece cryptosystem over hyperelliptic curves,” in *International Workshop on Algebraic and Combinatorial Coding Theory*, pp. 99 - 107, 2008.
- [69] A. Otmani, J.P. Tillich, and L. Dallot, “Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes,” *Special Issues of Mathematics in Computer Science*, vol. 3, no. 2, pp.129-140, January 2010.
- [70] R. Misoczki and P. S. L. M. Barreto, “Compact McEliece keys from Goppa codes,” *Cryptology ePrint Archive*, Report 2009/187, 2009.
- [71] T. P. Berger, P.-L. Cayrel, P. Gaborit, and A. Otmani, “Reducing key length of the McEliece cryptosystem,” in *Progress in Cryptology - AFRICACRYPT 2009*, ser. Lecture Notes in Computer Science. Springer Verlag, 2009, vol. 5580, pp. 77–97.
- [72] M. Baldi, F. Chiaraluce, R. Garello, and F. Mininni, “Quasi-cyclic low-density parity-check codes in the McEliece cryptosystem,” In *IEEE International Conference on Communications (ICC 2007)*, pp. 951-956, June 2007.
- [73] M. Baldi, M. Bianchi, F. Chiaraluce, J. Rosenthal, and D. Schipani, “Enhanced public key security for the McEliece cryptosystem,” 2011, <http://arxiv.org/abs/1108.2462>
- [74] H. M. Sun, “Improving the security of the McEliece public-key cryptosystem,” in *ASIACRYPT 1998*, ser. Lecture Notes in Computer Science. Springer Verlag, 1998, vol. 1514, pp. 200–213.
- [75] F. Strenzke, “Solutions for the Storage Problem of McEliece Public and Private Keys on Memory-constrained Platforms,” in *Proceedings of the 15th international conference on Information Security, ISC 2012*, Lecture Notes in Computer Science, Springer Berlin / Heidelberg (2012).
- [76] T. Eisenbarth, T. Güneysu, S. Heyse, and C. Paar, “MicroEliece: McEliece for Embedded Devices,” *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems (Berlin, Heidelberg)*, CHES '09, Springer-Verlag, 2009, pp. 49–64.
- [77] A. Shoufan, T. Wink, G. Molter, S. Huss, F. Strenzke, “A Novel Processor

- Architecture for McEliece Cryptosystem and FPGA Platforms,” *20th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2009)*, pp.98-105, July 2009.
- [78] S. Heyse, I. Von Maurich, T. Güneysu, “Smaller keys for code-based cryptography: QC-MDPC McEliece implementations on embedded devices,” in *Cryptographic Hardware and Embedded Systems (CHES '13)*, vol. 8086, pp. 273–292, Springer, Berlin, 2013.
- [79] F. Strenzke, “A smart card implementation of the McEliece PKC”, in *Proc. of the 4th IFIP WG 11.2 Internat. Conf. on Information Security Theory and Practices: Security and Privacy of Pervasive Systems and Smart Devices WISTP '10, Passau, Germany, 2010* (P. Samarati et al., eds.), *Lecture Notes in Comput. Sci.*, vol. 6033, Springer, Berlin, 2010, pp. 47–59.
- [80] D.J.C. MacKay, “Good error-correcting codes based on very sparse matrices,” in *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp.399-431, Mar. 1999.
- [81] I. Djurdjevic, J. Xu, K. Abdel-Ghaffar and S. Lin, “A class of low-density parity-check codes constructed based on Reed-Solomon codes with two information symbols,” *Communications Letters, IEEE* , vol.7, no.7, pp. 317,319, July 2003.
- [82] J. L. Fan, “Array codes as low-density parity check codes,” in *Proc. of the Intl. Symp. on Turbo Codes*, 2000, pp. 543–546.
- [83] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, Jr., “LDPC block and convolutional codes based on circulant matrices,” *IEEE Trans. on Inform. Theory*, vol. 50, no. 12, pp. 2966–2984, Dec. 2004.
- [84] R.M. Tanner, “A recursive approach to low complexity codes,” *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533-547, Sep. 1981.
- [85] H. Xiao, E. Eleftheriou and D. Arnold, “Regular and irregular progressive edge-growth tanner graphs,” *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 386-398, Jan. 2005.
- [86] H. Xiao and A. H. Banihashemi “Improved Progressive-Edge Growth (PEG) Construction of Irregular LDPC codes,” *IEEE Communications Letters*, vol. 8, no. 12, pp. 715-717, Dec. 2004.
- [87] T. Richardson, M. Shokrollahi and R. Urbanke, “Design of capacity-approaching irregular low-density parity check codes,” *IEEE Trans. Inform. Theory*, vol.47, pp. 619-637, Feb. 2001.
- [88] Z. Li, L. Chen, L. Zeng, S. Lin, and W. H. Fong, “Efficient Encoding of Quasi-Cyclic Low-Density Parity-Check Codes,” *IEEE Transactions on Communications*,

