INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]

MAJORITY-BASED AND HYBRID ITERATIVE
DECODING OF LOW-DENSITY PARITY-CHECK CODES

BY

Pirouz Zarrinkhat

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

OTTAWA-CARLETON INSTITUTE FOR ELECTRICAL AND COMPUTER ENGINEERING

FACULTY OF ENGINEERING

DEPARTMENT OF SYSTEMS AND COMPUTER ENGINEERING

CARLETON UNIVERSITY

OTTAWA, ONTARIO

APRIL 20TH, 2005

© Copyright by Pirouz Zarrinkhat, 2005

Library and
Archives Canada

Bibliothèque et
Archives Canada

0-494-06749-7

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

CARLETON UNIVERSITY
DEPARTMENT OF SYSTEMS AND COMPUTER ENGINEERING

The undersigned hereby recommend to the Faculty of Graduate Studies and Research the acceptance of the thesis "**Majority-Based and Hybrid Iterative Decoding of Low-Density Parity-Check Codes**" submitted by **Pirouz Zarrinkhat** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Dated: April 20th, 2005

Dr. Rafik A. Goubran
Chair, Department of Systems
and Computer Engineering

Dr. Marc P. C. Fossorier
External Examiner

Dr. Amirhossein Banihashemi
Thesis Supervisor

Abstract

This doctoral thesis presents new iterative decoding algorithms for low-density parity-check (LDPC) codes, and discusses the corresponding code design methods. We start by studying a class of hard-decision iterative decoding algorithms, referred to as *majority-based* (MB) algorithms, which are particularly attractive for their simple implementation. We show the great advantages that some MB decoding algorithms are capable of offering over such benchmarks as **Gallager's** decoding algorithm A ($\mathcal{G}_A$) and majority decoding algorithm. In particular, we demonstrate that while the asymptotic convergence speed of $\mathcal{G}_A$ is exponential with iteration number, for other MB algorithms, it is super-exponential. Moreover, we show that some MB algorithms outperform $\mathcal{G}_A$ and majority decoding algorithm by a large margin, in terms of threshold values, and derive tight bounds on these thresholds. The fast convergence and the better performance offered by MB algorithms, suggest a serious role for them in future coding schemes. We then introduce *hybrid* decoding algorithms in a general framework. Hybrid algorithms combine existing iterative decoding algorithms in order to improve the decoding performance and/or speed of convergence. We propose a family of hybrid algorithms, called *time-invariant* hybrid. These algorithms use a specific blend of different algorithms in each iteration. The blend, however, does not change by iteration. We demonstrate, through various examples, that hybrid algorithms are capable of substantially improving error performance and convergence speed at the expense of a small increase in the complexity of the decoder's structure. We obtain optimized hybrid algorithms (jointly with the optimized code structures in the case of irregular codes). We also compare (optimized) time-invariant hybrid

algorithms with better-known switch-type ones, and show that although switch-type hybrid algorithms outperform their time-invariant counterparts asymptotically, and when the channel parameter is perfectly known, the trend changes at finite block lengths and limited maximum number of iterations, or when a perfect knowledge of the channel is not available. This suggests that robust time-invariant hybrid algorithms could be practically more attractive than switch-type hybrid algorithm.

Acknowledgements

I would like to thank Dr. Amirhossein Banihashemi, my thesis supervisor, for his continuous support and encouragement throughout my Ph.D. program. He was always available for fruitful discussions, and his ideas and editorial efforts have greatly improved the quality of this dissertation.

I am thankful to Dr. Marc P. C. Fossorier of the University of Hawai'i at Manoa, Dr. Claude D'Amours of the University of Ottawa, Dr. Calvin Plett of the Department of Electronics, Dr. David D. Falconer, Dr. Halim Yanikomeroglu, and Dr. Ian D. Marsland of the Department of Systems and Computer Engineering, for their willingness to act as examining committee members, and for their helpful comments and advice.

I am also thankful to the Government (and the people) of Ontario, Carleton University, the David and Rachel Epstein Foundation, and BAE Systems Canada Inc., for their financial support of my work in the form of scholarships and a bursary.

I would also like to thank all my friends in our Department, especially Mohammadreza Yazdani, Arash Shokrani Chaharsoughi, Hua Xiao, and Christophe L. Herry, all the helpful staff members of the Department, especially Blazenka Power and Darlene Hébert, and the following individuals (and they know why): Johann S. Bach, Wolfgang A. Mozart, and Ludwig van Beethoven.

Last, but not least, I am truly grateful to my family, for their love and support.

Ottawa, April 20, 2005

Pirouz Zarrinkhat

To my family

Table of Contents

Table of Contents	vii
List of Tables	x
List of Figures	xi
List of Notations and Abbreviations	xvi
1 Introduction	1
1.1 Organization of the Thesis	3
2 Preliminaries and Background	5
2.1 Digital Communication Systems	7
2.2 Channel Models	9
2.2.1 Binary Erasure Channel (BEC)	9
2.2.2 Binary Symmetric Channel (BSC)	9
2.2.3 BSC with Erasure (BSC-E)	11
2.2.4 Gaussian Channel (GC)	11
2.2.5 Power-Limited GC (PLGC)	13
2.2.6 Binary Input PLGC (BIPLGC)	13
2.2.7 Band-Limited PLGC (BLPLGC)	13
2.3 Capacity for Non-Vanishing Error Probabilities	16
2.4 Channel Codes	18
2.4.1 Linear Block Codes	20
2.5 Low-Density Parity-Check (LDPC) Codes	21
2.5.1 Structure of LDPC Codes	22
2.5.2 Graphical Representations of LDPC Codes	23
2.6 Message-Passing Decoders	26
2.6.1 Algorithms for Discrete-Alphabet Channels	28

2.6.2	Algorithms for Continuous-Alphabet Channels	30
2.7	Threshold and Density Evolution	34
2.7.1	Symmetry Conditions	35
2.7.2	Formulation of Density Evolution	36
2.8	Approximations to Density Evolution	38
3	Majority-Based (MB) Decoding of Regular LDPC Codes	40
3.1	MB Algorithms	41
3.1.1	MB Algorithms: Formal Definition	41
3.1.2	MB Algorithms: Density Evolution	42
3.2	Dynamics of MB Algorithms	43
3.2.1	Threshold Characterizations and Bounds	43
3.2.2	Speed of Convergence	55
3.3	Simulation Results and Special Cases	59
3.4	Conclusion	66
4	Hybrid Decoding of Regular LDPC Codes	68
4.1	Hybrid Decoding Algorithms	69
4.1.1	Formal Definition	69
4.1.2	$\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ Algorithms	72
4.2	Convergence Properties of Hybrid Algorithms	79
4.2.1	Maximum Threshold and Convergence Speed of Hybrid Algorithms	79
4.2.2	Performance Comparison of MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ Algorithms (Maximum Threshold and Finite-Length Simulation)	89
4.2.3	Comparison Between MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ Algorithms in the Absence of Perfect Knowledge of the Channel	97
4.3	Conclusion	108
5	Hybrid Decoding of Irregular LDPC Codes	109
5.1	MB $\mathcal{H}_{TI}$ Algorithms for Decoding Irregular LDPC Codes	110
5.1.1	Density Evolution	110
5.1.2	Threshold Characterization and Bounds	113
5.1.3	Asymptotic Speed of Convergence	114
5.2	Optimal Degree Distribution Pairs	115
5.3	Comparisons with Optimized $\mathcal{G}_A$ and $\mathcal{G}_B$	118
5.4	Finite-Length Simulation Results	128
5.5	Conclusion	130

6 Concluding Remarks	133
6.1 Summary of Contributions	133
6.2 Possible Future Work	135
A Density Evolution for Min-Sum Decoding Algorithm	137
A.1 Derivation of Density Evolution Formulas for Min-Sum Decoding Algorithm	137
B Graphical Representation of a General Recursion	142
C Computation of Threshold for Two-Stage $\mathcal{H}_{ST}$ Algorithms	144
C.1 Convergence Region of MB Algorithms and Threshold of Two-Stage $\mathcal{H}_{ST}$ Algorithms	144
D Monotonicity of BSCs with Respect to MB $\mathcal{H}_{TI}$ Algorithms for Decoding Irregular Codes	149
E Stability Condition of the Fixed-Point Located at Zero for MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ Algorithms	151
Bibliography	153

List of Tables

3.1	Threshold values and smallest steady-points for different orders of MB algorithms	61
3.2	Number of iterations required to achieve an average fraction of errors equal to, or less than, 10^{-6} . The channel parameter is 90% of the smallest threshold amongst different orders of MB algorithms	64
4.1	Threshold values for MB algorithms, and maximum threshold values for MB $\mathcal{H}_{ST}$ and $\mathcal{H}_{TI}$ algorithms	91
4.2	Optimal blending vectors $\vec{\alpha}_{\{\text{MB } \mathcal{H}_{TI}\}}^*$ for MB $\mathcal{H}_{TI}$ algorithms	92
4.3	Optimal switching-sequences for MB $\mathcal{H}_{ST}$ algorithms	93
5.1	Maximum achievable rates and corresponding degree distribution pairs under MB $\mathcal{H}_{TI}$ algorithms	117
C.1	Threshold values for $\mathcal{C}^\infty(3, 6)$, $\mathcal{C}^\infty(4, 6)$, and $\mathcal{C}^\infty(4, 8)$, decoded by $k = 0, 1, \dots, 6$ iterations of min-sum, and then ∞ iterations of MB^0 . . .	148

List of Figures

2.1	Block diagram of a digital communication system.	8
2.2	Capacity of the BEC as a function of erasure probability.	10
2.3	Capacity of the BSC as a function of cross-over probability.	12
2.4	Capacity of the PLGC as a function of $\frac{P}{\sigma^2}$	14
2.5	Capacity of the BIPLGC as a function of $\frac{1}{\sigma^2}$	15
2.6	Bandwidth-efficiency diagram.	17
2.7	Shannon limit curves for rates $r = \frac{1}{3}, \frac{1}{2}$, and $\frac{2}{3}$, and for BSC.	19
2.8	Shannon limit curves for rates $r = \frac{1}{3}, \frac{1}{2}$, and $\frac{2}{3}$, and for BIPLGC.	19
2.9	The Tanner graph corresponding to the code given in Example 2.5.1.	25
2.10	Illustration of check-nodes and variable-nodes' operations throughout the iteration process: in iteration 0, check-nodes are inactive (a), and variable-nodes pass their channel messages (b); in iteration $\ell > 0$, check-nodes' outgoing messages are determined by their incoming messages in iteration $\ell - 1$ (c), and variable-nodes determine their outgoing messages based on their channel messages, as well as their incoming messages in iteration ℓ (d).	27
3.1	Graphical interpretation of density evolution for $\mathcal{C}^\infty(3, 6)$, decoded by MB^0 , and for $\varepsilon = 0.0350$ and 0.0450	45

3.2	Graphical interpretation of density evolution for $\mathcal{C}^\infty(4, 6)$, decoded by MB^1 , and for $\varepsilon = 0.0800$ and 0.1000 : (a) an example of a decreasing trend of average error probability that does not converge to zero; (b) an example of an increasing trend.	49
3.3	Starting-point loci for $\mathcal{C}^\infty(3, 4)$ and $\mathcal{C}^\infty(5, 10)$, decoded by MB^0 and MB^1 , respectively.	54
3.4	Comparison of convergence speed of different orders of MB decoding algorithms for $\mathcal{C}^\infty(8, 10)$, and for $\varepsilon = 1/63$, which is the threshold for MB^3	58
3.5	BER curves for a randomly constructed $(6, 10)$ -regular LDPC code of length 10^5 , decoded by different orders of MB algorithms: (a) horizontal axis represents E_b/N_0 in dB; (b) horizontal axis represents ε . (Note that MB^2 is the same as $\mathcal{G}_A$.)	67
4.1	A snapshot of a $(3, 5)$ -regular LDPC code of length 10 decoded by a hybrid algorithm which involves two constituent algorithms. White nodes, which comprise 60% of the variable-nodes and 50% of the check-nodes, process the messages according to the first algorithm, while black nodes treat the messages in accordance with the second one. . .	71
4.2	Graphical representation of density evolution for $\mathcal{C}^\infty(8, 9)$, decoded by MB^0 , MB^1 , MB^2 , MB^3 , and $\mathcal{H}_{TI}(MB^0, MB^1, MB^2, MB^3, \bar{\alpha}_o)$, with $\bar{\alpha}_o$ given in Example 4.1.2, over a BSC with crossover probability $\varepsilon_o = 0.0720$	75
4.3	Graphical representation of density evolution for $\mathcal{C}^\infty(8, 9)$, decoded by MB^0 , MB^1 , MB^2 , MB^3 , and $\mathcal{H}_{ST}(MB^0, MB^1, MB^2, MB^3, \{\bar{\alpha}_o^{(\ell)}\}_{\ell=0}^\infty)$, with $\{\bar{\alpha}_o^{(\ell)}\}_{\ell=0}^\infty$ given in Example 4.1.3, over a BSC with crossover probability $\varepsilon_o = 0.0720$. (Switching-points are shown by $\star$.)	76

- 4.4 Graphical representation of density evolution for $\mathcal{C}^\infty(8, 9)$, decoded by MB^0, MB^1, MB^2, MB^3 , and $\mathcal{H}_{ST}(MB^0, MB^1, MB^2, MB^3, \{\tilde{\alpha}_\circ^{(\ell)}\}_{\ell=0}^\infty)$, with $\{\tilde{\alpha}_\circ^{(\ell)}\}_{\ell=0}^\infty$ given in Example 4.1.4, over a BSC with crossover probability $\varepsilon_\circ = 0.0720$. Note that $\Delta_\circ$ starts with MB^2 and switches to MB^1 in iteration 5. 78
- 4.5 Graphical representation of density-evolution for $\mathcal{C}^\infty(8, 9)$, decoded by MB^0, MB^1, MB^2, MB^3 , and $\mathcal{H}_{ST}(MB^0, MB^1, MB^2, MB^3, \{\tilde{\alpha}_\circ^{(\ell)}\}_{\ell=0}^\infty)$, with $\{\tilde{\alpha}_\circ^{(\ell)}\}_{\ell=0}^\infty$ given in Example 4.1.5, over a BSC with crossover probability $\varepsilon_\circ = 0.0700$. Note that $\Delta_\circ$ starts with MB^2 and switches to MB^1 in iteration 4. 80
- 4.6 Convergence domains associated with channel parameters $\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*$, $0.9 \cdot \varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*$, and $0.8 \cdot \varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*$, for $\mathcal{C}^\infty(5, 10)$. Observe that $\mathbb{N}_{\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*} \subset \mathbb{N}_{0.9 \cdot \varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*} \subset \mathbb{N}_{0.8 \cdot \varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*}$ 85
- 4.7 Graphical representation of density evolution for $\mathcal{C}^\infty(6, 7)$, decoded by MB^0, MB^1, MB^2 , and $\mathcal{H}_{ST}(MB^0, MB^1, MB^2, \{\tilde{\alpha}_{\varepsilon_\circ}^{\mathcal{G}_B(\ell)}\}_{\ell=0}^\infty)$, with $\{\tilde{\alpha}_{\varepsilon_\circ}^{\mathcal{G}_B(\ell)}\}_{\ell=0}^\infty$ given in Example 4.2.3, over a BSC with crossover probability $\varepsilon_\circ = 0.0825$. Note that $\Delta_\circ$ starts with MB^2 , and switches to MB^1 and MB^0 after iterations 2 and 9, respectively. 90
- 4.8 BER curves for a randomly constructed $(8, 9)$ -regular LDPC code of length 9×10^3 , with no cycle of length 4, decoded by MB^0, MB^1, MB^2, MB^3 , $\Delta_{TI}^* = \mathcal{H}_{TI}(MB^0, MB^1, MB^2, MB^3, \tilde{\alpha}_{\{\text{MB } \mathcal{H}_{TI}\}}^*)$ (marked by $\triangleright$), and $\Delta_{ST}^* = \mathcal{H}_{ST}(MB^0, MB^1, MB^2, MB^3, \{\tilde{\alpha}_{\{\text{MB } \mathcal{H}_{ST}\}}^{(\ell)}\}_{\ell=0}^\infty)$ (marked by $\triangleleft$). 96
- 4.9 BER curves for a randomly constructed $(5, 8)$ -regular LDPC code of length 8×10^3 , decoded by $MB^0, MB^1, \mathcal{H}_{TI}(MB^0, MB^1, \tilde{\alpha}_{\{\text{MB } \mathcal{H}_{TI}\}}^*)$ (marked by $\triangleright$), and $\mathcal{H}_{ST}(MB^0, MB^1, \{\tilde{\alpha}_{\{\text{MB } \mathcal{H}_{ST}\}}^{(\ell)}\}_{\ell=0}^\infty)$ (marked by $\triangleleft$). 98

4.10	Comparison between the convergence speed of MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ algorithms, for $\mathcal{C}^\infty(4, 5)$, and for $\varepsilon = \Gamma \cdot \varepsilon_{\{\text{MB } \mathcal{H}_{ST}\}}^*$ ($\varepsilon_{\{\text{MB } \mathcal{H}_{ST}\}}^* = 0.0963$), $\Gamma = 0.25, 0.50, 0.75, 0.80, 0.90$, and 1.00 (left to right). (Both algorithms are set to achieve their maximum thresholds.)	100
4.11	Graphical representation of density-evolution for $\mathcal{C}^\infty(4, 5)$, decoded by $MB^0, MB^1, \Delta_{TI}^* = \mathcal{H}_{TI}(MB^0, MB^1, \bar{\alpha}_{\{\text{MB } \mathcal{H}_{TI}\}}^*)$, and $\Delta_{ST}^* = \mathcal{H}_{ST}(MB^0, MB^1, \{\bar{\alpha}_{\{\text{MB } \mathcal{H}_{ST}\}}^{*(\ell)}\}_{\ell=0}^\infty)$, for $\varepsilon_0 = 0.8 \cdot \varepsilon_{\{\text{MB } \mathcal{H}_{ST}\}}^* = 0.0770$. Note that Δ_{ST}^* starts with MB^1 , and switches to MB^0 in iterations 58. (The optimum switching time is in iteration 7: the dotted line shows the corresponding evolution of the average error probability for the optimal switching.)	101
4.12	Comparison between the convergence speed of MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ algorithms, for $\mathcal{C}^\infty(8, 9)$, and for $\varepsilon = \Gamma \cdot \varepsilon^*$ ($\varepsilon^* = 0.0756$), $\Gamma = 0.25, 0.50, 0.75, 0.90, 0.95$, and 1.00 (left to right). (Both algorithms are set to achieve their maximum thresholds.)	103
4.13	Comparison between the convergence speed of MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ algorithms, for $\mathcal{C}^\infty(4, 5)$. Subscripts and superscripts of the curve labels represent the actual and the estimated channel parameters, respectively ($TI_{\varepsilon_{act}}^{\varepsilon_{est}}, ST_{\varepsilon_{act}}^{\varepsilon_{est}}$). [Refer to Example 4.2.8.]	105
4.14	(a) Evolution of the average error probability for $\mathcal{C}^\infty(4, 5)$, decoded by an MB $\mathcal{H}_{ST}$ algorithm optimized for an estimated channel parameter $\varepsilon_{est} = 0.0900$, over a channel with actual parameter $\varepsilon_{act} = 0.0915$; (b) Improper switching results in divergence: the dotted line shows the evolution of the average error probability when the algorithm is optimized for the channel parameter $\varepsilon = 0.0915$	106
4.15	Comparison between the convergence speed of MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ algorithms, for $\mathcal{C}^\infty(8, 9)$. Subscripts and superscripts of the curve labels represent the actual and the estimated channel parameters, respectively ($TI_{\varepsilon_{act}}^{\varepsilon_{est}}, ST_{\varepsilon_{act}}^{\varepsilon_{est}}$). [Refer to Example 4.2.9.]	107

5.1	Graphical interpretation of density-evolution; (a) for $\mathcal{C}^\infty(\bar{\Lambda}_o, \bar{\rho}_o)$, decoded by the MB $\mathcal{H}_{TI}$ algorithm given in Example 5.3.1, assuming $\varepsilon = \varepsilon^*(\bar{\Lambda}_o, \bar{\rho}_o) = 0.0525$; (b) for $\mathcal{C}^\infty(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$, decoded by $\mathcal{G}_B$, assuming $\varepsilon = \varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B}) = 0.0577$. (See Example 5.3.2.)	124
5.2	Graphical interpretation of density-evolution; (a) for $\mathcal{C}^\infty(\bar{\Lambda}_o, \bar{\rho}_o)$, decoded by the optimized MB $\mathcal{H}_{TI}$ algorithm given in Example 5.3.1; (b) for $\mathcal{C}^\infty(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$, decoded by the $\mathcal{G}_B$ optimized for $\varepsilon = \varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B}) = 0.0577$; both under the assumption of $\varepsilon = \frac{3}{4} \cdot \varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B}) = 0.043275$. (See Example 5.3.2.)	125
5.3	Comparison between the convergence speed of the MB $\mathcal{H}_{TI}$ algorithm given in Example 5.3.1, and $\mathcal{G}_B$, over $\mathcal{C}^\infty(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$; (a) for $\varepsilon = \Gamma \cdot \varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$, with $\mathcal{G}_B$ optimized for $\varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$, and (b) for $\varepsilon = \Gamma \cdot \varepsilon^*(\bar{\Lambda}_o, \bar{\rho}_o)$, with $\mathcal{G}_B$ optimized for $\varepsilon^*(\bar{\Lambda}_o, \bar{\rho}_o)$: $\Gamma = 0.1, 0.2, \dots, 1.0$ - left to right. (See Example 5.3.2.)	129
5.4	BER curves for the MB $\mathcal{H}_{TI}$ algorithm given in Example 5.3.1 and the $\mathcal{G}_B$ s given in Example 5.3.2, for $\ell_{\max} = 10, 20, 50$, and 500. (See Example 5.4.1.)	131
A.1	Density evolution for $\mathcal{C}^\infty(3, 6)$, under min-sum decoding algorithm, over an BIPLGC with noise variance $\sigma^2 = 0.5$. (See Example A.1.1.) .	141
B.1	Illustration of the graphical representation of a recursion.	143
C.1	Convergence region of MB^0 for $\mathcal{C}^\infty(3, 6)$, and $p_{\varepsilon, \Delta_1}^{(k)}$ curves for $k = 1, 2, \dots, 6$ iterations of min-sum decoding algorithm.	147

List of Notations and Abbreviations

Notation	Description	First Use
C	Channel capacity	5
$\mathbf{u}$	Information sequence	7
$\mathbf{v}$	Encoder output sequence	7
$\mathcal{X}$	Input alphabet of a discrete channel	7
$\mathcal{Y}$	Output alphabet of a discrete channel	7
$P(y x)$	probability transition matrix of a discrete channel	7
y	An output symbol	7
x	An input symbol	7
$\hat{\mathbf{v}}$	Corrupted $\mathbf{v}$	7
$\hat{\mathbf{u}}$	Estimated $\mathbf{u}$	7
$\mathfrak{h}(\cdot)$	Binary (base-2) entropy function	9
BEC	Binary erasure channel	9
e	Erasure probability of a BEC	9
BSC	Binary symmetric channel	9
ε	Cross-over probability of a BSC	9
BSC-E	Binary symmetric channel with erasure	11
GC	G aussian channel	11
$\mathbb{Z}$	The set of integer numbers	11
x_t	A continuous-alphabet input symbol in discrete time t	11
$\mathbb{R}$	The set of real numbers	11

Notation	Description	First Use
y_t	A continuous-alphabet output symbol in discrete time t	11
z_t	A Gaussian noise symbol in discrete time t	11
σ	Noise standard deviation	11
AWGN	Additive white Gaussian noise	11
PLGC	Power-limited GC	13
P	Average power	13
BIPLGC	Binary input PLGC	13
$\varphi_\sigma(x)$	An auxiliary function of σ and x	13
BLPLGC	Band-limited PLGC	13
$x(t)$	A continuous-alphabet input symbol in continuous time t	13
$y(t)$	A continuous-alphabet output symbol in continuous time t	13
$z(t)$	A Gaussian noise waveform	13
$\otimes$	Convolution	13
$h(t)$	BLPLGC impulse response	13
B	Bandwidth	16
N_0	One-sided AWGN power spectral density	16
E_b	Energy per information bit	16
r	Code rate	16
η_e	Error probability	18
C_{η_e}	Capacity for non-vanishing error probability η_e	18
n	Length of a linear block code	20
k	Dimension of a linear block code	20
$\mathcal{C}$	A linear block code	21
$\mathbf{G}$	Generator matrix of a linear block code	21
$\mathbf{H}$	Parity-check matrix of a linear block code	21

Notation	Description	First Use
LDPC	Low-density parity-check	21
d_v	Degree of variable-nodes in a regular LDPC code	22
d_c	Degree of check-nodes in a regular LDPC code	22
$\mathcal{C}^n(d_v, d_c)$	Ensemble of (d_v, d_c) -regular LDPC codes of length n	23
iff	If and only if	24
$\oplus$	Addition in modulo-2	24
v	A variable-node	24
c	A check-node	24
ℓ	Iteration number	26
$\Psi_c^{(\ell)}(\cdot)$	Check-node message map in iteration ℓ	26
$\mathcal{O}$	Channel output alphabet	26
$\mathcal{M}$	Message alphabet	26
$\Psi_v^{(\ell)}(\cdot)$	Variable-node message map in iteration ℓ	26
$\mathcal{G}_A$	Gallager's decoding algorithm A	28
$\lceil \cdot \rceil$	Ceiling	29
$\mathcal{G}_B$	Gallager's decoding algorithm B	29
$\mathcal{RUM}$	Richardson, Urbanke, and Mitzenmacher's decoding algorithm	30
PMF	Probability mass function	31
$\otimes$	Check-node operation for sum-product decoding algorithm	31
LR	Likelihood ratio	32
LLR	Log-likelihood ratio	32
LD	Likelihood difference	32
BER	Bit error-rate	34
PDF	Probability density function	36
$q_e^{(\ell)}$	Check-to-variable message error probability in iteration ℓ	36

Notation	Description	First Use
MB	Majority-based	40
ω	Order of an MB algorithm	41
$\lfloor \cdot \rfloor$	Floor	41
MB^ω	MB algorithm of order ω	41
$p_{\epsilon, \epsilon}^{(\ell)}$	p_ϵ in iteration ℓ for channel parameter ϵ	42
x	A variable	42
$f_{\pm}^{d_v, \omega, d_c}(x)$	A function of d_v , ω , d_c , and x	42
ϵ^*	Threshold	42
$f_{\epsilon}^{\omega}(x)$	The curve representing density evolution for MB^ω , and for channel parameter ϵ	43
$\beta_{\epsilon}^{\omega}$	The sequence of $p_{\epsilon, \epsilon}^{(\ell)}$ values	44
y	A function of x	44
$\zeta^{\pm}(x)$	An auxiliary function	46
κ	A constant	51
$\phi^{\omega}(x)$	Starting-point loci for MB^ω	52
δ^*	Smallest positive steady-point	52
$\tilde{\epsilon}$	A function of d_v , ω , and ϵ	59
η	A function of d_v , ω , d_c , and $\tilde{\epsilon}$	59
$\mathfrak{O}(\cdot)$	Order	59
$\mathfrak{X}$	An auxiliary variable	60
$\mathfrak{T}^*$	A specific root of an auxiliary polynomial	60
$\varphi(x)$	An auxiliary function of x	62
$\mathcal{H}_{TI}$	Time-invariant hybrid	69
$\mathcal{H}_{ST}$	Switch-type hybrid	69
Δ	A message-passing decoding algorithm	69
$\tilde{\alpha}^{(\ell)}$	Variable-nodes' hybridization vector in iteration ℓ	70
$\tilde{\beta}^{(\ell)}$	Check-nodes' hybridization vector in iteration ℓ	70
$\{\tilde{\alpha}^{(\ell)}\}_{\ell=0}^{\infty}$	Variable-nodes' hybridization sequence	70

Notation	Description	First Use
$\{\vec{\beta}^{(\ell)}\}_{\ell=0}^{\infty}$	Check-nodes' hybridization sequence	70
$h_{\varepsilon}^{\vec{\alpha}}(x)$	The curve representing an MB $\mathcal{H}_{TI}$ algorithm	73
$\varepsilon_{\{\text{MB } \mathcal{H}\}}^*$	Maximum threshold for MB hybrid algorithms	82
$\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*$	Maximum threshold for MB $\mathcal{H}_{TI}$ algorithms	82
$\mathcal{N}_{\varepsilon}$	Convergence domain associated with ε	82
λ	A real number in $[0, 1]$	83
$\vec{\alpha}_{\{\text{MB } \mathcal{H}_{TI}\}}^*$	Threshold-maximizing hybridization vector	83
ι	Target error-rate	84
$\vec{\alpha}_{\varepsilon, \iota}$	Speed-maximizing hybridization vector	84
Ω	A specific order of MB algorithms	87
$\varepsilon_{\{\text{MB } \mathcal{H}_{ST}\}}^*$	Maximum threshold for MB $\mathcal{H}_{ST}$ algorithms	88
Γ	A coefficient	100
$\vec{\lambda}$	Variable-nodes' degree distribution vector	110
$\vec{\rho}$	Check-nodes' degree distribution vector	110
$(\vec{\lambda}, \vec{\rho})$	Degree distribution vector pair	110
$\mathcal{C}^n(\vec{\lambda}, \vec{\rho})$	Irregular LDPC code ensemble of length n , associated to $(\vec{\lambda}, \vec{\rho})$	110
$\vec{\lambda}_j$	Concentrated $\vec{\lambda}$	111
$\mathcal{C}^n(\vec{\lambda}_j, \vec{\rho})$	Left-concentrated LDPC code ensemble of length n , associated to $(\vec{\lambda}_j, \vec{\rho})$	111
$MB^{j, \omega, \vec{\rho}}$	MB algorithm of order ω , operating on $\mathcal{C}^n(\vec{\lambda}_j, \vec{\rho})$	111
$f_{\pm}^{j, \omega, \vec{\rho}}(x)$	A function of j , ω , $\vec{\rho}$, and x	111
$f_{\varepsilon}^{j, \omega, \vec{\rho}}(x)$	The curve representing density evolution for $MB^{j, \omega, \vec{\rho}}$, and for channel parameter ε	111
$\Lambda_{j, \omega}$	Fraction of degree- j nodes that process messages according to $MB^{j, \omega, \vec{\rho}}$	111
$\vec{\Lambda}$	Ordered sequence of $\Lambda_{j, \omega}$ values	112
$(\vec{\Lambda}, \vec{\rho})$	Degree distribution vector pair	112

Notation	Description	First Use
$\mathcal{C}^\infty(\vec{\Lambda}, \vec{\rho})$	Irregular LDPC code ensemble of length ∞ , associated to $(\vec{\Lambda}, \vec{\rho})$	112
$h_{\varepsilon}^{\vec{\Lambda}, \vec{\rho}}(x)$	The curve representing density evolution for $\mathcal{C}^\infty(\vec{\Lambda}, \vec{\rho})$	112
$\varepsilon^*(\vec{\Lambda}, \vec{\rho})$	Threshold associated to $(\vec{\Lambda}, \vec{\rho})$	113
$\gamma(\vec{\Lambda}, \vec{\rho})$	Maximum stable channel parameter associated to $(\vec{\Lambda}, \vec{\rho})$	113
$\delta(\vec{\Lambda}, \vec{\rho})$	Smallest positive steady-point associated to $(\vec{\Lambda}, \vec{\rho})$	113
θ	Speed determining parameter for MB $\mathcal{H}_{TI}$ algorithms	115
$r(\vec{\Lambda}, \vec{\rho})$	Design rate associated to $(\vec{\Lambda}, \vec{\rho})$	115
$\overline{d}_v$	Average variable-node degree	116
$\overline{d}_c$	Average check-node degree	116
D_v	Maximum variable-node degree	116
D_c	Maximum check-node degree	116
$\mathbf{S}_j$	Switching-sequence for degree- j variable-nodes	120
$h_{\varepsilon}^{\vec{\lambda}, \vec{\rho}}(x)$	The curve representing density evolution for $\mathcal{C}^\infty(\vec{\lambda}, \vec{\rho})$	120
$\check{f}_{\varepsilon}^{j, \vec{\rho}}(x)$	A function of j , $\vec{\rho}$, ε , and x	120
$\wp$	A function of j and ω	120
Θ	Speed determining parameter for MB $\mathcal{H}_{ST}$ algorithms	121
SNR	Signal-to-noise ratio	130
CDF	Cumulative distribution function	138
$\odot$	Check-node operation in density evolution for min-sum decoding algorithm	139
$\circ$	Composition (of two functions)	142
$\mathcal{U}$	A one-to-one correspondence	145
$Q(\cdot)$	The Q -function	145

Chapter 1

Introduction

‘Who should rule?’

Plato’s response, which may be regarded as the archetypal rejoinder of all authoritarian doctrines, is “ *Ο βασιλλιάς-φιλόσοφος.* ”²

Karl Raimund Popper’s unorthodox response to the above question is “ The vital question is not *‘Who should rule?’* but *‘How can we minimize misrule?’.* ”

* * * * *

Low-density parity-check codes, and the associated iterative algorithms that can be used to decode these codes, were introduced by **Gallager** in the early sixties [1–3]. Regrettably, in spite of **Gallager**’s promising results on the strength and the flexibility of low-density parity-check codes (and iterative decoding algorithms), they did not receive the attention they deserved, and were soon forgotten. Some thirty years later, following the successful introduction of turbo-codes [4], interest in low-density parity-check codes was rekindled, as researchers started to realize the advantages of these codes, and the attractive performance / complexity tradeoff that

¹ “ *Few, but ripe.* ”

² “ *the Philosopher-King.* ”

the corresponding iterative decoding algorithms are capable of offering. Since then, there has been a great amount of research devoted to low-density parity-check codes, and to their encoding and decoding. (See, e.g., [5–10] and the references therein.)

Low-density parity-check codes are linear block codes, and are specified by their sparse (low-density) parity-check matrices. Suboptimal, yet efficient, iterative decoding algorithms, whose complexity grows only linearly with code's block length, can naturally be described using simple graphs [11]. The decoding process is performed by iteratively exchanging messages between the nodes, and along the edges, of these graphs [12].

There is a wide range of iterative algorithms for decoding low-density parity-check codes, each offering a particular tradeoff between error performance and decoding complexity. In general, these algorithms can be divided into two classes, based on the size of their message alphabet set: algorithms for discrete-alphabet channels, and algorithms for continuous-alphabet channels.

Algorithms in the first class are of interest because of their extremely low complexity (per iteration), which makes them attractive for high-rate data communications, and for the applications where the decoder's processing power is limited. (Examples include **Gallager's** decoding algorithm A [3, p. 48] and majority decoding algorithm [13].) Algorithms in the second class, on the other hand, are capable of processing real-valued messages. Not surprisingly, they offer better performance, but of course, they are computationally more demanding. (Examples include sum-product decoding algorithm and min-sum decoding algorithm.)

This doctoral thesis is concerned with devising efficient and fast decoding algorithms for low-density parity-check codes, where the system constraints require the use of low-complexity hard-decision decoders. It describes some improved iterative decoding strategies, and addresses the effective methods that can be used to design

(regular and irregular) codes capable of getting the best out of these new (decoding) strategies. Some of the ideas discussed in the thesis, however, are general, and can be applied to soft-decision decoding algorithms as well.

1.1 Organization of the Thesis

This thesis reports the results obtained during the course of the author's studies at the Department of Systems and Computer Engineering, Carleton University, Canada. During the years, these results have been presented at a number of conferences and workshops. Parts of the material have also been published, or submitted for possible publication, in journals.

Chapter 2 serves as a summary of the basic concepts involved in the rest of the thesis. We review some of the fundamental ideas behind digital communications and channel coding theory. We also review low-density parity-check codes, and survey some well-known iterative decoding algorithms. Furthermore, we discuss some important design and analysis tools, such as density evolution, for these codes and algorithms.

In **Chapter 3**, we perform a comprehensive study of majority-based algorithms, which can be used to decode regular low-density parity-check codes, and are of interest because of their low complexity (per iteration). We formally introduce these algorithms, and present their analysis based on density evolution. We also investigate the convergence properties (including the convergence speed) of these algorithms, in detail. These results have been published in [14, 15].

Chapter 4 presents our results on hybrid algorithms for decoding regular low-density parity-check codes. We show that hybrid algorithms are capable of substantially improving error performance and convergence speed at the expense of a small

increase in the complexity of the decoder's structure. We introduce time-invariant hybrid algorithms, and investigate their dynamics. We compare time-invariant hybrid algorithms with the better-known switch-type hybrid algorithms, and demonstrate that switch-type algorithms are not nearly as robust as their time-invariant counterparts, with respect to channel variations. These results have been published in [16–21].

Chapter 5 presents our most recent results on hybrid decoding, which extends the application of time-invariant hybrid algorithms to irregular codes. These results have been presented in [22], and have been submitted for possible publication in *IEEE Trans. Commun.*

In **Chapter 6**, we provide a summary of our contributions, and suggest some directions for future work.

“ *La semplicità è la più alta forma di sofisticazione.* ”¹

– Leonardo di ser Piero da Vinci

Chapter 2

Preliminaries and Background

Very few, if any, works have profoundly altered our understanding of communication as much as **Shannon**’s two-part article, “A mathematical theory of communication” [23], has. In this article, amidst other things,² **Shannon** showed that, contrary to the then popular belief amongst communication theorists, it was possible to communicate reliably without sacrificing the transmission rate of information. More specifically, he defined the *capacity* C of a given channel as the maximum *mutual information* between the input and the output of that channel, taken over all possible channel input distributions, and demonstrated that by using *channel codes*, we can achieve all rates below this capacity.

This result, widely regarded as one of the basic theorems of information theory, gave birth to the subject of channel coding theory, which is devoted to developing practical capacity-approaching channel coding schemes. In fact, soon after **Shannon** presented his innovative ideas, researchers started a lasting quest for better and better channel codes, and for more and more efficient encoding / decoding methods. First, **Hamming** codes were introduced in 1947 [24].³ Then **Golay** codes were discovered

¹“ *Simplicity is the ultimate form of sophistication.* ”

²In [23], **Shannon** also founded the field of *data compression*: he stated the *source coding* theorem, gave simple examples of source codes, and even introduced the main ideas of *rate-distortion theory*.

³**Shannon** was aware of **Hamming**’s work when he wrote his article: an example of noncausal nature of scientific publications!

in 1949 [25]. Later, **Reed-Muller** codes were introduced in 1953 [26–28]. Shortly thereafter, in 1955, **Elias** discovered *convolutional* codes [29].⁴ Then *BCH* codes were introduced by **Hocquenghem** in 1959 [32], and independently by **Bose** and **Ray-Chaudhuri** in 1960 [33].⁵ Almost at the same time, **Reed-Solomon** codes were introduced [35]. Later, **Forney** introduced *concatenation* as a constructing method for long codes [36,37]. Afterwards, with a few exceptions (see, e.g., [38–40]), the efforts concentrated largely on generalizing and refining the above-mentioned encoding and decoding methods.