- vol. 54, no. 1, Jan. 2006, pp. 71–81.
- [89] C. Di, D. Proietti, I. E. Telatar, T. Richardson, and R. Urbanke, “Finite-length analysis of low-density parity-check codes on the binary erasure channel,” *IEEE Trans. Inform. Theory*, vol. 48, pp. 1570–1579, June 2002.
 - [90] H. Qi and N. Goertz, “Low-Complexity Encoding of LDPC Codes: A New Algorithm and its Performance,” available at http://publik.tuwien.ac.at/files/PubDat_166941.pdf, (06. 04. 2011).
 - [91] T. J. Richardson R. L. Urbanke, “Efficient encoding of low-density parity-check codes,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 638–656, Feb. 2001.
 - [92] F. Guilloud, E. Boutillon, and J. Danger, “Lambda-Min Decoding Algorithm of Regular and Irregular LDPC Codes,” in *Proc. 3rd International Symposium on Turbo Codes and Related Topics*, Brest, France, pp. 451-454, Sep. 2003.
 - [93] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, “Reduced complexity decoding of LDPC codes,” *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.
 - [94] V. Savin, “Self-corrected min-sum decoding of LDPC codes,” in *Proc. of IEEE Int. Symp. on Information Theory (ISIT)*, 2008, pp. 146–150.
 - [95] S.K. Chilappagari, D.V. Nguyen, B. Vasic, M.W. Marcellin, “Error correction capability of column-weight-three LDPC codes under the Gallager A algorithm - Part II,” *IEEE Transactions on Information Theory*, vol. 56, no. 6, pp. 2626-2639, June 2010.
 - [96] S.K. Chilappagari and B. Vasic, “Error-correction capability of column-weight-three LDPC codes,” *IEEE Transactions on Information Theory*, vol.55, no.5, pp. 2055-2061, May 2009.
 - [97] Y.Kou, S.Lin, and M.P.C Fossorier, “Low-density parity-check codes based on finite geometries: a rediscovery and new results,” *IEEE Trans.Inform. Theory*, pp. 2711–2736, vol. 47, Nov. 2001.
 - [98] J.Zhang, and M.P.C.Fossorier, “A modified weighted bit-flipping decoding of low density parity-check codes,” *IEEE Communications Letters*, pp. 165–167, vol. 8, Mar. 2004.
 - [99] M.Jiang, C.Zhao, Z.Shi, and Y.Chen, “An improvement on the modified weighted bit flipping decoding algorithm for LDPC codes,” *IEEE Communications Letters*, vol. 9, no. 9, pp. 814–816, 2005.