In the last decade or so of the last century, we saw communism collapse, and the bipolar world gave place to a unipolar (some say multipolar) world; we witnessed the gargantuan-scale growth of the internet which changed many aspects of our lives, for better or worse; we saw wireless communications grow beyond our most optimistic expectations; and the list goes on.

We also were witnesses to an important paradigm shift in channel coding theory, after the successful introduction of *turbo-codes* in 1993 [4], which revolutionized the field, and led to virtually error-free communications at rates close to capacity. The main concern of this work is to devise more efficient decoding algorithms for a family of turbo-like codes, called *low-density parity-check*.

This chapter, which serves as a summary of the basic concepts involved in the rest of the thesis, is organized as follows. We first briefly review some of the fundamental ideas behind digital communication theory and channel coding theory. We then review low-density parity-check codes, and investigate their structure. Afterwards, we survey some important *message-passing* algorithms for decoding low-density parity-check codes over both discrete- and continuous-alphabet channels. Finally, we briefly discuss some important analysis tools, such as *density evolution*, which can be used

⁴The first decoding algorithm for convolutional codes was devised in 1957 [30,31].

⁵The first decoding algorithm for BCH codes was devised in 1960 [34].

in design and performance evaluation of low-density parity-check codes.

2.1 Digital Communication Systems

Figure 2.1 represents the block diagram of the *digital* communication system with which we are dealing in this work. The *information source* generates an *information sequence* $\mathbf{u}$ of independent and equally likely binary symbols, each containing one bit of information. The *channel-encoder* then encodes $\mathbf{u}$ into a sequence $\mathbf{v}$ by adding redundancy in a controlled way. The sequence $\mathbf{v}$, in turn, is transmitted over the *discrete memoryless channel*.

Definition 2.1.1 (p. 184 in [41]). A *discrete channel* is defined to be a system consisting of an input alphabet $\mathcal{X}$, an output alphabet $\mathcal{Y}$, and a *probability transition matrix* $P(\mathbf{y}|\mathbf{x})$ that expresses the probability of observing the output symbol $\mathbf{y}$ given that the input symbol $\mathbf{x}$ is sent. The channel is *memoryless* if the probability distribution of the output is conditionally independent of previous channel inputs and channel outputs.

The channel noise corrupts $\mathbf{v}$, and $\tilde{\mathbf{v}}$, the corrupted version, is fed into the *channel-decoder*, which tries to recover $\mathbf{v}$, and therefrom $\mathbf{u}$. The output of the channel-decoder, $\hat{\mathbf{u}}$, is finally provided to the *information sink*.

Definition 2.1.2 (p. 190 in [41]). A discrete channel is said to be *symmetric* if the rows and columns of $P(\mathbf{y}|\mathbf{x})$ are permutations of each other. It is said to be *weakly symmetric* if the rows of $P(\mathbf{y}|\mathbf{x})$ are permutations of each other, and column sums are equal.

Figure 2.1: Block diagram of a digital communication system.

Remark 2.1.1. For a weakly symmetric channel,

$$C = \log_2 |\mathcal{Y}| - \mathfrak{h}(\text{row}) \text{ bits per channel use,} \quad (2.1.1)$$

where $\mathfrak{h}(\cdot)$ denotes the binary entropy function.

2.2 Channel Models

In this section, we introduce some important channel models, and give their capacities.

2.2.1 Binary Erasure Channel (BEC)

This simple asymmetric binary-input / ternary-output channel is specified by the probability transition matrix

$$\begin{bmatrix} 1 - \mathbf{e} & \mathbf{e} & 0 \\ 0 & \mathbf{e} & 1 - \mathbf{e} \end{bmatrix}. \quad (2.2.1)$$

$\mathbf{e}$ is called the *erasure* probability of the channel.

Remark 2.2.1.

$$C_{\text{BEC}} = 1 - \mathbf{e} \text{ bits per channel use,} \quad (2.2.2)$$

and is achieved when the input distribution is uniform [41, pp. 187–188].

Figure 2.2 depicts C_{BEC} versus $\mathbf{e}$.

2.2.2 Binary Symmetric Channel (BSC)

This symmetric binary-input / binary-output channel, which will be one of the focal points of this thesis, is specified by the probability transition matrix

$$\begin{bmatrix} 1 - \varepsilon & \varepsilon \\ \varepsilon & 1 - \varepsilon \end{bmatrix}. \quad (2.2.3)$$

Figure 2.2: Capacity of the BEC as a function of erasure probability.

ε is called the *cross-over* probability of the channel.

Remark 2.2.2.

$$C_{\text{BSC}} = 1 - \mathfrak{h}(1 - \varepsilon, \varepsilon) \text{ bits per channel use,} \quad (2.2.4)$$

and is achieved when the input distribution is uniform [41, pp. 186–187].

Figure 2.3 depicts C_{BSC} versus ε .

2.2.3 BSC with Erasure (BSC-E)

This asymmetric binary-input / ternary-output channel is specified by the probability transition matrix

$$\begin{bmatrix} 1 - \mathbf{e} - \varepsilon & \mathbf{e} & \varepsilon \\ \varepsilon & \mathbf{e} & 1 - \mathbf{e} - \varepsilon \end{bmatrix}. \quad (2.2.5)$$

Remark 2.2.3. It can be shown that

$$C_{\text{BSC-E}} = 1 - \mathbf{e} + \mathfrak{h}(1 - \mathbf{e}, \mathbf{e}) - \mathfrak{h}(1 - \mathbf{e} - \varepsilon, \mathbf{e}, \varepsilon) \text{ bits per channel use,} \quad (2.2.6)$$

and is achieved when the input distribution is uniform.

2.2.4 Gaussian Channel (GC)

The input and the output of this continuous-alphabet channel at time t , $t \in \mathbb{Z}$, denoted by $\mathbf{x}_t$, $\mathbf{x}_t \in \mathbb{R}$, and $\mathbf{y}_t$, $\mathbf{y}_t \in \mathbb{R}$, respectively, are related via $\mathbf{y}_t = \mathbf{x}_t + \mathbf{z}_t$, where $\mathbf{z}_t$ is a normally distributed random-variable with mean 0 and standard deviation σ . The additive Gaussian noise is assumed to be white (AWGN) and independent of the input. Without further conditions, the capacity of GC is infinity.

Figure 2.3: Capacity of the BSC as a function of cross-over probability.

2.2.5 Power-Limited GC (PLGC)

PLGC models many practical channels, including radio and satellite links.

Remark 2.2.4. Assuming an average power constraint P ,

$$C_{\text{PLGC}} = \frac{1}{2} \cdot \log_2 \left(1 + \frac{P}{\sigma^2} \right) \text{ bits per channel use,} \quad (2.2.7)$$

and is achieved when the input is a normally distributed random-variable with mean 0 and standard deviation $\sqrt{P}$ [41, pp. 241–242].

Figure 2.4 depicts C_{PLGC} versus $\frac{P}{\sigma^2}$.

2.2.6 Binary Input PLGC (BIPLGC)

This is the PLGC assuming that $x_t \in \{-1, +1\}$.

Remark 2.2.5.

$$C_{\text{BIPLGC}} = - \int_{-\infty}^{+\infty} \varphi_\sigma(x) \cdot \log_2 \varphi_\sigma(x) \cdot dx - \frac{1}{2} \cdot \log_2(2 \cdot \pi \cdot e \cdot \sigma^2) \text{ bits per channel use,} \quad (2.2.8)$$

where

$$\varphi_\sigma(x) = \frac{1}{\sqrt{8 \cdot \pi \cdot \sigma^2}} \left(e^{-\frac{(x-1)^2}{2 \cdot \sigma^2}} + e^{-\frac{(x+1)^2}{2 \cdot \sigma^2}} \right), \quad (2.2.9)$$

and is achieved when the input distribution is uniform [12].

Figure 2.5 depicts C_{BIPLGC} versus $\frac{1}{\sigma^2}$. Note that $C_{\text{BIPLGC}} \rightarrow 1$ as $\frac{1}{\sigma^2} \rightarrow \infty$.

2.2.7 Band-Limited PLGC (BLPLGC)

The input and the output of this continuous-alphabet channel at time t , $t \in \mathbb{R}$, denoted by $x(t)$, $x(t) \in \mathbb{R}$, and $y(t)$, $y(t) \in \mathbb{R}$, respectively, are related via $y(t) = (x(t) + z(t)) \otimes h(t)$, where $z(t)$ is the waveform of the white **Gaussian** noise, $h(t)$

Figure 2.4: Capacity of the PLGC as a function of $\frac{P}{\sigma^2}$.

Figure 2.5: Capacity of the BIPLGC as a function of $\frac{1}{\sigma^2}$.

is the impulse response of an ideal (bandpass) filter of bandwidth B , and $\otimes$ denotes *convolution*. BLPLGC models many practical channels, including radio networks and telephone lines.

Remark 2.2.6.

$$C_{\text{BLPLGC}} = B \cdot \log_2 \left(1 + \frac{P}{B \cdot N_0} \right) \text{ bits per second.} \quad (2.2.10)$$

where P is the average signal power, and $\frac{N_0}{2}$ is the noise two-sided power spectral density [41, pp. 247–250].

By substituting $C \cdot E_b$ (E_b denotes energy per information bit) for P in (2.2.10), we obtain

$$E_b/N_0 = \frac{2^{C/B} - 1}{C/B}. \quad (2.2.11)$$

This equation can be used to obtain the capacity boundary in the bandwidth-efficiency diagram shown in Figure 2.6. Note the $C/B \rightarrow 0$ as $E_b/N_0 \rightarrow \ln 2$ ($= -1.5917$ dB).

2.3 Capacity for Non-Vanishing Error Probabilities

The equations provided in the previous subsection can be used to find the maximum transmission rate for which we can have reliable communications, i.e., communications with vanishing error probability, over a fixed channel. Equivalently, they can be used to find the worst channel over which we can have reliable communications for a fixed transmission rate. For instance, in order to reliably communicate over a BEC with erasure probability $\mathbf{e} = \frac{1}{2}$, the rate should not exceed $C_{\text{BEC}}(\mathbf{e})|_{\mathbf{e}=\frac{1}{2}} = \frac{1}{2}$. On the other hand, for reliable communications over a BEC at rate $r = \frac{1}{2}$, the channel parameter should not exceed $C_{\text{BEC}}^{-1}(r)|_{r=\frac{1}{2}} = \frac{1}{2}$.

Figure 2.6: Bandwidth-efficiency diagram.

But what if we allow a non-vanishing error probability p_e ? What is the maximum transmission rate for which we can have communications with non-vanishing error probability p_e over a fixed channel? Or equivalently, what is the worst channel over which we can have communications with non-vanishing error probability p_e for a fixed transmission rate?

To answer the above questions, we use the *source channel separation* theorem [41, p. 215] which states that we can design the source code and the channel code separately, and combine the results to achieve optimal performance. We assume that a source code is used to compress the information with such a rate that generates a **Hamming** distortion of at most p_e . In basic rate-distortion theory, it is shown that the rate-distortion function $R(p_e)$ for the source, assuming a **Hamming** distortion measure, is given by $1 - \mathfrak{h}(p_e)$ [41, p. 342]. For reliable communications over the channel, we must have $r \cdot (1 - \mathfrak{h}(p_e)) \leq C$. As such

$$C_{p_e} = \frac{C}{1 - \mathfrak{h}(p_e)} \text{ bits per channel use.} \quad (2.3.1)$$

Using 2.3.1, we can easily see that for communications at rate r with non-vanishing error probability p_e over a channel with parameter π which has an increasing (decreasing) capacity $C(\pi)$ in π , π should (should not) exceed $C^{-1}(r \cdot (1 - \mathfrak{h}(p_e)))$. Figures 2.7 and 2.8 demonstrate these boundaries for BSC and BIPLGC, at rates $r = \frac{1}{3}$, $\frac{1}{2}$, and $\frac{2}{3}$.

2.4 Channel Codes

The idea in channel coding theory is to add redundancy to the information bits in such a way that they can be recovered later, even if the bits are corrupted by noise. The simplest way to protect information is to transmit each symbol several times over the

Figure 2.7: **Shannon** limit curves for rates $r = \frac{1}{3}$, $\frac{1}{2}$, and $\frac{2}{3}$, and for BSC.

Figure 2.8: **Shannon** limit curves for rates $r = \frac{1}{3}$, $\frac{1}{2}$, and $\frac{2}{3}$, and for BIPLGC.

channel. By repeating each symbol n times, a *repetition code* of length n is obtained. The decoding is performed by a simple majority vote. This coding scheme can achieve any arbitrary level of reliability, but is not efficient in terms of information rate, as for transmitting an information symbol, we send $n - 1$ extra symbols. Therefore, increasing n for improving the reliability, results in decreasing the rate. Fortunately, as promised by **Shannon**, it is possible to communicate reliably without sacrificing the transmission rate, or more formally,

Theorem 2.4.1 (Theorem 8.7.1 in [41]). *All rates below capacity C are achievable. Specifically, for every rate $r < C$, there exist a sequence of rate- r and length- n binary block codes with maximum probability of decoding error going to zero with increasing block length. Conversely, any sequence of rate- r and length- n binary block codes with maximum probability of error going to zero, must have $r < C$.*

Unfortunately, **Shannon** stopped short of telling us how to find these codes.⁶ The rest of this work, in fact, is about a family of binary block codes which fulfills **Shannon**'s old promise. We continue this chapter by (very) briefly introducing linear block codes.

2.4.1 Linear Block Codes

Definition 2.4.1. An (n, k) linear block code over a field $\mathbb{F}$ is a k -dimensional subspace of vector space $\mathbb{F}^n$. (Note that $r = \frac{k}{n}$.) A binary linear block code is a linear block code over $\mathbb{F}_2$.

A linear block code can be specified by any basis for this subspace, as well as any basis for the null-space (orthogonal complement) of this subspace.

⁶Perhaps this was not that bad after all! What else would generations of coding theorists do, had he found the codes as well?

Definition 2.4.2. A *generator matrix* for an (n, k) linear block code $\mathcal{C}$ is a $k \times n$ matrix $\mathbf{G}$ whose rows form a basis for $\mathcal{C}$.

Example 2.4.1. The generator matrix for the repetition code of length 3 is given by

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}.$$

◀

Definition 2.4.3. A *parity-check matrix* for an (n, k) linear block code $\mathcal{C}$ is a $(n - k) \times n$ matrix $\mathbf{H}$ whose rows form a basis for the null-space of $\mathcal{C}$.

Example 2.4.2. The parity-check matrix for the repetition code of length 3 is given by

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

◀

Remark 2.4.1. The null-space of the repetition subspace is the parity-check subspace, and vice versa.

2.5 Low-Density Parity-Check (LDPC) Codes

GALLAGER introduced LDPC codes in the early sixties [1–3]. Despite his extremely promising results, the codes were soon forgotten due to limited computational resources available at the time, which made the corresponding *iterative* decoding algorithms he had outlined, look impractical.⁷ Aside from a few notable exceptions,

⁷There have also been suggestions that LDPC codes were generally neglected because it was assumed that *concatenated* codes [36, 37] were superior for practical purposes.

such as [11, 42, 43], the field lay dormant for a long time. In fact, it was not until the introduction of turbo-codes that researchers realized the strength of iterative coding schemes and the attractive performance/complexity tradeoff that they offer. Since then, there has been a great amount of research devoted to LDPC codes (and turbo-codes), and the performance and the complexity of associated iterative decoding algorithms. (See, e.g., [5–10] and the references therein.)

It is worth mentioning that a number of ideas used by **Shannon** in outlining the proof of his channel-coding theorem, are incorporated in iterative coding schemes. In the proof (of the theorem), to symmetrize the average probability of error, it is calculated over a *random* choice of code-books, each describing a rate- r code of length n (which goes to infinity). In parallel, a random component is present in both LDPC coding schemes (the random structure of the code's parity-check matrix) and turbo-coding schemes (the random interleaver present in the encoder), and also, the code length goes to infinity.

2.5.1 Structure of LDPC Codes

LDPC codes are linear block codes, and therefore, can be fully described by their parity-check matrices. The matrices, however, as the term ‘low-density’ suggests, are sparse. (This sparseness allows the complexity of the corresponding iterative decoding algorithms to grow only linearly with the codes’ block length.) In particular, a binary (d_v, d_c) -regular LDPC code $\mathcal{C}$ has a parity-check matrix $\mathbf{H}$ with d_v and d_c ones in each column and row, respectively. In irregular LDPC codes, column and row weights are not constant, and are chosen according to two distributions [44–46]. The performance of irregular LDPC codes can be greatly enhanced, compared to regular ones, by carefully selecting these distributions.⁸

⁸For instance, under *sum-product* decoding algorithm, the best *irregular* rate-1/2 LDPC code performs within 0.0045 dB of the **Shannon** limit over BIPLGC [47].

Noting that the number of ones in the parity-check matrix of a (binary) (d_v, d_c) -regular LDPC code of length n and rate r equals $d_v \cdot n$ on the one hand, and $d_c \cdot m$, with $m \geq (1 - r) \cdot n$, on the other hand, we obtain

$$r \geq 1 - \frac{d_v}{d_c}. \quad (2.5.1)$$

$\mathcal{C}^n(d_v, d_c)$, the ensemble of (d_v, d_c) -regular LDPC codes of length n , corresponds to the set of all parity-check matrices of length n with column weight d_v and row weight d_c . (We also assume that parity-check matrices from this set are selected equally likely.) Following the common practice in the literature on LDPC codes [12, 44–46], we also consider ensembles of codes, rather than single instances of codes, throughout this work, to maintain the symmetry which is required for deriving many properties of these codes.

2.5.2 Graphical Representations of LDPC Codes

Graphical representation of LDPC codes was first introduced by **Gallager** [1, p. 46]. The idea, which was later generalized by **Tanner** [11], **Wiberg et al.** [48, 49], and **Kschischang et al.** [50], has proved extremely useful in understanding the dynamics of iterative decoding algorithms. In the following, we explain a graphical representation of codes which is widely referred to as ***Tanner graph***.

Tanner Graphs

A **Tanner graph** is a simple undirected graph with two non-empty sets of nodes, namely *variable-nodes* and *check-nodes*. The sets are disjoint, i.e., a variable-node is connected only to check-nodes and a check-node is connected only to variable-nodes. (Mathematically speaking, a **Tanner graph** is a complete bipartite graph.) The parity-check matrix of a linear block code, and hence the code itself, can be

represented by a **Tanner** graph if one assigns a variable-node to each code symbol, and a check-node to each parity-check equation. In the graph, an edge will connect the j^{th} variable-node to the i^{th} check-node if, and only if, (iff) the j^{th} code symbol participates in the i^{th} parity-check equation.

Example 2.5.1. Consider the linear block code $\mathcal{C}$ specified by the parity-check matrix

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

Clearly, for this code we have

$$x_1 \oplus x_4 \oplus x_5 \oplus x_7 = 0,$$

$$x_2 \oplus x_4 \oplus x_6 \oplus x_7 = 0,$$

$$x_3 \oplus x_5 \oplus x_6 \oplus x_7 = 0,$$

where $\oplus$ denotes modulo-2 addition. Illustrated in Figure 2.9 is the **Tanner** graph corresponding to this code. The code symbols are represented by the variable-nodes $v_1, v_2, v_3, v_4, v_5, v_6$, and v_7 on the left, and the parity-check equations are represented by the check-nodes c_1, c_2 , and c_3 on the right. ◀

Note that we can construct a parity-check matrix, and therefore a code, based on a bipartite graph, if it does not contain parallel edges. Consider now a bipartite graph with n variable-nodes of degree d_v and $(1 - r) \cdot n$ check-nodes of degree d_c . This graph completely specifies a (d_v, d_c) -regular LDPC code. The set of all such graphs, when equipped with a uniform probability distribution, is called the ensemble of (d_v, d_c) -regular graphs. Note that the ensembles of (d_v, d_c) -regular LDPC codes and (d_v, d_c) -regular (**Tanner**) graphs are essentially equivalent, and can be used interchangeably.

Figure 2.9: The **Tanner** graph corresponding to the code given in Example 2.5.1.

2.6 Message-Passing Decoders

In this section, we briefly review some important message-passing decoding algorithms [12]. These suboptimal, yet efficient, algorithms can naturally be described using **Tanner** graphs. The decoding process is performed by exchanging messages between variable-nodes and check-nodes along the edges of the code's **Tanner** graph, iteratively. Letting ℓ denote the iteration number, at the beginning (in iteration $\ell = 0$), variable-nodes and check-nodes are assigned some initial messages based on the channel outputs, and in iteration $\ell \geq 1$, every check-node (respectively, variable-node) processes the messages it has received from its neighboring variable-nodes (respectively, check-nodes), along with its initial message, if any, to produce new messages which it then sends back to its neighboring variable-nodes (respectively, check-nodes).

An important property of good message-passing decoding algorithms is that the outgoing message along an edge is independent of the incoming messages previously received along that edge. Using turbo-code terminology, in good message-passing decoding algorithms, only *extrinsic* messages are passed along edges of the **Tanner** graph.⁹

Formally, a message-passing decoder, operating over $\mathcal{C}^n(d_v, d_c)$, based on the principle of ‘allowing only extrinsic information to propagate,’ is specified by the four message maps $\Psi_c^{(0)} : \mathcal{O} \rightarrow \mathcal{M}$, $\Psi_v^{(0)} : \mathcal{O} \rightarrow \mathcal{M}$, $\Psi_c^{(\ell)} : \mathcal{M}^{d_v-1} \rightarrow \mathcal{M}$, and $\Psi_v^{(\ell)} : \mathcal{O} \times \mathcal{M}^{d_c-1} \rightarrow \mathcal{M}$, where $\mathcal{O}$ and $\mathcal{M}$ represent the channel output alphabet and the message alphabet, respectively. Figure 2.10 illustrates the operations in variable-nodes and check-nodes under a typical message-passing algorithm. (Variable-nodes and check-nodes are of degrees j and i , respectively.)

There is a wide range of iterative message-passing algorithms for decoding LDPC

⁹This is only true under the assumption of having a tree-like decoding neighborhood [12], and also having independent initial messages.

Figure 2.10: Illustration of check-nodes and variable-nodes' operations throughout the iteration process: in iteration 0, check-nodes are inactive (a), and variable-nodes pass their channel messages (b); in iteration $\ell > 0$, check-nodes' outgoing messages are determined by their incoming messages in iteration $\ell - 1$ (c), and variable-nodes determine their outgoing messages based on their channel messages, as well as their incoming messages in iteration ℓ (d).

codes, each offering a particular tradeoff between error performance and decoding complexity. In general, these algorithms can be divided into two classes based on the size of their message alphabet set.

2.6.1 Algorithms for Discrete-Alphabet Channels

In the following, we briefly describe several simple algorithms (suitable) for discrete-alphabet channels. These algorithms are of importance because of their extremely low complexity, which makes them attractive for high-rate data communications. (Other interesting examples can be found in [12].)

Gallager's Decoding Algorithm A ($\mathcal{G}_A$)

$\mathcal{G}_A$, which was first introduced by Gallager in [3, p. 48], can be used over BSC. Assuming that $\mathcal{O} = \mathcal{M} = \{-1, +1\}$, this simple algorithm, whose message maps do not depend on the iteration number, is specified as follows. In iteration zero, check-nodes are inactive,

$$\Psi_c^{(0)} = \text{Null}, \quad (2.6.1)$$

and variable-nodes pass their channel messages,

$$\Psi_v^{(0)}(\mathbf{m}_0) = \mathbf{m}_0. \quad (2.6.2)$$

In iteration $\ell \geq 1$, check-nodes pass the product of their extrinsic incoming messages in iteration $\ell - 1$,

$$\Psi_c^{(\ell)}(\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_{d_c-1}) = \prod_{i=1}^{d_c-1} \mathbf{m}_i, \quad (2.6.3)$$

and variable-nodes pass their channel messages unless all of their extrinsic incoming messages in iteration ℓ disagree with the corresponding channel messages, in which

case the negative of these channel messages are passed,

$$\Psi_v^{(\ell)}(\mathbf{m}_0, \mathbf{m}_1, \dots, \mathbf{m}_{d_v-1}) = \begin{cases} -\mathbf{m}_0, & \text{if } |\{j : \mathbf{m}_j = -\mathbf{m}_0\}| = d_v - 1, \\ \mathbf{m}_0, & \text{otherwise.} \end{cases} \quad (2.6.4)$$

Note that $\mathcal{G}_A$ is *universal* in that it does not require knowledge of the channel parameter.

Majority Decoding Algorithm

The message maps for this simple algorithm, introduced in [13], are the same as those for $\mathcal{G}_A$, except that

$$\Psi_v^{(\ell \geq 1)}(\mathbf{m}_0, \mathbf{m}_1, \dots, \mathbf{m}_{d_v-1}) = \begin{cases} -\mathbf{m}_0, & \text{if } |\{j : \mathbf{m}_j = -\mathbf{m}_0\}| \geq \lceil d_v/2 \rceil, \\ \mathbf{m}_0, & \text{otherwise.} \end{cases} \quad (2.6.5)$$

In words, variable-nodes pass the decisions agreed on by the majority of their extrinsic incoming messages, and use their corresponding channel messages to break ties. (Note that the message maps do not depend on the iteration number.)

Similar to $\mathcal{G}_A$, majority decoding algorithm is universal.

Gallager's Decoding Algorithm B ($\mathcal{G}_B$)

Compared to the previous algorithms, $\mathcal{G}_B$, also due to Gallager [3, p. 50], is more complex, but more efficient. Its message maps are the same as those for majority decoding algorithm, except that

$$\Psi_v^{(\ell \geq 1)}(\mathbf{m}_0, \mathbf{m}_1, \dots, \mathbf{m}_{d_v-1}) = \begin{cases} -\mathbf{m}_0, & \text{if } |\{j : \mathbf{m}_j = -\mathbf{m}_0\}| \geq b_\ell, \\ \mathbf{m}_0, & \text{otherwise.} \end{cases} \quad (2.6.6)$$

In words, variable-nodes pass their channel messages unless at least b_ℓ (out of $d_v - 1$) extrinsic incoming messages disagree with the corresponding channel messages, in which case the negative of these channel messages are passed. (Note that this message map depends explicitly on the iteration number.)

The optimal choice of b_ℓ , as we will see later, is a function of the channel parameter. Therefore, $\mathcal{G}_B$ is *not* universal in general.¹⁰

Amongst binary message-mapping algorithms, $\mathcal{G}_B$ is shown to have the minimum message error-rate in each iteration, which is equivalent to having the maximum threshold [3, 51, 52].

Richardson, Urbanke, and Mitzenmacher's Decoding Algorithm ($\mathcal{RUM}$)

This algorithm, which was independently introduced in [12] and [53], is more complex, but better performing, compared to the previous algorithms, and can be used over BSC and BSC-E. Assuming that $\mathcal{M} = \{-1, 0, +1\}$, the corresponding message maps are the same as those for $\mathcal{G}_B$, except that

$$\Psi_v^{(\ell \geq 1)}(\mathbf{m}_0, \mathbf{m}_1, \dots, \mathbf{m}_{d_v-1}) = \text{sgn} \left(\nu_\ell \cdot \mathbf{m}_0 + \sum_{j=1}^{d_v-1} \mathbf{m}_j \right), \quad (2.6.7)$$

where $\text{sgn}(\cdot)$ is the signum function. In words, variable-nodes take the (real) sums of their extrinsic incoming messages, plus their corresponding channel messages, appropriately weighted, and pass the signs of these sums. (Note that this message map depends explicitly on the iteration number.)

Similar to $\mathcal{G}_B$, $\mathcal{RUM}$ is *not* universal in general.

2.6.2 Algorithms for Continuous-Alphabet Channels

Algorithms for continuous-alphabet channels are capable of processing real-valued messages. It is no surprise, therefore, that the performance of these algorithms is better than that of the algorithms (suitable) for discrete-alphabet channels. But of course they are computationally more demanding. In the following, we describe two

¹⁰Hereafter, unless otherwise specified, $\mathcal{G}_B$ refers to the cases where b_ℓ is chosen optimally.

important such algorithms, namely *sum-product* decoding algorithm and *min-sum* decoding algorithm.

Sum-Product Decoding Algorithm

Sum-product algorithm, in general, attempts to compute, either exactly or approximately, various marginal functions associated with a global function which factors into a product of local functions [50]. The algorithm operates by computing various *sums* and *products*, hence the name ‘sum-product.’

Numerous instances of sum-product algorithm have long been known in many different fields. (Refer to [54] for a brief history of successive reinventions of the algorithm’s variants in research communities.) Examples include **Gallager’s** *a posteriori* probability (APP) decoding algorithm [3, pp. 41–47], an algorithm developed by **Chang** and **Hancock** for maximum APP (MAP) symbol detection on inter-symbol interference (ISI) channels [55], the well-known *forward-backward* algorithm by **Welsh** and **Baum** for tracking hidden **Markov** models [56], the well-known BCJR algorithm in coding literature for minimizing bit error probability [40], and **Pearl’s** *belief propagation* algorithm for **Bayesian** networks [57].

In the following, we focus on the application of sum-product algorithm in iterative decoding schemes. The probability mass function (PMF) for a binary random variable x may be represented by the vector (p_{+1}, p_{-1}) , where $p_{+1} \equiv P(x = +1)$ and $p_{-1} \equiv P(x = -1)$. Sum-product decoding algorithm is now specified as follows. When *extrinsic* messages $(p_{+1,1}, p_{-1,1}), (p_{+1,2}, p_{-1,2}), \dots, (p_{+1,d_c-1}, p_{-1,d_c-1})$ arrive at a check-node of degree d_c , the outgoing message will be

$$(p_{+1,d_c}, p_{-1,d_c}) = \bigotimes_{i=1}^{d_c-1} (p_{+1,i}, p_{-1,i}), \quad (2.6.8)$$

where

$$\bigotimes_{i=1}^d (p_{+1,i}, p_{-1,i}) \equiv (p_{+1,1}, p_{-1,1}) \otimes (p_{+1,2}, p_{-1,2}) \otimes \cdots \otimes (p_{+1,d}, p_{-1,d}), \quad (2.6.9)$$

and

$$(p_{+1,1}, p_{-1,1}) \otimes (p_{+1,2}, p_{-1,2}) \equiv (p_{+1,1} \cdot p_{+1,2} + p_{-1,1} \cdot p_{-1,2}, p_{+1,1} \cdot p_{-1,2} + p_{-1,1} \cdot p_{+1,2}). \quad (2.6.10)$$

Similarly, when extrinsic messages $(p_{+1,1}, p_{-1,1}), (p_{+1,2}, p_{-1,2}), \dots, (p_{+1,d_v-1}, p_{-1,d_v-1})$ arrive at a variable-node of degree d_v whose channel message is $(p_{+1,0}, p_{-1,0})$, the outgoing message will be

$$(p_{+1,d_v}, p_{-1,d_v}) = \left(\frac{\prod_{j=0}^{d_v-1} p_{+1,j}}{\prod_{j=0}^{d_v-1} p_{+1,j} + \prod_{j=0}^{d_v-1} p_{-1,j}}, \frac{\prod_{j=0}^{d_v-1} p_{-1,j}}{\prod_{j=0}^{d_v-1} p_{+1,j} + \prod_{j=0}^{d_v-1} p_{-1,j}} \right). \quad (2.6.11)$$

Since $p_{+1} + p_{-1} = 1$, the message vector (p_{+1}, p_{-1}) can be parameterized by a single value. Examples include likelihood ratio (LR), log-likelihood ratio (LLR), and likelihood difference (LD), defined as p_{+1}/p_{-1} , $\ln(p_{+1}/p_{-1})$, and $p_{+1} - p_{-1}$, respectively. In the LLR domain, it is not difficult to see that the sum-product decoding algorithm is specified as follows [12, 50]. In iteration zero, check-nodes are inactive,

$$\Psi_c^{(0)} = \text{Null}, \quad (2.6.12)$$

and variable-nodes pass their channel messages (which are conditioned LLRs),

$$\Psi_v^{(0)}(\mathbf{m}_0) = \mathbf{m}_0. \quad (2.6.13)$$

In iteration $\ell \geq 1$, check-nodes pass messages obtained from processing their extrinsic incoming messages in iteration $\ell - 1$ according to the mapping

$$\Psi_c^{(\ell)}(\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_{d_c-1}) = 2 \cdot \tanh^{-1} \left(\prod_{i=1}^{d_c-1} \tanh \frac{\mathbf{m}_i}{2} \right), \quad (2.6.14)$$

and variable-nodes pass the sum of their extrinsic incoming messages in iteration ℓ , plus their corresponding channel messages,

$$\Psi_v^{(\ell)}(\mathbf{m}_0, \mathbf{m}_1, \dots, \mathbf{m}_{d_v-1}) = \mathbf{m}_0 + \sum_{j=1}^{d_v-1} \mathbf{m}_j. \quad (2.6.15)$$

Assuming a BIPLGC model, the respective channel message m_0 of a transmitted codeword bit x , which is the LLR of x conditioned on the corresponding received symbol y , is obtained as follows.

$$\begin{aligned}
 m_0 &= \ln \frac{P(x = +1|y)}{P(x = -1|y)} \\
 &\stackrel{(i)}{=} \ln \frac{f_{y|x}(y|x = +1) \cdot P(x = +1)/f_y(y)}{f_{y|x}(y|x = -1) \cdot P(x = -1)/f_y(y)} \\
 &\stackrel{(ii)}{=} \ln \frac{f_{y|x}(y|x = +1)}{f_{y|x}(y|x = -1)} \\
 &\stackrel{(iii)}{=} \frac{2 \cdot y}{\sigma^2}, \tag{2.6.16}
 \end{aligned}$$

where (i), (ii), and (iii) follow from **Bayes'** rule, the assumption of the equality of $P(x = +1)$ and $P(x = -1)$, and the properties of a normal distribution (σ^2 is the variance of AWGN.) As can be seen, sum-product decoding algorithm requires the knowledge of the channel parameter, and in this sense, is *not* universal.

Min-Sum Decoding Algorithm

In channel coding, we are, now and again, interested in determining which valid configuration (codeword) has the largest APP, rather than determining the APPs for individual symbols. When all codewords are a priori equally likely, this amounts to maximum-likelihood (ML) decoding of codewords. **Viterbi** algorithm is a well-known ML decoding method for convolutional codes [38], and an efficient ML sequence detection (MLSD) tool for finite-state **Markov** sequences, with application to ISI channels [39]. Amongst message-passing algorithms, min-sum,¹¹ which may be regarded as a *bidirectional Viterbi* algorithm, is known to perform ML decoding on cycle-free **Tanner** graphs [48, 49, 54]. It can also be shown that this algorithm is an approximation to sum-product decoding algorithm. (For more details, refer to [50] and [58].)

¹¹This algorithm is sometimes referred to as *max-sum* [54].

In the LLR domain, the corresponding message maps of min-sum decoding algorithm are the same as those for sum-product decoding algorithm, except that

$$\Psi_c^{(\ell \geq 1)}(\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_{d_c-1}) = \text{sgn} \left(\prod_{i=1}^{d_c-1} \mathbf{m}_i \right) \cdot \min_{i: 1 \leq i \leq d_c-1} \{|\mathbf{m}_i|\}. \quad (2.6.17)$$

Assuming a BIPLGC, similar to the case for sum-product decoding algorithm, we obtain $\mathbf{m}_0 = \frac{2y}{\sigma^2}$, where y is the corresponding received symbol, and σ^2 is the variance of AWGN. Interestingly, for min-sum decoding algorithm, σ^2 can be factored out from all message maps, and can be ignored. In other words, min-sum decoding algorithm does not require the knowledge of the channel parameter, and is universal.

2.7 Threshold and Density Evolution

For a given code ensemble and a given message-passing decoding algorithm, the *threshold* is the maximum channel parameter¹² with the property that for all channel parameters strictly less than this threshold, the expected fraction of incorrect messages (passed along the edges of the codes' **Tanner** graphs) approaches zero as the number of iterations increases [12]. *Density evolution* is an efficient method for tracking the evolution of the densities of these messages, and predicting the asymptotic (convergence) behavior of iterative algorithms in decoding ensembles of random codes [12]. In order to apply density evolution, we assume that we are dealing with independent channel messages, and we are having a tree-like decoding neighborhood [12], to maintain statistical independence amongst messages throughout the iterative process. However, concentration results of [45] will guarantee that for sufficiently large block lengths, almost every randomly constructed code has a bit error-rate (BER) waterfall region very close to the threshold, even though the code's **Tanner** graph

¹²This is under the assumption that the channel condition worsens with increasing the channel parameter.

may not necessarily be cycle-free. In the following, we first briefly review the so-called *symmetry conditions*, and then describe density evolution method for $\mathcal{G}_B$.

2.7.1 Symmetry Conditions

In [12], it is shown that under the following symmetry conditions, the probability of message¹³ error is independent of the transmitted codeword.

Channel, Check-Node, and Variable-Node Symmetry Conditions

Definition 2.7.1. A binary-input channel is said to be (output) symmetric if

$$P(y = y|x = +1) = P(y = -y|x = -1). \quad (2.7.1)$$

(For continuous output channels, (2.7.1) becomes $f_{y|x}(y|x = +1) = f_{y|x}(-y|x = -1)$.)

Definition 2.7.2. A check-node message map is said to be symmetric if

$$\Psi_c^{(\ell)}(m_1, m_2, \dots, -m_i, \dots, m_{d_c-1}) = -\Psi_c^{(\ell)}(m_1, m_2, \dots, m_i, \dots, m_{d_c-1}), \quad (2.7.2)$$

for every $i \in \{1, 2, \dots, d_c - 1\}$

Definition 2.7.3. A variable-node message map is said to be symmetric if

$$\Psi_v^{(\ell)}(-m_0, -m_1, \dots, -m_{d_v-1}) = -\Psi_v^{(\ell)}(m_0, m_1, \dots, m_{d_v-1}). \quad (2.7.3)$$

The above symmetry conditions are fulfilled for all the channels introduced in the beginning of this chapter, and all the message-passing algorithms described in this chapter. As such, message error probabilities are independent of the transmitted codewords. In the sequel, we will assume that the all-one codeword has been sent, and a discrete message is erroneous if it is negative. This, as we will see, greatly simplifies the analysis and notations in the rest of the work.

¹³This refers to the messages that are passed along the edges of the **Tanner** graph.

2.7.2 Formulation of Density Evolution

In this subsection, focusing on $\mathcal{G}_B$, density evolution formulas are derived. Similar discussions for some other message-passing decoding algorithms, including sum-product decoding algorithm, can be found in [12]. It should be noted that for $\mathcal{G}_B$, we are dealing with PMFs, in our derivation, rather than probability density functions (PDFs).¹⁴

Density Evolution for $\mathcal{G}_B$

In the following, we use density evolution method to track error PMF under $\mathcal{G}_B$ as a function of iteration number. (The channel of interest is binary symmetric.) We first state the following useful lemma without proof.

Lemma 2.7.1 (Lemma 4.1 in [3]). *Consider a sequence of L independent binary digits in which the l^{th} digit is -1 with probability p_l . Then the probability that an even number of digits are -1 is*

$$\frac{1 + \prod_{l=1}^L (1 - 2 \cdot p_l)}{2}.$$

Corollary 2.7.1. *In the above sequence, the probability that an odd number of digits are -1 is*

$$\frac{1 - \prod_{l=1}^L (1 - 2 \cdot p_l)}{2}.$$

Applying this corollary to (2.6.3), using (2.6.6), and letting the average fractions of erroneous check-to-variable messages and variable-to-check messages in iteration $\ell \geq 1$ be denoted by $q_e^{(\ell)}$ and $p_e^{(\ell)}$, respectively, one can see that

$$q_e^{(\ell)} = \frac{1 - (1 - 2 \cdot p_e^{(\ell-1)})^{d_c-1}}{2}, \quad (2.7.4)$$

¹⁴In Appendix A, as an exercise, density evolution is formulated for min-sum decoding algorithms. There, PDFs are dealt with.

and

$$\begin{aligned}
 p_e^{(\ell)} &= p_e^{(0)} \cdot \left(1 - \sum_{k=b_\ell}^{d_v-1} \binom{d_v-1}{k} \cdot (1 - q_e^{(\ell)})^k \cdot (q_e^{(\ell)})^{d_v-1-k} \right) \\
 &\quad + (1 - p_e^{(0)}) \cdot \sum_{k=b_\ell}^{d_v-1} \binom{d_v-1}{k} \cdot (q_e^{(\ell)})^k \cdot (1 - q_e^{(\ell)})^{d_v-1-k}. \quad (2.7.5)
 \end{aligned}$$

The first term in the right-hand side of (2.7.5) represents the cases where the channel message is incorrect and less than b_ℓ (out of $d_v - 1$) extrinsic messages are correct. The second term, on the other hand, represents the cases where the channel message is correct but at least b_ℓ (out of $d_v - 1$) extrinsic messages are incorrect.

Combining (2.7.4) and (2.7.5), we obtain, as in [3, p. 50],

$$\begin{aligned}
 p_e^{(\ell \geq 1)} &= p_e^{(0)} \cdot \left(1 - \sum_{k=b_\ell}^{d_v-1} \binom{d_v-1}{k} \right. \\
 &\quad \cdot \left(\frac{1 + (1 - 2 \cdot p_e^{(\ell-1)})^{d_c-1}}{2} \right)^k \cdot \left(\frac{1 - (1 - 2 \cdot p_e^{(\ell-1)})^{d_c-1}}{2} \right)^{d_v-1-k} \Big) \\
 &\quad + (1 - p_e^{(0)}) \cdot \sum_{k=b_\ell}^{d_v-1} \binom{d_v-1}{k} \\
 &\quad \cdot \left(\frac{1 - (1 - 2 \cdot p_e^{(\ell-1)})^{d_c-1}}{2} \right)^k \cdot \left(\frac{1 + (1 - 2 \cdot p_e^{(\ell-1)})^{d_c-1}}{2} \right)^{d_v-1-k} \quad (2.7.6)
 \end{aligned}$$

The optimal choice of b_ℓ for minimizing $p_e^{(\ell)}$, determined in [3], is the smallest integer b satisfying

$$\frac{1 - p_e^{(0)}}{p_e^{(0)}} \leq \left(\frac{1 + (1 - 2 \cdot p_e^{(\ell-1)})^{d_c-1}}{1 - (1 - 2 \cdot p_e^{(\ell-1)})^{d_c-1}} \right)^{2 \cdot b - d_v + 1}. \quad (2.7.7)$$

Noting that $p_e^{(0)}$ is equal to the channel parameter ε , it is now clear that $\mathcal{G}_B$ requires knowledge of the channel parameter and thus is *not* universal.

2.8 Approximations to Density Evolution

As mentioned earlier, density evolution is an important tool for analyzing the asymptotic behavior of message-passing decoding algorithms. It can, however, be computationally intensive for complex decoding algorithms such as sum-product and min-sum. In these cases, simpler approximate methods, which are inevitably less accurate, are desirable.

For sum-product decoding algorithm, when used over BIPLGC, a number of simpler one-dimensional analysis methods have been proposed based on the interesting observation in [49] that the PDF of LLR messages (passed along **Tanner** graph edges throughout the iterative process) is approximately **Gaussian** (**Gaussian** approximation) [59–61]. Although quite accurate for variable-to-check messages, this approximation is not that precise for check-to-variable messages. In [62], a more accurate one-dimensional analysis method is introduced. The key point in the new method is that check-to-variable messages are not assumed to have a **Gaussian** distribution, instead, their density is computed in each iteration.

Another well-known and widely-used analysis method for iterative algorithms is *extrinsic information transfer* (EXIT) *chart* method [59,63–66]. In EXIT chart analysis, instead of tracking the evolution of the density of messages, we track the evolution of a single parameter (usually, the mutual information between the messages passed and the transmitted bits), iteration by iteration. This method, in general, is not as accurate as density evolution. It is, nevertheless, attractive because of its simplicity, in addition to producing reasonably accurate results. Furthermore, EXIT charts permit visualizing the behavior of iterative decoding algorithms, which is of high importance in both performance analysis and code design. We can also use EXIT chart-like graphs to track the error probability of the messages, iteration by iteration

[14, 15, 17–21, 51, 52, 62, 67].¹⁵ For some iterative algorithms, such as those focused on in Chapters 3, 4, and 5, analysis based on tracking the error probability, using EXIT chart-like graphs, is equivalent to analysis based on density evolution, as the density of messages can be specified by a single parameter which is related to the error probability. We leave more detailed treatment of this subject for the next chapter, where it is addressed under the title of ‘graphical representation of density evolution.’

¹⁵This approach was introduced originally by Gallager [3, p. 49].

“ Non sempre la verità corrisponde all’opinione della maggioranza. ”¹

– Giovanni Paolo II

Chapter 3

Majority-Based (MB) Decoding of Regular LDPC Codes

In this chapter, we present a comprehensive study of a family of *binary* message-passing algorithms for decoding *regular* LDPC codes, referred to as *majority-based* (MB) algorithms. $\mathcal{G}_A$, which is regarded as a benchmark for binary message-passing decoding algorithms [68–70], and majority decoding algorithm, both introduced in Chapter 2, belong to this family. MB algorithms, which are the building-blocks of $\mathcal{G}_B$, work based on a generalized majority-decision rule, and their simple implementation makes them potentially attractive for many high-rate communication applications.² Throughout this chapter, we show the great advantages that some of these algorithms are capable of offering over $\mathcal{G}_A$ and majority decoding algorithm. In particular, we demonstrate that the error probability for MB algorithms converges to zero very fast, and in many cases, much speedier than it does for $\mathcal{G}_A$. This makes MB algorithms good candidates for high speed applications, such as optical communication systems, where the processing delay has to be minimized. Moreover, our study shows that some MB algorithms appear to outperform $\mathcal{G}_A$ and majority decoding algorithm by a large margin, in the sense that they proffer much better (noise) threshold values. The fast convergence and the better performance offered by MB algorithms other than $\mathcal{G}_A$

¹ “ The truth is not always the same as the majority decision. ”

²It is noteworthy that other hard-decision decoding algorithms based on ideas similar to those behind MB algorithms have been discussed in [71, 72].

and majority decoding algorithm, suggests a serious role for them in future coding schemes where the system constraints require the use of hard-decision decoders.

The outline of this chapter is as follows. In Section 3.1, we present the formal definition of MB algorithms, and explain the recursion describing the evolution of the average error probability for regular LDPC code ensembles. We will then present, in Section 3.2, the analysis based on density evolution. This includes discussions on the fixed-points of MB algorithms and the stability of the fixed-point located at zero (corresponding to average error probability of zero). We also investigate the convergence properties of these algorithms, and provide characterizations and tight upper bounds for the threshold. Section 3.3 contains numerical results on threshold values of MB algorithms and more discussions on the convergence speed of these algorithms. Some concluding remarks are given in Section 3.4.

3.1 MB Algorithms

3.1.1 MB Algorithms: Formal Definition

Consider $\mathcal{C}^n(d_v, d_c)$. For a non-negative integer $\omega \leq \lfloor d_v/2 \rfloor - 1$, the MB algorithm of order ω , denoted by MB^ω , which can be used to decode $\mathcal{C}^n(d_v, d_c)$ over a BSC, is specified by the same message maps as those for $\mathcal{G}_B$, except that

$$\Psi_v^{(\ell \geq 1)}(\mathbf{m}_0, \mathbf{m}_1, \dots, \mathbf{m}_{d_v-1}) = \begin{cases} -\mathbf{m}_0, & \text{if } |\{j : \mathbf{m}_j = -\mathbf{m}_0\}| \geq \lfloor d_v/2 \rfloor + \omega, \\ \mathbf{m}_0, & \text{otherwise,} \end{cases} \quad (3.1.1)$$

where all messages have an alphabet space of $\mathcal{M} = \{-1, +1\}$.³

In words, variable-nodes pass their channel messages unless at least $\lfloor d_v/2 \rfloor + \omega$ extrinsic incoming messages disagree with the corresponding channel messages, in

³Note that for MB algorithms, variable-node and check-node symmetry conditions are fulfilled.

which case the negative of these channel messages are passed. Note that the value of ω determines the strength of the disagreement required for the change. Extreme cases are majority decoding algorithm ($\omega = 0$) and $\mathcal{G}_A$ ($\omega = \lfloor d_v/2 \rfloor - 1$), equivalent to majority and unanimous votes, respectively. It is worth mentioning that MB algorithms are universal, as their message maps do not depend on the iteration number.

3.1.2 MB Algorithms: Density Evolution

We assume that MB^ω is used to decode $\mathcal{C}^n(d_v, d_c)$ over a BSC with crossover probability ε , and use density evolution to track the evolution of the average fraction of erroneous messages (i.e., messages whose signs do not agree with the signs of their corresponding transmitted bits) with iterations. (In order to apply density evolution, we assume a tree-like decoding neighborhood.)

Letting the average fractions of erroneous variable-to-check messages in iteration $\ell \geq 1$ be denoted by $p_{e,\varepsilon}^{(\ell)}$, one can see, as discussed in density evolution for $\mathcal{G}_B$, that

$$p_{e,\varepsilon}^{(\ell \geq 1)} = p_{e,\varepsilon}^{(0)} \cdot \left(1 - f_+^{d_v, \omega, d_c} \left(p_{e,\varepsilon}^{(\ell-1)}\right)\right) + \left(1 - p_{e,\varepsilon}^{(0)}\right) \cdot f_-^{d_v, \omega, d_c} \left(p_{e,\varepsilon}^{(\ell-1)}\right), \quad (3.1.2)$$

where $p_{e,\varepsilon}^{(0)} = \varepsilon$, and

$$f_{\pm}^{d_v, \omega, d_c}(x) \equiv \sum_{k=\lfloor d_v/2 \rfloor + \omega}^{d_v-1} \binom{d_v-1}{k} \cdot \left(\frac{1 \pm (1-2 \cdot x)^{d_c-1}}{2}\right)^k \cdot \left(\frac{1 \mp (1-2 \cdot x)^{d_c-1}}{2}\right)^{d_v-1-k}. \quad (3.1.3)$$

The first term in the right-hand side of (3.1.2) represents the cases where the channel message is incorrect and less than $\lfloor d_v/2 \rfloor + \omega$ (out of $d_v - 1$) extrinsic messages are correct. The second term, on the other hand, represents the cases where the channel message is correct but at least $\lfloor d_v/2 \rfloor + \omega$ (out of $d_v - 1$) extrinsic messages are incorrect.

As mentioned in Chapter 2, for a given (d_v, d_c) pair and a given order ω , the threshold $\varepsilon^*(d_v, \omega, d_c)$, or simply ε^* , is defined as the supremum of all values of ε in

the interval $[0, 0.5]$ such that $\lim_{\ell \rightarrow \infty} p_{e,\varepsilon}^{(\ell)} = 0$. We will see in the next section that $p_{e,\varepsilon}^{(\ell)}$ is an increasing function of ε . It then follows that $\forall \varepsilon < \varepsilon^*$, $\lim_{\ell \rightarrow \infty} p_{e,\varepsilon}^{(\ell)} = 0$. The threshold, in general, can be computed numerically. However, as we will show later, in some cases the threshold can be determined analytically. (Similar results have been presented for $\mathcal{G}_A$ in [68, 70].) We would like to remind that although the density evolution analysis is based on the assumption of random cycle-free **Tanner** graphs with given degrees for variable-nodes and check-nodes, concentration results of [45] will guarantee that for sufficiently large block lengths, almost every randomly constructed code has a BER waterfall region very close to the threshold, even though the code's **Tanner** graph may not necessarily be cycle-free.

3.2 Dynamics of MB Algorithms

In this section, we analyze the dynamics of MB algorithms via density evolution. We use a graphical approach to visualize the convergence behavior of these algorithms. This approach is particularly helpful in discussing the threshold values and the speed of convergence. One of the main results in this chapter is Theorem 3.2.3, which provides a tight upper bound on the threshold of MB algorithms.

3.2.1 Threshold Characterizations and Bounds

To investigate the evolution of the recursion (3.1.2) throughout the iterations, we define the function

$$f_{\varepsilon}^{\omega}(x) \equiv \varepsilon \cdot (1 - f_{+}^{\omega}(x)) + (1 - \varepsilon) \cdot f_{-}^{\omega}(x).^4 \quad (3.2.1)$$

⁴Clearly, the functions $f_{+}^{\omega}(x)$, $f_{-}^{\omega}(x)$, and $f_{\varepsilon}^{\omega}(x)$ also depend on d_v and d_c . However, we deal with this dependency implicitly, as our focus is mainly on the dependency of these functions on ω , and in the case of $f_{\varepsilon}^{\omega}(x)$, on ε .

We are interested in the properties of $f_\varepsilon^\omega(x)$ in the interval $[0, 0.5]$, as $p_{e,\varepsilon}^{(\ell \geq 1)} = f_\varepsilon^\omega(p_{e,\varepsilon}^{(\ell-1)})$ and $p_{e,\varepsilon}^{(0)} = \varepsilon$, and therefore the sequence $(p_{e,\varepsilon}^{(0)}, p_{e,\varepsilon}^{(1)}, p_{e,\varepsilon}^{(2)}, \dots)$ and the sequence $\beta_\varepsilon^\omega \equiv (\varepsilon, f_\varepsilon^\omega(\varepsilon), f_\varepsilon^\omega(f_\varepsilon^\omega(\varepsilon)), \dots)$ are identical. We use this relationship to give a graphical interpretation of density evolution as follows. For a given order ω and a given channel parameter ε , consider the curve $y = f_\varepsilon^\omega(x)$ in the x - y plane. We start from the point $(\varepsilon, f_\varepsilon^\omega(\varepsilon))$ on the curve. From this *starting-point*, we move horizontally towards the line $y = x$ to reach the point $(f_\varepsilon^\omega(\varepsilon), f_\varepsilon^\omega(\varepsilon))$ on the line. We then move vertically towards the curve $y = f_\varepsilon^\omega(x)$ to reach the point $(f_\varepsilon^\omega(\varepsilon), f_\varepsilon^\omega(f_\varepsilon^\omega(\varepsilon)))$ on the curve. By repeating these horizontal and vertical steps, we obtain all the values in the sequence β_ε^ω , or equivalently, in the sequence $(p_{e,\varepsilon}^{(0)}, p_{e,\varepsilon}^{(1)}, p_{e,\varepsilon}^{(2)}, \dots)$. (Refer to Appendix B.) Clearly, the properties of the function $f_\varepsilon^\omega(x)$, and its relative location with respect to the line $y = x$, determine the convergence behavior of this recursion.

Example 3.2.1. *Figure 3.1 graphically interprets density evolution for $\mathcal{C}^\infty(3, 6)$, decoded by MB^0 , and for channel parameters $\varepsilon = 0.0350$ and 0.0450 . As can be seen, depending on the channel parameter, we can have a decreasing ($\varepsilon = 0.0350$) or an increasing ($\varepsilon = 0.0450$) trend of average error probability with iterations. For a starting-point on the line $y = x$, the algorithm is stuck right from the beginning, and there will be no decreasing or increasing trend.* ◀

In the following, we discuss some of the properties of $f_\varepsilon^\omega(x)$ which are useful in understanding the dynamics of MB algorithms.

Lemma 3.2.1. *For any given order $\omega_\circ$ and any given parameter $x_\circ \in [0, 0.5]$, $f_\varepsilon^{\omega_\circ}(x_\circ)$ is increasing in ε , i.e., $\frac{\partial}{\partial \varepsilon} f_\varepsilon^{\omega_\circ}(x_\circ) \geq 0$.*

Proof. It can be shown that

$$\frac{\partial}{\partial \varepsilon} f_\varepsilon^{\omega_\circ}(x_\circ) = 1 - (f_+^{\omega_\circ}(x_\circ) + f_-^{\omega_\circ}(x_\circ)). \quad (3.2.2)$$

Figure 3.1: Graphical interpretation of density evolution for $C^\infty(3, 6)$, decoded by MB^0 , and for $\epsilon = 0.0350$ and 0.0450 .

To show that (3.2.2) is non-negative, we first define

$$\zeta^+(x) \equiv \frac{1 + (1 - 2 \cdot x)^{d_v-1}}{2}, \quad (3.2.3)$$

and

$$\zeta^-(x) \equiv \frac{1 - (1 - 2 \cdot x)^{d_v-1}}{2}. \quad (3.2.4)$$

Clearly, for any $x_o \in [0, 0.5]$,

$$0.5 \leq \zeta^+(x_o) \leq 1,$$

$$0 \leq \zeta^-(x_o) \leq 0.5,$$

and

$$\zeta^+(x_o) + \zeta^-(x_o) = 1.$$

Thus

$$\begin{aligned} (\zeta^+(x_o) + \zeta^-(x_o))^{d_v-1} &= \sum_{k=0}^{d_v-1} \binom{d_v-1}{k} \cdot (\zeta^+(x_o))^k \cdot (\zeta^-(x_o))^{d_v-1-k} \\ &\stackrel{(i)}{=} \sum_{k=0}^{\lfloor d_v/2 \rfloor - \omega_o - 1} \binom{d_v-1}{k} \cdot (\zeta^+(x_o))^k \cdot (\zeta^-(x_o))^{d_v-1-k} \\ &\quad + \sum_{k=\lfloor d_v/2 \rfloor - \omega_o}^{\lceil d_v/2 \rceil + \omega_o - 1} \binom{d_v-1}{k} \cdot (\zeta^+(x_o))^k \cdot (\zeta^-(x_o))^{d_v-1-k} \\ &\quad + \sum_{k=\lceil d_v/2 \rceil + \omega_o}^{d_v-1} \binom{d_v-1}{k} \cdot (\zeta^+(x_o))^k \cdot (\zeta^-(x_o))^{d_v-1-k} \\ &= 1. \end{aligned} \quad (3.2.5)$$

According to (3.1.3), the first and the third terms on the right-hand side of (i) are equal to $f_-^\omega(x_o)$ and $f_-^\omega(x_o)$, respectively, and the second term is non-negative. Using (3.2.2), we can now conclude that $\frac{\partial}{\partial \varepsilon} f_\varepsilon^{\omega_o}(x_o) \geq 0$. $\square$

Remark 3.2.1. It can be seen that $f_{\varepsilon}^{\omega_{\circ}}(x_{\circ})$ is a strictly increasing function of ε , i.e., $\frac{\partial}{\partial \varepsilon} f_{\varepsilon}^{\omega_{\circ}}(x_{\circ}) > 0$, if the second term in (i) is positive, or equivalently if $x_{\circ} \neq 0$ and the upper limit of this summation term is greater than, or equal to, its lower limit, i.e., either $d_{\mathbf{v}}$ is odd, or $\omega_{\circ} \neq 0$.

Lemma 3.2.2. *For any given order $\omega_{\circ}$ and any given channel parameter $\varepsilon_{\circ} \in [0, 0.5]$, $f_{\varepsilon_{\circ}}^{\omega_{\circ}}(x)$ is increasing in x , i.e., $\frac{\partial}{\partial x} f_{\varepsilon_{\circ}}^{\omega_{\circ}}(x) \geq 0$.*

Proof. It is not difficult to see that with $\zeta^+(x)$ and $\zeta^-(x)$ defined as (3.2.3) and (3.2.4), respectively, we have

$$\begin{aligned} \frac{\partial}{\partial x} f_{\varepsilon_{\circ}}^{\omega_{\circ}}(x) &= (\lceil d_{\mathbf{v}}/2 \rceil + \omega_{\circ}) \cdot \binom{d_{\mathbf{v}} - 1}{\lceil d_{\mathbf{v}}/2 \rceil + \omega_{\circ}} \cdot (d_{\mathbf{c}} - 1) \cdot (1 - 2 \cdot x)^{d_{\mathbf{c}} - 2} \\ &\quad \cdot \left(\varepsilon_{\circ} \cdot (\zeta^+(x))^{\lceil d_{\mathbf{v}}/2 \rceil + \omega_{\circ} - 1} \cdot (\zeta^-(x))^{\lfloor d_{\mathbf{v}}/2 \rfloor - \omega_{\circ} - 1} \right. \\ &\quad \left. + (1 - \varepsilon_{\circ}) \cdot (\zeta^+(x))^{\lfloor d_{\mathbf{v}}/2 \rfloor - \omega_{\circ} - 1} \cdot (\zeta^-(x))^{\lceil d_{\mathbf{v}}/2 \rceil + \omega_{\circ} - 1} \right), \end{aligned} \quad (3.2.6)$$

and the right-hand side of (3.2.6) is non-negative in the interval of interest, $[0, 0.5]$. $\square$

Remark 3.2.2. It should be noted that $\frac{\partial}{\partial x} f_{\varepsilon_{\circ}}^{\omega_{\circ}}(x) > 0, \forall x \in (0, 0.5)$. It is equal to zero at $x = 0.5$ for any $\omega_{\circ}$ and $\varepsilon_{\circ}$, and also at $x = 0$ for any $\varepsilon_{\circ}$ and $\omega_{\circ} \neq \lfloor d_{\mathbf{v}}/2 \rfloor - 1$. For $\mathcal{G}_A$ ($\omega_{\circ} = \lfloor d_{\mathbf{v}}/2 \rfloor - 1$), the derivative at $x = 0$ is only zero if $\varepsilon_{\circ} = 0$.

The next proposition follows from Lemma 3.2.1 and Lemma 3.2.2 by finite induction.

Proposition 3.2.1. *For any given order $\omega_{\circ}$, $p_{\varepsilon, \varepsilon}^{(\ell \geq 1)} = f_{\varepsilon}^{\omega_{\circ}}(p_{\varepsilon, \varepsilon}^{(\ell-1)})$ is increasing in ε . In other words, MB algorithms respect physical degradation ordering of BSCs.*

Also, the next two propositions follow from Lemma 3.2.2.

Proposition 3.2.2. *For a given $MB^{\omega_\circ}$ and a given channel parameter $\varepsilon_\circ \in [0, 0.5]$, the sequence $\mathcal{B}_{\varepsilon_\circ}^{\omega_\circ}$ is strictly decreasing, constant, or strictly increasing if the starting-point $(\varepsilon_\circ, f_{\varepsilon_\circ}^{\omega_\circ}(\varepsilon_\circ))$ on the curve $y = f_{\varepsilon_\circ}^{\omega_\circ}(x)$ is below, on, or above the line $y = x$, respectively.*

Proposition 3.2.3 (Convergence to Fixed-Points). *For a given $MB^{\omega_\circ}$ and a given channel parameter $\varepsilon_\circ \in [0, 0.5]$, the sequence $\mathcal{B}_{\varepsilon_\circ}^{\omega_\circ}$ converges to a crossing-point (a point at which the curve $y = f_{\varepsilon_\circ}^{\omega_\circ}(x)$ and the line $y = x$ intersect). A crossing-point is in fact a fixed-point of the recursion given in (3.1.2).*

It is noteworthy that having the starting-point below the line $y = x$, as the next example demonstrates, is no guarantee for the decreasing average probability of error to converge to zero.

Example 3.2.2. *Figure 3.2 depicts the graphical interpretation of density evolution for $C^\infty(4, 6)$, decoded by MB^1 , and for channel parameters $\varepsilon = 0.0800$ and 0.1000 . Figure 3.2(a) shows a decreasing trend that does not converge to zero and is trapped at a crossing-point, while Figure 3.2(b) demonstrates an increasing trend which also converges to a crossing-point. ◀*

The following theorem describes the condition for the convergence of the average error probability to zero.

Theorem 3.2.1. *For a given order $\omega_\circ$ and a given channel parameter $\varepsilon_\circ \in [0, 0.5]$, $\mathcal{B}_{\varepsilon_\circ}^{\omega_\circ}$ is a decreasing sequence which converges to zero iff the curve $y = f_{\varepsilon_\circ}^{\omega_\circ}(x)$ is below the line $y = x$, $\forall x \in (0, \varepsilon_\circ]$.*

Proof. For sufficiency, if the curve $y = f_{\varepsilon_\circ}^{\omega_\circ}(x)$ is below the line $y = x$, $\forall x \in (0, \varepsilon_\circ]$, then combining Propositions 3.2.2 and 3.2.3, one can see that the probability of error

Figure 3.2: Graphical interpretation of density evolution for $C^\infty(4,6)$, decoded by MB^1 , and for $\epsilon = 0.0800$ and 0.1000 : (a) an example of a decreasing trend of average error probability that does not converge to zero; (b) an example of an increasing trend.

decreases and will converge to the only crossing-point in the interval $[0, \varepsilon_0]$, which is $x = 0$. For necessity, we show that if the curve $y = f_{\varepsilon_0}^{\omega_0}(x)$ is not below the line $y = x$, $\forall x \in (0, \varepsilon_0]$, then the sequence $\beta_{\varepsilon_0}^{\omega_0}$ does not converge to zero. Suppose that this is indeed the case (i.e., the curve $y = f_{\varepsilon_0}^{\omega_0}(x)$ is not below the line $y = x$, $\forall x \in (0, \varepsilon_0]$). Then, for some $x \in (0, \varepsilon_0]$, the curve $y = f_{\varepsilon_0}^{\omega_0}(x)$ is either above, or on the line $y = x$. If ε_0 is amongst such x , then according to Proposition 3.2.2, the trend is not decreasing. Otherwise, the starting-point $(\varepsilon, f_{\varepsilon}^{\omega_0}(\varepsilon))$ is below the line $y = x$, and since the curve $y = f_{\varepsilon_0}^{\omega_0}(x)$ is continuous, there must exist at least one crossing-point in the interval $(0, \varepsilon_0]$. In this case, according to Propositions 3.2.2 and 3.2.3, the rightmost of such crossing-points entraps the decreasing average probability of error, and prevents it from converging to zero. $\square$

Combining Proposition 3.2.1 and the definition of threshold ε^* , it can be seen that $\forall \varepsilon < \varepsilon^*$, $\lim_{\ell \rightarrow \infty} p_{\varepsilon, \varepsilon}^{(\ell)} = 0$. In addition, one can combine Theorem 3.2.1 and Lemma 3.2.1 to see, from a slightly different perspective, that if $\beta_{\varepsilon_0}^{\omega_0}$ is a decreasing sequence which converges to zero for a given MB^{ω_0} and a given channel parameter $\varepsilon_0 \in [0, 0.5]$, then $\forall \varepsilon \in [0, \varepsilon_0]$, the sequence $\beta_{\varepsilon}^{\omega_0}$ will also be decreasing, and will converge to zero. We can thus provide an alternative definition for the threshold ε^* of an MB^{ω_0} as follows.

Proposition 3.2.4 (Characterization of Threshold). *For any given MB^{ω_0} with a threshold ε^* , we have*

$$\varepsilon^* = \sup \{ \varepsilon \in [0, 0.5] \mid f_{\varepsilon}^{\omega_0}(x) < x, \forall x \in (0, \varepsilon] \}. \quad (3.2.7)$$

It is noteworthy that the above characterization closely resembles the alternative description of the threshold for $\mathcal{G}_A$ on BSC, given in [70], and that of binary erasure channel (BEC) and its associated iterative decoding algorithm, presented in [46].

As $f_\varepsilon^\omega(0) = 0$, $x_o = 0$ is always a fixed-point of (3.1.2). The following theorem describes the stability of this important fixed-point.

Theorem 3.2.2 (Stability of the Fixed-Point of MB^{ω_o} Located at Zero). *Consider $C^\infty(d_v, d_c)$. For any MB^{ω_o} with $\omega_o < \lfloor d_v/2 \rfloor - 1$ and any channel parameter $\varepsilon_o \in [0, 0.5]$, there exists a constant $\kappa = \kappa(d_v, d_c, \omega_o, \varepsilon_o) > 0$, such that $\forall x \in (0, \kappa)$, $f_{\varepsilon_o}^{\omega_o}(x) < x$, i.e., the fixed-point $x_o = 0$ is stable.⁵ For $\mathcal{G}_A$ ($\omega_o = \lfloor d_v/2 \rfloor - 1$), if $\varepsilon_o \cdot (d_v - 1) \cdot (d_c - 1) < 1$, or $\varepsilon_o \cdot (d_v - 1) \cdot (d_c - 1) = 1$ and $d_v > 3$, then there exists a constant $\kappa = \kappa(d_v, d_c, \varepsilon_o) > 0$, such that $\forall x \in (0, \kappa)$, $f_{\varepsilon_o}^{\lfloor d_v/2 \rfloor - 1}(x) < x$, i.e., the fixed-point $x_o = 0$ is stable. On the other hand, if $\varepsilon_o \cdot (d_v - 1) \cdot (d_c - 1) > 1$, or $\varepsilon_o \cdot (d_v - 1) \cdot (d_c - 1) = 1$ and $d_v = 3$, then there exists a constant $\kappa = \kappa(d_v, d_c, \varepsilon_o) > 0$, such that $\forall x \in (0, \kappa)$, $f_{\varepsilon_o}^{\lfloor d_v/2 \rfloor - 1}(x) > x$, i.e., the fixed-point $x_o = 0$ is unstable.⁶*

Proof. We know that $f_{\varepsilon_o}^{\omega_o}(0) = 0$, and based on (3.2.6), we have

$$\frac{\partial}{\partial x} f_{\varepsilon_o}^{\omega_o}(x) \Big|_{x=0} = \begin{cases} \varepsilon_o \cdot (d_v - 1) \cdot (d_c - 1), & \text{if } \omega_o = \lfloor d_v/2 \rfloor - 1, \\ 0, & \text{otherwise.} \end{cases} \quad (3.2.8)$$

For sufficiently small values of x , the convergence behavior is determined by the linear term in the Taylor series of $f_{\varepsilon_o}^{\omega_o}(x)$ around zero. For any MB^{ω_o} with $\omega_o < \lfloor d_v/2 \rfloor - 1$, we thus have $f_{\varepsilon_o}^{\omega_o}(x) < x$ for sufficiently small values of $x \neq 0$. For $\mathcal{G}_A$, this is only the case if $\varepsilon_o \cdot (d_v - 1) \cdot (d_c - 1) < 1$, or $\varepsilon_o \cdot (d_v - 1) \cdot (d_c - 1) = 1$ and $d_v > 3$. $\square$

The following corollary, which is the same as the upper bound on the threshold of $\mathcal{G}_A$ given in [70] for the special case of regular codes, follows from Theorems 3.2.1 and 3.2.2.

⁵The stability condition can equivalently be stated as the existence of a constant $\kappa > 0$ such that if $p_{\varepsilon_o, \kappa_o}^{(\ell_o)} < \kappa$ for some ℓ_o , then $\lim_{\ell \rightarrow \infty} p_{\varepsilon_o, \kappa_o}^{(\ell)} = 0$.

⁶Similar results for sum-product can be found in [46, 73–75].

Corollary 3.2.1. *For $\mathcal{G}_A$, the threshold ε^* satisfies*

$$\varepsilon^* \leq \frac{1}{(d_v - 1) \cdot (d_c - 1)}. \quad (3.2.9)$$

It should be clear to the reader by now that the starting-point $(\varepsilon, f_\varepsilon^\omega(\varepsilon))$ has a major role in the convergence properties of MB algorithms. In the following, we provide a concise discussion on this role.

Let us first define the function

$$\phi^\omega(x) \equiv x \cdot (1 - f_+^\omega(x)) + (1 - x) \cdot f_-^\omega(x), \quad (3.2.10)$$

in the interval $[0, 0.5]$, with polynomial functions $f_+^\omega(x)$ and $f_-^\omega(x)$ given by (3.1.3). We notice that $\phi^\omega(x) = f_\varepsilon^\omega(x)|_{\varepsilon=x}$. The curve $y = \phi^\omega(x)$ is, in fact, the locus of the starting-points of MB^ω for different channel parameters x . We call any crossing-point of the curve $y = \phi^\omega(x)$ and the line $y = x$, a *steady-point*. Obviously, the sequence β_ε^ω remains constant if the starting-point is a steady-point. The points $(0, 0)$ and $(0.5, 0.5)$ are two steady-points of all MB algorithms for any ensemble of regular LDPC codes. As the two coordinates for a steady-point are equal, hereafter, we identify a steady-point with a single value.

Example 3.2.3. *Depicted in Figure 3.3, are the loci of starting-points for $C^\infty(3, 4)$ and $C^\infty(5, 10)$, decoded by MB^0 and MB^1 , respectively. The figure shows that in general, in addition to the points 0 and 0.5, there may exist other steady-points. ◀*

We denote the smallest positive steady-point by δ^* . It is, as we will see later, an upper bound on the threshold. We continue with a few results on $\phi^\omega(x)$ and the steady-points of MB algorithms. We start with the following corollary, which follows from Lemmas 3.2.1 and 3.2.2, and the fact that $\phi^\omega(x) = f_\varepsilon^\omega(x)|_{\varepsilon=x}$, and thus $\frac{d}{dx}\phi^\omega(x) = \frac{\partial}{\partial \varepsilon}f_\varepsilon^\omega(x)|_{\varepsilon=x} + \frac{\partial}{\partial x}f_\varepsilon^\omega(x)|_{\varepsilon=x}$.

Corollary 3.2.2. *For any given order ω , $\phi^{\omega\circ}(x)$ is increasing in x , i.e., $\forall x \in [0, 0.5]$, $\frac{d}{dx}\phi^{\omega\circ}(x) \geq 0$.*

It follows from Remarks 3.2.1 and 3.2.2 that with the exception of $x = 0.5$, d_ω even, and $\omega = 0$, $\phi^\omega(x)$ is a strictly increasing function of x in the interval $(0, 0.5]$, i.e., $\frac{d}{dx}\phi^\omega(x) > 0$.

Remark 3.2.3. Combining (3.2.2) and (3.2.6), it is easy to see that $\frac{d}{dx}\phi^\omega(x)|_{x=0} = 0$, and also $\frac{d}{dx}\phi^\omega(x)|_{x=0.5} < 1$. The curve $y = \phi^\omega(x)$ also crosses the line $y = x$ at $x = 0$ and $x = 0.5$. Putting these together, we conclude that the (continuous) curve $y = \phi^\omega(x)$ must cross the line $y = x$ at least once more in the interval $(0, 0.5)$, and therefore, in addition to $x = 0$ and $x = 0.5$, there exist at least one other steady-point. Thus, $\delta^* < 0.5$.

Note that $\forall \varepsilon \in (0, \delta^*)$, the starting-point is below the line $y = x$, and the next theorem follows from Proposition 3.2.4.

Theorem 3.2.3. *The smallest positive steady-point δ^* is an upper bound for the threshold ε^* , i.e., $\varepsilon^* \leq \delta^*$.*

As we will see later, the upper bound of Theorem 3.2.3 is achieved for many MB algorithms applied to a wide range of regular LDPC code ensembles, i.e., we have $\varepsilon^* = \delta^*$. This provides an alternative characterization for threshold and an easier method to compute it. When the bound is not achieved, i.e., when $\varepsilon^* < \delta^*$, then for $\varepsilon^* < \varepsilon < \delta^*$, the trend of the sequence β_ε^ω is decreasing, but $\lim_{\ell \rightarrow \infty} p_{\varepsilon, \varepsilon}^{(\ell)} > 0$. (Refer to Figure 3.2(a).)

One can combine Corollary 3.2.1 and Theorem 3.2.3 to conclude the following corollary.

Figure 3.3: Starting-point loci for $C^\infty(3, 4)$ and $C^\infty(5, 10)$, decoded by MB^0 and MB^1 , respectively.

Corollary 3.2.3 (Lemma 4 in [70]). *For $\mathcal{G}_A$, the threshold ε^* satisfies*

$$\varepsilon^* \leq \min \left\{ \frac{1}{(d_v - 1) \cdot (d_c - 1)}, \delta^* \right\}. \quad (3.2.11)$$

3.2.2 Speed of Convergence

It is easy to see that for a given channel parameter ε_o , the convergence speed of a MB^ω at any iteration depends on how far the curve $y = f_{\varepsilon_o}^\omega(x)$ is from the line $y = x$ in that iteration. The farther the curve, the faster the trend. This explains the slow convergence of $\mathcal{G}_A$ in decoding $\mathcal{C}^\infty(d_v, d_c)$ over a channel with a parameter $\varepsilon < \varepsilon_o$ and close to $\frac{1}{(d_v-1)(d_c-1)}$. [Refer to Table 3.2.] In this case, there is only a narrow tunnel between the curve and the line in the vicinity of $x = 0$.

The following theorem partially orders MB algorithms in terms of their convergence speed.

Theorem 3.2.4. *Suppose that for a given order ω_o , a given channel parameter ε_o , and a given parameter x_o , we have $f_{\varepsilon_o}^{\omega_o}(x_o) \leq f_{\varepsilon_o}^{\omega_o+1}(x_o)$. Then $f_{\varepsilon_o}^{\omega_1}(x) \leq f_{\varepsilon_o}^{\omega_2}(x)$, $\forall x \in [0, x_o]$ and for all $\omega_o \leq \omega_1 \leq \omega_2$.*

To prove this theorem, we first state and prove the following lemma.

Lemma 3.2.3. *Given an order ω_o , a channel parameter ε_o , and a parameter x_o , we have the inequality*

$$\frac{1 - \varepsilon_o}{\varepsilon_o} \leq \left(\frac{1 + (1 - 2 \cdot x_o)^{d_c-1}}{1 - (1 - 2 \cdot x_o)^{d_c-1}} \right)^{2 \cdot (\lceil d_v/2 \rceil + \omega_o) - d_v + 1}, \quad (3.2.12)$$

iff $f_{\varepsilon_o}^{\omega_o}(x_o) \leq f_{\varepsilon_o}^{\omega_o+1}(x_o)$.