- [100] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, "Gradient descent bit flipping algorithms for decoding LDPC codes," *IEEE Trans. Commun.*, vol. 58, no. 6, pp.1610–1614, 2010.
- [101] N. Miladinovic and M. Fossorier, "Improved bit-flipping decoding of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 51, no. 4, pp. 1594–1606, Apr. 2005.
- [102] D. V. Nguyen, M. W. Marcellin, B. Vasic, "Two-bit bit flipping decoding of LDPC codes," *Proc. IEEE Int. Symp. on Inform. Theory, St. Petersburg, Russia*, Jul. 31 - Aug. 5 2011, pp. 1995 – 1999.
- [103] G. Sundararajan, C. Winstead, and E. Boutillon, "Noisy gradient descent bit-flip decoding for LDPC codes," *IEEE Trans. Commun.*, vol. 62, no. 10, pp. 3385–3400, Oct. 2014.
- [104] Proakis, John G, "Digital communications," *McGraw-Hill, New York*, 1995.
- [105] R. M. Tanner, D. Sridhara, and T. Fuja, "A Class of Group-Structured. LDPC Codes," in *ISCTA*, Ambleside, England, 2001.
- [106] S.Y. Chung, G. Forney, T. Richardson, R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Communications Letters*, vol. 5, pp. 58-60, 2001.
- [107] T. Richardson, "Error floors of LDPC codes," in *Proc. 41st Allerton Conf. on Communications, Control, and Computing*, (Allerton House, Monticello, Illinois, USA), October 1–3 2003.
- [108] B. Vasic, S. K. Chilappagari, D. V. Nguyen, and S. K. Planjery, "Trapping set ontology," in *Proc. Allerton Conf. on Commun., Control, and Computing*, Sep. 2009.
- [109] S. Ghosh, K. Roy, "Parameter variation tolerance and error resiliency: New design paradigm for the nanoscale era," *Proceedings of IEEE*, vol. 98, no. 10, pp. 1718–1751, Oct. 2010.
- [110] B. Vasic and S. K. Chilappagari, "An information theoretical framework for analysis and design of nanoscale fault-tolerant memories based on low-density parity-check codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 11, pp. 2438–2446, Nov. 2007.
- [111] S. Chilappagari, M. Ivkovic, and B. Vasic, "Analysis of one step majority logic decoders constructed from faulty gates," in *Proc. 2006 IEEE International Symposium on Information Theory*, July, pp. 469–473.
- [112] S. Brkic, P. Ivanis, and B. Vasic, "Analysis of one-step majority logic decoding under

- correlated data-dependent gate failures,” in *Proc. 2014 IEEE International Symposium on Information Theory (ISIT)*, June, pp. 2599–2603.
- [113] L. Varshney, “Performance of LDPC Codes Under Faulty Iterative Decoding,” *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4427–4444, July 2011.
- [114] C. Kameni Ngassa, V. Savin, and D. Declercq, “Min-Sum-based decoders running on noisy hardware,” in *Proc. 2013 IEEE GLOBECOM*, Dec. 2013.
- [115] C.-H. Huang and L. Dolecek, “Analysis of finite-alphabet iterative decoders under processing errors,” in *Proc. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May, pp. 5085–5089.
- [116] S. Tabatabaei Yazdi, H. Cho, and L. Dolecek, “Gallager B Decoder on Noisy Hardware,” *IEEE Transactions on Communications*, vol. 61, no. 5, pp. 1660–1673, May 2013.
- [117] F. Leduc-Primeau and W. Gross, “Faulty Gallager-B decoding with optimal message repetition,” in *Proc. 50th Annual Allerton Conference on Communication, Control, and Computing*, pp. 549–556, Oct. 2012.
- [118] L. Danjean and S. K. Planjery, [http://www2.engr.arizona.edu/~vasiclab/tools/LDPC Code List](http://www2.engr.arizona.edu/~vasiclab/tools/LDPCCodeList).
- [119] K. Le, D. Declercq, C. Spagnol, P. Ivanis, B. Vasic, “Efficient Realization Of Probabilistic Gradient Descent Bit Flipping Decoders,” *Proc. 21-st IEEE International Conference on Electronics Circuits and Systems (ISCAS 2015)*, Lisbon, Portugal, May 24 - 27 2015, pp. 1-4.
- [120] T. Richardson and R. Urbanke, *Modern Coding Theory*, Cambridge University Press, 2008.
- [121] J. Stern, A method for finding codewords of small weight, In G. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, vol. 388 of Lecture Notes in Computer Science, pp. 106–113. Springer, 1989.
- [122] X. Y. Hu, E. Eleftheriou, D. M. Arnold, and A. Dholakia, “Efficient implementations of the sum-product algorithm for decoding LDPC codes,” In *Proc. IEEE Global Telecommunications Conference (GLOBE-COM '01)*, vol. 2, pp. 1036–1036, San Antonio, TX, November 2001.
- [123] J. C. Faugère, A. Otmani, L. Perret and J. P. Tillich, “Algebraic Cryptanalysis of Compact McEliece’s Variants - Toward a Complexity Analysis,” In *International Conference on Symbolic Computation and Cryptography*, SCC 2010, pp. 45-56, 2010.