Proof. $f_{\varepsilon_o}^{\omega_o}(x_o) \leq f_{\varepsilon_o}^{\omega_o+1}(x_o)$

$$\begin{aligned} & \stackrel{(3.2.1)}{\iff} (1 - \varepsilon_o) \cdot (f_{-}^{\omega_o}(x_o) - f_{-}^{\omega_o+1}(x_o)) \leq \varepsilon_o \cdot (f_{+}^{\omega_o}(x_o) - f_{+}^{\omega_o+1}(x_o)) \\ & \stackrel{(3.1.3)}{\iff} \frac{1 - \varepsilon_o}{\varepsilon_o} \leq \left(\frac{1 + (1 - 2 \cdot x_o)^{d_c-1}}{1 - (1 - 2 \cdot x_o)^{d_c-1}} \right)^{2 \cdot (\lceil d_v/2 \rceil + \omega_o) - d_v + 1}. \end{aligned}$$

□

Proof of Theorem 3.2.4. Since $f_{\varepsilon_o}^{\omega_o}(x_o) \leq f_{\varepsilon_o}^{\omega_o+1}(x_o)$, according to Lemma 3.2.3, the order ω_o , the channel parameter ε_o , and the parameter x_o satisfy (3.2.12). This inequality remains satisfied if ω_o is increased, or x_o is decreased. Hence, according to Lemma 3.2.3 again, $f_{\varepsilon_o}^{\omega_1}(x) \leq f_{\varepsilon_o}^{\omega_2}(x)$, $\forall x \in [0, x_o]$ and for all $\omega_o \leq \omega_1 \leq \omega_2$. □

It is noteworthy that for any given order ω_o and any given channel parameter ε_o , (3.2.12) is satisfied for sufficiently small values of x_o . In other words, *between two MB algorithms used to decode the same ensemble of regular LDPC codes over the same channel, the one with smaller order eventually converges faster at the later stages of decoding*. Also noteworthy is the close relationship between (3.2.12) and (2.7.7). We will further discuss this relation in Chapter 4.

Example 3.2.4. Figure 3.4 illustrates Theorem 3.2.4. In this figure, we have depicted the line $y = x$ and the curves $y = f_{\varepsilon}^{\omega}(x)$, for $C^{\infty}(8, 10)$, for $\omega = 3, 2, 1$, and 0, and for $\varepsilon = \frac{1}{(8-1) \cdot (10-1)} = 1/63$ (which is the threshold for MB^3 , i.e., $\mathcal{G}_A$). As can be seen, for instance, $f_{1/63}^1(0.0278) \leq f_{1/63}^2(0.0278)$, and in accordance with Theorem 3.2.4, $f_{1/63}^1(x) \leq f_{1/63}^2(x) \leq f_{1/63}^3(x)$, $\forall x \in [0, 0.0278]$. The partial ordering of these curves in the intervals $[0, 0.0017]$, $[0.0018, 0.0113]$, $[0.0114, 0.0185]$,

$[0.0186, 0.0278]$, $[0.0279, 0.0345]$, $[0.0346, 0.0495]$, and $[0.0496, 0.5]$, are

$$\begin{aligned}
 f_{1/63}^0(x) &< f_{1/63}^1(x) < f_{1/63}^2(x) < f_{1/63}^3(x), \\
 f_{1/63}^1(x) &< f_{1/63}^0(x) < f_{1/63}^2(x) < f_{1/63}^3(x), \\
 f_{1/63}^1(x) &< f_{1/63}^2(x) < f_{1/63}^0(x) < f_{1/63}^3(x), \\
 f_{1/63}^1(x) &< f_{1/63}^2(x) < f_{1/63}^3(x) < f_{1/63}^0(x), \\
 f_{1/63}^2(x) &< f_{1/63}^1(x) < f_{1/63}^3(x) < f_{1/63}^0(x), \\
 f_{1/63}^2(x) &< f_{1/63}^3(x) < f_{1/63}^1(x) < f_{1/63}^0(x),
 \end{aligned}$$

and

$$f_{1/63}^3(x) < f_{1/63}^2(x) < f_{1/63}^1(x) < f_{1/63}^0(x),$$

respectively. Notice that the ordering of the curves in the vicinity of $x = 0$ is $0 < 1 < 2 < 3$, while it is exactly the other way around in the vicinity of $x = 0.5$. It is also interesting to note that over a channel with this parameter, the average error probability of 10^{-6} is achieved after 3, 2, 4, and 28567 iterations for orders $\omega = 0, 1, 2$, and 3, respectively. This enormous difference is expected, since unlike the other curves, the tunnel between the curve $y = f_{1/63}^3(x)$ and the line $y = x$ is very narrow in the vicinity of $x = 0$. (Table 3.2 presents similar comparisons for more codes.) ◀

We finish this section by the following theorem which discusses the convergence speed of MB algorithms in the vicinity of the fixed-point at zero.

Theorem 3.2.5. *For $\varepsilon < \varepsilon^*$, the probability of error goes to zero exponentially with the number of iterations ℓ for $\mathcal{G}_A$, and super-exponentially for other MB algorithms, provided that ℓ is large enough.*

Figure 3.4: Comparison of convergence speed of different orders of MB decoding algorithms for $C^\infty(8, 10)$, and for $\varepsilon = 1/63$, which is the threshold for MB^3 .

Proof. Considering only the most dominant term in the expansion of (3.2.1) for sufficiently small values of x , it is easily verifiable that

$$f_\varepsilon^\omega(x) \approx \tilde{\varepsilon} \cdot \binom{d_v - 1}{\lfloor d_v/2 \rfloor - \omega} \cdot ((d_c - 1) \cdot x)^{\lfloor d_v/2 \rfloor - \omega}, \quad (3.2.13)$$

where $\tilde{\varepsilon} = \varepsilon$, if $\omega \neq 0$, or d_v is odd; and $= 1$, otherwise. One can then deduce from the application of $p_{\varepsilon, \varepsilon}^{(\ell)} = f_\varepsilon^\omega(p_{\varepsilon, \varepsilon}^{(\ell-1)})$, $\ell \geq 1$, recursively, with the initial condition $p_{\varepsilon, \varepsilon}^{(0)} = \varepsilon$, that for sufficiently small values of ε , and with $\eta \equiv \tilde{\varepsilon} \cdot \binom{d_v - 1}{\lfloor d_v/2 \rfloor - \omega} \cdot (d_c - 1)^{\lfloor d_v/2 \rfloor - \omega}$, $p_{\varepsilon, \varepsilon}^{(\ell)}$ is $\mathcal{O}(\eta^\ell)$, if $\omega = \lfloor d_v/2 \rfloor - 1$; and $\mathcal{O}\left((\eta^{1/(\lfloor d_v/2 \rfloor - \omega - 1)} \cdot \varepsilon)^{(\lfloor d_v/2 \rfloor - \omega)^\ell}\right)$, if $\omega < \lfloor d_v/2 \rfloor - 1$; as $\ell \rightarrow \infty$. ($\mathcal{O}(\cdot)$ denotes order.) In other words, $p_{\varepsilon, \varepsilon}^{(\ell)}$ reduces exponentially, or super-exponentially with parameter $\lfloor d_v/2 \rfloor - \omega$, in ℓ , respectively.⁷ This is of course with the conditions that ε is less than ε^* , and also small enough to satisfy $\eta \cdot \varepsilon^{\lfloor d_v/2 \rfloor - \omega - 1} < 1$. The latter condition, however, can be relaxed by noticing the fact that $\varepsilon < \varepsilon^* \Rightarrow \lim_{\ell \rightarrow \infty} p_{\varepsilon, \varepsilon}^{(\ell)} = 0$, which implies that if ℓ is large enough, we will be in close enough vicinity of $x = 0$ on the curve $f_\varepsilon^\omega(x)$ such that the probability of error goes to zero with same exponents as those mentioned above. $\square$

It should be noted that for lower order MB algorithms, the asymptotic convergence of error probability towards zero is faster, as the parameter $\lfloor d_v/2 \rfloor - \omega$ in the exponent is larger.

3.3 Simulation Results and Special Cases

In this section, we compute MB algorithms' threshold values for some regular LDPC code ensembles. We also show that in many cases $\varepsilon^* = \delta^*$, and thus the threshold can be computed analytically. Moreover, we provide simulation results to compare MB algorithms in terms of their speed of convergence.

⁷Similar results for the special case of $\omega = 0$ are given in [13, 76].

Example 3.3.1. *In this example, we analytically compute the threshold when $C^\infty(4, 5)$ is decoded by MB^0 . For these parameters, as we will prove in Theorem 3.3.1, $\varepsilon^* = \delta^*$. The threshold is thus the smallest real root δ^* of $\phi^0(x) - x$ in the interval $(0, 0.5)$. Applying the change of variable $\mathfrak{X} = 1 - 2 \cdot x$, we rewrite $\phi^0(x) - x$ as $\mathfrak{X}^{12} - 3 \cdot \mathfrak{X}^4 + 2 \cdot \mathfrak{X}$. Computing δ^* is then equivalent to finding the largest real root $\mathfrak{T}^*$ of this new polynomial in the interval $(0, 1)$. We obtain $\mathfrak{T}^* = 0.94790954676824$, and thus $\varepsilon^* = \delta^* = \frac{1-\mathfrak{T}^*}{2} = 0.02604522661588$. (Note that obtaining exact thresholds numerically is usually a cumbersome task.) ◀*

Table 3.1 lists ε^* and δ^* for some (d_v, d_c) pairs, and for different orders of MB algorithms. It also lists $\frac{1}{(d_v-1) \cdot (d_c-1)}$ values. As can be seen, for many (d_v, d_c) pairs, $\mathcal{G}_A$ ($\omega = \lfloor d_v/2 \rfloor - 1$) is not the best choice in terms of the threshold value. As a matter of fact, in some cases, this algorithm has the smallest threshold amongst all the orders of MB algorithms. (Refer to the last six rows of Table 3.1.) Moreover, as we will see later, other orders also appear to be, in many cases, considerably faster than $\mathcal{G}_A$.

Table 3.1 also shows that for many ensembles, the bounds on the threshold value (given in Theorem 3.2.3 and Corollary 3.2.3) are tight, i.e., we usually have $\varepsilon^* = \delta^*$, for $\omega \neq \lfloor d_v/2 \rfloor - 1$, and $\varepsilon^* = \min \left\{ \frac{1}{(d_v-1) \cdot (d_c-1)}, \delta^* \right\}$, for $\omega = \lfloor d_v/2 \rfloor - 1$. It should be noted, however, that there exist other examples where the bounds cannot be achieved. See, for instance, the underlined entries in the last three rows of Table 3.1. For these cases, the curve $y = f_\varepsilon^\omega(x)$ touches the line $y = x$ somewhere in the interval $(0, \varepsilon^*)$. The following theorems cover some of the cases for which the bounds are achieved.

Theorem 3.3.1. *Consider $C^\infty(d_v, d_c)$, and suppose that this ensemble is decoded by MB^0 . If the variable-node degree is an even number, then $\varepsilon^* = \delta^*$.*

Proof. Using (3.1.3), it is easy to see that for an even d_v , $f_+^0(x) + f_-^0(x) = 1$. Therefore, $f_\varepsilon^0(x) = \varepsilon \cdot (1 - f_+^0(x)) + (1 - \varepsilon) \cdot f_-^0(x) = f_-^0(x)$ and $\phi^0(x) = x \cdot (1 - f_+^0(x)) + (1 -$

Table 3.1: Threshold values and smallest steady-points for different orders of MB algorithms

(d_v, d_c)	$\varepsilon_{MB^0}^*, \delta_{MB^0}^*$	$\varepsilon_{MB^1}^*, \delta_{MB^1}^*$	$\varepsilon_{MB^2}^*, \delta_{MB^2}^*$	$\varepsilon_{MB^3}^*, \delta_{MB^3}^*$	$\frac{1}{(d_v-1) \cdot (d_c-1)}$
(3, 4)	0.1069, 0.1069	–	–	–	0.1666
(3, 5)	0.0611, 0.0611	–	–	–	0.1250
(3, 6)	0.0394, 0.0394	–	–	–	0.1000
(3, 7)	0.0275, 0.0275	–	–	–	0.0833
(3, 8)	0.0202, 0.0202	–	–	–	0.0714
(3, 9)	0.0155, 0.0155	–	–	–	0.0625
(3, 10)	0.0122, 0.0122	–	–	–	0.0555
(4, 5)	0.0260, 0.0260	0.0833, 0.1265	–	–	0.0833
(4, 6)	0.0159, 0.0159	0.0666, 0.0911	–	–	0.0666
(4, 7)	0.0107, 0.0107	0.0555, 0.0695	–	–	0.0555
(4, 8)	0.0077, 0.0077	0.0476, 0.0552	–	–	0.0476
(4, 9)	0.0058, 0.0058	0.0416, 0.0451	–	–	0.0416
(4, 10)	0.0045, 0.0045	0.0370, 0.0378	–	–	0.0370
(5, 6)	0.0650, 0.0650	0.0500, 0.1244	–	–	0.0500
(5, 7)	0.0482, 0.0482	0.0416, 0.0978	–	–	0.0416
(5, 8)	0.0374, 0.0374	0.0357, 0.0796	–	–	0.0357
(5, 9)	0.0301, 0.0301	0.0312, 0.0666	–	–	0.0312
(5, 10)	0.0248, 0.0248	0.0277, 0.0568	–	–	0.0277
(6, 7)	0.0306, 0.0306	0.0806, 0.0806	0.0333, 0.1181	–	0.0333
(6, 8)	0.0236, 0.0236	0.0648, 0.0648	0.0285, 0.0975	–	0.0285
(6, 9)	0.0188, 0.0188	0.0536, 0.0536	0.0250, 0.0824	–	0.0250
(6, 10)	0.0155, 0.0155	0.0453, 0.0453	0.0222, 0.0710	–	0.0222
(7, 8)	0.0494, 0.0494	0.0854, 0.0854	0.0238, 0.1113	–	0.0238
(7, 9)	0.0405, 0.0405	0.0717, 0.0717	0.0208, 0.0948	–	0.0208
(7, 10)	0.0339, 0.0339	0.0615, 0.0615	0.0185, 0.0822	–	0.0185
(8, 9)	0.0296, 0.0296	0.0601, 0.0601	<u>0.0719</u> , 0.0859	0.0178, 0.1049	0.0178
(8, 10)	0.0247, 0.0247	0.0511, 0.0511	<u>0.0640</u> , 0.0741	0.0158, 0.0913	0.0158
(9, 10)	0.0409, 0.0409	0.0653, 0.0653	<u>0.0552</u> , 0.0844	0.0138, 0.0990	0.0138

$x) \cdot f_-^0(x) = f_-^0(x)$. Based on the definitions of ε^* and δ^* , this means that ε^* must be equal to δ^* . $\square$

It is worth noting that for even d_v values, MB^0 uses the channel messages only once at the beginning: when the initialization is done, these messages play no further role, and the variable-nodes only pass the majority decision of incoming extrinsic messages. This independence of decoding process (f curves) from ε , is, in fact, the underlying cause of the equality of ε^* and δ^* .

For regular LDPC code ensembles with odd variable-node degrees, we could neither prove a similar general result, nor find any counterexample. In the next theorem, however, we prove that $\varepsilon^* = \delta^*$ for the special case of $d_v = 3$.

Theorem 3.3.2. *For the ensemble of $(3, d_c)$ -regular LDPC codes, decoded by $\mathcal{G}_A(MB^0)$, $\varepsilon^* = \delta^*$.*

Proof. It is easily verifiable that for $d_c = 4, 5$, and 6 , we have $\varepsilon^* = \delta^*$. In the following, we prove the equality for $d_c > 6$. By using **Descartes'** rule of signs,⁸ one can show that for $d_v = 3$, and any $\varepsilon \in [0, 0.5]$, the polynomial $f_\varepsilon^0(x) - x$ has at most *three* real zeros in $[0, 0.5]$.⁹ If $\varepsilon^* < \delta^*$, then, according to Proposition 3.2.4, $f_\varepsilon^0(x)$ *touches* the line $y = x$ somewhere in $[0, \varepsilon^*)$ at least *once*, and thus $f_\varepsilon^0(x) - x$ will have at least *two* zeros in $[0, \varepsilon^*)$. Moreover, for $d_c = 6$, $f_\varepsilon^0(0)$ crosses the line $y = x$ at *two* different points in $(0, 0.5)$. It can also be shown that for all $\varepsilon \in [0, 0.5]$, $f_\varepsilon^0(x)$

⁸**Descartes'** rule of signs states that if $\varphi(x) = \sum_{i=1}^n c_i \cdot x^{p_i}$, with $0 \leq p_1 < p_2 < \dots < p_n$, and with all coefficients $c_i \neq 0$, then the number of positive real zeros of φ is upper bounded by the number of sign changes between consecutive coefficients c_i [77].

⁹By defining a new variable $\mathfrak{X} = 1 - 2 \cdot x$, $\varphi(x) = f_\varepsilon^0(x) - x$ can be rewritten as $\varphi(\mathfrak{X}) = -(1 - 2 \cdot \varepsilon) + 2 \cdot \mathfrak{X} - 2 \cdot \mathfrak{X}^{d_c-1} + (1 - 2 \cdot \varepsilon) \cdot \mathfrak{X}^{2 \cdot (d_c-1)}$. Applying **Descartes'** rule to $\varphi(\mathfrak{X})$, it is easy to see that $\varphi(\mathfrak{X})$ has at most three positive real zeros for $\varepsilon < 0.5$, and at most one positive real zero for $\varepsilon = 0.5$. This is equivalent to saying that $\varphi(x)$ has at most three real zeros in $(-\infty, 0.5)$, and therefore in $[0, 0.5]$. As $\mathfrak{X} = 0$ ($\equiv x = 0.5$) is a simple zero of $\varphi(\mathfrak{X})$ for $\varepsilon = 0.5$, and is not a zero of $\varphi(\mathfrak{X})$ otherwise, the interval $[0, 0.5]$ can be extended to $[0, 0.5]$.

is increasing with d_c . Being increasing with d_c , ε , and x , having a derivative of zero at $x = 0.5$, and having a value less than ε^* at $x = \varepsilon^*$, the curve $y = f_{\varepsilon^*}^0(x)$, for $d_c > 6$, has no choice but to *cross* the line $y = x$ at *two* different points in $[0, 0.5]$, in addition to the *touch-induced* zeros of $f_{\varepsilon^*}^0(x) - x$. Therefore, the number of zeros will add up to at least *four*, which contradicts **Descartes'** rule of signs. The only way to avoid this contradiction is to have $\varepsilon^* = \delta^*$. $\square$

In Table 3.1, we observe that for $\omega = \lfloor d_v/2 \rfloor - 1$, i.e., when $\mathcal{G}_A$ is used, ε^* is equal to $\frac{1}{(d_v-1)(d_c-1)}$, and less than δ^* , for all (d_v, d_c) pairs with $d_v > 3$. It should be noted that this is not the case in general. A counterexample is the ensemble of $(4, 11)$ -regular LDPC codes, decoded by MB^0 . In this case, $\varepsilon^* = \delta^* = 0.0322 < \frac{1}{(4-1)(11-1)} = 1/30$.

To compare the convergence speed of MB algorithms, we proceed as follows. For a given (d_v, d_c) pair, we compute the smallest number of iterations that each order of the algorithm needs to achieve an average fraction of incorrect messages equal to, or less than, 10^{-6} when the channel parameter is 90% of the smallest value amongst the thresholds of different orders. (We choose 90% arbitrarily, with the consideration that it results in both a relatively large channel parameter, and a relatively fast convergence, even for the algorithm with the smallest threshold.) The results are listed in Table 3.2. It can be seen that those cases in which $\varepsilon^* = \delta^*$, enjoy a very fast convergence, in general. The convergence speed of some of these cases is more than 20 times faster than the cases for which $\varepsilon^* < \delta^*$. On the contrary, we see that $\mathcal{G}_A$ usually has a slow convergence. This happens when the curve $y = f_{\varepsilon^*}^\omega(x)$ for $\mathcal{G}_A$ touches the line $y = x$ at zero, and the channel parameter ε is also close to ε^* . In this case the tunnel between $y = f_{\varepsilon}^\omega(x)$ and $y = x$ is very narrow in the vicinity of $x = 0$, which results in the slow decrease of error probability with iterations.¹⁰

¹⁰In Table 3.2, it is observed that for cases with $d_v = 4$, MB^1 , equivalent to $\mathcal{G}_A$, is faster than MB^0 . The reason is that in these cases, the threshold for MB^0 , and thus the selected channel parameter, is much smaller than the threshold of $\mathcal{G}_A$. This means that for $\mathcal{G}_A$, the tunnel between $y = f_{\varepsilon}^\omega(x)$ and $y = x$ is wide open in the whole interval $(0, \varepsilon)$, including the vicinity of 0.

Table 3.2: Number of iterations required to achieve an average fraction of errors equal to, or less than, 10^{-6} . The channel parameter is 90% of the smallest threshold amongst different orders of MB algorithms

(d_v, d_c)	$\omega = 0$	$\omega = 1$	$\omega = 2$	$\omega = 3$
(3, 4)	29	–	–	–
(3, 5)	20	–	–	–
(3, 6)	16	–	–	–
(3, 7)	14	–	–	–
(3, 8)	13	–	–	–
(3, 9)	12	–	–	–
(3, 10)	11	–	–	–
(4, 5)	8	8	–	–
(4, 6)	8	7	–	–
(4, 7)	7	6	–	–
(4, 8)	7	5	–	–
(4, 9)	7	5	–	–
(4, 10)	7	4	–	–
(5, 6)	5	85	–	–
(5, 7)	5	82	–	–
(5, 8)	6	82	–	–
(5, 9)	6	61	–	–
(5, 10)	6	41	–	–
(6, 7)	5	4	47	–
(6, 8)	5	3	31	–
(6, 9)	5	3	23	–
(6, 10)	5	3	20	–
(7, 8)	3	3	78	–
(7, 9)	3	3	75	–
(7, 10)	3	3	75	–
(8, 9)	3	2	3	73
(8, 10)	3	2	3	71
(9, 10)	2	2	3	79

Example 3.3.2. Figure 3.5 depicts BER curves versus both E_b/N_0 ¹¹ and ε , for a randomly constructed $(6, 10)$ -regular LDPC code of length 10^5 , with no cycle of length 4 in its **Tanner** graph, decoded by MB^0 , MB^1 , and MB^2 ($\mathcal{G}_A$). For all simulation results, the maximum number of iterations is set to 500, and for each point, enough number of codewords are simulated to generate at least 10 codeword errors.¹² The threshold values for these algorithms are 0.0155 ($E_b/N_0 = 7.6464$ dB), 0.0453 ($E_b/N_0 = 5.5383$ dB), and $\frac{1}{(6-1)(10-1)} = 1/45$ ($E_b/N_0 = 7.0343$ dB), respectively. As can be seen in Figure 3.5, the waterfall regions of BER curves associated with lower-order MB algorithms are very sharp, and occur at close proximity to the theoretical threshold for these algorithms. For $\mathcal{G}_A$, however, the waterfall region is not nearly as sharp. At a BER of about 10^{-6} , MB^0 and MB^1 outperform $\mathcal{G}_A$ by more than 2.7 dB and 0.5 dB, respectively. This is while the theoretical threshold for MB^0 is even worse than that of $\mathcal{G}_A$ by more than 0.6 dB. This, which suggests that for $\mathcal{G}_A$, the convergence of the ensemble performance to its average by increasing the block length, predicted by concentration results of [45], happens much slower compared to the other two algorithms, can be loosely explained as follows. Amongst MB algorithms, due to the shape of its f curve, $\mathcal{G}_A$ has a relatively slow convergence speed and needs more iterations to achieve a certain (low) error probability. For a low error probability, therefore, the (cycle-free) supporting tree [12] in decoding under $\mathcal{G}_A$, has to be deeper. This thus exacerbates the performance deviation from the asymptotic limit under finite block lengths, where the graph inevitably has cycles with relatively short lengths. It is worth mentioning that in some applications having a gradual slope in waterfall region can be considered an advantage. This is particularly the case for wireless applications where E_b/N_0 could change over a rather wide range.

¹¹This is the equivalent E_b/N_0 assuming that the BSC is a model for BIPLGC with a 1-bit quantizer.

¹²For many simulation points at higher BERs, at least 200 codeword errors are generated. Reducing this to 10 for just a few points at the bottom of the curves was due to the limitation in computational resources.

At $BER = 10^{-5}$, the average number of iterations for converged cases for MB^0 , MB^1 , and $\mathcal{G}_A$ are about 5.60, 9.21, and 13.29, respectively. These numbers are increased to about 6.19, 10.79, and 37.95, respectively, for BER of about 10^{-3} . These results show the slower convergence of $\mathcal{G}_A$ compared to the other two algorithms. ◀

3.4 Conclusion

MB algorithms, which include $\mathcal{G}_A$ and majority decoding algorithm as special cases, are of interest mainly due to their low complexity (per iteration). By performing a detailed study of these algorithms, we showed that many of MB algorithms had larger noise thresholds compared to $\mathcal{G}_A$ and majority decoding algorithm. We presented tight analytical upper bounds on these thresholds, and proved that in many cases, these bounds were achieved. We also highlighted another attractive feature of MB algorithms, i.e., their generally fast convergence. We showed that an MB algorithm's convergence speed was determined by its order, and explained why many MB algorithms outpaced $\mathcal{G}_A$ substantially. In particular, we proved that while the asymptotic convergence speed of $\mathcal{G}_A$ was exponential with iteration number, for other MB algorithms, it was super-exponential. As a result of this work, we expect that more attention would be paid to MB algorithms other than the widely popular $\mathcal{G}_A$. The fast convergence of MB algorithms, along with their relatively large noise thresholds, make them good candidates for *hybrid* algorithms in which different decoding algorithms are combined to achieve better performance/complexity tradeoffs, compared to those achievable by individual algorithms [13, 16–21, 45, 51, 52, 67, 76]. (Refer to Chapter 4.)

Figure 3.5: BER curves for a randomly constructed $(6, 10)$ -regular LDPC code of length 10^5 , decoded by different orders of MB algorithms: (a) horizontal axis represents E_b/N_0 in dB; (b) horizontal axis represents ϵ . (Note that MB^2 is the same as $\mathcal{G}_A$.)

“There’s nothing remarkable about it. All one has to do is to hit the right keys at the right time and the instrument plays itself.”

– Johann Sebastian Bach

Chapter 4

Hybrid Decoding of Regular LDPC Codes

In this chapter, we discuss the idea of combining existing iterative (decoding) algorithms in order to devise more effective *hybrid* algorithms. Hybrid algorithms were first introduced by Gallager in the construction of $\mathcal{G}_B$, and have been since revisited in literature such as [13, 45, 51, 52, 76]. We refer to these algorithms as *switch-type* hybrid, since they switch between different constituent algorithms through the iteration process. Here, we propose a new family of hybrid algorithms, called *time-invariant* hybrid. These algorithms use a specific blend of different algorithms in each iteration. The blend, however, does not change by iteration, hence, the name ‘time-invariant.’

The outline of this chapter is as follows. In Section 4.1, we provide a formal definition for hybrid algorithms, and demonstrate, through a few examples, that hybrid algorithms are capable of improving error performance and convergence speed¹ at the expense of a small increase in the complexity of the decoder’s structure. We then investigate the dynamics of hybrid algorithms in Section 4.2, and compare time-invariant and switch-type hybrid algorithms with their constituents, and also with

¹The term *convergence speed* means speed of convergence of error probability to zero (or to a specified value).

each other, and show that although $\mathcal{G}_B$, as an MB switch-type hybrid algorithm, provides the best threshold and the fastest convergence amongst all MB hybrid algorithms, it can be outperformed by MB time-invariant hybrid algorithms at finite block lengths. We also show that MB switch-type hybrid algorithms are not nearly as robust as their time-invariant counterparts, with respect to channel variations. Some concluding remarks are given in Section 4.3.

4.1 Hybrid Decoding Algorithms

We start in Section 4.1.1 by providing a formal definition for hybrid algorithms in a general framework. In Section 4.1.2, we consider two categories of hybrid algorithms, referred to as *time-invariant* hybrid ($\mathcal{H}_{TI}$) algorithms and *switch-type* hybrid ($\mathcal{H}_{ST}$) algorithms. We then give a few examples showing the great improvement in error performance and convergence speed that these hybrid algorithms can provide compared to their constituents.

4.1.1 Formal Definition

Consider $\mathcal{C}^n(d_v, d_c)$, and suppose that $\Delta_0, \Delta_1, \dots, \Delta_{N-1}$ are N message-passing algorithms which can be used to decode $\mathcal{C}^n(d_v, d_c)$ over a given channel. Each algorithm Δ_i , $0 \leq i \leq N-1$, is specified in iteration $\ell \geq 0$ by $\Psi_{i,v}^{(\ell)}$ and $\Psi_{i,c}^{(\ell)}$, its variable-node and check-node message maps, respectively. Assuming that all messages have the same alphabet space $\mathcal{M}$,² we can formulate a new message-passing decoding algorithm Δ which is defined based on $\Delta_0, \Delta_1, \dots, \Delta_{N-1}$ as follows. In iteration ℓ of Δ , variable-nodes and check-nodes are randomly partitioned into N groups according to

²Message alphabets can be different provided that each algorithm is able to interpret the messages it receives.

PMF vectors $\vec{\alpha}^{(\ell)} = (\alpha_0^{(\ell)}, \alpha_1^{(\ell)}, \dots, \alpha_{N-1}^{(\ell)})$ and $\vec{\beta}^{(\ell)} = (\beta_0^{(\ell)}, \beta_1^{(\ell)}, \dots, \beta_{N-1}^{(\ell)})$, respectively, and the nodes in group i , $0 \leq i \leq N-1$, process the messages in accordance with the constituent algorithm Δ_i . The new algorithm is said to be the hybrid of $\Delta_0, \Delta_1, \dots, \Delta_{N-1}$ according to hybridization sequences $\{\vec{\alpha}^{(\ell)}\}_{\ell=0}^{\infty}$ and $\{\vec{\beta}^{(\ell)}\}_{\ell=0}^{\infty}$, and this is denoted by

$$\Delta = \mathcal{H} \left(\Delta_0, \Delta_1, \dots, \Delta_{N-1}, \{\vec{\alpha}^{(\ell)}\}_{\ell=0}^{\infty}, \{\vec{\beta}^{(\ell)}\}_{\ell=0}^{\infty} \right). \quad (4.1.1)$$

Example 4.1.1. *A snapshot of a hybrid decoding process is shown in Figure 4.1. The process involves two constituent algorithms. In the captured iteration, variable-nodes and check-nodes have been partitioned into two groups according to PMF vectors (0.6, 0.4) and (0.5, 0.5), respectively, and the nodes in each group treat the messages in accordance with one of the two algorithms. (The two groups have been differentiated by their colors.)* ◀

Note that we can symbolically specify Δ with message maps

$$\Psi_v^{(\ell)} = \sum_{i=0}^{N-1} \alpha_i^{(\ell)} \cdot \Psi_{i,v}^{(\ell)}, \quad (4.1.2)$$

and

$$\Psi_c^{(\ell)} = \sum_{i=0}^{N-1} \beta_i^{(\ell)} \cdot \Psi_{i,c}^{(\ell)}. \quad (4.1.3)$$

One can regard hybrid decoding algorithms as *irregular* decoding algorithms, in analogy to *irregular* LDPC codes. While in the latter, the decoding algorithm is fixed but the degrees of variable-nodes or check-nodes are variable, in the former, the decoding algorithm is variable (different nodes perform according to different algorithms) but variable-nodes and check-nodes have fixed degrees. It is well known that introducing irregularity into code structure improves the performance of iterative coding schemes [44–47]. Interestingly, as will be shown in this chapter, introducing

Figure 4.1: A snapshot of a $(3, 5)$ -regular LDPC code of length 10 decoded by a hybrid algorithm which involves two constituent algorithms. White nodes, which comprise 60% of the variable-nodes and 50% of the check-nodes, process the messages according to the first algorithm, while black nodes treat the messages in accordance with the second one.

irregularity into the decoding can also improve the performance of iterative coding schemes.

4.1.2 $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ Algorithms

The above definition of hybrid algorithms is quite general. It is, in fact, general to such an extent that makes analyzing the behavior of these algorithms very complex. In this work, we focus only on two specific, yet important, combining methods whose analyses are manageable. In the first method, the constituent algorithms are blended together in a *time-invariant* manner ($\mathcal{H}_{TI}$). In this case, $\tilde{\alpha}^{(\ell)}$ and $\tilde{\beta}^{(\ell)}$ are independent of ℓ .³ In the second method, combining the constituent algorithms is performed by *switching* between them ($\mathcal{H}_{ST}$). The sequences $\{\tilde{\alpha}^{(\ell)}\}_{\ell=0}^{\infty}$ and $\{\tilde{\beta}^{(\ell)}\}_{\ell=0}^{\infty}$ in this case consist of elementary vectors. (In other words, in each iteration ℓ of an $\mathcal{H}_{ST}$ algorithm, all the nodes of one type (variable or check) process messages according to only one of the constituent algorithms.) It is worth mentioning that $\mathcal{G}_B$ and the algorithms discussed in [13, 16, 51, 52, 76] belong to this category.

In the following, we continue with a few illustrative examples to show how hybrid algorithms can improve error performance and convergence speed. In all the examples, as well as the rest of the chapter, we assume that hybrid algorithms' constituents belong to the family of MB algorithms. More precisely, we assume that $\mathcal{C}^{\infty}(d_v, d_c)$ is decoded over a BSC, using

$$\Delta = \mathcal{H} \left(MB^0, MB^1, \dots, MB^{\lfloor d_v/2 \rfloor - 1}, \{\tilde{\alpha}^{(\ell)}\}_{\ell=0}^{\infty} \right). \quad (4.1.4)$$

(Note that since all MB algorithms have the same check-node message map, we have dropped $\{\tilde{\beta}^{(\ell)}\}_{\ell=0}^{\infty}$ from the notation.) Such hybrid algorithms are referred to as *MB*

³It should be noted that the results on $\mathcal{H}_{TI}$ algorithms, obtained in the rest of this chapter, are still valid even if the partitioning of the nodes remains unchanged throughout the iteration process, i.e., even if the same group of nodes perform the same operation in every iteration, as these are 'average' results on the ensembles of codes, rather than a particular realization of codes.

hybrid algorithms. The assumption of MB constituents makes the analysis simpler since keeping track of the expected fraction of incorrect messages throughout the iteration process can be performed using closed form formulas. In principle, however, other algorithms can also be used as constituents. The analysis would become more tedious for such cases.⁴

As shown in Chapter 3, for a given channel parameter ε , an MB algorithm of order ω , $0 \leq \omega \leq \lfloor d_v/2 \rfloor - 1$, can be represented by a simple curve $f_\varepsilon^\omega(x)$. Similarly, the curve

$$h_\varepsilon^{\vec{\alpha}}(x) = \sum_{\omega=0}^{\lfloor d_v/2 \rfloor - 1} \alpha_\omega \cdot f_\varepsilon^\omega(x), \quad (4.1.5)$$

which is a linear combination of $f_\varepsilon^\omega(x)$ curves, represents the $\mathcal{H}_{TI}$ algorithm devised by blending MB algorithms according to vector $\vec{\alpha} = (\alpha_0, \alpha_1, \dots, \alpha_{\lfloor d_v/2 \rfloor - 1})$. By an argument similar to that in Chapter 3, it can be shown that the sequence of error probabilities for such an $\mathcal{H}_{TI}$ algorithm is monotonic, i.e., it is strictly decreasing, constant, or strictly increasing iff $h_\varepsilon^{\vec{\alpha}}(\varepsilon)$ is less than, equal to, or greater than ε , respectively. It can also be shown that the sequence converges to zero with iterations iff $h_\varepsilon^{\vec{\alpha}}(x) < x, \forall x \in (0, \varepsilon]$.

Example 4.1.2. Consider $\mathcal{C}^\infty(8, 9)$ and the four MB algorithms that can be used to decode this ensemble over a BSC, namely MB^0 , MB^1 , MB^2 , and MB^3 . Also, consider $\Delta_\circ$, the $\mathcal{H}_{TI}$ algorithm that blends these MB algorithms according to $\vec{\alpha}_\circ = (0, 0.2, 0.8, 0)$. Figure 4.2 depicts the curves representing $\Delta_\circ$ and its constituents for a channel with $\varepsilon_\circ = 0.0720$. One can see that the necessary and sufficient condition for the convergence of error probability to zero, i.e., $f_{\varepsilon_\circ}^\omega(x) < x, \forall x \in (0, \varepsilon_\circ]$, is not satisfied for either one of constituent MB algorithms. (It can be easily verified that the average probability of error increases with iterations for MB^0 and MB^1 , while for MB^2 and MB^3 , although the average error probability decreases with iterations, it

⁴In Appendix C, a hybrid algorithm with some constituents other than MB algorithms, is studied.

does not approach zero.) On the other hand, as the portrayed evolution of the expected fraction of erroneous messages for $\Delta_\circ$ shows, the average probability of error for this $\mathcal{H}_{TI}$ algorithm converges to zero by increasing the number of iterations, and there is still room for $\Delta_\circ$ to cope with even worse channels. ◀

Much the same as $\mathcal{H}_{TI}$ algorithms, an $\mathcal{H}_{ST}$ algorithm can be represented by a combination of the curves representing its constituents. This combination, however, is piecewise instead of linear.

Example 4.1.3. Consider the same scenario as in the previous example, but suppose that the hybrid decoding algorithm is switch-type this time. More specifically, let $\Delta_\circ$, the $\mathcal{H}_{ST}$ algorithm in use, be specified by

$$\bar{\alpha}_\circ^{(\ell)} = \begin{cases} (0, 0, 1, 0), & \text{if } \ell \leq 5, \\ (0, 1, 0, 0), & \text{if } 6 \leq \ell \leq 9, \\ (0, 0, 0, 1), & \text{if } \ell = 10, \\ (1, 0, 0, 0), & \text{if } 11 \leq \ell. \end{cases}$$

Fig. 4.3 portrays the curves representing MB^0 , MB^1 , MB^2 , MB^3 , and $\Delta_\circ$ for a channel with $\varepsilon_\circ = 0.0720$. Notice that $\Delta_\circ$ is represented by a piecewise combination of the curves representing its constituents. The figure further portrays the evolution of the expected fraction of incorrect messages for $\Delta_\circ$, which, unlike those for the constituent MB algorithms, does converge to zero with iterations. As can be seen, the sequence of error probabilities in this example is not monotonic.⁵ We can therefore conclude that in contrast to the case for $\mathcal{H}_{TI}$ algorithms, the sequence of error probabilities is not necessarily monotonic for $\mathcal{H}_{ST}$ algorithms. Also, similar to the previous example, one can see that $\Delta_\circ$ has still more room to cope with even worse channels. ◀

⁵Notice that the sequence of error probabilities is strictly decreasing in all but one iteration, number 10, where it increases.

Figure 4.2: Graphical representation of density evolution for $C^\infty(8, 9)$, decoded by MB^0 , MB^1 , MB^2 , MB^3 , and $\mathcal{H}_{TI}(MB^0, MB^1, MB^2, MB^3, \vec{\alpha}_o)$, with $\vec{\alpha}_o$ given in Example 4.1.2, over a BSC with crossover probability $\epsilon_o = 0.0720$.

Figure 4.3: Graphical representation of density evolution for $\mathcal{C}^\infty(8, 9)$, decoded by MB^0, MB^1, MB^2, MB^3 , and $\mathcal{H}_{ST}(MB^0, MB^1, MB^2, MB^3, \{\vec{\alpha}_o^{(\ell)}\}_{\ell=0}^\infty)$, with $\{\vec{\alpha}_o^{(\ell)}\}_{\ell=0}^\infty$ given in Example 4.1.3, over a BSC with crossover probability $\varepsilon_o = 0.0720$. (Switching-points are shown by $\star$.)

Given that all switchings between constituent algorithms occur immediately after the crossing-points of their corresponding curves are passed, one can see that an $\mathcal{H}_{ST}$ algorithm can be represented by a continuous curve. For such cases, the sequence of error probabilities is monotonic, and we can show, by an argument similar to the one in Chapter 3, that this sequence converges to zero with iterations iff the above-mentioned continuous curve is below the line $y = x$, in the interval of interest.

Example 4.1.4. *Consider the same scenario as in the previous example, but let $\Delta_{\circ}$ be specified by*

$$\vec{\alpha}_{\circ}^{(\ell)} = \begin{cases} (0, 0, 1, 0), & \text{if } \ell \leq 4, \\ (0, 1, 0, 0), & \text{if } 5 \leq \ell. \end{cases}$$

Here, $\Delta_{\circ}$ can be represented by a continuous curve. Fig. 4.4 depicts the curves representing $\Delta_{\circ}$ and its constituents for a channel with $\varepsilon_{\circ} = 0.0720$. The figure also portrays the evolution of the expected fraction of erroneous messages for $\Delta_{\circ}$, which converges monotonically to zero with iterations. ◀

The following is an example of how a hybrid algorithm can outpace its constituents in convergence.

Example 4.1.5. *Consider the same scenario as in the previous example, but let $\Delta_{\circ}$ be specified by*

$$\vec{\alpha}_{\circ}^{(\ell)} = \begin{cases} (0, 0, 1, 0), & \text{if } \ell \leq 3, \\ (0, 1, 0, 0), & \text{if } 4 \leq \ell. \end{cases}$$

Fig. 4.5 depicts the curves representing $\Delta_{\circ}$ and its constituents for a channel with $\varepsilon_{\circ} = 0.0700$, and also portrays the evolution of the expected fraction of incorrect messages for $\Delta_{\circ}$, which converges to zero with iterations. It is noteworthy that while $\Delta_{\circ}$ achieves an error-rate of 10^{-8} after iteration 10, MB^2 , its only constituent capable of decreasing error probability to zero, requires at least 36 iterations to achieve the same target error-rate. The reason for the slower convergence of MB^2 is that the

Figure 4.4: Graphical representation of density evolution for $\mathcal{C}^\infty(8,9)$, decoded by MB^0 , MB^1 , MB^2 , MB^3 , and $\mathcal{H}_{ST}(MB^0, MB^1, MB^2, MB^3, \{\vec{\alpha}_o^{(\ell)}\}_{\ell=0}^\infty)$, with $\{\vec{\alpha}_o^{(\ell)}\}_{\ell=0}^\infty$ given in Example 4.1.4, over a BSC with crossover probability $\varepsilon_o = 0.0720$. Note that Δ_o starts with MB^2 and switches to MB^1 in iteration 5.

tunnel between $f_{\varepsilon_0}^2(x)$ and $y = x$ is much narrower compared to the tunnel between $h_{\varepsilon_0}^{\{\bar{\alpha}_0^{(\ell)}\}_{\ell=0}^\infty}(x)$ and $y = x$. ◀

Before concluding this section, we provide a comparison between the algorithms in Examples 4.1.2 to 4.1.5, in terms of their speed of convergence. We have computed the required number of iterations for each algorithm to achieve a target error-rate of 10^{-8} . The results, which demonstrate faster convergence of the $\mathcal{H}_{ST}$ algorithms, are 29, 15, and 12 iterations for the $\mathcal{H}_{TI}$, the first $\mathcal{H}_{ST}$, and the second $\mathcal{H}_{ST}$, respectively.

4.2 Convergence Properties of Hybrid Algorithms

In this section we will study the convergence properties of hybrid algorithms. We start in Section 4.2.1 by defining the *maximum threshold* of MB hybrid algorithms. We then continue by investigating the convergence behavior of MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ algorithms. In Sections 4.2.2 and 4.2.3, we compare MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ algorithms, and show that MB $\mathcal{H}_{TI}$ algorithms are inferior, in terms of performance and convergence speed, to MB $\mathcal{H}_{ST}$ algorithms if perfect knowledge of the channel is available. However, in the absence of such knowledge, MB $\mathcal{H}_{TI}$ algorithms appear to be far less sensitive to channel conditions, and thus can be practically more attractive.

4.2.1 Maximum Threshold and Convergence Speed of Hybrid Algorithms

For a message-passing algorithm, as mentioned in Chapter 2, the supremum of all channel parameters for which the probability of error converges to zero with iterations is called threshold. In the following theorem, we will demonstrate the monotonicity of BSCs with respect to MB hybrid algorithms.

Figure 4.5: Graphical representation of density-evolution for $\mathcal{C}^\infty(8, 9)$, decoded by MB^0, MB^1, MB^2, MB^3 , and $\mathcal{H}_{ST}(MB^0, MB^1, MB^2, MB^3, \{\tilde{\alpha}_\circ^{(\ell)}\}_{\ell=0}^\infty)$, with $\{\tilde{\alpha}_\circ^{(\ell)}\}_{\ell=0}^\infty$ given in Example 4.1.5, over a BSC with crossover probability $\varepsilon_\circ = 0.0700$. Note that $\Delta_\circ$ starts with MB^2 and switches to MB^1 in iteration 4.

Theorem 4.2.1. *The convergence of error probability of an MB hybrid algorithm to zero for a channel parameter ε implies its convergence to zero for every channel parameter $\varepsilon' < \varepsilon$.*

Proof. Consider $\mathcal{C}^\infty(d_v, d_c)$, and suppose that this ensemble is decoded by $\Delta = \mathcal{H}(MB^0, MB^1, \dots, MB^{\lfloor d_v/2 \rfloor - 1}, \{\tilde{\alpha}^{(\ell)}\}_{\ell=0}^\infty)$ over two BSCs with crossover probabilities ε and $\varepsilon' < \varepsilon$. Let further $\{p_{\varepsilon, \varepsilon}^{(\ell)}\}_{\ell=0}^\infty$ and $\{p_{\varepsilon, \varepsilon'}^{(\ell)}\}_{\ell=0}^\infty$ denote the sequences of expected fraction of incorrect messages with iteration number for channel parameters ε and ε' , respectively, and also assume that in a given iteration ℓ_o , $p_{\varepsilon, \varepsilon'}^{(\ell_o)} \leq p_{\varepsilon, \varepsilon}^{(\ell_o)}$. Then

$$\begin{aligned}
 p_{\varepsilon, \varepsilon'}^{(\ell_o+1)} &\stackrel{(i)}{=} \sum_{\omega=0}^{\lfloor d_v/2 \rfloor - 1} \alpha_{\omega}^{(\ell_o+1)} \cdot f_{\varepsilon'}^{\omega} \left(p_{\varepsilon, \varepsilon'}^{(\ell_o)} \right) \\
 &\stackrel{(ii)}{\leq} \sum_{\omega=0}^{\lfloor d_v/2 \rfloor - 1} \alpha_{\omega}^{(\ell_o+1)} \cdot f_{\varepsilon}^{\omega} \left(p_{\varepsilon, \varepsilon'}^{(\ell_o)} \right) \\
 &\stackrel{(iii)}{\leq} \sum_{\omega=0}^{\lfloor d_v/2 \rfloor - 1} \alpha_{\omega}^{(\ell_o+1)} \cdot f_{\varepsilon}^{\omega} \left(p_{\varepsilon, \varepsilon}^{(\ell_o)} \right) \\
 &\stackrel{(iv)}{=} p_{\varepsilon, \varepsilon}^{(\ell_o+1)},
 \end{aligned} \tag{4.2.1}$$

where equalities (i) and (iv) follow from (3.1.2), (3.2.1), (4.1.2), and (4.1.3), and inequalities (ii) and (iii) are based on the fact that $f_{\varepsilon}^{\omega}(x)$ is increasing in ε and x , respectively. (See Lemmas 3.2.1 and 3.2.2.) As $p_{\varepsilon, \varepsilon'}^{(0)} = \varepsilon' < \varepsilon = p_{\varepsilon, \varepsilon}^{(0)}$, we can now use finite induction to deduce that $p_{\varepsilon, \varepsilon'}^{(\ell)} \leq p_{\varepsilon, \varepsilon}^{(\ell)}$, $\ell \geq 0$, and subsequently conclude that $\lim_{\ell \rightarrow \infty} p_{\varepsilon, \varepsilon'}^{(\ell)} = 0$ is implied by $\lim_{\ell \rightarrow \infty} p_{\varepsilon, \varepsilon}^{(\ell)} = 0$. $\square$

We can therefore infer that for all channel parameters less than the threshold of an MB hybrid algorithm, the expected fraction of erroneous messages passed in iteration ℓ approaches zero as ℓ tends to infinity, and for all channel parameters greater than the threshold, this fraction is bounded away from zero. For MB hybrid algorithms, the threshold can be maximized over all sequences $\{\tilde{\alpha}^{(\ell)}\}_{\ell=0}^\infty$; the *maximum threshold*

is denoted by $\varepsilon_{\{\text{MB } \mathcal{H}\}}^*$, and is obtained, in general, by exhaustive search. For MB $\mathcal{H}_{TI}$ algorithms, in particular, we can formulate the maximum threshold as

$$\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^* = \max_{\vec{\alpha}} \left\{ \sup \left\{ \varepsilon \in [0, 0.5] \mid h_{\varepsilon}^{\vec{\alpha}}(x) < x, \forall x \in (0, \varepsilon] \right\} \right\}. \quad (4.2.2)$$

In the following, we provide another characterization for $\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*$ which makes its computation slightly simpler. We will also elaborate on the convergence behavior of MB $\mathcal{H}_{TI}$ algorithms, and show that maximizing their convergence speed is equivalent to solving a non-linear optimization problem. For MB $\mathcal{H}_{ST}$ algorithms, formulation of the maximum threshold is slightly more complicated. However, as we will see later, there exists a straightforward method for obtaining the optimal *switching-sequence*. There, we will also discuss MB $\mathcal{H}_{ST}$ algorithms' speed of convergence, and see that the switching-sequence which minimizes the error probability in each iteration results in the fastest convergence.⁶

MB $\mathcal{H}_{TI}$ Algorithms

Consider $\mathcal{C}^\infty(d_v, d_c)$ and all the MB $\mathcal{H}_{TI}$ algorithms that can be used to decode this ensemble, either successfully or unsuccessfully, over a BSC with parameter ε . It is clear that the convergence of the error probability to zero with iterations for an MB $\mathcal{H}_{TI}$ algorithm depends on the corresponding PMF vector $\vec{\alpha}$, as well as the value of channel parameter ε . Any PMF vector $\vec{\alpha} = (\alpha_0, \alpha_1, \dots, \alpha_{\lfloor d_v/2 \rfloor - 1})$ represents a single point in the $(\lfloor d_v/2 \rfloor - 1)$ -dimensional hyperplane defined by $\sum_{\omega=0}^{\lfloor d_v/2 \rfloor - 1} \alpha_\omega = 1$. Depending on ε , the average probability of error converges to zero with iterations for some points in this hyperplane. We call the set of all such points *convergence domain* associated with ε , and denote it by $\mathfrak{R}_\varepsilon$. In the following, we show that $\mathfrak{R}_\varepsilon$ is *convex*.

Theorem 4.2.2. *Convergence domain of an MB $\mathcal{H}_{TI}$ algorithm is a convex set.*

⁶We leave some details on MB hybrid algorithms' speed of convergence for Chapter 5, where they will be discussed in a general framework.

Proof. To prove this theorem, we will show that the convergence domain of an MB $\mathcal{H}_{TI}$ algorithm associated with an arbitrary channel parameter, contains all the line segments connecting any pair of its points. Let Δ and Δ' be two MB $\mathcal{H}_{TI}$ algorithms specified by the PMF vectors $\vec{\alpha}$ and $\vec{\alpha}'$, and be represented by the curves $h_{\varepsilon}^{\vec{\alpha}}(x)$ and $h_{\varepsilon}^{\vec{\alpha}'}(x)$ for a given channel parameter ε , respectively. Consider now Δ''_{λ} , the MB $\mathcal{H}_{TI}$ algorithm specified by the vector $\vec{\alpha}''_{\lambda} = \lambda \cdot \vec{\alpha} + (1 - \lambda) \cdot \vec{\alpha}'$ for some non-negative $\lambda \leq 1$, and let this algorithm be represented by the curve $h_{\varepsilon}^{\vec{\alpha}''_{\lambda}}(x)$ for ε . Then

$$\begin{aligned}
 h_{\varepsilon}^{\vec{\alpha}''_{\lambda}}(x) &\stackrel{(i)}{=} \sum_{\omega=0}^{\lfloor d_v/2 \rfloor - 1} \alpha''_{\lambda, \omega} \cdot f_{\varepsilon}^{\omega}(x) \\
 &= \sum_{\omega=0}^{\lfloor d_v/2 \rfloor - 1} (\lambda \cdot \alpha_{\omega} + (1 - \lambda) \cdot \alpha'_{\omega}) \cdot f_{\varepsilon}^{\omega}(x) \\
 &= \lambda \cdot \sum_{\omega=0}^{\lfloor d_v/2 \rfloor - 1} \alpha_{\omega} \cdot f_{\varepsilon}^{\omega}(x) + (1 - \lambda) \cdot \sum_{\omega=0}^{\lfloor d_v/2 \rfloor - 1} \alpha'_{\omega} \cdot f_{\varepsilon}^{\omega}(x) \\
 &\stackrel{(ii)}{=} \lambda \cdot h_{\varepsilon}^{\vec{\alpha}}(x) + (1 - \lambda) \cdot h_{\varepsilon}^{\vec{\alpha}'}(x), \tag{4.2.3}
 \end{aligned}$$

where equalities (i) and (ii) follow from (4.1.5). Assume now that the probability of error for Δ and Δ' converges to zero for the channel parameter ε . This implies that $h_{\varepsilon}^{\vec{\alpha}}(x) < x \wedge h_{\varepsilon}^{\vec{\alpha}'}(x) < x, \forall x \in (0, \varepsilon]$. Using (4.2.3), we can therefore conclude that $h_{\varepsilon}^{\vec{\alpha}''_{\lambda}}(x) < x, \forall x \in (0, \varepsilon]$, from which the convergence of error probability to zero for Δ''_{λ} over a BSC with parameter ε immediately follows. The above argument is valid $\forall \lambda \in [0, 1]$, and $\forall \vec{\alpha}, \vec{\alpha}' \in \mathfrak{N}_{\varepsilon}$, which means $\mathfrak{N}_{\varepsilon}$ is convex. $\square$

It is also not difficult to see that $\mathfrak{N}_{\varepsilon} \subset \mathfrak{N}_{\varepsilon'}$ if $\varepsilon' < \varepsilon$, and thus by increasing ε , the convergence domain shrinks. In the limit, as ε tends to the maximum threshold $\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*$, the domain shrinks to a single-element set containing $\vec{\alpha}_{\{\text{MB } \mathcal{H}_{TI}\}}^*$.⁷

⁷This can be used to give another characterization for threshold as follows. $\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^* = \sup \{\varepsilon \in [0, 0.5] | \mathfrak{N}_{\varepsilon} \neq \emptyset\}$.

Example 4.2.1. Consider $\mathcal{C}^\infty(5, 10)$. The maximum threshold $\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*$ for this ensemble is equal to 0.0374, which is achieved using $\vec{\alpha}_{\{\text{MB } \mathcal{H}_{TI}\}}^* = (0.2589, 0.7411)$. (Note that the thresholds for constituent algorithms MB^0 and MB^1 are 0.0248 and 0.0277, respectively, both considerably smaller than $\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*$.) Figure 4.6 illustrates the convergence domains associated with channel parameters $\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*$, $0.9 \cdot \varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*$, and $0.8 \cdot \varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*$. Notice that the hyperplane and the convergence domains here are one-dimensional (line segments). Also, notice that $\mathbb{N}_{\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*} \subset \mathbb{N}_{0.9 \cdot \varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*} \subset \mathbb{N}_{0.8 \cdot \varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*}$. ◀

It is worth mentioning that if the objective is to maximize the algorithm's speed of convergence at a target error-rate ι for a channel parameter ε , the problem of finding the optimal vector $\vec{\alpha}_{\varepsilon, \iota}$, which does not necessarily belong to $\mathbb{N}_\varepsilon$, can be formulated as the minimization problem

- minimize $\ell \geq 0$,
- subject to $\alpha_\omega \geq 0$ and $\sum_{\omega=0}^{\lfloor d_v/2 \rfloor - 1} \alpha_\omega = 1$ and $\underbrace{h_\varepsilon^{\vec{\alpha}} \circ h_\varepsilon^{\vec{\alpha}} \circ \dots \circ h_\varepsilon^{\vec{\alpha}}(\varepsilon)}_{\ell \text{ times}} \leq \iota$.⁸

Example 4.2.2. Consider $\mathcal{C}^\infty(5, 10)$ and a target error probability of $\iota = 0.01$. For channel parameters $\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^* = 0.0374$, $0.9 \cdot \varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^* = 0.0336$, and $0.8 \cdot \varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^* = 0.0299$, we obtain $\vec{\alpha}_{0.0374, 0.01} = (0.2042, 0.7958)$, $\vec{\alpha}_{0.0336, 0.01} = (0.1821, 0.8179)$, and $\vec{\alpha}_{0.0299, 0.01} = (0.1137, 0.8863)$, respectively. The corresponding minimum numbers of iterations are 19, 8, and 5, respectively. It is interesting to note that $\vec{\alpha}_{0.0374, 0.01}$ does not belong to the convergence domain $\mathbb{N}_{0.0374} = \{(0.2589, 0.7411)\}$. ◀

⁸It is worth mentioning that the optimization problems here, highly resemble the optimization problems involved in designing irregular LDPC codes.

Figure 4.6: Convergence domains associated with channel parameters $\varepsilon_{\{MB \mathcal{H}_{TI}\}}^*$, $0.9 \cdot \varepsilon_{\{MB \mathcal{H}_{TI}\}}^*$, and $0.8 \cdot \varepsilon_{\{MB \mathcal{H}_{TI}\}}^*$, for $C^\infty(5, 10)$. Observe that $\mathcal{N}_{\varepsilon_{\{MB \mathcal{H}_{TI}\}}^*} \subset \mathcal{N}_{0.9 \cdot \varepsilon_{\{MB \mathcal{H}_{TI}\}}^*} \subset \mathcal{N}_{0.8 \cdot \varepsilon_{\{MB \mathcal{H}_{TI}\}}^*}$.

MB $\mathcal{H}_{ST}$ Algorithms

To study the convergence properties of MB $\mathcal{H}_{ST}$ algorithms, we first review the results on partial ordering of MB algorithms' f curves, derived in Chapter 3. For $\mathcal{C}^\infty(d_v, d_c)$, and for a BSC with parameter ε , we showed that

$$\frac{1-\varepsilon}{\varepsilon} \leq \left(\frac{1+(1-2 \cdot x)^{d_c-1}}{1-(1-2 \cdot x)^{d_c-1}} \right)^{2 \cdot (\lceil d_v/2 \rceil + \omega) - d_v + 1} \iff f_\varepsilon^\omega(x) \leq f_\varepsilon^{\omega+1}(x), \omega < \lfloor d_v/2 \rfloor - 1, \quad (4.2.4)$$

and also

$$f_\varepsilon^\omega(x_\circ) < f_\varepsilon^{\omega+1}(x_\circ) \implies f_\varepsilon^{\omega'}(x) < f_\varepsilon^{\omega''}(x), \forall \omega', \omega'' | \omega \leq \omega' < \omega'' \wedge \forall x \leq x_\circ, \quad (4.2.5)$$

and

$$f_\varepsilon^{\omega-1}(x_\circ) > f_\varepsilon^\omega(x_\circ) \implies f_\varepsilon^{\omega''}(x) > f_\varepsilon^{\omega'}(x), \forall \omega', \omega'' | \omega \geq \omega' > \omega'' \wedge \forall x \geq x_\circ. \quad (4.2.6)$$

As mentioned in Chapter 3, for any ε , there exists a neighborhood of $x = 0$ in which $f_\varepsilon^0(x) < f_\varepsilon^1(x) < \dots < f_\varepsilon^{\lfloor d_v/2 \rfloor - 1}(x)$, and similarly, for any ε , we have $f_\varepsilon^0(x) > f_\varepsilon^1(x) > \dots > f_\varepsilon^{\lfloor d_v/2 \rfloor - 1}(x)$ in the close vicinity of $x = 0.5$. By carefully examining the left-hand side of (4.2.4), one will notice that the same inequality is used in (2.7.7) to find the optimal choice of $\mathcal{G}_B$'s parameter b_ℓ in order to minimize the probability of error. Based on (4.2.4) and (4.2.5), choosing the smallest b_ℓ satisfying (2.7.7) is equivalent to switching in each iteration to the MB algorithm that has the minimum $f_\varepsilon^\omega(p_e^{(\ell-1)})$, where $p_e^{(\ell-1)}$ represents the average error probability in the preceding iteration. Using our terminology, therefore, $\mathcal{G}_B$ is an MB $\mathcal{H}_{ST}$ algorithm which switches, in each iteration, to the constituent MB algorithm whose curve is below all the other curves. This switching-sequence, which maximizes the speed of convergence, is denoted by $\{\bar{\alpha}_\varepsilon^{\mathcal{G}_B(\ell)}\}_{\ell=0}^\infty$ hereafter. $\mathcal{G}_B$ can then be represented by the curve

$$h_\varepsilon^{\{\bar{\alpha}_\varepsilon^{\mathcal{G}_B(\ell)}\}_{\ell=0}^\infty}(x) = \min_{\omega} \{f_\varepsilon^\omega(x), 0 \leq \omega \leq \lfloor d_v/2 \rfloor - 1\}. \quad (4.2.7)$$

As the curve for $\mathcal{G}_B$ is always the minimum of the curves for MB algorithms, one can see that this algorithm achieves the maximum threshold amongst all MB $\mathcal{H}_{ST}$ algorithms [51]. In the following, we prove the stronger result that $\mathcal{G}_B$ is optimal in minimizing the probability of error, and maximizing the threshold amongst *all* MB hybrid algorithms.

Theorem 4.2.3. *$\mathcal{G}_B$ has the largest threshold amongst all MB hybrid algorithms.*

Proof. Consider $\mathcal{C}^\infty(d_v, d_c)$, and suppose that over BSCs, the maximum threshold $\varepsilon_{\{\text{MB } \mathcal{H}\}}^*$ for this ensemble is achieved by an MB hybrid algorithm Δ , specified by $\{\tilde{\alpha}^{(\ell)}\}_{\ell=0}^\infty$. Let $\{p_e^{(\ell)}\}_{\ell=0}^\infty$ and $\{p_{e, \mathcal{G}_B}^{(\ell)}\}_{\ell=0}^\infty$ denote the sequences of expected fraction of erroneous messages with iteration number, corresponding to Δ and $\mathcal{G}_B$, respectively, for a channel parameter $\varepsilon < \varepsilon_{\{\text{MB } \mathcal{H}\}}^*$. Let further Ω denote the order of MB algorithm to which we switch in iteration $\ell_o + 1$, according to $\mathcal{G}_B$, and assume that in a given iteration ℓ_o , $p_{e, \mathcal{G}_B}^{(\ell_o)} \leq p_e^{(\ell_o)}$. Then

$$\begin{aligned}
 p_{e, \mathcal{G}_B}^{(\ell_o+1)} &\stackrel{\text{(i)}}{=} f_\varepsilon^\Omega \left(p_{e, \mathcal{G}_B}^{(\ell_o)} \right) \\
 &\stackrel{\text{(ii)}}{\leq} \sum_{\omega=0}^{\lfloor d_v/2 \rfloor - 1} \alpha_\omega^{(\ell_o+1)} \cdot f_\varepsilon^\omega \left(p_{e, \mathcal{G}_B}^{(\ell_o)} \right) \\
 &\stackrel{\text{(iii)}}{\leq} \sum_{\omega=0}^{\lfloor d_v/2 \rfloor - 1} \alpha_\omega^{(\ell_o+1)} \cdot f_\varepsilon^\omega \left(p_e^{(\ell_o)} \right) \\
 &\stackrel{\text{(iv)}}{=} p_e^{(\ell_o+1)},
 \end{aligned} \tag{4.2.8}$$

where equalities (i) and (iv) follow from (3.1.2) and (3.2.1), inequality (ii) is resulted from $f_\varepsilon^\Omega(p_{e, \mathcal{G}_B}^{(\ell_o)}) \stackrel{(4.2.7)}{=} \min_{\omega} \left\{ f_\varepsilon^\omega(p_{e, \mathcal{G}_B}^{(\ell_o)}), 0 \leq \omega \leq \lfloor d_v/2 \rfloor - 1 \right\}$, and inequality (iii) is based on the fact that $f_\varepsilon^\omega(x)$ is increasing in x . (See Lemma 3.2.2.) Now as $p_{e, \mathcal{G}_B}^{(0)} = p_e^{(0)} = \varepsilon$, we can conclude by finite induction that $p_{e, \mathcal{G}_B}^{(\ell)} \leq p_e^{(\ell)}$, $\ell \geq 0$ and thus $\lim_{\ell \rightarrow \infty} p_{e, \mathcal{G}_B}^{(\ell)} \leq \lim_{\ell \rightarrow \infty} p_e^{(\ell)}$. But $\lim_{\ell \rightarrow \infty} p_e^{(\ell)} = 0, \forall \varepsilon < \varepsilon_{\{\text{MB } \mathcal{H}\}}^*$. Therefore, $\lim_{\ell \rightarrow \infty} p_{e, \mathcal{G}_B}^{(\ell)} = 0, \forall \varepsilon < \varepsilon_{\{\text{MB } \mathcal{H}\}}^*$.

$\varepsilon_{\{\text{MB } \mathcal{H}\}}^*$, which is equivalent to saying that the maximum threshold is also achieved by $\mathcal{G}_B$. $\square$

To find the maximum threshold of MB $\mathcal{H}_{ST}$ algorithms, we, therefore, look for the supremum value of ε for which the probability of error for $\mathcal{G}_B$ goes to zero with iterations, or equivalently

$$\varepsilon_{\{\text{MB } \mathcal{H}_{ST}\}}^* = \sup \left\{ \varepsilon \in [0, 0.5] \mid h_{\varepsilon}^{\{\tilde{\alpha}_{\varepsilon}^{\mathcal{G}_B(\ell)}\}_{\ell=0}^{\infty}}(x) < x, \forall x \in (0, \varepsilon] \right\}. \quad (4.2.9)$$

We let $\{\tilde{\alpha}_{\{\text{MB } \mathcal{H}_{ST}\}}^{*(\ell)}\}_{\ell=0}^{\infty}$ denote the switching-sequence corresponding to the above maximization.

Example 4.2.3. Consider $\mathcal{C}^{\infty}(6, 7)$ and the three MB algorithms that can be used to decode this ensemble over a BSC, namely, MB^0 , MB^1 , and MB^2 , and let the channel parameter be $\varepsilon_{\circ} = 0.0825$. Consider further $\Delta_{\circ}$, the MB $\mathcal{H}_{ST}$ algorithm which is specified by

$$\tilde{\alpha}_{\varepsilon_{\circ}}^{\mathcal{G}_B(\ell)} = \begin{cases} (0, 0, 1), & \text{if } \ell \leq 2, \\ (0, 1, 0), & \text{if } 3 \leq \ell \leq 9, \\ (1, 0, 0), & \text{if } 10 \leq \ell. \end{cases}$$

This switching-sequence, which corresponds to $\mathcal{G}_B$ and has been obtained from (2.7.7), results in the fastest convergence of error probability to zero for $\varepsilon_{\circ}$. Portrayed in Figure 4.7 are the curves representing these algorithms, as well as the evolution of the expected fraction of incorrect messages for $\Delta_{\circ}$. It can be seen that error probability for $\Delta_{\circ}$ converges to zero very fast; for instance, our calculations show that an average error probability of 10^{-8} is achieved only after iteration 12. Note that for every x greater than the rightmost switching-point we have $f_{0.0825}^2(x) < f_{0.0825}^1(x) < f_{0.0825}^0(x)$. This ordering changes to $f_{0.0825}^1(x) < f_{0.0825}^2(x) < f_{0.0825}^0(x)$, and subsequently to $f_{0.0825}^1(x) < f_{0.0825}^0(x) < f_{0.0825}^2(x)$, between the two switching-points, and finally to $f_{0.0825}^0(x) < f_{0.0825}^1(x) < f_{0.0825}^2(x)$, in the vicinity of $x = 0$.

For this ensemble, $\varepsilon_{\{\text{MB } \mathcal{H}_{ST}\}}^* = 0.0858$ and is achieved by starting with MB^2 , and switching to MB^1 and MB^0 in iterations 6 and 24, respectively. ◀

4.2.2 Performance Comparison of MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ Algorithms (Maximum Threshold and Finite-Length Simulation)

Table 4.1 lists the threshold values of MB algorithms, and the maximum threshold values of MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ algorithms for several (d_v, d_c) pairs. The corresponding hybridization sequences are given in Tables 4.2 and 4.3. More specifically, Table 4.2 lists $\vec{\alpha}_{\{\text{MB } \mathcal{H}_{TI}\}}^*$ vectors,⁹ and Table 4.3 contains information about $\{\vec{\alpha}_{\{\text{MB } \mathcal{H}_{ST}\}}^*\}_{\ell=0}^\infty$ vectors. The last row of Table 4.2, for instance, is interpreted as follows. In order to achieve the maximum threshold, in each iteration, 45.09% of the variable nodes should operate under MB^1 , and the remaining 54.91% should treat the messages according to MB^2 . Likewise, the last row of Table 4.3, for instance, is interpreted as follows. In order to achieve the maximum threshold, we should start with MB^2 in iteration 1, and switch to MB^1 and MB^0 in iterations 9 and 24, respectively. As can be seen, both hybrid algorithms have significantly higher (maximum) thresholds compared to their constituents,¹⁰ which is an indication of their superior performance.

In the following examples, we also present some simulation results comparing the performance of MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ algorithms to that of their constituents, for LDPC codes of finite lengths.

⁹Notice that $\vec{\alpha}_{\{\text{MB } \mathcal{H}_{TI}\}}^* = (\alpha_{\{\text{MB } \mathcal{H}_{TI}\},0}^*, \alpha_{\{\text{MB } \mathcal{H}_{TI}\},1}^*, \dots, \alpha_{\{\text{MB } \mathcal{H}_{TI}\},\lfloor d_v/2 \rfloor - 1}^*)$.

¹⁰One can see that compared to the highest thresholds of constituent algorithms, improvements of up to 39.74% for MB $\mathcal{H}_{TI}$ algorithms, and up to 53.84% for MB $\mathcal{H}_{ST}$ algorithms are obtained.

Figure 4.7: Graphical representation of density evolution for $C^\infty(6,7)$, decoded by MB^0 , MB^1 , MB^2 , and $\mathcal{H}_{ST}(MB^0, MB^1, MB^2, \{\vec{\alpha}_{\epsilon_0}^{G_B(\ell)}\}_{\ell=0}^\infty)$, with $\{\vec{\alpha}_{\epsilon_0}^{G_B(\ell)}\}_{\ell=0}^\infty$ given in Example 4.2.3, over a BSC with crossover probability $\epsilon_0 = 0.0825$. Note that Δ_0 starts with MB^2 , and switches to MB^1 and MB^0 after iterations 2 and 9, respectively.

Table 4.1: Threshold values for MB algorithms, and maximum threshold values for MB $\mathcal{H}_{ST}$ and $\mathcal{H}_{TI}$ algorithms

(d_v, d_c)	$\varepsilon_{MB^0}^*$	$\varepsilon_{MB^1}^*$	$\varepsilon_{MB^2}^*$	$\varepsilon_{MB^3}^*$	$\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*$	$\varepsilon_{\{\text{MB } \mathcal{H}_{ST}\}}^*$
(4, 5)	0.0260	0.0833	–	–	0.0947	0.0963
(4, 6)	0.0159	0.0666	–	–	0.0724	0.0747
(4, 7)	0.0107	0.0555	–	–	0.0586	0.0611
(4, 8)	0.0077	0.0476	–	–	0.0491	0.0516
(4, 9)	0.0058	0.0416	–	–	0.0423	0.0447
(4, 10)	0.0045	0.0370	–	–	0.0371	0.0378
(5, 6)	0.0650	0.0500	–	–	0.0822	0.0865
(5, 7)	0.0482	0.0416	–	–	0.0640	0.0686
(5, 8)	0.0374	0.0357	–	–	0.0520	0.0566
(5, 9)	0.0301	0.0312	–	–	0.0436	0.0480
(5, 10)	0.0248	0.0277	–	–	0.0374	0.0416
(6, 7)	0.0306	0.0806	0.0333	–	0.0851	0.0858
(6, 8)	0.0236	0.0648	0.0285	–	0.0691	0.0704
(6, 9)	0.0188	0.0536	0.0250	–	0.0576	0.0593
(6, 10)	0.0155	0.0453	0.0222	–	0.0491	0.0510
(7, 8)	0.0494	0.0854	0.0238	–	0.0858	0.0858
(7, 9)	0.0405	0.0717	0.0208	–	0.0724	0.0724
(7, 10)	0.0339	0.0615	0.0185	–	0.0623	0.0624
(8, 9)	0.0296	0.0601	0.0719	0.0178	0.0756	0.0756
(8, 10)	0.0247	0.0511	0.0640	0.0158	0.0664	0.0664
(9, 10)	0.0409	0.0653	0.0552	0.0138	0.0701	0.0702

Table 4.2: Optimal blending vectors $\bar{\alpha}_{\{\text{MB } \mathcal{H}_{TI}\}}^*$ for MB $\mathcal{H}_{TI}$ algorithms

(d_v, d_c)	$\alpha_{\{\text{MB } \mathcal{H}_{TI}\},0}^*$	$\alpha_{\{\text{MB } \mathcal{H}_{TI}\},1}^*$	$\alpha_{\{\text{MB } \mathcal{H}_{TI}\},2}^*$	$\alpha_{\{\text{MB } \mathcal{H}_{TI}\},3}^*$
(4, 5)	0.1205	0.8795	—	—
(4, 6)	0.0799	0.9201	—	—
(4, 7)	0.0522	0.9478	—	—
(4, 8)	0.0312	0.9688	—	—
(4, 9)	0.0158	0.9842	—	—
(4, 10)	0.0031	0.9969	—	—
(5, 6)	0.3925	0.6075	—	—
(5, 7)	0.3499	0.6501	—	—
(5, 8)	0.3146	0.6854	—	—
(5, 9)	0.2847	0.7153	—	—
(5, 10)	0.2592	0.7408	—	—
(6, 7)	0.0003	0.6114	0.3883	—
(6, 8)	0.0000	0.5869	0.4131	—
(6, 9)	0.0002	0.5665	0.4333	—
(6, 10)	0.0002	0.5482	0.4516	—
(7, 8)	0.0000	0.9265	0.0735	—
(7, 9)	0.0001	0.8542	0.1457	—
(7, 10)	0.0005	0.8039	0.1956	—
(8, 9)	0.0000	0.1588	0.8412	0.0000
(8, 10)	0.0000	0.1269	0.8731	0.0000
(9, 10)	0.0000	0.4509	0.5491	0.0000

Table 4.3: Optimal switching-sequences for MB $\mathcal{H}_{ST}$ algorithms

(d_v, d_c)	MB^3	MB^2	MB^1	MB^0
(4, 5)	–	–	1	58
(4, 6)	–	–	1	54
(4, 7)	–	–	1	98
(4, 8)	–	–	1	79
(4, 9)	–	–	1	107
(4, 10)	–	–	1	52
(5, 6)	–	–	1	18
(5, 7)	–	–	1	14
(5, 8)	–	–	1	15
(5, 9)	–	–	1	15
(5, 10)	–	–	1	17
(6, 7)	–	1	6	24
(6, 8)	–	1	8	27
(6, 9)	–	1	8	23
(6, 10)	–	1	7	20
(7, 8)	–	1	5	87
(7, 9)	–	1	4	38
(7, 10)	–	1	5	33
(8, 9)	–	1	20	30
(8, 10)	–	1	24	34
(9, 10)	–	1	9	24

Example 4.2.4. Depicted in Figure 4.8, are BER curves for a randomly constructed $(8, 9)$ -regular LDPC code of length 9×10^3 , with no cycle of length 4, decoded by MB^0 , MB^1 , MB^2 , MB^3 , and $\Delta_{TI}^* = \mathcal{H}_{TI}(MB^0, MB^1, MB^2, MB^3, \vec{\alpha}_{\{MB \mathcal{H}_{TI}\}}^*)$ and $\Delta_{ST}^* = \mathcal{H}_{ST}(MB^0, MB^1, MB^2, MB^3, \{\vec{\alpha}_{\{MB \mathcal{H}_{ST}\}}^{(\ell)}\}_{\ell=0}^\infty)$, where $\vec{\alpha}_{\{MB \mathcal{H}_{TI}\}}^*$ is equal to $(0, 0.15885, 0.84115, 0)$, and $\{\vec{\alpha}_{\{MB \mathcal{H}_{ST}\}}^{(\ell)}\}_{\ell=0}^\infty$ is specified by

$$\vec{\alpha}_{\{MB \mathcal{H}_{ST}\}}^{(\ell)} = \begin{cases} (0, 0, 1, 0), & \text{if } \ell \leq 19, \\ (0, 1, 0, 0), & \text{if } 20 \leq \ell \leq 29, \\ (1, 0, 0, 0), & \text{if } 30 \leq \ell. \end{cases}$$

For all simulation results, the maximum number of iterations is set to 500, and for each point, enough number of codewords are simulated to generate at least 200 codeword errors.¹¹ The threshold values for MB algorithms are $\varepsilon_{MB^0}^* = 0.0296$, $\varepsilon_{MB^1}^* = 0.0601$, $\varepsilon_{MB^2}^* = 0.0719$, and $\varepsilon_{MB^3}^* = 0.0178$, and the maximum threshold values for MB hybrid algorithms, $\varepsilon_{\{MB \mathcal{H}_{ST}\}}^*$ and $\varepsilon_{\{MB \mathcal{H}_{TI}\}}^*$, are both equal to 0.0756. The figure shows that the waterfall regions of BER curves associated with MB hybrid algorithms, as well as those associated with MB^0 , MB^1 , and MB^2 , are sharp and occur at close proximity to the theoretical threshold for these algorithms. For MB^3 , however, the waterfall region is not as sharp, but still starts at a point close to $\varepsilon_{MB^3}^*$. The figure also shows that Δ_{TI}^* slightly outperforms Δ_{ST}^* in spite of both algorithms having the same maximum threshold. This suggests that for the MB $\mathcal{H}_{ST}$ algorithm, the convergence of the code performance to cycle-free case by increasing the block length, predicted by concentration results [45], happens slower compared to the MB $\mathcal{H}_{TI}$ algorithm, which can be considered a drawback of the former algorithm. It is worth mentioning that in our simulations, except for MB^2 , we did not observe any early error floor for the algorithms. At $BER = 10^{-5}$, the average numbers of iterations for converged cases are about 3.9, 5.4, 12.7, 5.3, 15.7, and 17.5, for MB^0 .