- [124] M. P. C. Fossorier, "Quasicyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1788–1793, Aug. 2004.
- [125] M. Baldi, "QC-LDPC Code-Based Cryptography," *Springer Briefs in Electrical and Computer Engineering*, Springer 2014, ISBN 978-3-319-02555-1, pp. 1-120.
- [126] M. Baldi, F. Chiaraluce, "New Quasi Cyclic Low Density Parity Check codes based on Difference Families," In *Proc. Int. Symp. Commun. Theory and Appl., (ISCTA 05)*, Ambleside, UK, pp 244–249, July 2005.
- [127] T. A. Berson, "Failure of the McEliece public-key cryptosystem under message-resend and related-message attack," In *CRYPTO '97, LNCS*, Vol. 1294, pp 213-220, 1997.
- [128] M. P.C. Fossorier, K. Kobara, H. Imai, "Modeling Bit Flipping Decoding Based on Non Orthogonal Check Sums with Application to Iterative Decoding Attack of McEliece Crypto-System," *IEEE Trans. Inform. Theory*, vol. 53, no. 1, pp. 402-411, 2007.
- [129] M. Hiroto, M. Mohri, and M. Morii, "A probabilistic computation method for the weight distribution of low-density parity-check codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Adelaide, Australia, Sep. 2005, pp. 2166–2170.
- [130] P. Camion and J. Patarin, "The knapsack hash function proposed at CRYPTO '89 can be broken," In *D.W. Davies, editor, Advances in Cryptology - EUROCRYPT '91*, vol. 547 of LNCS, pages 39–53. Springer, 1991.
- [131] J. S. Leon, "A probabilistic algorithm for computing minimum weights of large error correcting codes," *IEEE Transactions on Information Theory*, vol. 34, pp. 1354–1359, 1988.
- [132] A. Becker, A. Joux, A. May, A. Meurer, "Decoding random binary linear codes in $2n/20$: How $1 + 1 = 0$ improves information set decoding," In *EUROCRYPT 2012*, Lecture Notes in Computer Science, vol. 7237, Springer Verlag, pp. 520–536, 2012.
- [133] M. R. Yazdani, S. Hemati, A. H. Banihashemi, "Improving belief propagation on graphs with cycles," *IEEE Communications Letters*, vol. 8, no. 1, pp. 57-59, 2004.
- [134] D. V. Nguyen, S. K. Chilappagari, B. Vasic, and M. W. Marcellin, "On the construction of structured LDPC codes free of small trapping sets," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2280-2302, Apr. 2012.

Прилог 1.

Изјава о ауторству

Потписани-а OMRAN AL KASHEED

број уписа 5049/2011

Изјављујем

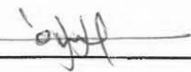
да је докторска дисертација под насловом

АЛГОРИТМИ ДЕЦИДОВАЊА МАЛЕ КОМПЛЕКСНОСТИ
ПОГОВНИ ЗА ПРИМЕНУ У АСИМЕТРИЧНИМ КРИПТОСИСТЕМИМА

- резултат сопственог истраживачког рада,
- да предложена дисертација у целини ни у деловима није била предложена за добијање било које дипломе према студијским програмима других високошколских установа,
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио интелектуалну својину других лица.

Потпис докторанда

У Београду, 8.10.2015.



Прилог 2.

**Изјава о истоветности штампане и електронске
верзије докторског рада**

Име и презиме аутора OMMAN AL RASHEED

Број уписа 5049/2011

Студијски програм ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Наслов рада АЛГОРИТМИ ДЕКРИПТОВАЊА МАЛЕ КОМПЛЕКСНОСТИ
ПОГОВНИ ЗА ПРИМЕНУ У АСИМЕТРИЧНИМ КРИПТОСИСТЕМИМА

Ментор проф. др ПРЕДРАГ ИВАНЧИШ

Потписани OMMAN AL RASHEED


изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла за објављивање на порталу Дигиталног репозиторијума Универзитета у Београду.

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис докторанда

У Београду, 8. 10. 2015.



Прилог 3.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

АЛГОРИТМИ ЗА ВЕНДОВАЊА МАЛЕ КОМПЛЕКСНОСТИ
ПОГОВНИ ЗА ПРИМЕНУ У АСИМЕТРИЧНИМ КРИПТОСИСТЕМИМА

која је моје ауторско дело.

Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Београду могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

1. Ауторство
2. Ауторство - некомерцијално
3. Ауторство – некомерцијално – без прераде
4. Ауторство – некомерцијално – делити под истим условима
5. Ауторство – без прераде
6. Ауторство – делити под истим условима

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на полеђини листа).

Потпис докторанда

У Београду, 8.10.2015.