¹¹At very low BERs, due to limitation in computational resources, we reduced 200 to 10 for 16 out of 99 points reported here.

MB^1 , MB^2 , MB^3 , Δ_{TI}^* , and Δ_{ST}^* , respectively. Again, Δ_{TI}^* is slightly superior to Δ_{ST}^* . (These results should be interpreted with caution, as these algorithms operate over different ranges of channel parameters. It is not fair, for instance, to judge the convergence speed of MB^3 against that of Δ_{ST}^* solely based on the average number of iterations each algorithm requires to achieve a BER of 10^{-5} , as the latter algorithm achieves the target BER over a channel which is more than eight-times worse than that of the former algorithm ($\varepsilon_{\Delta_{ST}^*} \approx 0.064$ compared to $\varepsilon_{MB^3} \approx 0.008$). Clearly, the better-performing algorithm requires considerably smaller number of iterations over a significantly better channel, such as the one with $\varepsilon = 0.008$. The comparison is fair only if the algorithms operate over more or less the same range of channel parameters, such as the case for Δ_{TI}^* and Δ_{ST}^* .) ◀

Example 4.2.5. Figure 4.9 depicts BER curves for a randomly constructed $(5, 8)$ -regular LDPC code of rate $\frac{3}{8}$ and length 8×10^3 , with no cycle of length 4, decoded by MB^0 , MB^1 , $\Delta_{TI}^* = \mathcal{H}_{TI}(MB^0, MB^1, \bar{\alpha}_{\{MB \mathcal{H}_{TI}\}}^*)$, where $\bar{\alpha}_{\{MB \mathcal{H}_{TI}\}}^* = (0.3146, 0.6854)$, and $\Delta_{ST}^* = \mathcal{H}_{ST}(MB^0, MB^1, \{\bar{\alpha}_{\{MB \mathcal{H}_{ST}\}}^{*(\ell)}\}_{\ell=0}^\infty)$, where $\{\bar{\alpha}_{\{MB \mathcal{H}_{ST}\}}^{*(\ell)}\}_{\ell=0}^\infty$ is specified by

$$\bar{\alpha}_{\{MB \mathcal{H}_{ST}\}}^{*(\ell)} = \begin{cases} (0, 1), & \text{if } \ell \leq 14, \\ (1, 0), & \text{if } 15 \leq \ell. \end{cases}$$

(Similar to the previous example, the maximum number of iterations is 500, and the number of generated codeword errors for each point is 200.) The threshold values for MB algorithms are $\varepsilon_{MB^0}^* = 0.0374$, and $\varepsilon_{MB^1}^* = 0.0357$, and the maximum threshold values for MB hybrid algorithms are $\varepsilon_{\{MB \mathcal{H}_{TI}\}}^* = 0.0520$, and $\varepsilon_{\{MB \mathcal{H}_{ST}\}}^* = 0.0566$. The figure shows that the sharp waterfall regions of BER curves associated with MB hybrid algorithms and MB^0 , occur close to the corresponding threshold values. For MB^1 , however, the waterfall region is not nearly as sharp (but still starts at a point close to $\varepsilon_{MB^1}^*$). The figure also shows that the difference between the error performances

Figure 4.8: BER curves for a randomly constructed $(8,9)$ -regular LDPC code of length 9×10^3 , with no cycle of length 4, decoded by MB^0 , MB^1 , MB^2 , MB^3 , $\Delta_{TI}^* = \mathcal{H}_{TI}(MB^0, MB^1, MB^2, MB^3, \tilde{\alpha}_{\{MB \mathcal{H}_{TI}\}}^*)$ (marked by $\triangleright$), and $\Delta_{ST}^* = \mathcal{H}_{ST}(MB^0, MB^1, MB^2, MB^3, \{\tilde{\alpha}_{\{MB \mathcal{H}_{ST}\}}^{(\ell)}\}_{\ell=0}^\infty)$ (marked by $\triangleleft$).

corresponding to Δ_{TI}^* and Δ_{ST}^* is slightly less than the difference between $\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*$ and $\varepsilon_{\{\text{MB } \mathcal{H}_{ST}\}}^*$. This, again, suggests that for the MB $\mathcal{H}_{ST}$ algorithm, the convergence of the code performance to cycle-free case, happens slower compared to the MB $\mathcal{H}_{TI}$ algorithm. (We did not observe any early error floor for these algorithms.) ◀

4.2.3 Comparison Between MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ Algorithms in the Absence of Perfect Knowledge of the Channel

Comparison of $\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*$ and $\varepsilon_{\{\text{MB } \mathcal{H}_{ST}\}}^*$ values in Table 4.1 shows that MB $\mathcal{H}_{TI}$ algorithms are slightly inferior, in terms of performance, to their switch-type counterparts. At finite lengths, however, as demonstrated in Example 4.2.4, the trend can change with $\mathcal{H}_{TI}$ algorithms slightly outperforming $\mathcal{H}_{ST}$ algorithms. Moreover, in the following, we show that $\mathcal{H}_{TI}$ algorithms are more robust and can better cope with unknown or changing channel conditions. The following examples illustrate these points.

Example 4.2.6 (Unknown Channel Parameter). Consider $\mathcal{C}^\infty(4, 5)$, and the two MB algorithms that can be used to decode this ensemble over a BSC, namely MB^0 and MB^1 . For this ensemble, the maximum thresholds for MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ algorithms are $\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^* = 0.0947$ and $\varepsilon_{\{\text{MB } \mathcal{H}_{ST}\}}^* = 0.0963$, respectively. (The former is obtained by using vector $\vec{\alpha}_{\{\text{MB } \mathcal{H}_{TI}\}}^* = (0.1208, 0.8792)$, while the latter is achieved by starting with MB^1 , and switching to MB^0 after iteration 57.) Assuming that the channel parameter is unknown, and our main concern is to ensure the convergence of MB $\mathcal{H}_{TI}$ or $\mathcal{H}_{ST}$ algorithms for as wide a range of channel parameters as possible, we set both algorithms to achieve their maximum thresholds.¹² Figure 4.10 portrays error-rate

¹²As is shown in Theorem 4.2.1, the convergence of error probability of an MB hybrid algorithm to zero for a given channel parameter ε implies its convergence to zero for every parameter $\varepsilon' < \varepsilon$. By setting these algorithms to achieve their maximum thresholds, we indeed guarantee their convergence for all channel parameters below the maximum thresholds.

Figure 4.9: BER curves for a randomly constructed $(5, 8)$ -regular LDPC code of length 8×10^3 , decoded by MB^0 , MB^1 , $\mathcal{H}_{Tl}(MB^0, MB^1, \tilde{\alpha}_{\{MB \mathcal{H}_{Tl}\}}^*)$ (marked by $\triangleright$), and $\mathcal{H}_{ST}(MB^0, MB^1, \{\tilde{\alpha}_{\{MB \mathcal{H}_{ST}\}}^{(\ell)}\}_{\ell=0}^\infty)$ (marked by $\triangleleft$).

curves versus iteration number, for the two algorithms and for several channel parameters. One can see that although the convergence of MB $\mathcal{H}_{ST}$ algorithms is faster than that of MB $\mathcal{H}_{TI}$ algorithms when the channel parameter is close to the maximum threshold, the latter can outpace the former for a wide range of target error probabilities, and for a wide range of channel parameters less than the maximum threshold. In order to explain this result, we have depicted in Figure 4.11 the curves representing MB^0 , MB^1 , and $\Delta_{TI}^* = \mathcal{H}_{TI}(MB^0, MB^1, \vec{\alpha}_{\{MB \mathcal{H}_{TI}\}}^*)$ for a channel with $\varepsilon_o = 0.8 \cdot \varepsilon_{\{MB \mathcal{H}_{ST}\}}^*$. We have also portrayed the evolution of the average error probability for $\Delta_{ST}^* = \mathcal{H}_{ST}(MB^0, MB^1, \{\vec{\alpha}_{\{MB \mathcal{H}_{ST}\}}^{(\ell)}\}_{\ell=0}^\infty)$ over the same channel. As can be seen, because of the non-optimality of our setting for this channel parameter, the MB $\mathcal{H}_{ST}$ algorithm fails to switch to the right constituent algorithm at the right time; it switches from MB^1 to MB^0 in iteration 58, instead of iteration 7, and therefore is outpaced by the MB $\mathcal{H}_{TI}$ algorithm for a rather wide range of target error probabilities.

It is worth mentioning that for a target error probability of, say, $\iota_o = 10^{-4}$, the optimal PMF vector for the fastest convergence of MB $\mathcal{H}_{TI}$ algorithms is $\vec{\alpha}_{\varepsilon_o, \iota_o} = (0.1558, 0.8442)$. The curve $h_{\varepsilon_o}^{II}(x)$ corresponding to this vector is also plotted in Figure 4.11. As can be seen, this curve is rather close to the curve for $\Delta_{TI}^*(h_{\varepsilon_o}^I(x))$. This explains the fast convergence of Δ_{TI}^* even though this hybrid algorithm has not been specifically designed for the fastest convergence over this channel. ◀

Example 4.2.7 (Unknown Channel Parameter). Consider $\mathcal{C}^\infty(8, 9)$, and the four MB algorithms that can be used to decode this ensemble over a BSC, namely MB^0 , MB^1 , MB^2 , and MB^3 . For this ensemble, $\varepsilon_{\{MB \mathcal{H}_{TI}\}}^* = \varepsilon_{\{MB \mathcal{H}_{ST}\}}^* = 0.0756$, and is achieved, for the MB $\mathcal{H}_{TI}$ algorithm, by using vector $\vec{\alpha}_{\{MB \mathcal{H}_{TI}\}}^* = (0, 0.15885, 0.84115, 0)$, and for the MB $\mathcal{H}_{ST}$ algorithm, by starting with MB^2 , and switching to MB^1 and MB^0 , after 19 and 29 iterations. Figure 4.12 depicts error-rate curves versus iteration number, for the two hybrid algorithms, and for several channel parameters. As can be

Figure 4.10: Comparison between the convergence speed of MB $\mathcal{H}_{Tl}$ and $\mathcal{H}_{ST}$ algorithms, for $\mathcal{C}^\infty(4, 5)$, and for $\varepsilon = \Gamma \cdot \varepsilon_{\{\text{MB } \mathcal{H}_{ST}\}}^*$ ($\varepsilon_{\{\text{MB } \mathcal{H}_{ST}\}}^* = 0.0963$), $\Gamma = 0.25, 0.50, 0.75, 0.80, 0.90$, and 1.00 (left to right). (Both algorithms are set to achieve their maximum thresholds.)

Figure 4.11: Graphical representation of density-evolution for $\mathcal{C}^\infty(4, 5)$, decoded by MB^0 , MB^1 , $\Delta_{TI}^* = \mathcal{H}_{TI}(MB^0, MB^1, \bar{\alpha}_{\{MB \mathcal{H}_{TI}\}}^*)$, and $\Delta_{ST}^* = \mathcal{H}_{ST}(MB^0, MB^1, \{\bar{\alpha}_{\{MB \mathcal{H}_{ST}\}}^{*(\ell)}\}_{\ell=0}^\infty)$, for $\epsilon_0 = 0.8 \cdot \epsilon_{\{MB \mathcal{H}_{ST}\}}^* = 0.0770$. Note that Δ_{ST}^* starts with MB^1 , and switches to MB^0 in iterations 58. (The optimum switching time is in iteration 7: the dotted line shows the corresponding evolution of the average error probability for the optimal switching.)

seen, again, the MB $\mathcal{H}_{TI}$ algorithm outpaces the MB $\mathcal{H}_{ST}$ algorithm for all parameters less than, say, about 95% of ε^* . ◀

Our extended experiences with different ensembles, indicate that the aforementioned phenomenon is typical of MB hybrid algorithms.¹³

Example 4.2.8 (Error in Estimating Channel Parameter). *Consider again $\mathcal{C}^\infty(4, 5)$, MB^0 , and MB^1 , and suppose that while the actual channel parameter is $\varepsilon_{act} = 0.0915$, due to changes in channel condition, or estimation errors, we underestimate the channel parameter, and set MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ algorithms for their fastest convergence at target error-rate of 10^{-6} over a channel with estimated parameter $\varepsilon_{est} = 0.0900$. (For MB $\mathcal{H}_{TI}$ algorithms, we have $\bar{\alpha}_{0.0900, 10^{-6}} = (0.1353, 0.8647)$ and $\bar{\alpha}_{0.0915, 10^{-6}} = (0.1310, 0.8690)$. For MB $\mathcal{H}_{ST}$ algorithms, as mentioned earlier, $\mathcal{G}_B$ provides the optimal switching-sequence. Here, the fastest convergence is achieved by starting with MB^1 , and switching to MB^0 after iteration 11 for $\varepsilon = 0.0900$, and after iteration 13 for $\varepsilon = 0.0915$.) The two algorithms are compared in Figure 4.13. (The subscripts and the superscripts of the curve labels represent ε_{act} and ε_{est} , respectively.) One can see that while underestimation of the channel parameter results in the failure of the MB $\mathcal{H}_{ST}$ algorithm to converge (see $ST_{0.0915}^{0.0900}$ in this figure, and also see Figure 4.14), it has a small effect on the performance of the MB $\mathcal{H}_{TI}$ algorithm. (Compare $TI_{0.0915}^{0.0900}$ with $TI_{0.0915}^{0.0915}$, and observe that the number of iterations required to achieve an error-rate of 10^{-6} , increases by less than 15% ($281 \rightarrow 320$) due to the underestimation of the channel parameter.) Also, comparing $TI_{0.0915}^{0.0947}$ ($\varepsilon_{\{MB \mathcal{H}_{TI}\}}^* = 0.0947$) with $TI_{0.0915}^{0.0915}$, shows that the performance of the MB $\mathcal{H}_{TI}$ algorithm designed to achieve the maximum threshold $\varepsilon_{\{MB \mathcal{H}_{TI}\}}^*$, on a channel with parameter $\varepsilon = 0.0915$, is only slightly inferior to that of the MB $\mathcal{H}_{TI}$ algorithm particularly optimized for this channel (and for target error probability of 10^{-6}), requiring*

¹³Note that for the finite-length simulation results presented earlier, the settings are optimal for maximum thresholds. This situation is much the same as the one in Examples 4.2.6 and 4.2.7.

Figure 4.12: Comparison between the convergence speed of MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ algorithms, for $\mathcal{C}^\infty(8,9)$, and for $\varepsilon = \Gamma \cdot \varepsilon^*$ ($\varepsilon^* = 0.0756$), $\Gamma = 0.25, 0.50, 0.75, 0.90, 0.95$, and 1.00 (left to right). (Both algorithms are set to achieve their maximum thresholds.)

less than 15% increase in the number of iterations at 10^{-6} ($281 \rightarrow 318$). This is in contrast to the case for MB $\mathcal{H}_{ST}$ algorithms: The comparison between $ST_{0.0915}^{0.0963}$ ($\varepsilon_{\{\text{MB } \mathcal{H}_{ST}\}}^* = 0.0963$) and $ST_{0.0915}^{0.0915}$ indicates an increase of nearly 180% in the number of iterations required to achieve an error-rate of 10^{-6} ($22 \rightarrow 62$). ◀

Example 4.2.9 (Error in Estimating Channel Parameter). Consider $\mathcal{C}^n(8, 9)$ again, and suppose that while $\varepsilon_{\text{act}} = 0.0720$, we set MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ algorithms for their fastest convergence over a channel with (estimated) parameter $\varepsilon_{\text{est}} = 0.0705$. Figure 4.15 compares the two algorithms. As can be seen, similar to the case in the previous example, underestimation of the channel parameter results in the failure of the MB $\mathcal{H}_{ST}$ algorithm to converge (see $ST_{0.0720}^{0.0705}$). It hardly affects, however, the performance of the MB $\mathcal{H}_{TI}$ algorithm. (Compare $TI_{0.0720}^{0.0705}$ with $TI_{0.0720}^{0.0720}$.) In addition, comparing $TI_{0.0720}^{0.0756}$ with $TI_{0.0720}^{0.0720}$, shows that the performance of the MB $\mathcal{H}_{TI}$ algorithm designed to achieve the maximum threshold $\varepsilon^* = 0.0756$, on a channel with $\varepsilon = 0.0720$ is almost as good as that of the MB $\mathcal{H}_{TI}$ algorithm particularly optimized for this channel. This, however, is not the case for the MB $\mathcal{H}_{ST}$ algorithm, as the comparison between $ST_{0.0720}^{0.0756}$ and $ST_{0.0720}^{0.0720}$ indicates. ◀

Examples 4.2.6 to 4.2.9 show that, unlike MB $\mathcal{H}_{ST}$ algorithms, the optimal design for MB $\mathcal{H}_{TI}$ algorithms is universal, in the sense that the MB $\mathcal{H}_{TI}$ algorithm designed for $\varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*$, performs close to optimal, in terms of convergence speed, on a wide range of channels with $\varepsilon < \varepsilon_{\{\text{MB } \mathcal{H}_{TI}\}}^*$. In the following example, we examine the effect of underestimated channel parameter at finite block lengths.

Example 4.2.10. Assuming that the channel parameter is $\varepsilon_{\text{est}} = 0.0900$, we use the optimized hybridization sequences obtained in Example 4.2.8 for the fastest convergence of MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ algorithms, and simulate the performance of the algorithms over a channel with $\varepsilon_{\text{act}} = 0.0915$ for a randomly constructed $(4, 5)$ -regular LDPC code

Figure 4.13: Comparison between the convergence speed of MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ algorithms, for $\mathcal{C}^\infty(4, 5)$. Subscripts and superscripts of the curve labels represent the actual and the estimated channel parameters, respectively $(TI_{\varepsilon_{act}}^{est}, ST_{\varepsilon_{act}}^{est})$. [Refer to Example 4.2.8.]

Figure 4.14: (a) Evolution of the average error probability for $\mathcal{C}^\infty(4, 5)$, decoded by an MB $\mathcal{H}_{ST}$ algorithm optimized for an estimated channel parameter $\varepsilon_{est} = 0.0900$, over a channel with actual parameter $\varepsilon_{act} = 0.0915$; (b) Improper switching results in divergence: the dotted line shows the evolution of the average error probability when the algorithm is optimized for the channel parameter $\varepsilon = 0.0915$.

Figure 4.15: Comparison between the convergence speed of MB $\mathcal{H}_{Tl}$ and $\mathcal{H}_{ST}$ algorithms, for $\mathcal{C}^\infty(8, 9)$. Subscripts and superscripts of the curve labels represent the actual and the estimated channel parameters, respectively ($Tl_{\varepsilon_{act}}^{\varepsilon_{est}}$, $ST_{\varepsilon_{act}}^{\varepsilon_{est}}$). [Refer to Example 4.2.9.]

of length 10^4 , with no cycle of length 4. We set the maximum number of iterations to 500, and continue the simulation until 200 codeword errors are generated. For the MB $\mathcal{H}_{TI}$ algorithm, BER and word error-rate (WER) are 4.51×10^{-3} and 1.03×10^{-1} , respectively. The corresponding figures for the MB MB $\mathcal{H}_{ST}$ algorithm are 1.73×10^{-1} and 4.96×10^{-1} , which are much worse than those of the MB $\mathcal{H}_{TI}$ algorithm. ◀

4.3 Conclusion

A general framework for hybrid decoding algorithms was introduced. Using density evolution, and by focusing on MB hybrid algorithms, we showed the great improvement in threshold and speed of convergence that hybrid algorithms could provide compared to their constituents. We also investigated convergence properties of these algorithms in detail, and provided simulation results at finite block lengths. Our results indicate that although MB $\mathcal{H}_{ST}$ algorithms outperform MB $\mathcal{H}_{TI}$ algorithms asymptotically (at very large block lengths), and when the channel parameter is perfectly known, the trend changes at finite block lengths, or when a perfect knowledge of the channel is not available. This suggests that MB $\mathcal{H}_{TI}$ algorithms could be practically more attractive than MB $\mathcal{H}_{ST}$ algorithms. Although in this chapter we have only considered regular LDPC codes, (many of) our results can be extended to irregular codes, as seen in the next chapter.

*“The secret of ugliness consists not in irregularity, but in being
uninteresting.”*

– Ralph Waldo Emerson

Chapter 5

Hybrid Decoding of Irregular LDPC Codes

In Chapter 4, restricting our attention to regular LDPC codes, and to MB algorithms, we proposed $\mathcal{H}_{TI}$ algorithms. We also demonstrated that for regular LDPC codes, MB $\mathcal{H}_{TI}$ algorithms are much more robust against the changes of the channel parameter compared to $\mathcal{G}_B$, which is, in fact, the optimal $\mathcal{H}_{ST}$ algorithm, in both performance and convergence speed, with MB constituents.

On the other hand, it is well known that introducing irregularity into LDPC codes' structure can improve their performance, compared to their regular counterparts. Due to this fact, we are naturally motivated to study the performance of MB $\mathcal{H}_{TI}$ algorithms in decoding irregular codes. In fact, we expect to gain more by jointly optimizing the node degrees in LDPC codes' structure, and the constituent algorithms' hybridization sequences.

In this chapter, we first extend the application of MB $\mathcal{H}_{TI}$ algorithms to irregular codes. We then find optimal degree distributions for irregular LDPC codes decoded by MB $\mathcal{H}_{TI}$ algorithms. As part of this process, for each variable-node degree, we obtain optimal partitioning of variable-nodes of that degree according to MB algorithms of different orders associated to that degree. Finally, we compare MB $\mathcal{H}_{TI}$ algorithms with $\mathcal{G}_A$ and $\mathcal{G}_B$, applied to optimized irregular LDPC codes. Our results demonstrate

the superiority of MB $\mathcal{H}_{TI}$ algorithms over $\mathcal{G}_A$ in terms of noise threshold. We also show that in the absence of the perfect knowledge of the channel, MB $\mathcal{H}_{TI}$ algorithms can beat $\mathcal{G}_B$ in terms of decoding time (convergence speed). Moreover, our study indicates that, unlike the case for $\mathcal{G}_B$, the optimal design of MB $\mathcal{H}_{TI}$ algorithms is universal in the sense that the algorithm designed for optimal threshold also performs very well in terms of convergence speed over a wide range of channel parameters. Throughout this chapter, we follow [44–46] for basic definitions.

5.1 MB $\mathcal{H}_{TI}$ Algorithms for Decoding Irregular LDPC Codes

This section is organized as follows. Using density evolution, we first analyze MB $\mathcal{H}_{TI}$ algorithms applied to irregular LDPC code ensembles. We then provide characterizations for, and give tight upper bounds on, the thresholds of MB $\mathcal{H}_{TI}$ algorithms. At the end, we investigate the speed of convergence of (irregular LDPC code ensembles') error probability to zero, under MB $\mathcal{H}_{TI}$ algorithms.

5.1.1 Density Evolution

An irregular LDPC code is generally characterized by a *degree distribution* vector pair $(\vec{\lambda}, \vec{\rho})$, where the l^{th} element of $\vec{\lambda}$ (respectively, $\vec{\rho}$), denotes the fraction of edges in the code's Tanner graph that are adjacent to *variable-nodes* (respectively, *check-nodes*) of degree l [44–46].¹ Following the common practice in the literature on LDPC codes, we also consider ensembles of codes, rather than single instances of codes, throughout this paper. We denote by $\mathcal{C}^n(\vec{\lambda}, \vec{\rho})$ the ensemble of irregular LDPC codes of length n associated to $(\vec{\lambda}, \vec{\rho})$.

¹We assume that the first element of both vectors equals zero, i.e., $\lambda_1 = \rho_1 = 0$.

For a fixed check-node degree distribution vector $\vec{\rho}$ and a given *concentrated* variable-node degree distribution vector $\vec{\lambda}_j$ (whose j^{th} element is 1), similar to the case for regular codes, there exist $\lfloor j/2 \rfloor$ different MB algorithms which can be used to decode $\mathcal{C}^n(\vec{\lambda}_j, \vec{\rho})$ over a BSC. In particular, $MB^{j,\omega,\vec{\rho}}$ denotes the MB algorithm of order ω , $0 \leq \omega \leq \lfloor j/2 \rfloor - 1$.

Defining $\rho(x) \equiv \sum_{i \geq 2} \rho_i \cdot x^{i-1}$, it can be shown that for a given channel crossover probability ε , and under $MB^{j,\omega,\vec{\rho}}$, the recursion

$$p_{\varepsilon,\varepsilon}^{(\ell \geq 1)} = p_{\varepsilon,\varepsilon}^{(0)} \cdot \left(1 - f_+^{j,\omega,\vec{\rho}}(p_{\varepsilon,\varepsilon}^{(\ell-1)})\right) + \left(1 - p_{\varepsilon,\varepsilon}^{(0)}\right) \cdot f_-^{j,\omega,\vec{\rho}}(p_{\varepsilon,\varepsilon}^{(\ell-1)}), \quad (5.1.1)$$

where

$$f_{\pm}^{j,\omega,\vec{\rho}}(x) \equiv \sum_{k=\lfloor j/2 \rfloor + \omega}^{j-1} \binom{j-1}{k} \cdot \left(\frac{1 \pm \rho(1-2 \cdot x)}{2}\right)^k \cdot \left(\frac{1 \mp \rho(1-2 \cdot x)}{2}\right)^{j-1-k}, \quad (5.1.2)$$

describes the evolution of the average fraction of erroneous (variable to check) messages with iteration number ℓ for $\mathcal{C}^\infty(\vec{\lambda}_j, \vec{\rho})$, if initiated by $p_{\varepsilon,\varepsilon}^{(0)} = \varepsilon$. This recursion can be represented by the curve

$$f_{\varepsilon}^{j,\omega,\vec{\rho}}(x) = \varepsilon \cdot \left(1 - f_+^{j,\omega,\vec{\rho}}(x)\right) + (1 - \varepsilon) \cdot f_-^{j,\omega,\vec{\rho}}(x). \quad (5.1.3)$$

By arguments similar to those in Chapter 3, where $\mathcal{C}^\infty(\vec{\lambda}_j, \vec{\rho}_i)$ was dealt with, it can be proven that the curve $f_{\varepsilon}^{j,\omega,\vec{\rho}}(x)$ is increasing in both ε and x . (See Lemmas 3.2.1 and 3.2.2.)

In [70], the performance of code ensembles with non-concentrated variable-node degree distributions under $\mathcal{G}_A$ is investigated. For $\mathcal{G}_A$, the variable-nodes of degree j operate solely under $MB^{j,\lfloor j/2 \rfloor - 1, \vec{\rho}}$. Here, we study a more general case where different variable-nodes of the same degree are allowed to operate under different orders of MB algorithms in a time-invariant hybrid framework. Consider variable-nodes of degree $j \geq 2$, to which a fraction λ_j of edges are connected. Suppose that a fraction $\frac{\Lambda_{j,\omega}}{\lambda_j}$ of these degree- j nodes process messages according to $MB^{j,\omega,\vec{\rho}}$, $0 \leq \omega \leq \lfloor j/2 \rfloor - 1$.

Define vector $\vec{\Lambda}$ as the ordered sequence of $\Lambda_{j,\omega}$ values in j , where for each j , ω sweeps values between 0 and $\lfloor j/2 \rfloor - 1$.² This vector, together with $\vec{\rho}$, not only characterizes the corresponding irregular code ensemble, but also has the corresponding decoding algorithm embedded in it.

One can see that for a given channel crossover probability ε and under the MB $\mathcal{H}_{TI}$ algorithm specified by $\vec{\Lambda}$ and $\vec{\rho}$, the recursion

$$p_{e,\varepsilon}^{(\ell \geq 1)} = \sum_{j \geq 2} \sum_{\omega=0}^{\lfloor j/2 \rfloor - 1} \Lambda_{j,\omega} \cdot \left(p_{e,\varepsilon}^{(0)} \cdot \left(1 - f_{+}^{j,\omega,\vec{\rho}} \left(p_{e,\varepsilon}^{(\ell-1)} \right) \right) + \left(1 - p_{e,\varepsilon}^{(0)} \right) \cdot f_{-}^{j,\omega,\vec{\rho}} \left(p_{e,\varepsilon}^{(\ell-1)} \right) \right), \quad (5.1.4)$$

gives the expected fraction of incorrect messages with iterations for $\mathcal{C}^{\infty}(\vec{\Lambda}, \vec{\rho})$, if initiated by $p_{e,\varepsilon}^{(0)} = \varepsilon$. Recursion (5.1.4) can be represented by the curve

$$h_{\varepsilon}^{\vec{\Lambda}, \vec{\rho}}(x) = \sum_{j \geq 2} \sum_{\omega=0}^{\lfloor j/2 \rfloor - 1} \Lambda_{j,\omega} \cdot f_{\varepsilon}^{j,\omega,\vec{\rho}}(x), \quad (5.1.5)$$

which is a linear combination of $f_{\varepsilon}^{j,\omega,\vec{\rho}}(x)$ curves given in (5.1.3), and thus is increasing in both ε and x .

Arguments similar to those in Chapter 3 can be used to prove that the sequence $\beta_{\varepsilon} = \{p_{e,\varepsilon}^{(\ell)}\}_{\ell=0}^{\infty}$ is strictly decreasing, constant, or strictly increasing iff $h_{\varepsilon}^{\vec{\Lambda}, \vec{\rho}}(\varepsilon)$ is less than, equal to, or greater than ε , respectively. Furthermore, it can be shown that β_{ε} converges to the first encountered crossing-point of the curve $y = h_{\varepsilon}^{\vec{\Lambda}, \vec{\rho}}(x)$ and the line $y = x$. Crossing-points are fixed-points of recursion (5.1.4). As $h_{\varepsilon}^{\vec{\Lambda}, \vec{\rho}}(0) = 0$, $\forall \varepsilon$, $\forall \vec{\Lambda}$, $\forall \vec{\rho}$, $x = 0$ is always a fixed-point of the recursion, and we can conclude that

²Note that $\vec{\Lambda}(1) = 0$, and for $l \geq 2$, $\vec{\Lambda}(l)$ denotes the fraction of edges that are adjacent to those degree- $j(l)$ variable-nodes which process the messages in accordance with the MB algorithm of order $\omega(l)$, where $j(l)$ is the minimum between the smallest even and odd integers whose squares are greater than or equal to $4 \cdot l - 4$ and $4 \cdot l - 3$, respectively, and $\omega(l) = l - \lfloor (j-1)/2 \rfloor \cdot \lceil (j-1)/2 \rceil - 2$. (E.g., if $l = 7$, we obtain $j = 5$ and $\omega = 1$.)

Theorem 5.1.1. β_ε is a decreasing sequence which converges to the fixed-point zero iff $h_\varepsilon^{\vec{\Lambda}, \vec{\rho}}(x) < x, \forall x \in (0, \varepsilon]$.

(The formal proof is similar to that of Theorem 3.2.1 in Chapter 3 and thus is omitted.)

5.1.2 Threshold Characterization and Bounds

Using the fact that $h_\varepsilon^{\vec{\Lambda}, \vec{\rho}}(x)$ in (5.1.5) is increasing in both ε and x , we can show that the convergence of an MB $\mathcal{H}_{TI}$ algorithm's error probability to zero for a channel parameter ε implies its convergence to zero for every channel parameter $\varepsilon' < \varepsilon$. (Refer to Appendix D.) For a given pair $(\vec{\Lambda}, \vec{\rho})$, the threshold $\varepsilon^*(\vec{\Lambda}, \vec{\rho})$ is defined as the supremum of all values of ε for which the sequence β_ε converges to zero, or equivalently

$$\varepsilon^*(\vec{\Lambda}, \vec{\rho}) = \sup \left\{ \varepsilon \geq 0 \mid h_\varepsilon^{\vec{\Lambda}, \vec{\rho}}(x) < x, \forall x \in (0, \varepsilon) \right\}. \quad (5.1.6)$$

In Appendix E, we show that the supremum value of ε satisfying the inequality $\frac{\partial}{\partial x} h_\varepsilon^{\vec{\Lambda}, \vec{\rho}}(0) < 1$, which is, in fact, the stability condition for the fixed-point zero, is given by

$$\gamma(\vec{\Lambda}, \vec{\rho}) = R \left(\frac{1 - \Lambda_{2,0} \cdot \rho'(1)}{\rho'(1) \cdot \sum_{j \geq 3} \Lambda_{j, \lfloor j/2 \rfloor - 1} \cdot (j-1)} \right), \quad (5.1.7)$$

where $R(\cdot)$ is the unit ramp function, defined by $R(x) = x$, for $x > 0$; and $= 0$, otherwise. Using this result, and letting $\delta(\vec{\Lambda}, \vec{\rho})$ denote the smallest root of $h_x^{\vec{\Lambda}, \vec{\rho}}(x) - x = 0$ in $(0, 0.5)$, we can deduce that

Theorem 5.1.2 (Upper Bounds on Threshold).

$$\varepsilon^*(\vec{\Lambda}, \vec{\rho}) \leq \min \left\{ \delta(\vec{\Lambda}, \vec{\rho}), \gamma(\vec{\Lambda}, \vec{\rho}) \right\}. \quad (5.1.8)$$

It can be seen that this bound, which is similar to the bound derived in [70] for the special case of $\omega = \lfloor j/2 \rfloor - 1$, is tight for many ensembles of irregular LDPC

codes. (E.g., for the rate-1/2 ensemble specified by $\Lambda_{4,1} = 0.8203$, $\Lambda_{7,1} = 0.1652$, $\Lambda_{19,1} = 0.0145$, $\rho_8 = 0.2600$, and $\rho_9 = 0.7400$, we have $\varepsilon^*(\vec{\Lambda}, \vec{\rho}) = \delta(\vec{\Lambda}, \vec{\rho}) = \gamma(\vec{\Lambda}, \vec{\rho}) = 0.0525$.)

5.1.3 Asymptotic Speed of Convergence

In Chapter 3, it is shown that under $MB^{j,\omega}$, the asymptotic (as $\ell \rightarrow \infty$) convergence of $\mathcal{C}^\infty(\vec{\lambda}_j, \vec{\rho}_i)$'s probability of error to zero for a channel parameter $\varepsilon < \varepsilon^*(\vec{\lambda}_j, \omega, \vec{\rho}_i)$ is exponential or super-exponential, in ℓ , for $\omega = \lfloor j/2 \rfloor - 1$ or $\omega < \lfloor j/2 \rfloor - 1$, respectively. It is not difficult to see that this result is extendable to code ensembles with concentrated variable-node and non-concentrated check-node degree distributions. In fact, by considering $\mathcal{C}^\infty(\lambda_j, \vec{\rho})$ and taking into account only the most dominant term in the expansion of (5.1.3) for sufficiently small values of x , it can be shown that

$$f_\varepsilon^{j,\omega,\vec{\rho}}(x) \approx \tilde{\varepsilon} \cdot \binom{j-1}{\lfloor j/2 \rfloor - \omega} \cdot (\rho'(1) \cdot x)^{\lfloor j/2 \rfloor - \omega}, \quad (5.1.9)$$

where $\tilde{\varepsilon} = \varepsilon$, if $\omega \neq 0$, or j is odd; and $= 1$, otherwise. One can then deduce from the application of $p_{e,\varepsilon}^{(\ell)} = f_\varepsilon^{j,\omega,\vec{\rho}}(p_{e,\varepsilon}^{(\ell-1)})$, $\ell \geq 1$, recursively, with the initial condition $p_{e,\varepsilon}^{(0)} = \varepsilon$, that for sufficiently small values of ε , and with $\eta \equiv \tilde{\varepsilon} \cdot \binom{j-1}{\lfloor j/2 \rfloor - \omega} \cdot \rho'(1)^{\lfloor j/2 \rfloor - \omega}$, $p_{e,\varepsilon}^{(\ell)}$ is $\mathcal{O}(\eta^\ell)$, if $\omega = \lfloor j/2 \rfloor - 1$; and $\mathcal{O}\left((\eta^{1/(\lfloor j/2 \rfloor - \omega - 1)} \cdot \varepsilon)^{(\lfloor j/2 \rfloor - \omega)^\ell}\right)$, if $\omega < \lfloor j/2 \rfloor - 1$; as $\ell \rightarrow \infty$. In other words, $p_{e,\varepsilon}^{(\ell)}$ reduces exponentially, or super-exponentially with parameter $\lfloor j/2 \rfloor - \omega$, in ℓ , respectively. This is of course with the conditions that ε is less than $\varepsilon^*(\vec{\lambda}_j, \omega, \vec{\rho})$ and is also small enough to satisfy $\eta \cdot \varepsilon^{\lfloor j/2 \rfloor - \omega - 1} < 1$. The latter condition, however, can be relaxed by noticing the fact that $\varepsilon < \varepsilon^*(\vec{\lambda}_j, \omega, \vec{\rho}) \Rightarrow \lim_{\ell \rightarrow \infty} p_e^{(\ell)} = 0$, which implies that if ℓ is large enough, we will be in close enough vicinity of $x = 0$ on the curve $f_\varepsilon^{j,\omega,\vec{\rho}}$ such that the probability of error goes to zero with the same exponents as mentioned earlier.³ These results are summarized in the

³Note that $\varepsilon^*(\vec{\lambda}_2, 0, \vec{\rho}) = 0$. Under $MB^{2,0,\vec{\rho}}$, therefore, the error probability does not converge to zero for $\varepsilon > 0$.

following theorem.

Theorem 5.1.3. *Under $\text{MB}^{j,\omega,\vec{\rho}}$ operating on $\mathcal{C}^\infty(\lambda_j, \vec{\rho})$, the asymptotic convergence of $p_e^{(\ell)}$ to zero is exponential or super-exponential, in ℓ , for $\omega = \lfloor j/2 \rfloor - 1$ or $\omega < \lfloor j/2 \rfloor - 1$, respectively, provided that $\varepsilon < \varepsilon^*(\vec{\lambda}_j, \omega, \vec{\rho})$.*

These results can be extended to MB $\mathcal{H}_{TI}$ algorithms. In fact, it is not difficult to infer from (5.1.5) that the convergence of error probability to zero for an MB $\mathcal{H}_{TI}$ algorithm is *at least* as fast as that of its slowest converging constituent algorithm. The next proposition, therefore, immediately follows.

Proposition 5.1.1. *The asymptotic convergence of error probability to zero for the MB $\mathcal{H}_{TI}$ algorithm operating on $\mathcal{C}^\infty(\vec{\Lambda}, \vec{\rho})$ is at least exponential in ℓ , provided that $\varepsilon < \varepsilon^*(\vec{\Lambda}, \vec{\rho})$.*

To see how fast precisely the convergence of the MB $\mathcal{H}_{TI}$ algorithm associated with $\mathcal{C}^\infty(\vec{\Lambda}, \vec{\rho})$ is, assuming that $\varepsilon < \varepsilon^*(\vec{\Lambda}, \vec{\rho})$, we shall first compute

$$\theta \equiv \min_{j,\omega:\Lambda_{j,\omega}>0} \{ \lfloor j/2 \rfloor - \omega \}. \quad (5.1.10)$$

The convergence of $p_e^{(\ell)}$ to zero is then at least exponential in ℓ , if $\theta = 1$; and at least super-exponential with parameter θ in ℓ , if $\theta > 1$.

5.2 Optimal Degree Distribution Pairs

For a given distribution pair $(\vec{\Lambda}, \vec{\rho})$, the fraction of edges that are adjacent to variable-nodes of degree j is $\sum_\omega \Lambda_{j,\omega}$. The *design rate* associated with $(\vec{\Lambda}, \vec{\rho})$ is then

$$r(\vec{\Lambda}, \vec{\rho}) = 1 - \frac{\sum_{i \geq 2} \rho_i \cdot \frac{1}{i}}{\sum_{j \geq 2} \sum_{\omega=0}^{\lfloor j/2 \rfloor - 1} \Lambda_{j,\omega} \cdot \frac{1}{j}}. \quad (5.2.1)$$

Notice that

$$\overline{d_v} \equiv \left(\sum_{j=2} \sum_{\omega=0}^{\lfloor j/2 \rfloor - 1} \Lambda_{j,\omega} \cdot \frac{1}{j} \right)^{-1}, \quad (5.2.2)$$

and

$$\overline{d_c} \equiv \left(\sum_i \rho_i \cdot \frac{1}{i} \right)^{-1}, \quad (5.2.3)$$

are the average variable-node and check-node degrees, respectively. Since different degree distributions result in different rates and thresholds, it is natural to search for those that maximize the threshold for a given rate, or maximize the rate for a given channel parameter, under the condition that average error probability tends to zero asymptotically. We take the second approach in this work. Letting the maximum variable-node and check-node degrees be denoted by D_v and D_c , respectively, the search can be formulated as the optimization problem

- minimize $\frac{\sum_{i=2}^{D_c} \rho_i \cdot \frac{1}{i}}{\sum_{j=2}^{D_v} \sum_{\omega=0}^{\lfloor j/2 \rfloor - 1} \Lambda_{j,\omega} \cdot \frac{1}{j}},$
- subject to
 - $\rho_i \geq 0$, $2 \leq i \leq D_c$, and $\sum_{i=2}^{D_c} \rho_i = 1$,
 - $\Lambda_{j,\omega} \geq 0$, $2 \leq j \leq D_v$, $0 \leq \omega \leq \lfloor j/2 \rfloor - 1$, and $\sum_{j=2}^{D_v} \sum_{\omega=0}^{\lfloor j/2 \rfloor - 1} \Lambda_{j,\omega} = 1$,
 - $\sum_{j=2}^{D_v} \sum_{\omega=0}^{\lfloor j/2 \rfloor - 1} \Lambda_{j,\omega} \cdot f_{\varepsilon}^{j,\omega,\vec{\rho}}(x) < x$, $\forall x \in (0, \varepsilon]$.

There are many methods for solving such optimization problems. In this work, we have simply used `fseminf` M-file of MATLAB's Optimization Toolbox. (This M-file finds the global minimum of a constrained multi-variable non-linear scalar function.)

Table 5.1 provides maximum achievable rates under MB $\mathcal{H}_{TI}$ algorithms for several channel parameters, restricting the search to concentrated check-node degree distributions (i.e., $\vec{\rho} = \vec{\rho}_i$). Better results may be obtained by lifting this restriction

Table 5.1: Maximum achievable rates and corresponding degree distribution pairs under MB $\mathcal{H}_{TI}$ algorithms

ε	r	$\vec{\Lambda}$	$\vec{\rho}$
0.0000	1.0000	—	—
0.0250	0.6875	$\Lambda_{4,1}=0.9601,$ $\Lambda_{5,1}=0.0259,$ $\Lambda_{13,1}=0.0081,$ $\Lambda_{15,1}=0.0039,$ $\Lambda_{19,1}=0.0003,$ $\Lambda_{21,1}=0.0017^a$	$\rho_{13}=1.0000$
0.0500	0.5164	$\Lambda_{4,1}=0.8329,$ $\Lambda_{7,1}=0.1408,$ $\Lambda_{17,1}=0.0058,$ $\Lambda_{19,1}=0.0205^b$	$\rho_9=1.0000$
0.0750	0.3701	$\Lambda_{3,0}=0.2150,$ $\Lambda_{4,1}=0.7454,$ $\Lambda_{6,1}=0.0396^b$	$\rho_6=1.0000$
0.1000	0.2642	$\Lambda_{3,0}=0.3674,$ $\Lambda_{4,1}=0.5878,$ $\Lambda_{18,1}=0.0179,$ $\Lambda_{19,1}=0.0269^b$	$\rho_5=1.0000$
0.1250	0.1906	$\Lambda_{3,0}=0.7689,$ $\Lambda_{4,1}=0.2044,$ $\Lambda_{18,1}=0.0267^b$	$\rho_4=1.0000$
0.1500	0.1135	$\Lambda_{3,0}=0.6405,$ $\Lambda_{4,1}=0.2514,$ $\Lambda_{19,1}=0.1081^b$	$\rho_4=1.0000$
0.1750	0.0687	$\Lambda_{2,0}=0.1476,$ $\Lambda_{3,0}=0.8524^b$	$\rho_3=1.0000$
0.2000	0.0368	$\Lambda_{2,0}=0.0765,$ $\Lambda_{3,0}=0.9235^b$	$\rho_3=1.0000$

^a $D_v = 27$ ^b $D_v = 19$

and increasing the values of D_v and D_c . It should be noted, however, that increasing D_v or D_c by 1, increases the search space dimension by $\lfloor D_v/2 \rfloor$ or 1, respectively.

5.3 Comparisons with Optimized $\mathcal{G}_A$ and $\mathcal{G}_B$

In Chapter 4, the superiority of MB $\mathcal{H}_{TI}$ algorithms over simple MB algorithms (e.g., $\mathcal{G}_A$) and MB $\mathcal{H}_{ST}$ algorithms (e.g., $\mathcal{G}_B$), in terms of error performance and robustness against channel variations, respectively, was shown. In that chapter, the focus was on regular LDPC codes. In this section, we show that MB $\mathcal{H}_{TI}$ algorithms possess similar advantages for irregular codes: they can outperform $\mathcal{G}_A$ (applied to optimally designed irregular LDPC codes), and can outpace $\mathcal{G}_B$ (optimally designed for irregular LDPC codes) when no perfect a priori knowledge of the channel condition is in hand.

In the sequel, we demonstrate the strength of MB $\mathcal{H}_{TI}$ algorithms by providing two examples. In the first example, we obtain a pair of degree distributions whose threshold under MB $\mathcal{H}_{TI}$ algorithms is greater than the best achievable threshold under $\mathcal{G}_A$ for the same rate.⁴ In the second example, which is more elaborate, we investigate how the lack of perfect knowledge of the channel affects the performance of MB $\mathcal{H}_{TI}$ algorithms compared to MB $\mathcal{H}_{ST}$ algorithms (in particular, compared to $\mathcal{G}_B$), and examine the pros and cons of each category of algorithms. In both examples, the codes are of rate 1/2. Results and trends similar to those reported in this section are observed for other code rates as well.

Example 5.3.1. *It is shown in [70] that amongst rate-1/2 LDPC codes, the best performance under $\mathcal{G}_A$ belongs to $\mathcal{C}^\infty(\tilde{\lambda}_{\mathcal{G}_A}, \tilde{\rho}_{\mathcal{G}_A})$, where $\tilde{\lambda}_{\mathcal{G}_A} = (0, 0, 0.1115, 0.8885)$*

⁴Notice that this result is not surprising, as $\mathcal{G}_A$ is a special case of MB $\mathcal{H}_{TI}$ algorithms where degree- j variable-nodes operate solely under the MB algorithm of order $\lfloor j/2 \rfloor - 1$. For instance, the MB $\mathcal{H}_{TI}$ algorithms specified by the last two degree distribution pairs in Table 5.1, are, in fact, optimized $\mathcal{G}_A$ with the constraint of concentrated check-node degree distribution.

and $\vec{\rho}_{\mathcal{G}_A} = (0, 0, 0, 0, 0, 0, 0.2600, 0.7400)$. For this ensemble, we have $\varepsilon^*(\vec{\lambda}_{\mathcal{G}_A}, \vec{\rho}_{\mathcal{G}_A}) = 0.0513$, $\overline{d_v} \approx 3.85$, and $\overline{d_c} \approx 7.7$. Using MB $\mathcal{H}_{TI}$ algorithms, however, better rates with asymptotical zero error probability can be achieved for $\varepsilon = 0.0513$. (We can, e.g., attain a rate of $r = 0.5076 > 1/2$ with the ensemble specified by $\Lambda_{4,1} = 0.8363$, $\Lambda_{7,1} = 0.1637$, $\rho_8 = 0.2400$, and $\rho_9 = 0.7600$.) This is a clear indication of the possibility of improving the threshold for rate-1/2 codes under MB $\mathcal{H}_{TI}$ algorithms. To find the optimal threshold based on our optimization tool for the rate, we do as follows. We choose a channel parameter ε' slightly greater than $\varepsilon = 0.0513$, for which we maximize the rate. If the new rate is still greater than 1/2, we (incrementally) increase ε' and repeat the rate maximization step. Otherwise, ε' is (incrementally) decreased and rate is maximized. By iterating this rather naïve procedure, we will finally end up with a rate-1/2 pair of distributions whose threshold is greater than $\varepsilon = 0.0513$.

Using the above-described method under the constraints of $D_v = 19$ and $D_c = 9$, we find that the best rate-1/2 ensemble is specified by $\Lambda_{\circ 4,1} = 0.8203$, $\Lambda_{\circ 7,1} = 0.1652$, $\Lambda_{\circ 19,1} = 0.0145$, $\rho_{\circ 8} = 0.2600$, and $\rho_{\circ 9} = 0.7400$. For this ensemble, we have $\varepsilon^*(\vec{\Lambda}_{\circ}, \vec{\rho}_{\circ}) = 0.0525 > 0.0513$, $\overline{d_v} \approx 4.35$, and $\overline{d_c} \approx 8.7$. Notice that the average variable-node and check-node degrees are slightly greater than, but still comparable to, those of $\mathcal{C}^{\infty}(\vec{\lambda}_{\mathcal{G}_A}, \vec{\rho}_{\mathcal{G}_A})$, which implies that the complexity per iteration of the two cases are almost the same. ◀

Before starting the second example, the reader should be reminded of the following facts about $\mathcal{G}_B$.

$\mathcal{G}_B$ is a well-known $\mathcal{H}_{ST}$ algorithm. The switching strategy for $\mathcal{G}_B$ is such that it guarantees the best threshold and the fastest convergence of error probability to zero (among MB hybrid algorithms), provided that the exact value of the channel parameter is known. This (switching) strategy, which was first derived

in [3] for regular codes, and was later generalized to irregular codes in [45], is specified as follows. In iteration $\ell \geq 1$, and for degree- j variable-nodes, the smallest integer $\omega_{\ell,j}$ satisfying

$$\frac{1-\varepsilon}{\varepsilon} \leq \left(\frac{1 + \rho(1 - 2p_{e,\varepsilon}^{(\ell-1)})}{1 - \rho(1 - 2p_{e,\varepsilon}^{(\ell-1)})} \right)^{2 \cdot \lceil j/2 \rceil - j + 2 \cdot \omega_{\ell,j} + 1}, \quad (5.3.1)$$

is the order of the MB algorithm which is selected for the nodes' operation. (Should the value of $\omega_{\ell,j}$ be greater than $\lceil j/2 \rceil - 1$, the nodes will send their channel messages.) Notice that this strategy is, in fact, equivalent to switching in each iteration to the MB algorithm that provides the greatest reduction in probability of error. It is worth mentioning that the switching-sequence $\mathbf{S}_j \equiv \{\omega_{\ell,j}\}_{\ell=1}^{\infty}$ is only a function of $j \bmod 2$, as $2 \cdot \lceil j/2 \rceil - j$ in the exponent of (5.3.1) is equal to 0 or 1 for even or odd values of j , respectively [52]. Also mention-worthy is the fact that $\mathbf{S}_j$ is a decreasing sequence which eventually settles down to zero, i.e., $\omega_{\ell,j} = 0, \forall j$, and for ℓ sufficiently large.

It can be seen that when $\mathcal{C}^{\infty}(\bar{\lambda}, \bar{\rho})$ is decoded by $\mathcal{G}_B$ over a channel with parameter ε , the curve

$$h_{\varepsilon}^{\bar{\lambda}, \bar{\rho}}(x) = \sum_{j \geq 2} \lambda_j \cdot \check{f}_{\varepsilon}^{j, \bar{\rho}}(x), \quad (5.3.2)$$

where $\check{f}_{\varepsilon}^{j, \bar{\rho}}(x) \equiv \min \left\{ \varepsilon, \min_{\omega} \{ f_{\varepsilon}^{j, \omega, \bar{\rho}}(x) \} \right\}$ represents the recursion describing the evolution of average error probability [52]. Based on (5.3.1), we can easily obtain the iterations where variable-nodes of a particular degree switch the order of the algorithm according to which they operate: the switching-point corresponding to a change of order from $\omega + 1$ to ω is given by

$$p_{e,\varepsilon,\omega} = \frac{1}{2} \cdot \left(1 - \rho^{-1} \left(\frac{(1-\varepsilon)^{\omega} - \varepsilon^{\omega}}{(1-\varepsilon)^{\omega} + \varepsilon^{\omega}} \right) \right), \quad (5.3.3)$$

where $\wp = \frac{1}{2 \cdot \omega + 1}$, for even-degree variable-nodes; and $= \frac{1}{2 \cdot (\omega + 1)}$, for odd-degree variable-nodes. Let the switching-points for even-degree and odd-degree variable-nodes be denoted by $p_{e,\varepsilon,\omega}^{even}$ and $p_{e,\varepsilon,\omega}^{odd}$, respectively. One can see from (5.3.3) that $p_{e,\varepsilon,\omega}^{even} < p_{e,\varepsilon,\omega}^{odd} < p_{e,\varepsilon,\omega+1}^{even} < p_{e,\varepsilon,\omega+1}^{odd}$, for $\omega \geq 0$ [52].

Similar to the case for MB $\mathcal{H}_{TI}$ algorithms, it can be proven that the asymptotic convergence of error probability to zero for $\mathcal{G}_B$ is exponential or super-exponential in ℓ . More precisely, with Θ defined as $\min_{j:\lambda_j > 0} \{\lfloor j/2 \rfloor\}$, it can be shown that as $\ell \rightarrow \infty$, $p_e^{(\ell)}$ reduces exponentially in ℓ , if $\Theta = 1$ (i.e., $\lambda_2 > 0$ or $\lambda_3 > 0$); and super-exponentially in ℓ , with parameter Θ , if $\Theta > 1$, provided that $\varepsilon < \varepsilon^*(\vec{\lambda}_{\mathcal{G}_B}, \vec{\rho}_{\mathcal{G}_B})$.

As (5.3.3) indicates, under $\mathcal{G}_B$, the availability of perfect knowledge of the channel plays a crucial role in determining the optimum switching strategy. In the following example, we compare an optimized $\mathcal{G}_B$ with the optimal MB $\mathcal{H}_{TI}$ algorithm given in Example 5.3.1, and show that the latter can better cope with unknown channel conditions.

Example 5.3.2. Consider $\mathcal{C}^\infty(\vec{\Lambda}_\circ, \vec{\rho}_\circ)$, the rate-1/2 ensemble optimized for MB $\mathcal{H}_{TI}$ algorithms in Example 5.3.1, and assume that $\varepsilon = \varepsilon^*(\vec{\Lambda}_\circ, \vec{\rho}_\circ) = 0.0525$. Figure 5.1(a) depicts the curve $h(x) = h_{0.0525}^{\vec{\Lambda}_\circ, \vec{\rho}_\circ}(x)$, which represents the evolution of expected fraction of errors when the above ensemble is decoded by the corresponding MB $\mathcal{H}_{TI}$ algorithm. As can be seen in the figure, the average probability of error approaches zero throughout the iteration process, with slower convergence in the vicinity of $p_e = 0.0525$ and $p_e = 0$. Depicted in Figure 5.1(a) are also the curves $f_4(x) = f_{0.0525}^{4,1,\vec{\rho}_\circ}(x)$, $f_7(x) = f_{0.0525}^{7,1,\vec{\rho}_\circ}(x)$, and $f_{19}(x) = f_{0.0525}^{19,1,\vec{\rho}_\circ}(x)$, which represent density evolution for the (left) concentrated ensembles $\mathcal{C}^\infty(\vec{\Lambda}_4, \vec{\rho}_\circ)$, $\mathcal{C}^\infty(\vec{\Lambda}_7, \vec{\rho}_\circ)$, and $\mathcal{C}^\infty(\vec{\Lambda}_{19}, \vec{\rho}_\circ)$, under $MB^{4,1,\vec{\rho}_\circ}$, $MB^{7,1,\vec{\rho}_\circ}$, and $MB^{19,1,\vec{\rho}_\circ}$, respectively. Note that

$$h(x) \stackrel{(5.1.5)}{=} \Lambda_{\circ 4,1} \cdot f_4(x) + \Lambda_{\circ 7,1} \cdot f_7(x) + \Lambda_{\circ 19,1} \cdot f_{19}(x).$$

Consider in addition $\mathcal{C}^\infty(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$, where $\lambda_{\mathcal{G}_B 3} = 0.1234$, $\lambda_{\mathcal{G}_B 4} = 0.5551$, $\lambda_{\mathcal{G}_B 16} = 0.3215$, and $\rho_{\mathcal{G}_B 10} = 1.0000$. This rate-1/2 ensemble, for which we have $\varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B}) = 0.0577$, $\bar{d}_v = 5$, and $\bar{d}_c = 10$, has been optimized for $\mathcal{G}_B$ amongst ensembles with $D_v = 16$ and $D_c = 10$ [45]. Using (5.3.1), it can be seen that this threshold is achieved as follows.

- ◇ For degree-16 variable-nodes, we start with $MB^{16,2,\bar{\rho}_{\mathcal{G}_B}}$ and switch to $MB^{16,1,\bar{\rho}_{\mathcal{G}_B}}$ and $MB^{16,0,\bar{\rho}_{\mathcal{G}_B}}$ in iterations 5 and 102, respectively.
- ◇ For degree-4 variable-nodes, we keep passing the channel messages until iteration 5, when we switch to $MB^{4,1,\bar{\rho}_{\mathcal{G}_B}}$. We later switch to $MB^{4,0,\bar{\rho}_{\mathcal{G}_B}}$ in iteration 102.
- ◇ For degree-3 variable-nodes, we start with passing the channel messages until iteration 43, when we switch to $MB^{3,0,\bar{\rho}_{\mathcal{G}_B}}$.

Assume now that $\varepsilon = \varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B}) = 0.0577$. Portrayed in Figure 5.1(b) are the curves $h(x) = h_{0.0577}^{\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B}}(x)$, $f_3(x) = \check{f}_{0.0577}^{3,\bar{\rho}_{\mathcal{G}_B}}(x)$, $f_4(x) = \check{f}_{0.0577}^{4,\bar{\rho}_{\mathcal{G}_B}}(x)$, and $f_{16}(x) = \check{f}_{0.0577}^{16,\bar{\rho}_{\mathcal{G}_B}}(x)$, which represent the evolution of expected error probability under $\mathcal{G}_B$ for the ensembles $\mathcal{C}^\infty(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$, $\mathcal{C}^\infty(\bar{\lambda}_3, \bar{\rho}_{\mathcal{G}_B})$, $\mathcal{C}^\infty(\bar{\lambda}_4, \bar{\rho}_{\mathcal{G}_B})$, and $\mathcal{C}^\infty(\bar{\lambda}_{16}, \bar{\rho}_{\mathcal{G}_B})$, respectively. Note that

$$h(x) \stackrel{(5.3.2)}{=} \lambda_{\mathcal{G}_B 3} \cdot f_3(x) + \lambda_{\mathcal{G}_B 4} \cdot f_4(x) + \lambda_{\mathcal{G}_B 16} \cdot f_{16}(x).$$

The figure shows that the average error probability approaches zero as the number of iterations goes to infinity. The convergence (of average error probability to zero), however, is slower in the vicinity of switching-points, which are

- $s^1: p_e = 0.0442$, where even-degree variable-nodes switch the order of their algorithms from 2 to 1,
- $s^2: p_e = 0.0273$, where odd-degree variable-nodes switch the order from 1 to 0,

and

- $s^3: p_e = 0.0067$, where even-degree variable-nodes switch the order from 1 to 0.⁵

(As mentioned before, degree- j variable-nodes operating under $MB^{j \rightarrow \omega \bar{p}}$ will pass their channel messages should ω exceed $\lfloor j/2 \rfloor - 1$.)

Figure 5.1(b) also shows the curves labeled by $f^{0 \rightarrow 4}(x)$, $f^{5 \rightarrow 42}(x)$, $f^{43 \rightarrow 101}(x)$, and $f^{102 \rightarrow \infty}(x)$, which provide graphical representations of density evolution for $\mathcal{C}^\infty(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$, assuming that even-degree and odd-degree variable-nodes are operating according to MB algorithms of orders 2 and 1, 1 and 1, 1 and 0, and 0 and 0, respectively. From the maximum threshold achieving strategy mentioned earlier, it is easy to see that

$$h(x) = \begin{cases} f^{0 \rightarrow 4}(x), & \text{if } 0 \leq \ell \leq 4, \\ f^{5 \rightarrow 42}(x), & \text{if } 5 \leq \ell \leq 42, \\ f^{43 \rightarrow 101}(x), & \text{if } 43 \leq \ell \leq 101, \\ f^{102 \rightarrow \infty}(x), & \text{if } 102 \leq \ell. \end{cases} \quad (\dagger)$$

Note that $(\dagger)$ can alternatively be written as

$$h(x) = \min \{ f^{0 \rightarrow 4}(x), f^{5 \rightarrow 42}(x), f^{43 \rightarrow 101}(x), f^{102 \rightarrow \infty}(x) \}. \quad (\dagger\dagger)$$

It is worth mentioning that this alternative form, which takes into account all potentially useful (order) combinations of even-degree and odd-degree variable-nodes for $\varepsilon = \varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$, also gives the curve corresponding to the switching-sequence that is optimal in terms of convergence speed for $\varepsilon < \varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$.

Now suppose that we would like to decode $\mathcal{C}^\infty(\bar{\Lambda}_\circ, \bar{\rho}_\circ)$ and $\mathcal{C}^\infty(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$ over a channel with actual parameter $\varepsilon_{act} = \frac{3}{4} \cdot \varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B}) = 0.043275$, using the aforesaid MB $\mathcal{H}_{TI}$ algorithm and $\mathcal{G}_B$, respectively. Under these algorithms, as can be seen in Figure 5.2(a) and Figure 5.2(b), the probability of error for both ensembles still

⁵Switching-points are obtained using (5.3.3).

Figure 5.1: Graphical interpretation of density-evolution; (a) for $C^\infty(\bar{\Lambda}_o, \bar{\rho}_o)$, decoded by the MB $\mathcal{H}_{TI}$ algorithm given in Example 5.3.1, assuming $\varepsilon = \varepsilon^*(\bar{\Lambda}_o, \bar{\rho}_o) = 0.0525$; (b) for $C^\infty(\bar{\Lambda}_{G_B}, \bar{\rho}_{G_B})$, decoded by $\mathcal{G}_B$, assuming $\varepsilon = \varepsilon^*(\bar{\Lambda}_{G_B}, \bar{\rho}_{G_B}) = 0.0577$. (See Example 5.3.2.)

Figure 5.2: Graphical interpretation of density-evolution; (a) for $C^\infty(\tilde{\Lambda}_o, \tilde{\rho}_o)$, decoded by the optimized MB $\mathcal{H}_{TI}$ algorithm given in Example 5.3.1; (b) for $C^\infty(\tilde{\lambda}_{\mathcal{G}_B}, \tilde{\rho}_{\mathcal{G}_B})$, decoded by the $\mathcal{G}_B$ optimized for $\varepsilon = \varepsilon^*(\tilde{\lambda}_{\mathcal{G}_B}, \tilde{\rho}_{\mathcal{G}_B}) = 0.0577$; both under the assumption of $\varepsilon = \frac{3}{4} \cdot \varepsilon^*(\tilde{\lambda}_{\mathcal{G}_B}, \tilde{\rho}_{\mathcal{G}_B}) = 0.043275$. (See Example 5.3.2.)

converges to zero, which is not unexpected. In fact, according to Appendix D, by setting an MB $\mathcal{H}_{TI}$ algorithm to achieve its maximum threshold, its convergence for all channel parameters below the maximum threshold is secured. This is also true for $\mathcal{G}_B$. Having said these, it is observed in the figures that the convergence behavior in the two cases are very different. On the one hand, the MB $\mathcal{H}_{TI}$ algorithm exhibits a smooth and fast convergence. One can see in Figure 5.2(a) that the convergence tunnel between the curve $y = h(x)$ and the line $y = x$ is wide open over, roughly speaking, the entire interval $(0, \varepsilon_{act}]$, and therefore the probability of error approaches zero rapidly. (The tunnel is only narrow in close proximity of $x = 0$, resulting in a relatively slow convergence in the vicinity of $p_e = 0$.) For the $\mathcal{G}_B$, on the other hand, the convergence changes pace abruptly, and is generally slower. As portrayed in Figure 5.2(b), the convergence tunnel for the $\mathcal{G}_B$ is wide open over some subintervals of $(0, \varepsilon_{act}]$, and almost closed over some others. This results in a slow convergence, on average, compared to the MB $\mathcal{H}_{TI}$ algorithm.⁶ This is, of course, for the simple reason that the strategy optimized for $\varepsilon = \varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$ is highly suboptimal for $\varepsilon = \varepsilon_{act}$, and switching to the right constituent algorithms fails to happen at the right iterations: even-degree (respectively, odd-degree) variable-nodes switch their order from 2 to 1 to 0 (respectively, 1 to 0) in iterations 4 and 102 (respectively, iteration 43), instead of iterations 2 and 10 (respectively, iteration 4).⁷ Graphically speaking, while the optimum convergence tunnel (for $\varepsilon = \varepsilon_{act}$) is specified by the line $y = x$ and the curve $y = h(x)$ obtained according to $(\dagger\dagger)$, the actual one is bordered by $y = x$ and $y = h(x)$ obtained according to $(\dagger)$. Note that $\mathcal{G}_B$'s sensitivity to the availability of an accurate knowledge of the channel parameter is in sharp contrast to the capability of MB $\mathcal{H}_{TI}$ algorithms to adapt to the channel condition. This universality is a major advantage of MB $\mathcal{H}_{TI}$ algorithms over $\mathcal{G}_B$ for applications where the channel condition may vary

⁶The two algorithms achieve an error fraction of 10^{-4} in iterations 59 and 30, respectively.

⁷By proper switching in iterations 2, 4, and 10, an error fraction of 10^{-4} is achieved in iteration 13.

and/or a perfect knowledge of the channel is not available at the receiver.

To make our point more clear, we consider the situation where the channel parameter is unknown, and the main concern is to ensure the convergence of MB $\mathcal{H}_{TI}$ algorithm or $\mathcal{G}_B$ for as wide a range of channel parameters as possible. In this situation, we have to set both algorithms to achieve their maximum thresholds.⁸ Figure 5.3(a) depicts error-rate curves versus iteration number for the two algorithms, and for channel parameters $\varepsilon = \Gamma \cdot \varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$, $\Gamma = 0.1, 0.2, \dots, 1.0$. The figure shows that although the convergence of the $\mathcal{G}_B$ is faster than that of the MB $\mathcal{H}_{TI}$ algorithm when the channel parameter is close to $\varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$, the latter can outpace the former for a wide range of target error probabilities, and for a wide range of channel parameters less than $\varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$ (in fact, for about $\Gamma < 0.85$).

It may justly be argued that the previous setting favors the MB $\mathcal{H}_{TI}$ algorithm, whose maximum threshold is smaller, in terms of convergence speed. In fact, the advantage of the MB $\mathcal{H}_{TI}$ algorithm in convergence speed is partly counter-balanced by the wider range of convergence for the $\mathcal{G}_B$. It will be more fair, therefore, to compare the algorithms' speed of convergence while the convergence ranges are the same.

For this, we opt to set $\mathcal{G}_B$ for the fastest convergence over a channel with $\varepsilon = \varepsilon^*(\bar{\lambda}_o, \bar{\rho}_o)$. This is done by proper switchings in iterations 3, 8, and 21. The 'new' $\mathcal{G}_B$ only converges to zero error probability for $\varepsilon \leq 1.0086 \cdot \varepsilon^*(\bar{\lambda}_o, \bar{\rho}_o) = 0.0529$, which is roughly the same as the convergence range for the MB $\mathcal{H}_{TI}$ algorithm. (This is due to the fact that $\mathcal{G}_B$ is very sensitive to underestimating the channel parameter.) Now let us see how the two algorithms compare in the new situation. Figure 5.3(b) portrays the fraction of errors versus iteration number for the two algorithms, and for $\varepsilon = \Gamma \cdot \varepsilon^*(\bar{\lambda}_o, \bar{\rho}_o)$, $\Gamma = 0.1, 0.2, \dots, 1.0$. One can see in the figure that similar to the previous situation, the MB $\mathcal{H}_{TI}$ algorithm outspeeds the (new) $\mathcal{G}_B$ for a wide range of target error probabilities, and for a wide range of channel parameters less than

⁸The code ensembles of interest are $\mathcal{C}^\infty(\bar{\lambda}_o, \bar{\rho}_o)$ and $\mathcal{C}^\infty(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$, correspondingly.

$\varepsilon^*(\vec{\Lambda}_o, \vec{\rho}_o)$ (in fact, for about $\Gamma < 0.73$).⁹ Expectedly, however, the difference between the speeds is not as dramatic as before. ◀

It should be clear to the reader from the provided examples that MB $\mathcal{H}_{TI}$ algorithms, which are generalization of $\mathcal{G}_A$, can be more attractive than $\mathcal{G}_B$, in terms of speed of convergence, in practice, where the channel condition is not perfectly known. This is particularly important as the maximum number of iterations is always limited in realistic situations. In the following section, assuming that the channel parameter is unknown, we present some finite-length simulation results that compare the performance of MB $\mathcal{H}_{TI}$ algorithms and $\mathcal{G}_B$ for several choices of maximum number of iterations.

5.4 Finite-Length Simulation Results

In this section, some finite-length simulation results, comparing the performance of the optimized MB $\mathcal{H}_{TI}$ algorithm of Example 5.3.1 and the optimized $\mathcal{G}_B$ of Example 5.3.2, are presented.

Example 5.4.1. *The above mentioned algorithms have been used to decode two randomly constructed codes belonging to $\mathcal{C}^\infty(\vec{\Lambda}_o, \vec{\rho}_o)$ and $\mathcal{C}^\infty(\vec{\Lambda}_{\mathcal{G}_B}, \vec{\rho}_{\mathcal{G}_B})$, correspondingly. The codes are of length 5×10^4 , with no cycle of length 4 in their graph representations. Using Monte Carlo method, we have obtained the performance of the algorithms for maximum number of iterations $\ell_{\max} = 10, 20, 50$, and 500, and for a wide range of channel parameters. For each reported point, enough number of codewords have been simulated to generate 100 codeword errors.*

⁹In Figure 5.3, the depicted error fractions are greater than or equal to 10^{-4} , which may not seem particularly low. It, however, can be seen that this range translates to a bit error-rate (BER) of at least 10^{-6} .

Figure 5.3: Comparison between the convergence speed of the MB $\mathcal{H}_{TI}$ algorithm given in Example 5.3.1, and $\mathcal{G}_B$, over $\mathcal{C}^\infty(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$; (a) for $\varepsilon = \Gamma \cdot \varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$, with $\mathcal{G}_B$ optimized for $\varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$, and (b) for $\varepsilon = \Gamma \cdot \varepsilon^*(\bar{\lambda}_o, \bar{\rho}_o)$, with $\mathcal{G}_B$ optimized for $\varepsilon^*(\bar{\lambda}_o, \bar{\rho}_o)$: $\Gamma = 0.1, 0.2, \dots, 1.0$ - left to right. (See Example 5.3.2.)

Figure 5.4 depicts the BER curves for the three algorithms. As can be seen in the figure, for $\ell_{\max} = 500$, where all the algorithms have sufficient time to converge, $\mathcal{G}_B$ optimized for $\varepsilon = \varepsilon^*(\bar{\lambda}_{\mathcal{G}_B}, \bar{\rho}_{\mathcal{G}_B})$ has the best performance at low BERs, (closely) followed by the $\mathcal{G}_B$ optimized for $\varepsilon = \varepsilon^*(\bar{\Lambda}_o, \bar{\rho}_o)$. This order, however, changes at higher BERs or when $\ell_{\max}$ decreases. In fact, for $\ell_{\max} = 10, 20$, and 50 , the MB $\mathcal{H}_{TI}$ algorithm comfortably outperforms the first $\mathcal{G}_B$. The MB $\mathcal{H}_{TI}$ algorithm also easily beats the second $\mathcal{G}_B$ for $\ell_{\max} = 10$, and manages to maintain a small lead over it for $\ell_{\max} = 20$, for a wide range of channel parameters. Both the MB $\mathcal{H}_{TI}$ algorithm and the second $\mathcal{G}_B$ reach their full potential with $\ell_{\max} = 50$: the performance for $\ell_{\max} = 50$ is almost identical to that of $\ell_{\max} = 500$. This is while for the first $\mathcal{G}_B$, there is a large gap between the performance of $\ell_{\max} = 50$ and that of $\ell_{\max} = 500$. (It is worth mentioning that a careful inspection of simulation data shows that for smaller values of ε , the number of bit errors for each codeword error is (typically) very small for the MB $\mathcal{H}_{TI}$ algorithm, and as such, its performance in high signal-to-noise ratio (SNR) region can be greatly improved by properly using an extremely high-rate outer (channel) code capable of removing very small fractions of random errors. For $\mathcal{G}_B$, however, this is not the case, especially for smaller values of $\ell_{\max}$, i.e., when a codeword is in error, many of its bits are also in error.) Note that these results are in general agreement with our discussions in the previous section and show that in practical situations when the maximum number of iterations is limited, MB $\mathcal{H}_{TI}$ algorithms can be more attractive compared to MB $\mathcal{H}_{ST}$ algorithms. ◀

5.5 Conclusion

MB $\mathcal{H}_{TI}$ algorithms for decoding irregular LDPC codes were introduced. We analyzed the performance and the convergence speed of these algorithms using density

Figure 5.4: BER curves for the MB $\mathcal{H}_{Tl}$ algorithm given in Example 5.3.1 and the $\mathcal{G}_{BS}$ given in Example 5.3.2, for $\ell_{\max} = 10, 20, 50$, and 500. (See Example 5.4.1.)

evolution. Tight upper bounds on threshold were derived and it was shown that the asymptotic convergence speed of MB $\mathcal{H}_{TI}$ algorithms can be super-exponential, and is at least exponential, in the number of iterations. We jointly optimized the degree distributions of LDPC codes and the time-invariant assignment for MB $\mathcal{H}_{TI}$ algorithms, and showed the improvement in threshold that MB $\mathcal{H}_{TI}$ algorithms can provide compared to the optimized $\mathcal{G}_A$. We also compared MB $\mathcal{H}_{TI}$ algorithms with optimized switch-type algorithms ($\mathcal{G}_B$) in terms of performance and convergence speed. Our results indicate that although $\mathcal{G}_B$ outperforms MB $\mathcal{H}_{TI}$ algorithms when the channel parameter is perfectly known, the trend changes when such a knowledge is not available. For such cases, MB $\mathcal{H}_{TI}$ algorithms demonstrate great robustness in preserving their (near) optimality. Our asymptotic results are also confirmed by finite-length simulations.

“A conclusion is the place where you got tired of thinking.”

– Harold Fricklestein

Chapter 6

Concluding Remarks

In this chapter, we briefly review the contributions of this thesis, and suggest some directions for future research work.

6.1 Summary of Contributions

As part of the first (main) contribution of this work, we comprehensively studied a family of binary message-passing decoding algorithms for regular LDPC codes which we call majority-based (MB) algorithms. MB algorithms work based on a generalized majority-decision rule, and are particularly attractive for their remarkably simple implementation. (Both **Gallager’s** algorithm A ($\mathcal{G}_A$), regarded as a benchmark for binary message-passing decoding algorithms, and majority decoding algorithm, belong to the family of MB algorithms.) Using density evolution, we were able to derive many theoretical results on the asymptotic convergence behavior of MB algorithms. We employed a graphical interpretation of density evolution to visualize these results. We presented a new characterization for MB algorithms’ noise threshold, and showed that many MB algorithms have substantially larger threshold values compared to $\mathcal{G}_A$ and majority decoding algorithm. We also presented tight analytical upper bounds on these thresholds, and proved that in many cases these bounds are achieved. We also highlighted another attractive feature of MB algorithms, i.e., their generally fast

convergence. We showed that an MB algorithm's convergence speed was determined by its order, and explained why many MB algorithms outpaced $\mathcal{G}_A$ substantially. In particular, we proved that while the asymptotic convergence speed of $\mathcal{G}_A$ was exponential with iteration number, for other MB algorithms, it was super-exponential. The fast convergence and the better performance offered by MB algorithms other than $\mathcal{G}_A$ and majority decoding algorithm, suggests a serious role for them in future coding schemes, where the system constraints require the use of hard-decision decoders.

As part of the second (main) contribution of this work, we introduced hybrid decoding algorithms for regular LDPC codes in a general framework, and showed that by properly combining existing iterative decoding algorithms, it was possible to improve both error performance and convergence speed. We recognized many well-known iterative algorithms as instances of switch-type hybrid ($\mathcal{H}_{ST}$) algorithms, where the hybrid algorithm switches between different constituent algorithms. We then proposed time-invariant hybrid ($\mathcal{H}_{TI}$) algorithms. These algorithms use a specific blend of various constituent algorithms in each iteration. The blend, however, does not change by iteration, hence, the name time-invariant. We used density evolution for design of MB hybrid decoders for LDPC codes. We also investigated asymptotic convergence properties of MB hybrid algorithms in detail, and provided simulation results at finite block lengths. We then compared MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ algorithms, and showed that although MB $\mathcal{H}_{ST}$ algorithms could provide better threshold and faster convergence amongst all MB hybrid algorithms, they could be outperformed by MB $\mathcal{H}_{TI}$ algorithms in the absence of perfect knowledge of the channel parameter, or at finite block lengths. We also showed that MB $\mathcal{H}_{ST}$ algorithms were not nearly as robust as their time-invariant counterparts, with respect to channel variations.

The third (main) contribution of this work is the extension of the application of MB $\mathcal{H}_{TI}$ algorithms to irregular LDPC codes. Again, using density evolution, we analyzed the asymptotic performance and the asymptotic convergence speed of

these algorithms. We derived tight upper bounds on threshold, and showed that the asymptotic convergence speed of MB $\mathcal{H}_{TI}$ algorithms for irregular codes can be super-exponential, and is at least exponential, in the number of iterations. We also jointly optimized the degree distributions of LDPC codes and the time-invariant assignment for MB $\mathcal{H}_{TI}$ algorithms, and showed the improvement in threshold that MB $\mathcal{H}_{TI}$ algorithms can provide compared to the optimized $\mathcal{G}_A$ for irregular codes. Moreover, we compared MB $\mathcal{H}_{TI}$ algorithms with optimized switch-type algorithms (Gallager's algorithm B, $\mathcal{G}_B$) for irregular codes, in terms of performance and convergence speed. Our asymptotic results, also confirmed by finite-length simulations, indicate that similar to the case for regular codes, although $\mathcal{G}_B$ outperforms MB $\mathcal{H}_{TI}$ algorithms when the channel parameter is perfectly known, the trend changes when such knowledge is not available. (For such cases, MB $\mathcal{H}_{TI}$ algorithms demonstrate great robustness in preserving their (near) optimality.)

In brief, our results suggest that MB $\mathcal{H}_{TI}$ algorithms are among the best choices of binary message-passing algorithms available. They are universal, their complexity per iteration is minimal, and they provide a robust performance over a wide range of channel conditions.

6.2 Possible Future Work

In the continuation of this work, for short-term research, an interesting problem to look into, in the context of hybrid algorithms, is investigating the possibility of improving the decoding performance / complexity by taking into account in partitioning the nodes, the structure of the code, and the channel messages (as opposed to random partitioning). One example is reliability-based scheduled decoding of LDPC codes. In reliability-based scheduled decoding algorithms, unreliable messages are identified, and the decoder prevents them from propagating in early stages of iterative decoding

[78]. These algorithms may be regarded as $\mathcal{H}_{ST}$ algorithms, and can be analyzed (and designed) accordingly.

For long-term research, we suggest investigating the performance of $\mathcal{H}_{TI}$ algorithms whose constituents are not restricted to MB algorithms. In the case of (time-invariantly) blending sum-product and min-sum decoding algorithms, we expect to see similar reduction in the sensitivity to the channel variations as in MB $\mathcal{H}_{TI}$ algorithms (to which sum-product decoding algorithm is vulnerable [79]).

Another direction is to evaluate the performance of hybrid algorithms over more sophisticated channels such as fading channels.

We also suggest focusing on the analysis of the convergence behavior of finite-length codes, in order to explain such phenomena as the presence of error floors at short block lengths (for hybrid algorithms designed for irregular codes, in particular). A promising direction, we believe, is to use control tools developed for investigating the chaotic behavior of non-linear equation sets, for iterative algorithms.

Appendix A

Density Evolution for Min-Sum Decoding Algorithm

In this appendix, we formulate density evolution method for min-sum decoding algorithm. The reader can refer to [80] and [81] for a more comprehensive treatment of the subject.

A.1 Derivation of Density Evolution Formulas for Min-Sum Decoding Algorithm

In the following, we use density evolution method to track error PDF under min-sum decoding algorithm as a function of iteration number. In iteration zero, where check-nodes are inactive and variable-nodes pass their channel messages (see (2.6.12) and (2.6.13)), we have

$$f_c^{(0)} = \mathfrak{Null}, \quad (\text{A.1.1})$$

and

$$f_v^{(0)} = N\left(\frac{2}{\sigma^2}, \frac{4}{\sigma^2}\right). \quad (\text{A.1.2})$$

(Recall that $\mathbf{m}_0 = \frac{2 \cdot y}{\sigma^2}$, where $y = 1 + z$, and $z \sim N(0, \sigma^2)$.)

In order to find $f_c^{(\ell)}$, the PDF of check-nodes' outgoing messages in iteration $\ell \geq 1$ (see (2.6.17)), we first prove the following proposition, stated in [81] without proof.

Proposition A.1.1. *Suppose that X and Y are independent random variables having PDFs $f_X(x)$ and $f_Y(y)$ and cumulative distribution functions (CDFs) $F_X(\mathbf{x})$ and $F_Y(\mathbf{y})$, respectively. For the random variable $Z = \text{sgn}(X) \cdot \text{sgn}(Y) \cdot \min\{|X|, |Y|\}$, we have*

$$f_Z(z) = \begin{cases} f_X(z) \cdot (1 - F_Y(-z)) + f_X(-z) \cdot F_Y(z) \\ + f_Y(z) \cdot (1 - F_X(-z)) + f_Y(-z) \cdot F_X(z), & \text{if } z < 0, \\ f_X(z) \cdot (1 - F_Y(z)) + f_X(-z) \cdot F_Y(-z) \\ + f_Y(z) \cdot (1 - F_X(z)) + f_Y(-z) \cdot F_X(-z), & \text{if } z > 0. \end{cases} \quad (\text{A.1.3})$$

Proof. Assuming that $z < 0$, we have

$$P\{Z \leq z\} = P\left\{(-z \leq X \cap Y \leq z) \cup (X \leq z \cap -z \leq Y)\right\}, \quad (\text{A.1.4})$$

where $\cap$ and $\cup$ represent intersection and union, respectively. Equivalently, we have

$$F_Z(z) = (1 - F_X(-z)) \cdot F_Y(z) + F_X(z) \cdot (1 - F_Y(-z)). \quad (\text{A.1.5})$$

On the other hand, for $z > 0$, we have

$$\begin{aligned} P\{Z \leq z\} &= P\left\{ \begin{aligned} &(0 < X \leq z \cap 0 < Y) \\ &\cup (0 < Y \leq z \cap 0 < X) \\ &\cup (-z \leq X < 0 \cap Y < 0) \\ &\cup (-z \leq Y < 0 \cap X < 0) \end{aligned} \right\}, \end{aligned} \quad (\text{A.1.6})$$

or equivalently,

$$\begin{aligned}
 F_Z(\mathbf{z}) = & (F_X(\mathbf{z}) - F_X(0)) \cdot (1 - F_Y(0)) \\
 & + (F_Y(\mathbf{z}) - F_Y(0)) \cdot (1 - F_X(0)) \\
 & - (F_X(\mathbf{z}) - F_X(0)) \cdot (F_Y(\mathbf{z}) - F_Y(0)) \\
 & + (F_X(0) - F_X(-\mathbf{z})) \cdot F_Y(0) \\
 & + (F_Y(0) - F_Y(-\mathbf{z})) \cdot F_X(0) \\
 & - (F_X(0) - F_X(-\mathbf{z})) \cdot (F_Y(0) - F_Y(-\mathbf{z})). \tag{A.1.7}
 \end{aligned}$$

The PDF in (A.1.3) is now obtained by differentiating the CDFs in (A.1.5) and (A.1.7). $\square$

Summarizing (A.1.3) in the notation

$$f_Z = f_X \odot f_Y, \tag{A.1.8}$$

it now follows from (2.6.17) that

$$f_c^{(\ell)} = \underbrace{f_v^{(\ell-1)} \odot f_v^{(\ell-1)} \odot \dots \odot f_v^{(\ell-1)}}_{d_c-1}. \tag{A.1.9}$$

For obtaining $f_v^{(\ell)}$, the PDF of variable-nodes' outgoing messages in iteration $\ell \geq 1$ (see (2.6.15)), we use the fact that the PDF of the sum of two independent random variables equals the convolution of their respective PDFs. Having said this, it follows from (2.6.15) that

$$f_v^{(\ell)} = f_v^{(0)} \otimes \underbrace{f_c^{(\ell)} \otimes f_c^{(\ell)} \otimes \dots \otimes f_c^{(\ell)}}_{d_v-1}. \tag{A.1.10}$$

($\otimes$ denotes the convolution of two functions.)

Now, combining (A.1.9) and (A.1.10), we obtain

$$f_v^{(\ell \geq 1)} = f_v^{(0)} \otimes \left(\bigotimes_{j=1}^{d_v-1} \left(\bigodot_{i=1}^{d_c-1} f_v^{(\ell-1)} \right) \right), \tag{A.1.11}$$

where $\bigodot_{i=1}^I f \equiv \underbrace{f \odot f \odot \cdots \odot f}_I$ and $\bigotimes_{j=1}^J f \equiv \underbrace{f \otimes f \otimes \cdots \otimes f}_J$.

The average fraction of erroneous variable-to-check messages (in iteration ℓ) is obtained as follows.

$$p_e^{(\ell)} = \int_{-\infty}^0 f_v^{(\ell)}(x) \cdot dx. \quad (\text{A.1.12})$$

Example A.1.1. *Figure A.1 illustrates the evolution of the PDF of (variable-to-check) messages for $\mathcal{C}^\infty(3,6)$ under min-sum decoding algorithm. The AWGN variance is 0.5. Depicted are $f_v^{(\ell)}(x)$ for $\ell = 0, 1, \dots, 5$. One can clearly see that the PDF shifts to the right as ℓ increases, and the probability of message error, which is obtained from (A.1.12), decreases. (Note that $f_v^{(0)} = N(4, 8)$.)* ◀

5

Figure A.1: Density evolution for $\mathcal{C}^\infty(3,6)$, under min-sum decoding algorithm, over an BIPLGC with noise variance $\sigma^2 = 0.5$. (See Example A.1.1.)

Appendix B

Graphical Representation of a General Recursion

Exact or approximate graphical interpretations of density evolution, and related concepts, are widely used to visualize the behavior of iterative decoding algorithms [14, 15, 17–21, 51, 52, 59, 63–67]. This approach, introduced originally by Gallager [3, p. 48], is based on a simple graphical representation of recursions, and is particularly helpful in investigating iterative algorithms' threshold values and convergence speeds.

Consider a one-variable recursion of the general form $x^{\ell+1} = f(x^\ell)$, $\ell \geq 0$, initiated by $x^0 = x_0$. It is easy to see that for $\ell > 1$, $x^\ell = \underbrace{f \circ f \circ \cdots \circ f}_{\ell \text{ times}}(x_0)$, where the symbol $\circ$ denotes the composition of two functions. As illustrated in Fig. B.1, this can be represented by successive horizontal and vertical projections between the curve $y = f(x)$ and the line $y = x$. Clearly, the properties of the function f , and its relative location with respect to the line $y = x$, determine the convergence behavior of this recursion.

Figure B.1: Illustration of the graphical representation of a recursion.

Appendix C

Computation of Threshold for Two-Stage $\mathcal{H}_{ST}$ Algorithms

In this appendix, we focus on computing the threshold for two-stage $\mathcal{H}_{ST}$ algorithms. In such cases, only two constituent algorithms are involved: we start the iteration process with one, and after a finite number of iterations, we switch to the other one. For the sake of simplicity, we assume that the second algorithm is an MB algorithm.

C.1 Convergence Region of MB Algorithms and Threshold of Two-Stage $\mathcal{H}_{ST}$ Algorithms

Consider a two-stage $\mathcal{H}_{ST}$ algorithm Δ . We start with Δ_1 , the first constituent algorithm, and after k iterations, we switch to Δ_2 , the second constituent algorithm, which is an MB algorithm. To compute the threshold for Δ , we introduce the *convergence region* of Δ_2 . Let $p_e^{(\ell)}$ denote the average fraction of erroneous messages, sent from variable-nodes to check-nodes, at the end of iteration ℓ of Δ_2 . Now, consider the recursion expressing $p_e^{(\ell)}$ as a function of $p_e^{(\ell-1)}$ and $p_e^{(0)}$, and let both $p_e^{(0)}$ and $p_e^{(1)}$ be selected freely (as opposed to $p_e^{(1)}$ being a function of $p_e^{(0)}$). Clearly, for a given $(p_e^{(0)}, p_e^{(1)})$ pair, the sequence $\{p_e^{(\ell)}\}_{\ell=0}^{\infty}$ may, or may not, converge to zero. The

convergence region of Δ_2 is defined as the set of all $(p_e^{(0)}, p_e^{(1)})$ pairs in the $p_e^{(0)}-p_e^{(1)}$ plane such that $\lim_{\ell \rightarrow \infty} p_e^{(\ell)} = 0$.

As an example, Figure C.1 shows the triangle-shaped convergence region of MB^0 , or equivalently $\mathcal{G}_A$, for $\mathcal{C}^\infty(3, 6)$. Notice that this convergence region is limited to $p_e^{(0)} = 0.1$. Using Theorem 3.2.2, it can be shown that for $\mathcal{C}^\infty(d_v, d_c)$, the convergence region of $\mathcal{G}_A$ is limited to $\frac{1}{(d_v-1)(d_c-1)}$, while it extends to $p_e^{(0)} = 0.5$ for other MB algorithms.

Convergence regions can be conveniently used to compute the threshold for two-stage $\mathcal{H}_{ST}$ algorithms under consideration, as follows. Let $p_{e,\Delta_1}^{(k)}(\pi)$ denote for channel parameter π , the average probability of error at the end of iteration k for Δ_1 . Now, assuming that there is a one-to-one correspondence $\mathcal{U}$ between $p_e^{(0)}$ and π , i.e., $p_e^{(0)} = \mathcal{U}(\pi)$ and $\pi = \mathcal{U}^{-1}(p_e^{(0)})$, we can conclude that if $(p_e^{(0)}, p_{e,\Delta_1}^{(k)}(\mathcal{U}^{-1}(p_e^{(0)})))$ is in the convergence region of Δ_2 , then the probability of error approaches zero as the number of iterations for Δ_2 tends to infinity. So, to find the threshold for Δ , we need to find the supremum value of $p_e^{(0)}$ for which $(p_e^{(0)}, p_{e,\Delta_1}^{(k)}(\mathcal{U}^{-1}(p_e^{(0)})))$ is in the convergence region of Δ_2 . This can be easily done by plotting the curve $p_e^{(1)} = p_{e,\Delta_1}^{(k)}(\mathcal{U}^{-1}(p_e^{(0)}))$ in the convergence region plane of Δ_2 . The threshold is then at the intersection of this curve and the boundary of the convergence region.

Example C.1.1. Consider $\mathcal{C}^\infty(3, 6)$, and assume that the channel is BIPLGC, parameterized by its noise standard deviation. Let Δ_1 and Δ_2 be min-sum decoding algorithm and MB^0 , respectively. Note that in order to switch from min-sum decoding algorithm to MB^0 , we have to replace the outgoing messages of variable-nodes, as well as channel messages, by their signs. Besides the convergence region of MB^0 for $\mathcal{C}^\infty(3, 6)$, depicted in Figure C.1 are also the curves $p_e^{(1)} = p_{e,\Delta_1}^{(k)}(\mathcal{U}^{-1}(p_e^{(0)}))$, $k = 1, 2, \dots, 6$, where $\mathcal{U}(\cdot) = Q(\frac{\cdot}{\sigma})$. ($Q(\cdot)$ denotes the Q -function.) These curves are obtained using density evolution. (See Appendix A.) The figure clearly shows

that the threshold values of hybrid decoding algorithms increase as the number of soft iterations increases, which is not unexpected. This can be taken to mean that by using a few iterations of min-sum decoding algorithm first, and then switching to MB algorithms, a large portion of the performance gap between the better-performing, but more complex, min-sum decoding algorithm and a not-so-well-performing, but simpler, MB algorithm is covered. ◀

Table C.1 lists the threshold values for $\mathcal{C}^\infty(3, 6)$, $\mathcal{C}^\infty(4, 6)$, and $\mathcal{C}^\infty(4, 8)$, decoded by $k = 0, 1, \dots, 6$ iterations of min-sum decoding algorithm, and then ∞ iterations of MB^0 . Note that, for example, when $\mathcal{C}^\infty(4, 8)$ is decoded, by just adding one iteration of min-sum decoding algorithm to MB^0 , the threshold is improved by more than 2.34 dB, and after 6 iterations of min-sum decoding algorithm, the original difference of more than 5.19 dB between the two threshold values, is reduced to only less than 0.34 dB.

Figure C.1: Convergence region of MB^0 for $C^\infty(3,6)$, and $p_{e,\Delta_1}^{(k)}$ curves for $k = 1, 2, \dots, 6$ iterations of min-sum decoding algorithm.

Table C.1: Threshold values for $\mathcal{C}^\infty(3, 6)$, $\mathcal{C}^\infty(4, 6)$, and $\mathcal{C}^\infty(4, 8)$, decoded by $k = 0, 1, \dots, 6$ iterations of min-sum, and then ∞ iterations of MB^0

k	$\mathcal{C}^\infty(3, 6)$		$\mathcal{C}^\infty(4, 6)$		$\mathcal{C}^\infty(4, 8)$	
	ε^*	E_b/N_0^* (dB)	ε^*	E_b/N_0^* (dB)	ε^*	E_b/N_0^* (dB)
0	0.0394	4.8904	0.0159	6.6321	0.0077	7.6864
1	0.0569	3.9837	0.0525	5.9562	0.0322	5.3402
2	0.0672	3.5037	0.0788	4.7625	0.0519	4.2250
3	0.0747	3.1794	0.0958	4.0797	0.0651	3.5991
4	0.0804	2.9370	0.1066	3.6619	0.0736	3.2221
5	0.0850	2.7482	0.1137	3.3964	0.0793	2.9834
6	0.0888	2.5979	0.1183	3.2239	0.0830	2.8274
∞	0.1120	1.6979	0.1275	2.8853	0.0913	2.4933

Appendix D

Monotonicity of BSCs with Respect to MB $\mathcal{H}_{TI}$ Algorithms for Decoding Irregular Codes

In this appendix, we prove that the convergence of error probability of an MB $\mathcal{H}_{TI}$ algorithm to zero for a channel parameter ε implies its convergence to zero for every channel parameter $\varepsilon' < \varepsilon$. Consider the ensemble associated to the distribution pair $(\vec{\Lambda}, \vec{\rho})$, and suppose that this ensemble is decoded by the corresponding MB $\mathcal{H}_{TI}$ algorithm over two BSCs with crossover probabilities ε and $\varepsilon' < \varepsilon$. Consider further β_ε and $\beta_{\varepsilon'}$, the sequences of expected fraction of incorrect messages with iteration number for channel parameters ε and ε' , respectively, and also assume that in a given iteration ℓ_o , $p_{\varepsilon, \varepsilon'}^{(\ell_o)} \leq p_{\varepsilon, \varepsilon}^{(\ell_o)}$. Then

$$\begin{aligned}
 p_{\varepsilon, \varepsilon'}^{(\ell_o+1)} &\stackrel{(i)}{=} \sum_{j \geq 2} \sum_{\omega=0}^{\lfloor j/2 \rfloor - 1} \Lambda_{j, \omega} \cdot f_{\varepsilon'}^{j, \omega, \vec{\rho}} \left(p_{\varepsilon, \varepsilon'}^{(\ell_o)} \right) \\
 &\stackrel{(ii)}{=} \sum_{j \geq 2} \sum_{\omega=0}^{\lfloor j/2 \rfloor - 1} \Lambda_{j, \omega} \cdot f_{\varepsilon}^{j, \omega, \vec{\rho}} \left(p_{\varepsilon, \varepsilon'}^{(\ell_o)} \right) \\
 &\stackrel{(iii)}{=} \sum_{j \geq 2} \sum_{\omega=0}^{\lfloor j/2 \rfloor - 1} \Lambda_{j, \omega} \cdot f_{\varepsilon}^{j, \omega, \vec{\rho}} \left(p_{\varepsilon, \varepsilon}^{(\ell_o)} \right) \\
 &\stackrel{(iv)}{=} p_{\varepsilon, \varepsilon}^{(\ell_o+1)},
 \end{aligned} \tag{D.1}$$

where equalities (i) and (iv) follow from (5.1.5), and inequalities (ii) and (iii) are based on the fact that $h_{\varepsilon}^{\tilde{\Lambda}, \tilde{\rho}}(x)$ is increasing in ε and x , respectively. As, by assumption, $p_{e, \varepsilon'}^{(0)} = \varepsilon' < \varepsilon = p_{e, \varepsilon}^{(0)}$, we can use finite induction to deduce that $p_{e, \varepsilon'}^{(\ell)} \leq p_{e, \varepsilon}^{(\ell)}$, $\ell \geq 0$, and subsequently conclude that $\lim_{\ell \rightarrow \infty} p_{e, \varepsilon'}^{(\ell)} = 0$ is implied by $\lim_{\ell \rightarrow \infty} p_{e, \varepsilon}^{(\ell)} = 0$.

Appendix E

Stability Condition of the Fixed-Point Located at Zero for MB $\mathcal{H}_{TI}$ and $\mathcal{H}_{ST}$ Algorithms

In this appendix, we derive the stability condition of the fixed-point located at zero by calculating the supremum value of ε satisfying the inequality $\frac{\partial}{\partial x} h_{\varepsilon}^{\tilde{\Lambda}, \tilde{\rho}}(0) < 1$.

Redefining $\zeta^+(x) \equiv (1 + \rho(1 - 2 \cdot x))/2$ and $\zeta^-(x) \equiv (1 - \rho(1 - 2 \cdot x))/2$, it is not difficult to see from (5.1.3) that

$$\begin{aligned} \frac{\partial}{\partial x} f_{\varepsilon}^{j, \omega, \tilde{\rho}}(x) &= ([j/2] + \omega) \cdot \binom{j-1}{[j/2] + \omega} \cdot \frac{d}{dz} \rho(z) \Big|_{z=1-2 \cdot x} \\ &\quad \cdot \left(\varepsilon \cdot (\zeta^+(x))^{[j/2] + \omega - 1} \cdot (\zeta^-(x))^{[j/2] - \omega - 1} \right. \\ &\quad \left. + (1 - \varepsilon) \cdot (\zeta^+(x))^{[j/2] - \omega - 1} \cdot (\zeta^-(x))^{[j/2] + \omega - 1} \right). \end{aligned} \quad (\text{E.1})$$

Using (E.1), it can be seen that

$$\frac{\partial}{\partial x} f_{\varepsilon}^{j, \omega, \tilde{\rho}}(0) = \begin{cases} \rho'(1), & \text{if } j = 2, \\ \varepsilon \cdot (j-1) \cdot \rho'(1), & \text{if } j \geq 3 \text{ and } \omega = \lfloor j/2 \rfloor - 1, \\ 0, & \text{elsewise.} \end{cases} \quad (\text{E.2})$$

Therefore, it follows from (5.1.5) that

$$\frac{\partial}{\partial x} h_{\varepsilon}^{\tilde{\Lambda}, \tilde{\rho}}(0) = \Lambda_{2,0} \cdot \rho'(1) + \varepsilon \cdot \rho'(1) \cdot \sum_{j \geq 3} \Lambda_{j, \lfloor j/2 \rfloor - 1} \cdot (j-1), \quad (\text{E.3})$$

and thus

$$\frac{\partial}{\partial x} h_{\varepsilon}^{\vec{\lambda}, \vec{\rho}}(0) < 1 \Rightarrow \varepsilon < \frac{1 - \Lambda_{2,0} \cdot \rho'(1)}{\rho'(1) \cdot \sum_{j \geq 3} \Lambda_{j, \lfloor j/2 \rfloor - 1} \cdot (j - 1)}. \quad (\text{E.4})$$

This, together with $\varepsilon \geq 0$, results in (5.1.7).

It is worth mentioning that the condition we just derived is basically the same as the one derived in [70] for $\mathcal{G}_A$. The reason is the fact that $MB^{j\omega, \vec{\rho}}$ is unconditionally stable at fixed-point zero for all values of ω other than $\lfloor j/2 \rfloor - 1$. (See (E.2).) Equally mention-worthy is the close relation between the condition derived above, and the one derived in [52] for $\mathcal{G}_B$. We know that under $\mathcal{G}_B$, as iterations proceed, all the variable-nodes eventually switch to MB^0 , which is unconditionally stable for every variable-node degree $j \geq 4$. (See (E.2).) Consequently, (E.4) reduces to

$$\frac{\partial}{\partial x} h_{\varepsilon}^{\vec{\lambda}, \vec{\rho}}(0) < 1 \Rightarrow \varepsilon < \frac{1 - \lambda_2 \cdot \rho'(1)}{2 \cdot \lambda_3 \cdot \rho'(1)}, \quad (\text{E.5})$$

which is the generalized stability condition for $\mathcal{G}_B$ over $C^\infty(\vec{\lambda}, \vec{\rho})$.

Bibliography

- [1] R. G. Gallager, *Low density parity check codes*. Sc.D. thesis, Massachusetts Institute of Technology, 1960.
- [2] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 21–28, Jan. 1962.
- [3] R. G. Gallager, *Low-Density Parity-Check Codes*. No. 21 in Research Monographs Series, Cambridge, MA: M.I.T. Press, 1963.
- [4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes (1)," in *Conf. Rec. IEEE ICC '93*, vol. 2, (Geneva, Switzerland), pp. 1064–1070, May 1993.
- [5] Proc. Int'l Symp. Turbo Codes, ENST de Bretagne, (Brest, France), Sept. 1997.
- [6] Proc. Int'l Symp. Turbo Codes, ENST de Bretagne, (Brest, France), Sept. 2000.
- [7] "Special Issue on Codes on Graphs and Iterative Algorithms," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, Feb. 2001.
- [8] "The Turbo Principle: From Theory to Practice," *IEEE J. Select. Areas Commun.*, vol. 19, no. 5, May 2001.
- [9] "The Turbo Principle: From Theory to Practice II," *IEEE J. Select. Areas Commun.*, vol. 19, no. 9, Sept. 2001.
- [10] Proc. Int'l Symp. Turbo Codes, ENST de Bretagne, (Brest, France), Sept. 2003.

- [11] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, pp. 533–547, Sept. 1981.
- [12] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599–618, Feb. 2001.
- [13] K. S. Zigangirov and M. Lentmaier, "On the asymptotic iterative decoding performance of low-density parity-check codes," in *Proc. Int'l Symp. Turbo Codes & Related Topics*, (Brest, France), pp. 39–42, ENST de Bretagne, Sept. 2000.
- [14] P. Zarrinkhat and A. H. Banihashemi, "Density evolution and convergence properties of majority-based algorithms for decoding low-density parity-check (LDPC) codes," in *Proc. Allerton Conf. Commun., Control, and Computing*, (Allerton, IL), pp. 1425–1434, University of Illinois at Urbana-Champaign, Oct. 2002.
- [15] P. Zarrinkhat and A. H. Banihashemi, "Threshold values and convergence properties of majority-based algorithms for decoding regular low-density parity-check codes," *IEEE Trans. Commun.*, vol. 52, pp. 2087–2097, Dec. 2004.
- [16] A. H. Banihashemi, A. Nouh, and P. Zarrinkhat, "Hybrid (multistage) decoding of low-density parity-check (LDPC) codes," in *Proc. Allerton Conf. Commun., Control, and Computing*, (Allerton, IL), pp. 1437–1438, University of Illinois at Urbana-Champaign, Oct. 2002.
- [17] P. Zarrinkhat and A. H. Banihashemi, "Density evolution and convergence properties of hybrid iterative algorithms for decoding low-density parity-check codes," in *Proc. Can. Work. Inform. Theory*, (Waterloo, ON, Canada), pp. 40–43, University of Waterloo, May 2003.
- [18] P. Zarrinkhat and A. H. Banihashemi, "Hybrid decoding of low-density parity-check codes," in *Proc. Int'l Symp. Turbo Codes & Related Topics*, (Brest, France), pp. 503–506, ENST de Bretagne, Sept. 2003.

- [19] P. Zarrinkhat and A. H. Banihashemi, "Hybrid hard-decision iterative decoding of regular low-density parity-check codes," in *Conf. Rec. IEEE ICC '04*, (Paris, France), pp. 435–439, June 2004.
- [20] P. Zarrinkhat and A. H. Banihashemi, "Hybrid hard-decision iterative decoding of regular low-density parity-check codes," *IEEE Commun. Lett.*, vol. 8, pp. 250–252, Apr. 2004.
- [21] P. Zarrinkhat, A. H. Banihashemi, and H. Xiao, "Time-invariant and switch-type hybrid iterative decoding of low-density parity-check codes," *Ann. Télécommun. / Ann. Telecommun.*, vol. 60, n° 1-2, pp. 103–131, Jan.-Feb. 2005.
- [22] P. Zarrinkhat and A. H. Banihashemi, "Hybrid hard-decision iterative decoding of irregular low-density parity-check codes," in *Proc. Allerton Conf. Commun., Control, and Computing*, (Allerton, IL), p. accepted for publication, University of Illinois at Urbana-Champaign, Aug. 2004.
- [23] C. E. Shannon, "A mathematical theory of communication," *Bell Sys. Tech. J.*, vol. 27, pp. 379–423 & 623–656, July & Oct. 1948.
- [24] R. W. Hamming, "Error detecting and error correcting codes," *Bell Sys. Tech. J.*, vol. 29, pp. 147–160, Apr. 1950.
- [25] M. J. E. Golay, "Notes on digital coding," *Proc. IRE*, vol. 37, p. 657, June 1949.
- [26] D. E. Muller, "Metric properties of boolean algebra and their application to switching circuits," Report 46, Digital Computer Laboratory, University of Illinois, Apr. 1953.
- [27] D. E. Muller, "Applications of boolean algebra to switching circuit design and to error detection," *IRE Trans. Elec'c Comp.*, vol. EC-3, pp. 6–12, 1954.
- [28] I. S. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *IRE Trans. Inform. Theory*, pp. 38–49, Sept. 1954.
- [29] P. Elias, "Coding for noisy channels," *IRE Conv. Rec.*, p't 4, pp. 37–46, 1955.

- [30] J. M. Wozencraft, "Sequential decoding," Technical Report 325, Research Laboratory of Electronics, Massachusetts Institute of Technology, 1957.
- [31] J. M. Wozencraft and B. Reiffen, *Sequential Decoding*. Cambridge, MA: M.I.T. Press, 1961.
- [32] P. A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres*, vol. 2, pp. 147–156, Apr. 1959.
- [33] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error-correcting binary group codes," *Inform. Cont.*, vol. 3, pp. 68–79, Mar. 1960.
- [34] W. W. Peterson, "Encoding and error-correction procedures for the bose-chaudhuri codes," *IRE Trans. Inform. Theory*, vol. IT-6, pp. 459–470, Sept. 1960.
- [35] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *SIAM J. Appl. Math.*, vol. 8, pp. 300–304, June 1960.
- [36] G. D. Forney, *Concatenated codes*. Sc.D. thesis, Massachusetts Institute of Technology, 1965.
- [37] G. D. Forney, *Concatenated Codes*. No. 37 in Research Monographs Series, Cambridge, MA: M.I.T. Press, 1966.
- [38] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, no. 2, pp. 260–269, Apr. 1967.
- [39] G. D. Forney, "The Viterbi algorithm," in *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.
- [40] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.
- [41] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley Series in Telecommunications, John Wiley & Sons, Inc., 1991.

- [42] V. V. Zyablov and M. S. Pinsker, "Estimation of the error-correction complexity for gallager low-density codes," *problemy peredachi informatsii*, vol. 11, no. 1, pp. 23–36, Jan.-Mar. 1975.
- [43] G. A. Margulis, "Explicit constructions of graphs without short cycles and low density codes," *Combinatorica*, vol. 2, no. 1, pp. 71–78, 1982.
- [44] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 569–584, Feb. 2001.
- [45] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inform. Theory*, vol. 47, pp. 585–598, Feb. 2001.
- [46] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619–637, Feb. 2001.
- [47] S.-Y. Chung, G. D. Forney, T. J. Richardson, and R. L. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, pp. 58–60, Feb. 2001.
- [48] N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *Euro. Trans. Telecommun.*, vol. 6, no. 5, pp. 513–525, Sept.-Oct. 1995.
- [49] N. Wiberg, *Codes and decoding on general graphs*. PhD thesis, Linköping University, Sweden, 1996.
- [50] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498–519, Feb. 2001.
- [51] M. Ardakani and F. R. Kschischang, "Gear-shift decoding," in *Proc. Biennial Symp. Commun.*, (Kingston, ON, Canada), pp. 86–90, Queen's University, June 2002.

- [52] M. Ardakani and F. R. Kschischang, "On the optimality of gallager's decoding algorithm B," in *Proc. Int'l Symp. Turbo Codes & Related Topics*, (Brest, France), pp. 165–168, ENST de Bretagne, Sept. 2003.
- [53] M. Mitzenmacher, "A note on low density parity check codes for erasures and errors," Technical Note 1998-017, SRC, Dec. 1998.
- [54] G. D. Forney, "On iterative decoding and the two-way algorithm," in *Proc. Int'l Symp. Turbo Codes & Related Topics*, (Brest, France), pp. 12–25, ENST de Bretagne, Sept. 1997.
- [55] R. W. Chang and J. C. Hancock, "On receiver structures for channels having memory," *IEEE Trans. Inform. Theory*, vol. IT-12, pp. 463–468, Oct. 1966.
- [56] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Ann. Math. Stat.*, vol. 37, no. 6, pp. 1554–1563, Dec. 1966.
- [57] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Maeto, CA: Morgan Kaufmann Publishers, Second ed., 1988.
- [58] G. Ungerboeck, "Nonlinear equalization of binary signals in gaussian noise," *IEEE Trans. Commun. Thech.*, vol. COM-19, no. 6, pp. 1128–1137, Dec. 1971.
- [59] D. Divsalar, S. Dolinar, and F. Pollara, "Low complexity turbo-like codes," in *Proc. Int'l Symp. Turbo Codes & Related Topics*, (Brest, France), pp. 73–80, ENST de Bretagne, Sept. 2000.
- [60] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.
- [61] F. Lehmann and G. M. Maggio, "An approximate analytical model of the message passing decoder of LDPC codes," in *Proc. Int'l Symp. Inform. Theory '02*, (Lausanne, Switzerland), p. 31, June-July 2002.

- [62] M. Ardakani and F. R. Kschischang, "A more accurate one-dimensional analysis and design of LDPC codes," *IEEE Trans. Commun.*, vol. 52, pp. 2106–2114, Dec. 2004.
- [63] S. ten Brink, "Iterative decoding trajectories of parallel concatenated codes," in *Proc. 3rd IEEE/ITG Conf. Source and Channel Coding*, (Munich, Germany), pp. 75–80, Jan. 2000.
- [64] H. E. Gamal, *On the theory and applications of space-time and graph based codes*. PhD thesis, University of Maryland, College Park, 2000.
- [65] H. E. Gamal and A. R. Hammons, "Analyzing the turbo decoder using the Gaussian approximation," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 671–686, Feb. 2001.
- [66] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, pp. 1727–1737, Oct. 2001.
- [67] M. Ardakani, *Efficient analysis, design and decoding of low-density parity-check codes*. PhD thesis, University of Toronto, 2004.
- [68] L. Bazzi, T. Richardson, and R. Urbanke, "Exact thresholds and optimal codes for the binary symmetric channel and Gallager's decoding algorithm A," in *Proc. Int'l Symp. Inform. Theory '00*, (Sorrento, Italy), p. 203, June 2000.
- [69] N. Miladinovic and M. Fossorier, "Improved bit flipping decoding of low-density parity check codes," in *Proc. Int'l Symp. Inform. Theory '02*, (Lausanne, Switzerland), p. 229, June-July 2002.
- [70] L. Bazzi, T. J. Richardson, and R. L. Urbanke, "Exact thresholds and optimal codes for the binary-symmetric channel and Gallager's decoding algorithm A," *IEEE Trans. Inform. Theory*, vol. 50, pp. 2010–2021, Sept. 2004.
- [71] J. L. Massey, *Threshold decoding*. PhD thesis, Massachusetts Institute of Technology, 1962.

- [72] J. L. Massey, *Threshold Decoding*. No. 20 in Research Monographs Series, Cambridge, MA: M.I.T. Press, 1963.
- [73] D. Burshtein and G. Miller, "Bounds on the performance of belief propagation decoding," *IEEE Trans. Inform. Theory*, vol. 48, pp. 112–122, Jan. 2002.
- [74] A. Khandekar and R. McEliece, "A lower bound on the iterative decoding threshold of irregular LDPC code ensembles," in *Proc. CISS '02 [CD-ROM]*, (Princeton, NJ), Princeton University, March 2002.
- [75] C.-C. Wang, S. R. Kulkarni, and H. V. Poor, "Density evolution for asymmetric memoryless channels," in *Proc. Int'l Symp. Turbo Codes & Related Topics*, (Brest, France), pp. 121–124, ENST Bretagne, Sept. 2003.
- [76] M. Lentmaier, *Towards a theory of codes for iterative decoding*. PhD thesis, Lund University, 2003.
- [77] P. Henrici, *Applied and Computational Complex Analysis*, vol. 1. New York: John Wiley & Sons, 1974.
- [78] A. Nouh and A. H. Banihashemi, "Reliability-based schedule for bit-flipping decoding of low-density parity-check codes," *IEEE Trans. Commun.*, vol. 52, pp. 2038–2040, Dec. 2004.
- [79] H. Saeedi, A. H. Banihashemi, and Q. Hong, "Asymptotic performance analysis of LDPC codes with channel estimation error," in *IEEE VTC2005-Spring*, accepted for presentation, (Stockholm, Sweden), May-June 2005.
- [80] S.-Y. Chung, *On the construction of some capacity-approaching coding schemes*. PhD thesis, Massachusetts Institute of Technology, 2000.
- [81] A. Anastasopoulos, "A comparison between the sum-product and the min-sum iterative detection algorithms based on density evolution," in *Proc. GLOBECOM '01*, vol. 2, (San Antonio, TX), pp. 1021–1025, Nov. 2001.